UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Uma Abordagem *Bottom-Up* para a Integração Semântica de Esquemas XML

por

RONALDO DOS SANTOS MELLO

Tese submetida a avaliação,
como requisito parcial para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, julho de 2002.

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Mello, Ronaldo dos Santos

Uma Abordagem *Bottom-Up* para a Integração Semântica de Esquemas XML / por Ronaldo dos Santos Mello. — Porto Alegre: PPGC da UFRGS, 2002.

145 f.: il.

Tese (doutorado) — . Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2002. Orientador: Heuser, Carlos Alberto.

1. Dados Heterogêneos. 2. Integração de Esquemas. 3. Dados Semi-Estruturados. 4. XML. 5. DTD. I. Heuser, Carlos Alberto. II. Título.

Agradecimentos

- Agradeço primeiramente ao meu orientador, prof. Carlos Alberto Heuser, pela disponibilidade constante durante o meu curso de doutorado e pelos inúmeros e valiosos conselhos dados. Não posso deixar de agradecer também a plena confiança que depositou em mim, aceitando ser meu orientador na UFRGS sem nenhuma restrição, mesmo após alguns acidentes de percurso no início do meu doutoramento;

- Agradeço à profa. Silvana Castano, da Universidade de Milão, Itália, pela acolhida e supervisão dadas durante o meu período de doutorado sanduíche. Obrigado pelas várias sugestões!

- Agradeço à profa. Cláudia Bauzer Medeiros, da UNICAMP, minha orientadora no primeiro ano de curso, pela confiança depositada e pelo apoio constante;

- Agradeço à CAPES, pela bolsa de doutorado;

- Agradeço à UFSC e ao departamento de Informática e de Estatística, por permitirem o afastamento das minhas atividades acadêmicas para cursar o doutorado;

- Agradeço ao Instituto de Informática da UFRGS por proporcionar a infra-estrutura adequada para a realização do doutorado;

- Agradeço aos meus pais, Heleno e Ione, por permitirem que eu tenha alcançado esta meta da minha vida profissional e por toda a preocupação que sempre tiveram comigo; aos meus irmãos, Lisiane, Daniel e Renan, pelo companheirismo e apoio; e aos meus avós Adriano e Nadir, pelo carinho dado! Um beijo em vocês todos e um beijinho especial no meu sobrinho Fabrício!

- Agradeço aos meus colegas e amigos do grupo de banco de dados: Carina, Vanessa, Renata Galante, Rodrigo, Sandro, Adriana Roma, e tantos outros que tive a oportunidade de conviver. Agradeço pelas sugestões dadas durante o desenvolvimento da tese e também pelo apoio e amizade. Agradeço também ao Sérgio Mergen e à Lai Yu Chin pelo desenvolvimento do primeiro protótipo!

- Agradeço a todos vocês que ficaram na torcida e me transmitiram muitos fluidos positivos: Pedrinho, Tita, Patrícia, Vicente, Marcelo e Marlise, Márcia, Laís, Clara, Karla, Gleiber, Isabel, Liane, Cláudia S., Walter, Tadeu, Marco. Um abraço forte em todos vocês!

- Por fim, agradeço a Deus, pelo Seu grande amor e por permitir que tudo isto seja possível! Muito obrigado!

Sumário

Lista de Abreviaturas

ARTEMIS	Analysis and Reconciliation Tool Environment for Multiple Information Systems
BDF	Bancos de Dados Federados
CXPath	Conceptual *XPath*
DAML	DARPA Agent Markup Language
DOM	Document Object Model
DTD	Document Type Definition
ER	Entity-Relationship model
MCC	Modelo Conceitual Canônico
MDB	Multidatabase
OIL	Ontology Inference Language
ORM	Object with Role Model
RDF	Resource Description Framework
SIF	Sistema de Informação Federado
W3C	The World Wide Web Consortium
XML	eXtensible Markup Language

Lista de Figuras

Lista de Tabelas

Resumo

XML (*eXtensible Markup Language*) é um padrão atual para representação e intercâmbio de dados semi-estruturados na *Web*. Dados semi-estruturados são dados não-convencionais cujas instâncias de uma mesma fonte de dados podem ter representações altamente heterogêneas. Em função disto, um esquema para estes dados tende a ser extenso para suportar todas as alternativas de representação que um dado pode assumir.

Parte do grande volume de dados disponível hoje na *Web* é composto por fontes de dados heterogêneas XML sobre diversos domínios do conhecimento. Para realizar o acesso a estas fontes, aplicações na *Web* necessitam de um mecanismo de *integração de dados*. O objetivo principal deste mecanismo é disponibilizar um esquema de dados global representativo dos diversos esquemas XML das fontes de dados. Com base neste esquema global, consultas são formuladas, traduzidas para consultas sobre os esquemas XML, executadas nas fontes de dados e os resultados retornados à aplicação.

Esta tese apresenta uma abordagem para a integração semântica de esquemas XML relativos a um domínio de aplicação chamada BInXS. BInXS adota um processo *bottom-up* de integração, no qual o esquema global é definido para um conjunto de esquemas XML representados através de DTDs (*Document Type Definitions*). A vantagem do processo *bottom-up* é que todas as informações dos esquemas XML são consideradas no esquema global. Desta forma, toda a informação presente nas fontes de dados pode ser consultada.

O processo de integração de BInXS é baseado em um conjunto de regras e algoritmos que realizam a *conversão* de cada DTD para um esquema canônico conceitual e a posterior *integração semântica* propriamente dita destes esquemas canônicos. O processo é semi-automático pois considera uma eventual intervenção de um usuário especialista no domínio para validar ou confirmar alternativas de resultado produzidas automaticamente.

Comparada com trabalhos relacionados, BInXS apresenta as seguintes contribuições: (i) uma *representação canônica conceitual para esquemas XML* que é o resultado de uma análise detalhada do modelo XML; (ii) um *método de unificação* que lida com as particularidades da integração de dados semi-estruturados e; (iii) uma *estratégia de mapeamento* baseada em expressões de consulta *XPath* que possibilita uma tradução simples de consultas globais para consultas a serem executadas nas fontes de dados XML.

Palavras-chave: Dados Heterogêneos, Integração de Esquemas, Dados Semi-Estruturados, XML, DTD.

TITLE: "A BOTTOM-UP APPROACH FOR THE SEMANTIC INTEGRATION OF XML SCHEMATA"

Abstract

XML (*eXtensible Markup Language*) is a common standard for semistructured data representation and exchange over the *Web*. Semistructured data are non-conventional data with instances that may have highly heterogeneous representations even in a same data source. Because of this feature, semistructured schemata are usually large to support all possible data representations.

Part of the large amount of data available over the *Web* is comprised by heterogeneous XML data sources related to several domains. In order to be able to access those sources, *Web* applications require a *data integration mechanism*. The main purpose of such mechanism is to produce a global schema for a set of XML schemata defined in the data sources. Based on this global schema, queries are formulated, translated into queries over the XML schemata, executed at the data sources and the query results returned to the application.

This thesis presents an approach for the semantic integration of XML schemata related to a same application domain that is called BInXS. BInXS follows a *bottom-up* integration process that builds the global schema from a set of XML schemata specified through DTDs (*Document Type Definitions*). The advantage of the *bottom-up* process is that all information defined in the XML schemata are considered in the global schema. Thus, all information available at the XML data sources may be queried.

The integration process of BInXS is based on a set of rules and algorithms that perform the *conversion* of each DTD to a canonical conceptual schema and further provides the *semantic unification* of these canonical schemata. This process is semi-automatic because consider the intervention of a domain expert user to validate or to confirm results that are automatically generated.

Compared to related work, the contributions of BInXS are: (i) a *canonical conceptual representation for an XML schema* that is obtained from a detailed analysis of the XML model; (ii) an *unification process* that deals with the particularities of semistructured data integration; and (iii) a *mapping strategy* based on *XPath* query expressions that provides a simple translation of global queries into queries to be executed at the XML data sources.

Keywords: Heterogeneous Data, Schema Integration, Semistructured Data, XML, DTD.

1 Introdução

XML (*eXtensible Markup Language*) é um padrão atual para representação e intercâmbio de dados na *Web* [EXT 2002]. Diversos domínios de aplicação, como *referências bibliográficas* [DBL 2002, SIG 99] e *comércio eletrônico*, representam informação em XML. Na área de comérico eletrônico, por exemplo, diversos esforços de padronização estão em andamento e envolvem XML [CXM 2002, EBi 2002, GCI 2002].

Um dado em XML é um *dado semi-estruturado* [ABI 2000]. Dados semi-estruturados são dados não convencionais cuja representação pode ser altamente heterogênea mesmo para instâncias em uma mesma fonte de dados. Por isto, seus esquemas tendem a ser extensos para suportar as diversas representações alternativas que um dado pode assumir.

Considerando o grande volume de informação disponível na *Web*, várias fontes XML heterogêneas com dados sobre um mesmo domínio de aplicação podem existir. Para realizar o acesso a estas informações, aplicações na *Web* devem desenvolver *mecanismos de integração de dados* que permitam consultas cujo escopo de atuação seja este conjunto de fontes XML. Integração de esquemas e dados XML é um tópico atual de pesquisa na comunidade de banco de dados, sendo inclusive um dos focos de atuação do grupo de *Web Semântica* do consórcio *W3C*[1] [WOR 2002a].

O foco desta tese é *integração de esquemas XML*. Alguns trabalhos relacionados a este tópico são encontrados na literatura [ROD 2001, LUD 99, REY 2001, JEN 2001, DOA 2001]. De modo geral, todos adotam um modelo uniforme de representação (modelo canônico) para esquemas XML e posteriormente geram um esquema global ou um conjunto de visões resultante da integração destes esquemas. Mesmo trazendo contribuições para este tópico de pesquisa, estes trabalhos apresentam os seguintes problemas:

- A modelagem canônica não leva em conta todas as particularidades do modelo de dados XML. Alguns deles definem ainda modelos com fraco poder de expressão para representar a intenção semântica dos dados;
- O processo de integração segue abordagens tradicionais de integração de banco de dados heterogêneos, que não consideram algumas características de dados semi-estruturados como representações mistas (texto e estruturada) e a existência de alternativas de representação para um mesmo dado.

Esta tese propõe uma *abordagem* detalhada para a *integração de esquemas XML* que trata estes problemas. Esta abordagem é chamada de *BInXS* . Os pontos fortes de BInXS são:

- Uma *representação canônica conceitual* para esquemas XML, com relacionamentos de associação e herança que modelam a intenção semântica dos dados XML. Este esquema conceitual é obtido através de um processo de *conversão do esquema XML* que analisa detalhadamente o modelo de dados XML;

[1] A W3C é um consórcio formado por acadêmicos e empresas com o objetivo de desenvolver tecnologias para a *Web*, sendo responsável inclusive pela criação da linguagem XML.

- Um processo de *integração semântica* destes esquemas conceituais que considera a determinação de equivalências e a resolução de conflitos para representações semi-estruturadas. Em particular, são tratados casos específicos na unificação de dois tipos de elementos (texto e estruturados) e elementos com representações alternativas. Um *esquema conceitual global* é o resultado deste processo.

A definição de um esquema conceitual global exige que *mapeamentos* de dados sejam mantidos para os esquemas XML para fins de consulta. Outra contribuição de BInXS é uma *estratégia de definição de mapeamentos* para elementos e relacionamentos do esquema global através de expressões de navegação *XPath*[2]. Trabalhos relacionados geralmente definem mapeamentos através de *visões*, que são especificações em uma linguagem de consulta associadas a um elemento do esquema global que recuperam elementos correspondentes nas fontes de dados locais. No momento da tradução de uma consulta global, as visões correspondentes a todos os elementos envolvidos na consulta devem ser analisadas para a montagem da consulta a ser enviada para a fonte de dados. O processo de tradução torna-se assim mais complexo. Uma expresão *XPath* não deixa de ser uma visão para uma informação XML, porém a sua adoção facilita a tradução de consultas uma vez que os elementos e relacionamentos definidos em uma consulta global são diretamente substituidos por expressões de navegação *XPath* que estão de acordo com a nomenclatura e a estrutura hierárquica do esquema XML da fonte de dados.

As atividades de conversão de um esquema XML e integração semântica de esquemas, citadas anteriormente, constituem o *processo de integração* de esquemas XML de BInXS. Este processo está inserido no contexto de um *sistema mediador* que realiza consultas a um conjunto de fontes XML heterogêneas de um mesmo domínio de aplicação. BInXS segue uma abordagem *bottom-up* de integração, ou seja, o esquema global é construído a partir do conjunto de esquemas definido para as fontes XML. Esta estratégia considera no esquema global todas as representações de uma mesma informação existentes nas fontes XML, possibilitando consultas a todo o universo de informações do domínio. A abordagem proposta por BInXS é discutida nos demais capítulos desta tese.

Esta tese está organizada em mais seis capítulos. O capítulo 2 apresenta o estado da arte em integração de dados, com ênfase para a integração de dados semi-estruturados. O capítulo 3 introduz a abordagem proposta pela tese e apresenta uma visão geral do processo de integração. O capítulo 4 discute o modelo canônico e a estratégia de mapeamento do esquema global para esquemas XML. Neste capítulo é apresentada ainda a linguagem de especificação de esquemas globais adotada. Os capítulos 5 e 6 apresentam em detalhe as atividades de conversão de um esquema XML para um esquema conceitual canônico e a integração semântica de esquemas conceituais, respectivamente. Finalmente, o capítulo 7 é dedicado às conclusões.

[2]XPath é um padrão da W3C para o acesso a dados XML.

2 Integração de Dados

Este capítulo apresenta o contexto no qual a tese se insere: *integração de dados*. As áreas de pesquisa relacionadas são apresentadas, assim como seus inter-relacionamentos, visando delimitar o escopo do problema tratado pela tese. Dentro deste escopo são descritos alguns trabalhos relacionados que serão posteriormente comparados com a solução proposta pela tese.

O estudo sobre as áreas relacionadas à tese produziram duas publicações: um tutorial sobre dados semi-estruturados publicado nos anais do *XV Simpósio Brasileiro de Banco de Dados (SBBD'2000)* [MEL 2000a] e um artigo sobre a aplicação de ontologias e modelos conceituais a dados semi-estruturados publicado nos anais da *XXVI Conferencia Latinoamericana de Informática (CLEI'2000)* [MEL 2000b]. Esta última publicação apresenta de forma resumida os resultados obtidos com a realização do *exame de qualificação de doutorado* [MEL 2000].

2.1 Gerenciamento de Dados Heterogêneos

Uma área de pesquisa existente desde a década de oitenta na comunidade de banco de dados é a área de *gerenciamento de dados heterogêneos*. A pesquisa nesta área é motivada pela necessidade das organizações, devido aos avanços em computação distribuída e redes de computadores, de compartilhar informação entre diferentes plataformas e sistemas de gerenciamento de dados [ELM 99].

O desenvolvimento de sistemas de gerenciamento de dados heterogêneos conduz ao problema da garantia de acesso transparente a sistemas de banco de dados heterogêneos e autônomos[1]. Soluções para este problema devem ser ambientes computacionais que disponibilizem, para usuários e aplicações, uma visão unificada das informações desejadas e presentes nestes bancos de dados, resolvendo os conflitos inerentes à heterogeneidade. Diversos tipos de heterogeneidade podem ocorrer entre sistemas de bancos de dados, desde a heterogeneidade de *hardware* (diferentes sistemas operacionais e protocolos de acesso) e *software* (diferentes sistemas de bancos de dados, com diferentes modelos de dados) até a heterogeneidade no nível de definição dos dados (diferentes esquemas de representação e interpretações semânticas) [WIE 93].

Sistemas de gerenciamento de bancos de dados heterogêneos têm sido denominados de sistemas de *Bancos de Dados Federados* ou sistemas *Multidatabase*. Não há consenso quanto a arquitetura adotada por cada um destes sistemas, uma vez que definições contraditórias são encontradas na literatura [SHE 90, LIT 90, OSZ 99, KIM 95]. Apesar disto, de acordo com o estudo mais recente de Elmagarmid et.al. [ELM 99], as seguintes definições são comumente utilizadas:

- *Bancos de Dados Federados (BDF)*: sistema que possui um ou mais esquemas globais (chamados esquemas federados) resultantes da integração dos esquemas dos bancos de dados locais. Consultas são feitas sobre estes esquemas

[1] Um sistema de banco de dados é dito autônomo se possui controle independente sobre seus dados e as operações sobre estes dados, decidindo quais informações compartilhar com o sistema de gerenciamento de dados heterogêneos do qual ele é componente.

federados e traduzidas para os esquemas de cada banco de dados participante da federação.

- *Multidatabase (MDB)*: sistema que não possui esquemas globais e sim uma linguagem multidatabase que suporta consultas a múltiplos bancos de dados ao mesmo tempo.

Esquemas federados são geralmente definidos de forma manual. Quando estes esquemas são mantidos pelo próprio administrador do BDF, tem-se um *BDF fortemente acoplado*. Quando os esquemas federados são mantidos pelos usuários da federação, através da definição de visões sobre os esquemas dos bancos de dados participantes, tem-se um *BDF fracamente acoplado*. As vantagens de uma federação fortemente acoplada são a transparência de localização e distribuição dos bancos de dados e a uniformidade de representação semântica dos múltiplos esquemas locais. As vantagens de uma federação fracamente acoplada são a flexibilidade dos usuários para definir as visões desejadas e a menor complexidade na manutenção dos esquemas federados frente a mudanças nos esquemas locais, uma vez que apenas as visões afetadas necessitam ser atualizadas.

A arquitetura típica de um BDF e de um MDB são mostradas na figura 2.1 (a) e 2.1 (b), respectivamente. Um BDF apresenta uma arquitetura em cinco níveis que corresponde a cinco tipos de esquemas [SHE 90]. Com exceção dos esquemas locais dos bancos de dados, todos os demais são definidos de acordo com um modelo canônico. Um *modelo canônico* é um modelo de dados que uniformiza a representação dos esquemas locais dos bancos de dados. Tal uniformização facilita a tarefa de integração de esquemas, uma vez que a heterogeneidade dos modelos de dados locais é abstraída. Um modelo canônico deve ter um poder de expressão igual ou superior aos modelos de dados locais, para que a semântica dos esquemas locais seja capaz de ser representada no esquema federado.

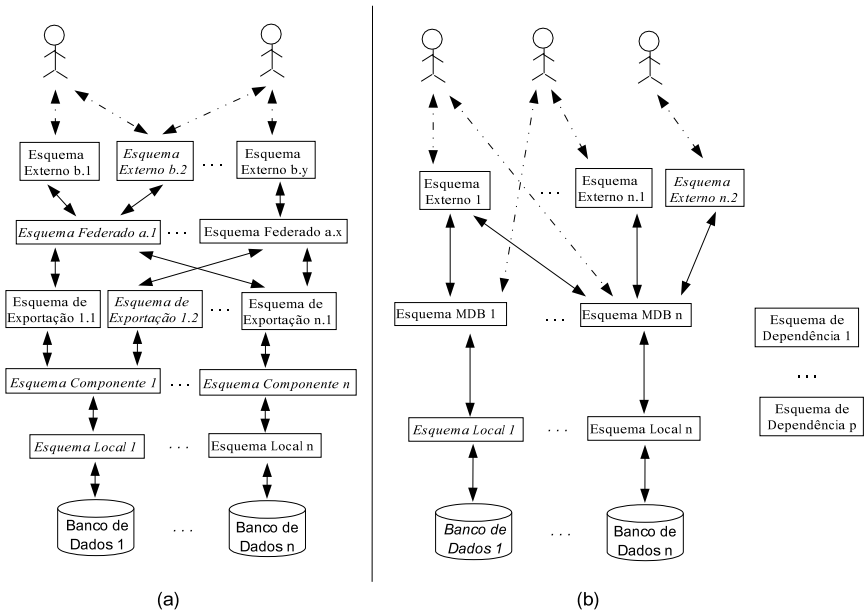


FIGURA 2.1 – Arquitetura tradicional de um banco de dados federado (a) e de um multidatabase (b)

A arquitetura de um BDF apresenta, no seu nível mais inferior, um conjunto de *esquemas locais* expressos no modelo de dados de cada banco de dados participante da federação. Um *esquema componente* é o resultado da tradução de um esquema local para o modelo canônico adotado, enquanto que um *esquema de exportação* representa uma porção do esquema componente que é compartilhado na federação. A integração de esquemas componentes resulta em um *esquema federado ou global*, a partir do qual diversos *esquemas externos* podem ser derivados, conforme o interesse de classes particulares de usuários e/ou aplicações. A arquitetura mínima de um BDF é aquela composta por esquemas locais, esquemas de exportação e um esquema federado.

A arquitetura de um MDB, dada a inexistência de um esquema global, permite o acesso direto a esquemas canônicos derivados de esquemas locais pelos usuários (figura 2.1 (b)) [LIT 90]. Estes esquemas podem ser *esquemas MDB*, que equivalem a esquemas componentes ou de exportação na arquitetura federada, ou *esquemas externos* definidos a partir de um ou mais esquemas MDB. Como não existe um esquema global, *esquemas de dependência* são criados para manter dependências entre esquemas locais, como equivalência de domínios de propriedades e *triggers* de atualização.

A abordagem MDB é considerada mais fracamente acoplada que um BDF fracamente acoplado uma vez que os esquemas locais podem ser vistos sem modificações, resultando em visões redundantes e heterogêneas de uma mesma informação. Apresenta as mesmas vantagens e desvantagens de uma federação fracamente acoplada, com o agravante de que os usuários do MDB devem entender os esquemas locais e resolver os conflitos de heterogeneidade no momento da definição de visões sobre múltiplos bancos de dados.

2.1.1 Integração de Esquemas

Independente da arquitetura utilizada por um sistema de gerenciamento de bancos de dados heterogêneos, a criação de esquemas globais ou externos exige que um processo de integração de esquemas seja realizado. *Integração de esquemas* é o processo de gerar um esquema integrado (esquema global) a partir de um conjunto de esquemas de entrada (esquemas locais), resolvendo as diversidades estruturais e semânticas existentes entre eles [BAT 86, ELM 99]. Este processo é complexo pois nem sempre a semântica dos esquemas locais está formalmente expressa ou documentada. Em função disto, ferramentas de integração de esquemas são *semi-automáticas* por natureza: elas podem, no máximo, sugerir conceitos candidatos a serem integrados e um usuário especialista aceita, rejeita ou modifica tal sugestão [RAH 2001].

O processo de integração de esquemas compreende as seguintes etapas, conforme mostra a figura 2.2 [BAT 86, ELM 99]:

1. *Pré-Integração*: análise dos esquemas locais a fim de definir a política de integração, como por exemplo, a ordem a ser seguida na integração dos esquemas e se a integração de um esquema é total ou parcial. Assume-se que todos os esquemas locais estão representados no mesmo modelo canônico;

2. *Comparação de Esquemas*: determinação de correspondências e conflitos entre conceitos de esquemas diferentes através da análise de suas características,

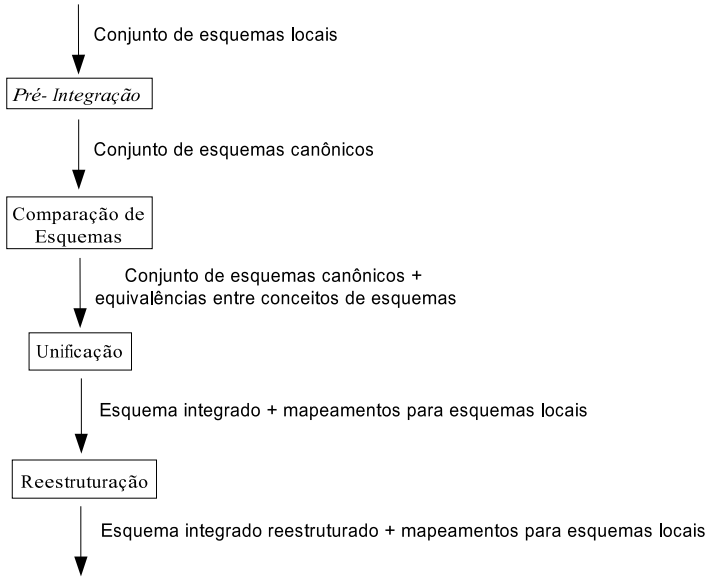


FIGURA 2.2 – Etapas de um processo de integração de esquemas

como atributos, relacionamentos e restrições de integridade. Relacionamentos semânticos inter-esquema, como generalização e agregação, são geralmente identificados;

3. *Unificação*: resolução de conflitos entre conceitos presentes nos esquemas locais e produção do esquema integrado ou global. Os seguintes critérios devem ser atendidos:

   - *Completude e corretude*: o esquema global deve conter todos os conceitos presentes nos esquemas locais de forma semanticamente correta;
   - *Minimalidade*: um conceito representado em mais de um esquema local deve ser representado uma única vez no esquema global (o conceito é unificado);
   - *Compreensão*: o esquema global deve ser de fácil compreensão para usuários e projetistas, ou seja, dentre várias possíveis representações, a mais clara deve ser escolhida.

   Informações de mapeamento entre conceitos no esquema global e esquemas locais são também geradas nesta etapa e mantidas em um catálogo específico;

4. *Reestruturação*: realização de atividades como inclusão de relações inter-esquema e simplificação do esquema integrado, como a remoção de relacionamentos redundantes, por exemplo. Relacionamentos inter-esquema podem ser eventualmente incluídos no esquema global na fase de Unificação.

Diversas taxonomias estão associadas a um processo de integração de esquemas. Estas taxonomias, mostradas na figura 2.3, dizem respeito a diferentes estratégias para a unificação de esquemas, técnicas de determinação de equivalências semânticas entre conceitos em esquemas diferentes, conflitos de heterogeneidade a

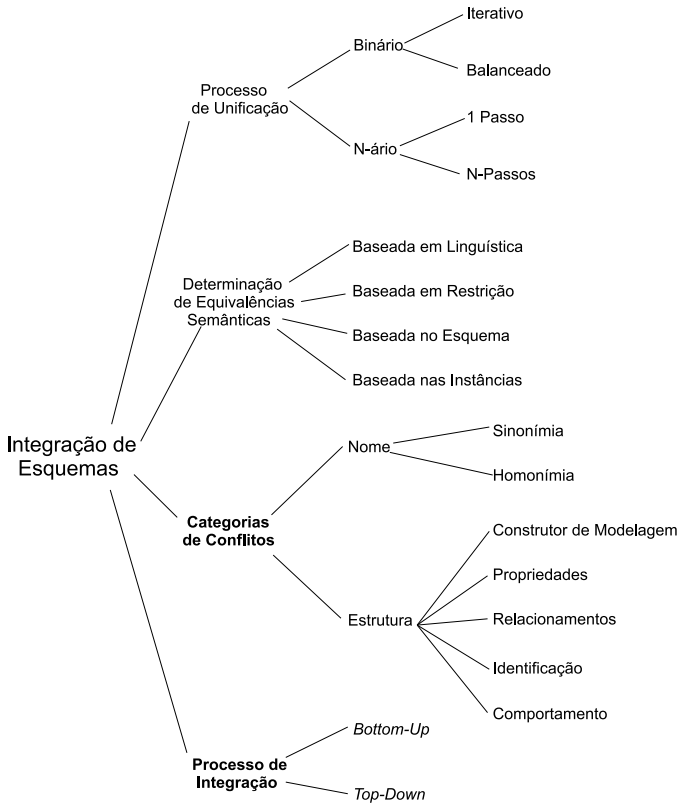


FIGURA 2.3 – Taxonomias de integração de esquemas

serem considerados na comparação de esquemas e a maneira como o processo de integração pode ser executado.

Na etapa de *Comparação de Esquemas*, diferentes técnicas podem ser aplicadas na determinação de equivalências semânticas entre conceitos de esquemas diferentes [RAH 2001]. Estas técnicas não são mutuamente exclusivas. Idealmente, todas elas deveriam ser suportadas por um processo de integração para aumentar o seu nível de automação e a sua qualidade.

A técnica *baseada em linguística* considera afinidade de nomes ou descrições textuais dos conceitos dos esquemas. Esta técnica geralmente se vale de informações auxiliares providas por *Thesauri* ou decisões prévias de unificação. A técnica *baseada em restrição* considera afinidade de características estruturais e de domínio associadas aos conceitos e suas propriedades.

Pode-se, ainda, analisar apenas informações sobre os esquemas (*baseada no esquema*) ou sobre os conteúdos dos dados (*baseada nas instâncias*) para determinar equivalências. No último caso geralmente são empregadas tecnologias de processamento de linguagem natural e aprendizado de máquina.

A etapa de *Comparação de Esquemas* necessita ainda determinar conflitos entre conceitos de esquemas diferentes. Em geral, duas categorias de conflitos são encontradas [BAT 86]. Conflitos de *nome* estão relacionados a nomenclatura dos conceitos. Deve-se considerar os casos de *sinonímia* (nomes diferentes dados a conceitos semanticamente equivalentes) e *homonímia* (nomes iguais dados a conceitos semanticamente diferentes). Dicionários léxicos são ferramentas importantes neste contexto para detectar não apenas sinônimos mas também outros tipos de relações

terminológicas inter-esquema, como generalização.

Conflitos de *estrutura* estão relacionados a representação dos conceitos. Esta categoria contempla diversos tipos de heterogeneidade, como *construtores de modelagem* (um mesmo conceito definido como uma entidade ou como um atributo, por exemplo), *propriedades* (conceitos com propriedades diferentes, propriedades com domínios diferentes, por exemplo), *relacionamentos* (diferenças de cardinalidade e opcionalidade), *identificação* (identificadores diferentes) e *comportamento* (diferentes políticas de atualização e restrições de integridade, por exemplo).

Na etapa de *Unificação*, duas estratégias de processamento podem ser aplicadas [BAT 86]:

- *Binária*: a unificação ocorre entre pares de esquemas locais, em vários passos. A vantagem desta estratégia é a redução da complexidade de comparação e unificação em cada passo. Esta estratégia é classificada ainda em *Iterativa* (unifica-se sequencialmente pares de esquemas) ou *Balanceada* (unifica-se em paralelo pares de esquemas). A primeira permite que se possa dar prioridade de unificação a esquemas com maior grau de afinidade semântica e a segunda reduz o tempo de processamento;

- *N-ária*: a unificação ocorre entre mais de dois esquemas, podendo ser realizada em um ou vários passos. A vantagem desta estratégia é a redução do número de passos de unificação. Caso a estratégia de vários passos seja utilizada, é possível dar prioridade de unificação a esquemas com maior grau de afinidade semântica.

O processo de integração de esquemas tradicional, mostrado na figura 2.2, é um processo *bottom-up*, uma vez que inicia pelos esquemas locais dos bancos de dados e finaliza com a definição do esquema global. Este processo, porém, pode ocorrer no sentido contrário, sendo chamado de processo *top-down* ou *integração de visões* [SHE 90].

Um processo *top-down* parte do pressuposto de que existem aplicações com requisitos de dados particulares e estas aplicações definem esquemas de dados que devem ser associados aos esquemas dos bancos de dados locais. As etapas de um processo *top-down* consistem em: *(i)* definir os esquemas externos (esquemas das aplicações); *(ii)* unificar estes esquemas, gerando o esquema global; *(iii)* identificar os bancos de dados onde os dados requiridos estão disponíveis e gerar os esquemas de exportação relativos a estes dados; *(iv)* determinar o mapeamento entre os conceitos do esquema global e os conceitos dos esquemas de exportação.

Uma abordagem de integração *top-down* reflete estritamente os requisitos de dados de uma ou mais aplicações. Conseqüentemente, a semântica dos esquemas locais é menos relevante que a semântica dos esquemas externos. Os esquemas externos podem inclusive definir dados abstratos ou derivados, como valores agregados ou agrupamentos de informações em faixas de valores.

Por outro lado, o processo *bottom-up*, justamente por levar em conta as semânticas locais, conduz a um esquema global mais natural para o domínio em questão. Com base neste esquema global, é possível uma gama mais ampla de consultas aos bancos de dados locais. Por isso, neste trabalho é considerado o processo tradicional de integração de dados heterogêneos.

2.2 Integração de Dados Semi-Estruturados

A pesquisa na área de gerenciamento de dados heterogêneos possibilitou o desenvolvimento de sistemas BDFs e MDBs, que permitem o acesso integrado a múltiplos bancos de dados heterogêneos. Entretanto, no decorrer da década de noventa, constatou-se que os requisitos para integração de dados mudaram. Problemas técnicos, relacionados ao controle da distribuição física e da gerência de redes, foram resolvidos com o uso da *Internet* e de tecnologias como CORBA e Java. Por outro lado, o número de fontes de dados aumentou consideravelmente com a popularização da *Web* como meio de publicação de informações [BUS 99].

Atualmente, grande parte dos dados disponíveis para acesso eletrônico não são mantidos em bancos de dados, mas em coleções de arquivos na *Web*. Isto porque a natureza destes dados (dados *Web*) é diferente dos dados tradicionais de bancos de dados, uma vez que instâncias de uma mesma informação em uma mesma fonte de dados podem ser diferentes. Tais dados são chamados de *dados semi-estruturados* [ABI 97, FLO 97, BUN 97, ABI 2000]. Tomando como exemplo um *site* na *Web* com informações curriculares sobre *docentes* de uma Universidade, cada docente pode apresentar as informações do seu *curriculum vitae* de uma maneira diferente. Um docente pode criar um pequeno texto informal descrevendo dados pessoais e experiência profissional, enquanto outro pode criar um documento organizado em seções e subseções, com referências para empresas e instituições onde trabalhou.

Dados semi-estruturados não são nem estritamente tipados, como dados tradicionais, nem completamente não-estruturados. Na verdade, são um misto de texto e estrutura. Um dado semi-estruturado pode ou não ter um esquema associado. Quando existe tal esquema, este tende a ser extenso para contemplar todas as irregularidades estruturais que as suas instâncias podem apresentar. Quando não existe um esquema, uma análise do dado deve ser feita para que a sua estrutura seja extraída. Independente da existência ou não de um esquema, um dado semi-estruturado é *auto-descritivo*, ou seja, a sua estrutura de representação faz parte da sua definição.

Um esquema para dados semi-estruturados é uma estrutura hierárquica na forma de um *grafo direcionado rotulado*. O vértice raiz da hierarquia identifica um objeto semi-estruturado propriamente dito, vértices internos identificam objetos componentes e arestas são referências para objetos componentes.

A figura 2.4 mostra um exemplo de um objeto semi-estruturado chamado *publications*. Este objeto mantém um conjunto de informações sobre artigos científicos.

O esquema dos objetos semi-estruturados da figura 2.4 pode ser deduzido a partir da estrutura de representação de *publications*. Objetos *article*, por exemplo, possuem *title*, *version*, *author* e *conference* como propriedades. A heterogeneidade de instâncias semi-estruturadas pode ser verificada para objetos do tipo *author*: o autor *Ronaldo Mello* possui *name*, várias ocorrências de *eMail* e *address* como propriedades. O autor *Carlos Heuser* possui *name*, *institution* e *rank* como propriedades.

2.2.1 XML

Um exemplo de dado semi-estruturado é um dado definido em XML. XML (*eXtensible Markup Language*) é um padrão popular do consórcio W3C (*World Wide*

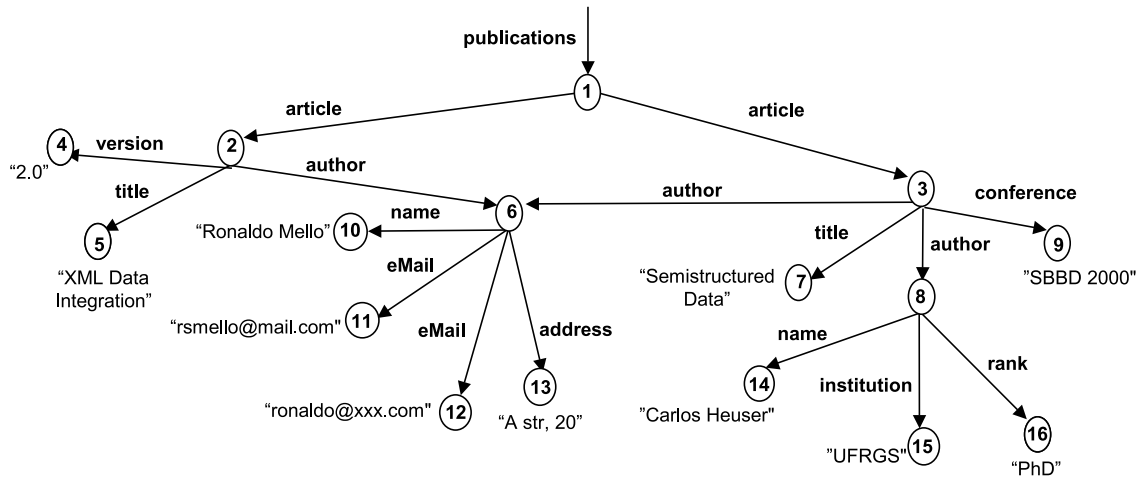


FIGURA 2.4 – Exemplo de um objeto semi-estruturado

Web Consortium) para publicação e intercâmbio de informações na *Web* [EXT 2002, BRA 2000]. XML é uma linguagem de marcação, ou seja, utiliza *tags* para delimitar informação. Diferente de HMTL, uma *tag* em XML tem caráter descritivo, ou seja, indica um significado para a informação delimitada por ela[2].

Um dado XML é chamado de *elemento*. Um elemento é formado por uma *tag* inicial e uma *tag* final que o delimitam e pelo seu *conteúdo* que pode ser um conjunto de elementos componentes, um conteúdo textual ou um misto de texto e elementos componentes. Um elemento pode ter *atributos*, que descrevem algumas de suas propriedades na *tag* inicial. Um documento XML é formado por um ou mais dados XML.

Os dados de um documento XML podem estar de acordo com um esquema definido através de uma *DTD*. Uma DTD (*Document Type Definition*) é uma gramática que restringe a forma como um conjunto de elementos pode ser organizado hierarquicamente. A cláusula `<!ELEMENT ...>` define um elemento e a cláusula `<!ATTLIST ...>` define um ou mais atributos de um elemento. A figura 2.5 (a) mostra um exemplo de uma DTD e a figura 2.5 (b) mostra um documento XML válido para esta DTD.

Os dados presentes no documento XML correspondem aos objetos semi-estruturados mostrados na figura 2.4. *publications* é o elemento raiz da hierarquia, sendo composto por um ou mais elementos *article*. Cada elemento *article*, por sua vez, possui um elemento *title*, um ou mais elementos *author* e um elemento opcional *conference*. *article* pode ter um atributo *version* associado. Um elemento *author* tem um elemento *name*, zero ou mais elementos *eMail* e *address* e elementos opcionais *institution* e *rank*. Elementos que possuem apenas um conteúdo textual são declarados como *#PCDATA*.

2.2.2 Arquiteturas Baseadas em Mediadores

A consideração de fontes de dados *Web* heterogêneas em sistemas de BDFs e MDBs introduz novos problemas de integração de esquemas pois as fontes de dados

[2]Em HTML, *tags* são instruções de formatação de dados utilizadas por *browsers Web*.

```
<!---------------------  DTD: Publ.DTD ---------------------->
<!ELEMENT publications(article+)>
<!ELEMENT article(title, author+, conference?)>
<!ATTLIST article version CDATA #IMPLIED>
<!ELEMENT author(name, institution?, eMail*, address*, rank?)>
<!ELEMENT conference(#PCDATA)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT institution(#PCDATA)>
<!ELEMENT eMail(#PCDATA)>
<!ELEMENT address(#PCDATA)>
<!ELEMENT rank(#PCDATA)>
```

(a)

```
<?XML version="1.0" encoding="UTF-8" ?>
<!DOCTYPE publications SYSTEM Publ.DTD>

<publications>
<article version = "2.0">
   <title>XML Data Integration</title>
   <author>
      <name>Ronaldo Mello</name>
      <eMail>rsmello@mail.com</eMail>
      <eMail>ronaldo@xxx.com</eMail>
      <address>A str.,20</address>
   </author>
</article>
<article>
   <title>Semistructured Data</title>
   <author>
      <name>Ronaldo Mello</name>
      <eMail>rsmello@mail.com</eMail>
      <eMail>ronaldo@xxx.com</eMail>
      <address>A str,20</address>
   </author>
   <author>
      <name>Carlos Heuser</name>
      <institution>UFRGS</institution>
      <rank>PhD</rank>
   </author>
</article>
</publications>
```

(b)

FIGURA 2.5 – Uma DTD (a) e um documento XML definido de acordo com esta DTD (b)

podem não ter um esquema associado. O mecanismo de visões, tipicamente adotado por sistemas MDB, torna-se inadequado uma vez que fontes de dados *Web* não são acessíveis por uma linguagem de consulta, dada a inexistência *a priori* de um sistema de gerenciamento de dados para estas fontes.

Para lidar com a problemática de integração de fontes semi-estruturadas foi empregada uma arquitetura de gerenciamento de dados heterogêneos chamada *arquitetura baseada em mediadores*. Esta arquitetura está associada a uma recente categoria de sistema de gerenciamento de dados heterogêneos, chamada de *sistemas de informação federados*(SIFs) [BUS 99].

A arquitetura baseada em mediadores é centrada em um conjunto de componentes independentes e interoperáveis chamados *mediadores* e *wrappers*. A figura 2.6 ilustra a arquitetura de um sistema baseado em mediadores.

A *camada de base* da arquitetura é formada por um conjunto de fontes de dados que não necessariamente são apenas sistemas de bancos de dados, mas também sistemas de arquivos, coleções de documentos e páginas HTML, dentre outras. A *camada de wrapper* tem por finalidade o controle de acesso a fontes de dados. Um

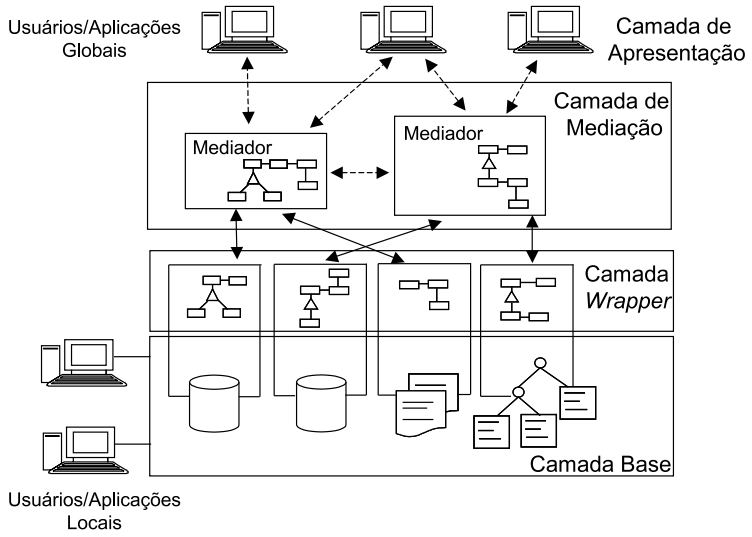


FIGURA 2.6 – Arquitetura de um sistema baseado em mediadores

wrapper é um módulo de *software* responsável por este controle, disponibilizando dados de uma fonte no formato desejado pela camada de mediação. Considerando a arquitetura de um BDF, a camada de *wrapper* atua na geração e controle de esquemas componentes e de exportação.

A *camada de mediação* é composta por um conjunto de mediadores que podem se comunicar entre si. Um *mediador*, como o próprio nome sugere, é um módulo de *software* que media entre o usuário ou aplicação na *camada de apresentação* e as fontes de dados [WIE 92]. Considerando a arquitetura de um BDF, esta camada atua na geração e controle de esquemas federados e externos e os mapeamentos necessários entre esquemas. Em geral, cada mediador gerencia um esquema federado e seus mapeamentos.

Um *SIF Baseado em Mediadores* (SIFBM) é um caso particular de um SIF que adota a arquitetura recém descrita. Na verdade, um SIF apresenta uma camada intermediária genérica, chamada *camada de federação*, entre a camada de *wrapper* e a camada de apresentação, responsável pelo gerenciamento da visão integrada dos esquemas de várias fontes de dados. Quando esta camada é composta por mediadores e o sistema permite apenas operações de consulta as fontes de dados, tem-se então um SIFBM [BUS 99]. O termo SIFBM se adequa melhor ao contexto de fontes de dados *Web*, que são *read-only* por natureza.

2.3 Trabalhos Relacionados

A pesquisa e o desenvolvimento na área de integração de dados semi-estruturados começaram a surgir na década passada. Projetos pioneiros foram o *TSIMMIS* [CHA 94, TSI 98] da Universidade de Stanford e o *Garlic* [CAR 95, GAR 99] do laboratório de pesquisas da IBM/Almaden. Ambos são sistemas que realizam um processo manual de integração com o suporte de ferramentas gráficas e linguagens de definição de visões integradas de dados. As abordagens utilizadas, entretanto, são diferentes. *TSIMMIS* segue a linha de MDB, não definindo um esquema global

e sim permitindo que o usuário defina seus mediadores, que agregam uma ou mais visões. *Garlic* segue a linha de BDF, na qual usuários especialistas definem um esquema global formado por classes de conceitos do domínio e correspondências entre estas classes e conceitos locais.

Recentemente, diversos outros trabalhos estão sendo propostos. O projeto *MIX* [LUD 99], a ferramenta *Xyleme* [REY 2001] e os sistemas *LSD* [DOA 2001], *DIXSE* [ROD 2001] e de *Jensen et. al.* [JEN 2001] são específicos para a integração de fontes XML. *MIX* adota a abordagem do *TSIMMIS*, não definindo um esquema global e construindo esquemas integrados a partir de DTDs locais através de uma linguagem de definição de visões. Já *Xyleme* e *LSD* realizam a geração de DTDs globais. *Xyleme* considera informação presente nos esquemas das DTDs locais e o auxílio de Thesauri. *LSD* considera informação presente nas instâncias XML e técnicas de aprendizado de máquina para deduzir correspondências entre elementos de DTDs, com base em uma integração manual preliminar. *DIXSE* e o sistema de *Jensen et. al.* constróem esquemas conceituais canônicos de DTDs para facilitar a integração semântica e provêem suporte posterior para o usuário realizar uma integração manual.

Outras propostas permitem a integração de fontes estruturadas e semi-estruturadas, como *YAT* [CLU 98], *Mc Brien et. al.* [MCB 2001], *Vdovjak et. al.* [VDO 2001], *CUPID* [MAD 2001], *Lim et. al.* [LIM 2001] e *MOMIS* [BEN 2001, BER 2001]. O ponto forte das três primeiras propostas é o auxílio à definição de mapeamentos entre esquemas canônicos. *YAT* e *Mc Brien et. al.* lidam com esquemas canônicos hierárquicos na forma de grafo. *YAT* oferece uma ferramenta gráfica para a criação dos mapeamentos e *Mc Brien et. al.* utiliza um modelo canônico, chamado HDM, como modelo intermediário de mapeamento. Todos os modelos das fontes locais são convertidos para HDM através de regras de mapeamento. Para o mapeamento de fontes XML para uma representação conceitual no modelo Entidade-Relacionamento (ER) [BAT 92], por exemplo, converte-se esquemas XML para HDM e estes para esquemas ER. *Vdovjak et. al.* possui uma linguagem chamada LMX para o mapeamento de esquemas locais e uma ontologia global previamente construída.

CUPID, *Lim et. al.* e *MOMIS* definem abstrações canônicas com maior poder de expressão semântica. *CUPID* e *Lim et. al.* lidam com grafos hierárquicos e relacionamentos de herança, composição e associação. *MOMIS* utiliza um modelo conceitual orientado a objetos com o suporte à união de representações disjuntas, que é adequado para a representação de dados semi-estruturados. Estas abordagens tem um caráter semi-automático, realizando uma integração automática inicial que considera informações do esquema, como a nomenclatura e a estrutura das informações.

As tabelas 2.1 e 2.2 comparam os trabalhos relacionados com relação as estratégias de integração de esquemas e o modelo canônico adotado. A maioria destas características vêm da taxonomia de integração de esquemas (seção 2.1.1). Os itens *análise de nomenclatura* e *análise estrutural* só são considerados se a integração é semi-automática, uma vez que são realizados por procedimentos automatizados.

Com base nestas tabelas, dois pontos importantes devem ser salientados. O primeiro é que grande parte dos trabalhos realiza ainda uma integração manual, exigindo que o usuário conheça alguma linguagem de definição de visões integradas ou alguma ferramenta de suporte à definição de mapeamentos ou esquemas globais.

TABELA 2.1 – Comparativo de trabalhos relacionados (I)

	TSIMMIS	Garlic	MIX	Xyleme	LSD	DIXSE	*Jensen*
Semi-Automático (A) ou Manual (M)	M	M	M	A	A	M	M
Análise de nomenclatura				√	√		
Analise estrutural				√	√		
Informações do esquema	√	√	√	√		√	√
Informações das instâncias					√		
Esquema global(G) ou visões integradas (V) ou mapeamentos entre esquemas locais (M)	V	G	V	G	G	G	G
Top-down (T) ou Bottom-up (B)	B	B	B	T	T	B	B
Processo de Integração	n-ário	n-ário	n-ário	binário	binário	n-ário	?
Modelo canônico	grafo	ODMG	DTD	DTD	DTD	conceitual	UML

TABELA 2.2 – Comparativo de trabalhos relacionados (II)

	YAT	*McBrien*	*Vdovjak*	CUPID	*Lim*	MOMIS
Semi-Automático (A) ou Manual (M)	M	M	M	A	A	A
Análise de nomenclatura				√	√	√
Analise estrutural				√		√
Informações do esquema	√	√	√	√	√	√
Informações das instâncias						
Esquema global(G) ou visões integradas (V) ou mapeamentos entre esquemas locais (M)	M	M	G	G	G	G
Top-down (T) ou Bottom-up (B)	-	-	T	B	B	B
Processo de Integração	binário	binário	binário	binário	binário	n-ário
Modelo canônico	árvore	grafo	conceitual	conceitual	grafo	conceitual

Tais abordagens têm a desvantagem de serem de baixa qualidade pois o nível de automatização é baixo. Outro ponto a salientar é o modelo canônico utilizado. Vários trabalhos lidam com representações lógicas de esquemas semi-estruturados, que podem ser as próprias DTDs ou estruturas de árvore ou grafo hierárquico derivados diretamente dos esquemas locais. Tais representações têm a desvantagem de considerar mais a organização estrutural do esquema em detrimento da semântica dos dados.

Neste sentido, *CUPID* e *MOMIS* são trabalhos mais completos pois apresentam um processo semi-automático de integração que realiza análise de informações dos esquemas locais e gera esquemas canônicos conceituais com um ou mais tipos de relacionamentos semânticos entre os dados. Porém, apresentam limitações no que diz respeito á integração de esquemas XML pois não tratam de forma completa a integração de elementos textuais com elementos estruturados, nem a integração de representações alternativas de elementos. No que diz respeito á modelagem conceitual de esquemas XML, *MOMIS* o define manualmente e *CUPID* nao modela representações alternativas de elementos nem filtra elementos semanticamente irrelevantes para o domínio.

DIXSE e o trabalho de *Vdokjak* também definem esquemas canônicos conceituais. Porém, o primeiro não modela relacionamentos de herança e o segundo define o esquema de forma manual, como *MOMIS* .

Apesar das vantagens de alguns trabalhos em relação a outros, o que se cons-

tata, até o momento da escrita desta tese, é que não existe uma proposta robusta para a integração semântica de esquemas XML. Isto porque verifica-se que dois importantes requisitos não são plenamente atendidos neste contexto. Estes requisitos são:

1. *Uma abstração canônica adequada a esquemas XML.* Por tal abstração entende-se:

   (a) um esquema conceitual para dados XML, com relacionamentos que capturem as suas intenções semânticas;

   (b) um esquema que leve em conta as particularidades do modelo da DTD, como representações alternativas de elementos, referências entre elementos, tipos de elementos especiais (EMPTY e ANY) e elementos mistos (agregam estrutura e texto). Estas particularidades não são totalmente consideradas nos trabalhos relacionados.

2. *Um processo de integração semântica adequado a esquemas XML.* Por tal processo entende-se:

   (a) um processo semi-automatizado, que aplica diversas técnicas de integração visando minimizar a intervenção do usuário, como por exemplo, a análise conjunta de informações do esquema (DTD) e de instâncias (documentos XML) associada à verificação de equivalências e conflitos de nomenclatura e de estrutura. Nenhum trabalho relacionado combina tais técnicas;

   (b) um processo que considera particularidades da integração de dados XML, como a unificação de elementos com representações alternativas. Esta questão não é explorada nos trabalhos relacionados.

Estas limitações encontradas nos trabalhos relacionados motivaram o desenvolvimento desta tese. A solução aqui proposta atende aos requisitos recém-mencionados. Esta solução é apresentada nos capítulos a seguir.

3 Uma Abordagem para Integração Semântica de Esquemas XML

Este capítulo apresenta a solução proposta para a integração de fontes de dados XML. Esta solução é uma abordagem *bottom-up* de integração semântica que considera um conjunto de esquemas definidos através de DTDs e documentos XML associados, gerando a partir deles um esquema global de referência para o acesso a fontes XML. Esta abordagem é utilizada por uma camada de mediação que faz parte da arquitetura de um sistema de gerenciamento de dados semi-estruturados heterogêneos.

A escolha por uma abordagem de integração para dados XML é justificada pela grande aceitação que o padrão XML tem encontrado nos meios acadêmico e industrial. Diversos domínios de aplicação que disponibilizam e manipulam dados na *Web*, como comércio eletrônico e bibliografia científica, têm definido padrões de esquemas XML para intercâmbio de dados. Porém, estes padrões são bastante heterogêneos, inexistindo mecanismos efetivos que propiciem uma visão integrada de suas informações.

A abordagem proposta é chamada de **BInXS**: uma sigla para ***B****ottom-up* ***In****tegration of* ***X****ML* ***S****chemata*.

Um artigo apresentando a visão geral de BInXS foi publicado nos anais do *International Workshop on Information Integration on the Web (WIIW'2001)* [MEL 2001].

3.1 Contexto

O processo de integração de BInXS é parte integrante de uma *camada de mediação* para o acesso a múltiplas fontes de dados XML. Esta camada tem dois propósitos: realizar a integração semântica dos esquemas das fontes e possibilitar a consulta as mesmas através de um esquema global.

A figura 3.1 apresenta a arquitetura do sistema na qual a camada de mediação está inserida. Esta é uma típica arquitetura baseada em mediadores, composta por três camadas: *wrapping*, mediação e apresentação.

A *camada de wrapping* é a camada inferior da arquitetura. Esta camada é composta por um conjunto de *wrappers*. Cada *wrapper* é responsável pelo acesso a dados de uma ou mais fontes XML que estejam organizadas de acordo com uma DTD. Por uma fonte XML entende-se uma base de documentos XML ou um conjunto de *sites* na *Web* contendo informações que são extraídas e organizadas no formato XML. Quando não existe uma DTD associada a uma fonte, o *wrapper* deve proceder a sua geração através da análise dos dados presentes no(s) seu(s) documento(s) XML.

A *camada de wrapping* realiza também consultas as fontes XML sob sua responsabilidade. Para tanto, recebe da camada de mediação especificações de consultas em uma linguagem de consulta para XML e retorna a mesma um documento XML com os resultados. Estas consultas respeitam a DTD local do *wrapper*. Detalhes adicionais sobre esta camada estão fora do escopo desta tese.

A *camada de mediação* é a camada intermediária da arquitetura. Esta camada é composta por um mediador que é responsável pelos processos de integração

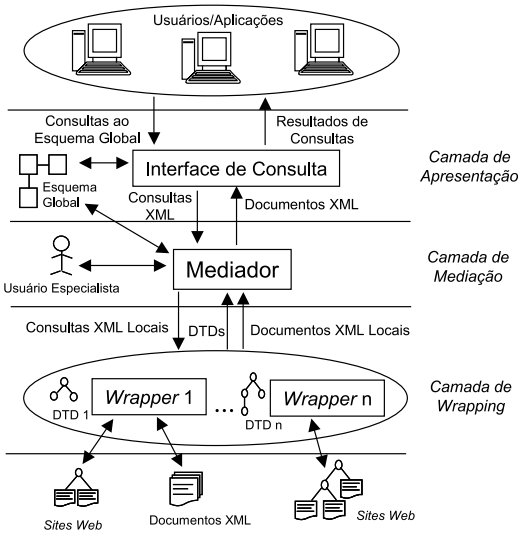


FIGURA 3.1 – Arquitetura proposta para um sistema baseado em mediador

semântica de esquemas XML e consulta a fontes XML. Nesta primeira versão da arquitetura, apenas um mediador é definido, considerando que o sistema atua sobre um único domínio de aplicação.

O processo de integração de BInXS é detalhado na próxima seção. Salienta-se apenas que é um processo semi-automático que exige a intervenção de um *usuário especialista* no domínio. O processo de consulta inicia com o recebimento de uma especificação de consulta em uma linguagem para XML enviada pela camada de apresentação. A consulta recebida está de acordo com o vocabulário do esquema global. Em seguida, esta consulta é decomposta e traduzida para o vocabulário e a estrutura da DTD local, sendo enviada para cada *wrapper*. Os resultados retornados por cada *wrapper* (documentos XML de acordo com as DTDs locais) são unificados, traduzidos para o vocabulário do esquema global e enviados como um único documento XML de resposta para a camada de apresentação. A atividade de consulta está fora do escopo desta tese.

A *camada de apresentação* é a camada superior da arquitetura. Esta camada é composta por uma interface de consulta através da qual usuários e aplicações realizam o acesso as fontes de dados XML. Esta interface disponibiliza uma visão do esquema global a partir do qual consultas declarativas podem ser formuladas, sendo esta consulta posteriormente traduzida para a sintaxe de uma linguagem de consulta para XML e enviada para a camada de mediação. No sentido oposto, o resultado de uma consulta, na forma de um documento XML, é devidamente apresentado na forma nativa XML ou em outro formato estrutural mais adequado, como uma tabela.

3.2 O Processo de Integração

O processo de integração de esquemas XML é o foco desta tese. BInXS adota uma abordagem semi-automática e *bottom-up*. A característica de ser semi-automático é inerente a um processo de integração semântica, na qual a intervenção

de um usuário especialista é necessária para validar a intenção semântica de um dado e de seus relacionamentos com outros dados. Neste processo, a intervenção do usuário é requerida em alguns casos para decidir por uma dada interpretação semântica durante as tarefas de abstração conceitual e unificação de esquemas XML.

A característica de ser *bottom-up* vem ao encontro do objetivo do processo, que é disponibilizar um esquema global que sirva como vocabulário de referência para a consulta a um conjunto de fontes XML. Do ponto de vista do gerenciamento de dados semi-estruturados, a manutenção deste esquema global é vantajosa pois deixa transparente para usuários e aplicações a alta heterogeneidade estrutural das fontes de dados. Além disso, o esquema global é um esquema conceitual rico pois considera a *intenção semântica de todas as fontes* que estão integradas e relacionamentos adicionais inter-esquema. Caso fosse utilizada uma abordagem *top-down* de integração, ter-se-ia um esforço maior em termos de tempo e complexidade para determinar *a priori* diversos esquemas conceituais particulares de cada aplicação e posteriormente tentar encontrar correspondências semânticas adequadas com os esquemas de cada fonte. A adoção de uma abordagem *bottom-up* não impede que esquemas externos adequados a aplicações sejam definidos a partir de um esquema global, sem recair no custo recém-descrito.

A adoção de um único esquema global de referência torna BInXS semelhante a um BDF fortemente acoplado. Considera-se adequada tal abordagem dada a vantagem da garantia da transparência de heterogeneidade das fontes semi-estruturadas, como salientado antes. Assim, evita-se que usuários e aplicações tenham conhecimento sobre esquemas locais de fontes XML (que são muitos, considerando a *Web*), como ocorre nas abordagens MDB e BDF fracamente acoplado.

O processo de integração de BInXS apresenta duas etapas, conforme ilustra a figura 3.2: Conversão da DTD e Integração Semântica.

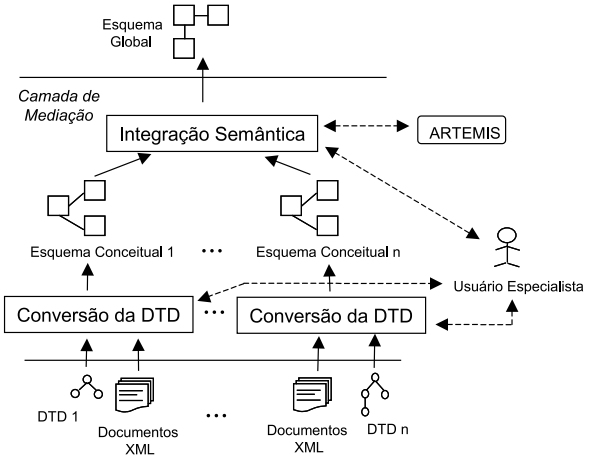


FIGURA 3.2 – Processo de integração de esquemas XML

A etapa de *Conversão da DTD* é responsável pela conversão de cada DTD associada a uma ou mais fontes XML em um esquema conceitual canônico. A utilização de um modelo conceitual como modelo canônico permite uma abstração da DTD que leva em conta a semântica dos seus dados, facilitando a etapa posterior de integração. O modelo canônico adotado é descrito na seção 4.1. Como nem

sempre é possível determinar a intenção do dado, a intervenção do usuário é requerida. Outra característica desta conversão é que não somente informação sobre os esquemas é analisada, mas também o conteúdo dos documentos XML (instâncias XML). Tal análise é necessária para auxiliar na definição de alguns pontos do esquema conceitual, como por exemplo, as cardinalidades inversas de relacionamentos entre elementos da DTD. Informações de mapeamento dos conceitos do esquema para elementos e atributos da DTD são gerados e mantidos em um catálogo. A etapa de *Conversão da DTD* é descrita no capítulo 5.

A etapa de *Integração Semântica* é responsável pela integração dos esquemas canônicos, gerando o esquema global e mapeamentos dos seus conceitos para elementos e atributos semanticamente equivalentes em cada DTD. O esquema global é também representado no modelo canônico. A tarefa de integração segue basicamente os passos do processo de integração *bottom-up* tradicional, sendo o usuário novamente solicitado para confirmar ou efetivar certas ações que ocorrem nas fases de comparação de esquemas e unificação. Na etapa de comparação de esquemas, particularmente, a ferramenta ARTEMIS [CAS 2001] é utilizada na determinação de afinidades semânticas entre conceitos de esquemas locais diferentes. A etapa de *Integração Semântica* é descrita no capítulo 6.

3.3 Visão Geral do Processo

Os capítulos 5 e 6 definem formalmente as regras e procedimentos que são utilizados pelas etapas do processo de integração de BInXS. Antes disto, esta seção apresenta uma visão geral do processo de integração, através de exemplos, com o objetivo de facilitar a compreensão de BInXS. Esta visão geral não tem a intenção de ilustrar todos os detalhes inerentes ao processo para não tornar exaustivo o número de exemplos a serem discutidos. Ao invés disto, os casos mais freqüentes são apresentados para que o princípio geral de funcionamento do processo seja entendido.

Utiliza-se como exemplo a DTD mostrada na figura 3.3 (*DTD1*), referente a um domínio bibliográfico. Esta DTD organiza algumas informações sobre *livros* (título, editora, tipo e dados de autores) e *anais* de conferências (ano e local).

3.3.1 Conversão da DTD

O processo de integração de BInXS inicia com a etapa de *conversão da DTD* para um esquema conceitual. A DTD1 e um conjunto de documentos XML associados são a entrada para esta etapa.

Pré-Processamento

Inicialmente, a DTD1 passa por um *pré-processamento*. Este pré-processamento modifica a sua especificação para facilitar a conversão para um esquema conceitual correspondente. O pré-processamento aplicado à DTD1 resulta na DTD1 *pré-processada* mostrada na figura 3.3. As modificações executadas no pré-processamento da DTD1 são as seguintes:

- *remoção manual* dos elementos `MyPublications` (linha 2) e `WriterList` (linha 6). Estes elementos são removidos por não serem significativos para o domínio

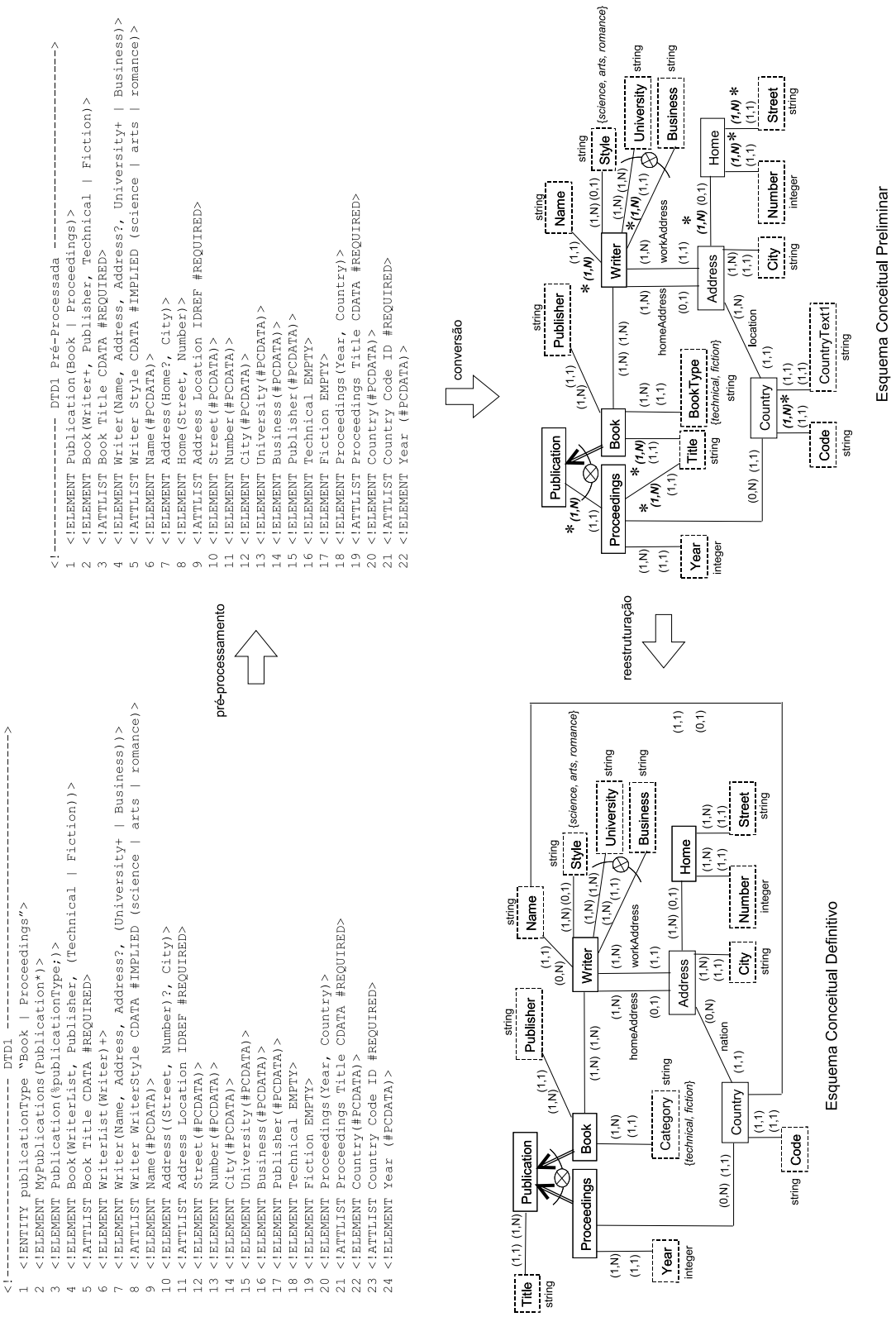


FIGURA 3.3 – Exemplo de aplicação da etapa de *Conversão da DTD*

em questão. Com esta remoção, os elementos que fazem referência a eles passam a conter os seus sub-elementos;

- *renomeação manual* de `WriterStyle` (linha 8) para um nome mais adequado `Style`;
- *substituição automática* da definição da entidade `publicationType` (linha 1) em todos os elementos e atributos que fazem referência a ela (elemento `Publication` (linha 3), neste caso). Com isto, as definições de elementos e atributos tornam-se completas;
- *remoção automática* de *estruturas aninhadas* dos elementos `Book` (linha 4), `Writer` (linha 7) e `Address` (linha 10). Estas remoções de aninhamentos são realizadas para facilitar a determinação de relacionamentos entre elementos da DTD no momento da conversão. Quando o aninhamento de um elemento E é removido, os sub-elementos deste aninhamento podem ou não ser mantidos na definição de E. Em caso negativo, cria-se um elemento adicional (*elemento virtual*) na DTD para manter o aninhamento. Para o elemento `Address` (linha 10) foi definido um elemento virtual `Home` na DTD1 pré-processada para manter o aninhamento `(Street, Number)?`.

Conversão

Uma vez obtida a DTD pré-processada, ocorre a *conversão* propriamente dita para um *esquema conceitual preliminar*. Um esquema conceitual é representado em um modelo canônico que provê dois tipos de *conceitos* e *relacionamentos* binários entre conceitos[1]. O princípio geral de conversão de elementos e atributos de uma DTD é ilustrado na figura 3.4.

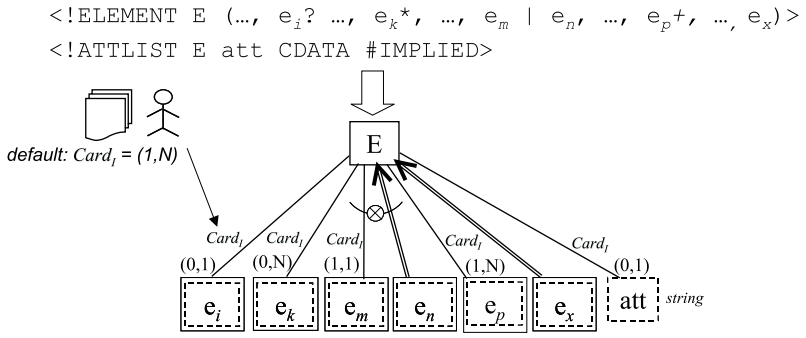


FIGURA 3.4 – Princípio geral de conversão de uma DTD

A idéia básica é que um elemento E composto apenas por sub-elementos e/ou atributos torna-se um *conceito não-léxico* (quadrado contínuo). Elementos com um conteúdo texto (`#PCDATA`) ou atributos tornam-se *conceitos léxicos* (quadrados tracejados). Um conceito léxico possui um *tipo de dado* (*string*, *integer*, *float*, *character*, *date*, *time* ou *datetime*) que pode ser determinado através de uma *análise de documentos XML*. Caso esta análise não seja conclusiva, o tipo *default string* é assumido.

[1] O modelo canônico é apresentado no capítulo 4.

Um conceito não-léxico possui *relacionamentos* com conceitos correspondentes aos seus sub-elementos e/ou atributos. Estes conceitos podem ser léxicos ou não-léxicos (representados por um quadrado contínuo e tracejado ao mesmo tempo pois podem assumir um destes dois tipos). Estes relacionamentos podem ser de *associação* (linha) ou de *herança* (seta do conceito especializado para o genérico). No caso de um relacionamento de associação, existem *restrições de cardinalidade* definidas em ambas as direções do relacionamento. A cardinalidade direta é determinada em função da existência ou não de operadores regulares (símbolos "?", "*", "+") definidos para os sub-elementos de *E*. A cardinalidade inversa ($Card_I$) necessita de um processo de *análise de documentos XML* e *validação de um usuário especialista* para ser determinada.

Quando existe uma *escolha* na definição de *E* (símbolo "|"), uma *disjunção* (curva com "X") é definida para os relacionamentos envolvidos. Os *atributos* de *E* tornam-se conceitos léxicos associados ao conceito correspondente a *E*, com cardinalidade direta obrigatória ou opcional, dependendo da sua definição.

O *esquema conceitual preliminar* obtido através da conversão da DTD1 pré-processada é mostrado na figura 3.3. O elemento inicial, *Publication* (linha 1), torna-se um conceito não-léxico, com relacionamentos disjuntos para os conceitos *Book* e *Proceedings* devido à escolha. Nesta conversão, uma heurística particular (*heurística de herança*) verifica, com o auxílio de um *Thesaurus*, se um elemento filho é um termo *mais específico* que o termo correspondente ao elemento pai. Esta heurística vale para *Publication-Book* e um relacionamento de herança é definido entre eles. O Thesaurus não valida tal heurística para *Publication-Proceedings*, sendo definido um relacionamento de associação entre eles. A cardinalidade do relacionamento no sentido *Publication→Proceedings* é *(1,1)* pois apenas uma instância obrigatória de *Proceedings* está definida na DTD como elemento filho de *Publication*. A cardinalidade inversa *Publication←Proceedings* tenta ser determinada através da análise de instâncias do elemento *Proceedings* nos documentos XML. Supondo que não tenha sido encontrada uma mesma instância do elemento *Proceedings* como filha de mais de uma instância de *Publication*, esta cardinalidade é definida como *(1,N)* por *default*. O símbolo "*"indica que ela deve ser confirmada posteriormente pelo usuário especialista.

O elemento *Book* (linha 2) torna-se um conceito não-léxico com relacionamentos de associação para os conceitos *Writer* (cardinalidade direta *(1,N)* devido ao operador "+") e *Publisher*. Uma outra heurística particular (*heurística de qualificação*) verifica se um elemento *E* possui um ou mais elementos filhos que são elementos do tipo `EMPTY` (vazio) sem atributos. Em caso positivo, assume-se que estes elementos são qualificações de *E*, tornando-se uma *enumeração* de valores permitidos para um conceito léxico especial *E+Type* associado a *E*. Tal situação ocorre para os elementos *Technical* e *Fiction* (linhas 16 e 17), que são considerados categorias de um livro. *Book* possui ainda um atributo obrigatório *Title* (linha 3) que se torna um conceito léxico com tipo de dado *string*.

O elemento *Writer* (linha 4) torna-se também um conceito não-léxico com relacionamentos de associação para os conceitos *Name*, *Address*, *University* e *Business*. Uma disjunção é novamente definida para os dois últimos relacionamentos devido à escolha. Para o relacionamento com o conceito *Address* surge um impasse sobre a intenção semântica deste relacionamento: como existem duas ocorrências do sub-elemento *Address*, é possível que:

1. um autor tenha um endereço obrigatório e um segundo endereço opcional (opcionalidade é indicada pelo operador "?"), sendo ambos endereços residenciais;

2. a mesma intenção anterior, sendo ambos endereços profisssionais;

3. um autor tem um endereço residencial obrigatório e um endereço profissional opcional;

4. um autor tem um endereço profissional obrigatório e um endereço residencial opcional.

Para resolver este impasse, a *intervenção do usuário especialista* é necessária. Supondo que a alternativa 4 tenha sido escolhida, definem-se dois *relacionamentos nomeados* para diferenciar endereços residenciais (*homeAddress*) e profissionais (*workAddress*).

Writer possui um atributo opcional *Style* (linha 5) que se torna um conceito léxico associado a ele. Este atributo possui uma *enumeração* de valores permitidos (*science*, *arts*, *romance*) que se torna uma enumeração associada ao conceito.

A conversão do elemento *Address* (linha 7) segue o princípio geral de conversão. A única novidade nesta conversão é que *Address* possui um atributo do tipo `IDREF` (linha 9). Este atributo define uma referência para o valor de um atributo do tipo `ID` de algum elemento da DTD. Nestas situações, uma *análise de documentos XML* é também realizada para verificar se o valor das referências `IDREF` de todas as instâncias de um tipo de elemento E_1 indicam um mesmo tipo de elemento E_2. Em caso positivo, estabelece-se um relacionamento de associação entre E_1 e E_2 no esquema conceitual. Supondo que o atributo *Location* de *Address* faz referência sempre a identificadores de instâncias de elementos *Country* (*Country* possui um atributo do tipo `ID` (linha 21)), uma associação *Address-Country* com nome *Location* é definida no esquema conceitual.

O elemento *Proceedings* (linha 18) gera um conceito não-léxico com associações para os conceitos *Year*, *Title* e *Country*. Como os conceitos *Title* e *Country* possuem associações com mais de um conceito (*Title*, por exemplo, associa-se também com *Book*), as cardinalidades diretas das suas associações com estes conceitos são definidas como *opcionais*. Tal definição é feita porque, por exemplo, uma ocorrência de *Title* é ou um título de livro ou um título de *Proceedings*.

O último elemento a ser convertido, *Country* (linha 20), apresenta ainda uma particularidade de conversão: apesar de ser um elemento `#PCDATA`, ele possui atributos, o que o torna um conceito não-léxico. Para fins de representação do seu conteúdo textual, um conceito léxico especial *CountryText* é definido e associado a ele, além da associação com o conceito *Code*.

Reestruturação

Um esquema conceitual preliminar não é considerado um esquema conceitual definitivo para uma DTD pois alguns *ajustes* automáticos ou manuais podem ser feitos. Estes ajustes dizem respeito a tarefas como modificação de tipo ou cardinalidade de relacionamento, generalização de relacionamentos e modificação de nomes de conceitos e relacionamentos.

Dado o esquema conceitual preliminar da figura 3.3, os seguintes ajustes são realizados:

- *cardinalidades inversas*: cardinalidades inversas pendentes ("*") são manualmente acertadas. Alguns exemplos são: *(1,N)* para *Business*→*Writer* e *(1,1)* para *Code*→*Country*;

- *nomenclaturas*: nomes de conceitos (principalmente conceitos especiais gerados na conversão) e relacionamentos são manualmente modificados para nomes mais adequados ao domínio, como *BookType* → *Category* e *CountryText1* → *Name*. A modificação *CountryText1* → *Name* gera uma associação entre *Country* e *Name*;

- *tipo de relacionamento*: um relacionamento de herança entre dois conceitos pode não ter sido identificado por um Thesaurus na fase de conversão e ter sido definido como uma associação. De forma contrária, um relacionamento de herança definido pode não ser adequado para o contexto do domínio. Neste caso, uma alteração manual de tipo de relacionamento se faz necessária. Um exemplo é a consideração de *Proceedings* como sendo uma especialização de *Publication*;

- *generalização de relacionamentos*: quando todos os conceitos especializados E_1, ..., E_n de um conceito genérico G possuem relacionamentos comuns e de mesmo tipo com um conceito X, os relacionamentos de E_1, ..., E_n com X podem ser automaticamente generalizados para um único relacionamento de G com X. Um exemplo é o relacionamento de *Title* com *Proceedings* e *Book*, que é generalizado para um relacionamento de *Publication*.

Uma vez concluídos os ajustes necessários, obtém-se o esquema conceitual definitivo para a DTD1, mostrado na figura 3.3.

3.3.2 Integração Semântica

Dado um conjunto de esquemas conceituais correspondente a um conjunto de DTDs, a etapa de *Integração Semântica* produz um *esquema conceitual global* que é o resultado da integração destes esquemas. O esquema conceitual para a DTD1, obtido na etapa anterior mais um esquema conceitual exemplo para uma DTD2 são a entrada para esta etapa. Estes esquemas conceituais são mostrados na figura 3.5.

Agrupamento de Conceitos Sinônimos

Inicialmente, os conceitos dos esquemas conceituais de entrada são analisados e são definidos *clusters de afinidade*. Cada cluster de afinidade mantém conceitos de esquemas conceituais diferentes que tenham equivalência semântica, ou seja, conceitos que sejam sinônimos. Esta análise de afinidade leva em conta o nome e os relacionamentos de conceitos não-léxicos, assim como o nome e o tipo de dado de conceitos léxicos. Um passo chamado *agrupamento de conceitos sinônimos* é responsável por esta tarefa. Este passo utiliza a ferramenta ARTEMIS para a definição dos clusters de afinidade. Clusters definidos automaticamente podem ser validados pelo usuário. Na figura 3.5 são mostrados os clusters de afinidade para os dois esquemas conceituais de entrada (os números 1 e 2 indicam a procedência do conceito - esquema conceitual 1 e 2, respectivamente).

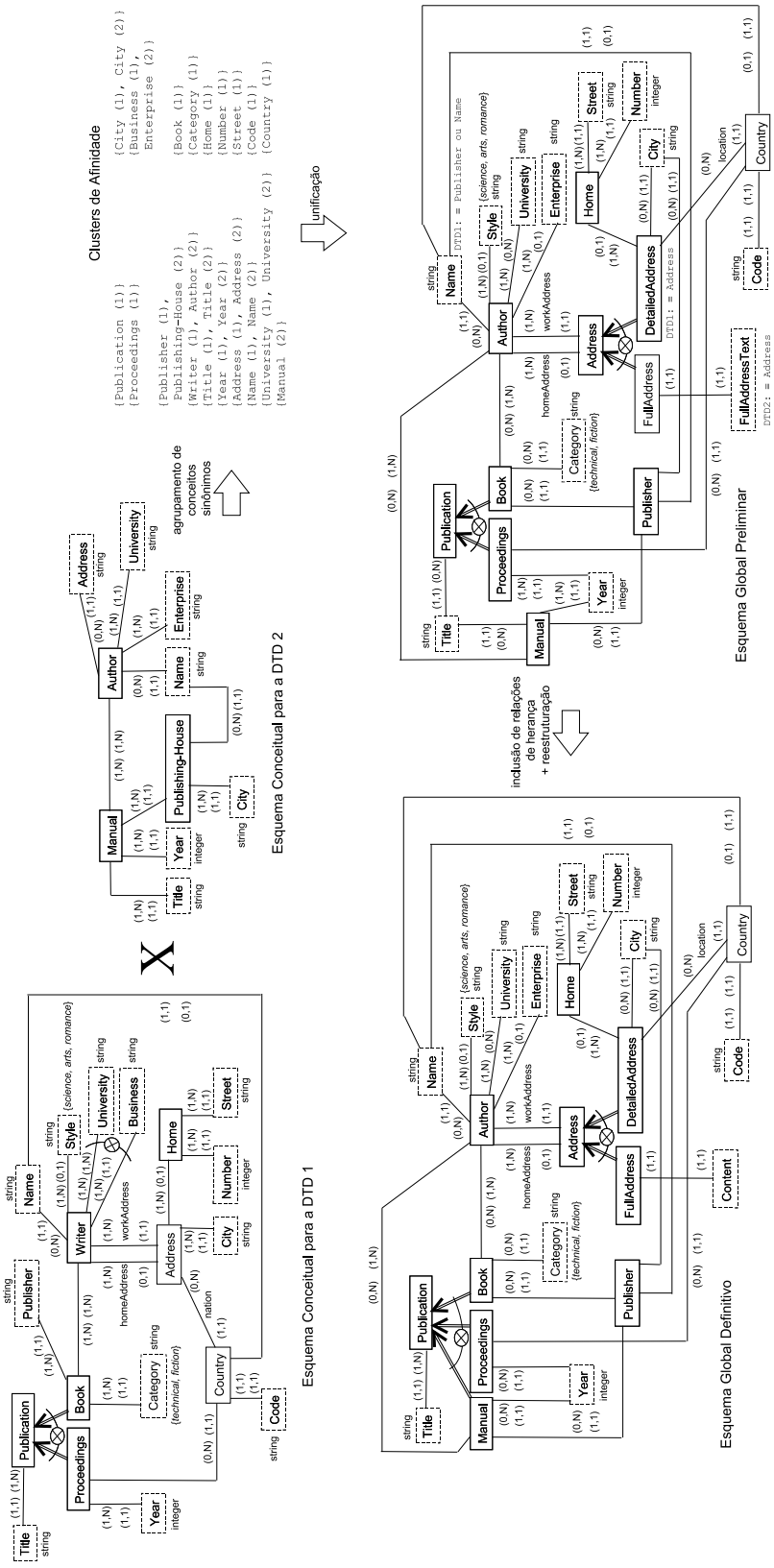


FIGURA 3.5 – Exemplo de aplicação da etapa de *Integração Semântica*

Unificação

Na seqüência, ocorre a *unificação* propriamente dita dos conceitos presentes em cada cluster, com a geração de um *esquema global preliminar* como resultado. Esta unificação resolve basicamente conflitos de nome, relacionamentos e disjunções de relacionamentos e considera os tipos de conceitos presentes em cada cluster. Quando um cluster é composto por apenas um conceito, este é diretamente representado no esquema global. Alguns exemplos são os clusters que mantêm os conceitos *Publication* e *Street*. Quando um cluster é composto por mais de um conceito, o *nome* do conceito global é o nome que ocorre mais vezes no cluster ou um nome escolhido pelo usuário.

Para fins de unificação, três tipos de cluster podem existir, considerando os tipos de conceitos que podem estar em um cluster. Quando um cluster é composto apenas por conceitos léxicos, tem-se um *cluster léxico*. A unificação de um cluster léxico produz um *conceito global léxico* cujo tipo de dado é o tipo mais genérico dentre todos os tipos de dados dos conceitos do cluster. Enumerações são unificadas caso todos os conceitos possuam enumeração. Alguns exemplos ocorrem para os clusters que mantém os conceitos *Title* e *Business + Enterprise* (ver esquema global preliminar na figura 3.5). No primeiro caso, produz-se um conceito global léxico *Title* com tipo *string* pois todos os conceitos possuem o mesmo nome e tipo de dado. No segundo caso, produz-se um conceito global léxico *Enterprise* (nome decidido pelo usuário) com tipo de dado comum *string*.

Quando um cluster é composto apenas por conceitos não-léxicos, tem-se um *cluster não-léxico*. A unificação de um cluster não-léxico produz um *conceito global não-léxico*. Um exemplo desta unificação ocorre para o cluster que mantém os conceitos *Writer* e *Author*. Como mostra a figura 3.5, um conceito global *Author* é produzido. Os relacionamentos de ambos os conceitos são considerados no conceito global. Relacionamentos de associação que são particulares apenas de um esquema conceitual são definidos como relacionamentos *opcionais*. Um exemplo é o relacionamento com *Book*, definido apenas no esquema conceitual 1. Ele se torna um relacionamento opcional (*(0,N)*) de *Author*.

Quando um relacionamento de mesmo tipo (associação ou herança) ocorre entre os conceitos de um cluster em mais de um esquema conceitual, este relacionamento é *unificado* e representado uma única vez no esquema global. Para relacionamentos de associação, valem as cardinalidades mais gerais dentre as cardinalidades direta e inversa dos relacionamentos envolvidos. Um exemplo é a unificação dos relacionamentos *Writer-University* e *Author-University* mostrado na figura 3.5. Ele produz um relacionamento *Author-University* no esquema global com cardinalidade direta *(1,N)* (a mais geral dentre as cardinalidades diretas *(1,N)* e *(1,1)*) e inversa *(1,N)*.

Na unificação de relacionamentos, é possível que um relacionamento de um conceito tenha correspondência com mais de um relacionamento de outro conceito no mesmo cluster. Um exemplo presente na figura 3.5 é o relacionamento *Author-Address* no esquema conceitual 2. Ele pode ter correspondência tanto com *Writer-homeAddress-Address* como com *Writer-workAddress-Address* no esquema conceitual 1. Nestas situações, o usuário deve decidir qual dos relacionamentos de um esquema tem a mesma intenção semântica do relacionamento de outro esquema. Supondo que o usuário afirme que *Author-Address* significa o endereço profissional do autor, este é unificado com *Writer-workAddress-Address*, que é mantido como um

relacionamento obrigatório e nomeado no esquema global.

Relacionamentos de um conceito não-léxico podem ser *disjuntos*. Na unificação de conceitos não-léxicos, se uma disjunção D de um conceito apresenta *zero* ou *um* relacionamento em comum com outros conceitos do mesmo cluster, D *é representada* no esquema global. Na unificação dos esquemas conceituais da figura 3.5, a disjunção dos relacionamentos de herança do conceito *Publication* é um exemplo. Esta disjunção é mantida no esquema global preliminar. Tal decisão se justifica pelo fato de não ocorrer conflito de disjunção entre dois ou mais relacionamentos de conceitos em esquemas conceituais diferentes.

Por outro lado, um conflito de disjunção existe para os relacionamentos de *Writer* e *Author* com *University* e *Enterprise* nos esquemas conceituais 1 e 2, considerando que *Business* no esquema conceitual 1 é semanticamente equivalente a *Enterprise* no esquema conceitual 2. No esquema conceitual 1, existe uma disjunção entre os dois conceitos e no esquema conceitual 2 não. Neste caso, a disjunção *não é representada* no esquema global pois não é válida para o esquema conceitual 2. Se os relacionamentos forem de associação, definem-se todos os relacionamentos com conflito de disjunção como *opcionais*. Assim, evita-se que os relacionamentos disjuntos em um dado esquema conceitual ocorram todos obrigatoriamente no nível global. Na figura 3.5, o esquema global preliminar define os relacionamentos *Author-University* e *Author-Enterprise* como opcionais.

O último caso de unificação ocorre para clusters chamados *mistos*, ou seja, clusters compostos por conceitos léxicos e não-léxicos. Quando existem conceitos léxicos e não-léxicos a serem unificados, opta-se por uma representação não-léxica pois esta fornece mais detalhes de representação para o conceito global correspondente. Uma unificação dos relacionamentos de todos os conceitos não-léxicos do cluster é realizada, como explicado anteriormente. Um exemplo mostrado na figura 3.5 é a unificação dos conceitos *Publisher* e *Publishing-House*, sendo o primeiro léxico e o segundo não-léxico. A representação não-léxica de *Publishing-House* é definida no nível global através do conceito unificado *Publisher*.

Uma questão a considerar na unificação mista é qual *correspondência* no nível global existe para cada conceito léxico do cluster misto, uma vez que o conceito global C_g correspondente a este cluster é o resultado da unificação apenas dos conceitos não-léxicos do cluster. Esta correspondência é determinada pelo usuário e considera três casos:

1. *um conceito léxico do cluster misto corresponde exatamente a um conceito global léxico associado a* C_g *(mapeamento um-para-um).* Um exemplo deste caso ocorre na unificação do cluster *Publisher* e *Publishing-House*. O conteúdo de *Publisher* no esquema 1 mantém o *nome* da editora. Logo, este contéudo equivale ao conteúdo do conceito léxico *Nome* associado a *Publisher* no esquema global. Esta correspondência é indicada na figura 3.5 através da referência aos elementos ou atributos que o conceito global *Nome* representa na DTD 1, sendo *Publisher* uma delas;

2. *um conceito léxico* c_l *do cluster misto não tem correspondência com conceitos léxicos do esquema global.* Neste caso, um conceito léxico global deve ser criado para suportar a representação de c_l. Este conceito é associado a C_g através de um relacionamento opcional;

3. *um conceito léxico do cluster misto corresponde a mais de um conceito global léxico associado a C_g (mapeamento um-para-muitos).* Um exemplo deste caso ocorre na unificação do cluster que mantém os conceitos *Address*. O conceito léxico *Address* definido no esquema conceitual 2 mantém um endereço completo como conteúdo. Já no esquema conceitual 1, o conceito não-léxico *Address* está associado a conceitos que definem partes de um endereço. Por não ser possível estabelecer uma correspondência um-para-um para *Address* no esquema conceitual 2, é definido um conceito global *Address* que generaliza estas representações alternativas de endereço. O conceito global especializado *FullAddress* corresponde ao endereço léxico do esquema conceitual 2 e o conceito especializado *DetailedAddress* corresponde ao endereço não-léxico do esquema conceitual 1, conforme indicado na figura 3.5. O conceito global *FullAddress* torna-se um conceito não-léxico pois relacionamentos de herança devem ocorrer sempre entre conceitos deste tipo. O conteúdo de *FullAddress* é representado pelo conceito léxico especial chamado *FullAddressText*.

Uma vez unificados todos os tipos de clusters de conceitos sinônimos, obtém-se o *esquema global preliminar* mostrado na figura 3.5.

Inclusão de Relações de Herança

Como o esquema global preliminar é o resultado da unificação de diversos esquemas conceituais, é possível que pares de conceitos advindos de esquemas conceituais diferentes tenham um relacionamento de herança. Este relacionamento pode ser relevante para o domínio. Para tanto, o passo de *inclusão de relações de herança* verifica, novamente com o auxílio de um Thesaurus, se tais relacionamentos existem. Aqueles que forem considerados relevantes pelo usuário são definidos no esquema global.

Um exemplo ocorre no esquema global preliminar da figura 3.5 para os conceitos *Publication* e *Manual*. Ambos são provenientes de esquemas conceituais diferentes e, de acordo com o Thesaurus, *Publication* é um termo mais geral que *Manual*. Considerando a aprovação do usuário e também o fato de que todos os relacionamentos de associação de *Publication* são também relacionamentos de *Manual*, define-se um relacionamento de herança entre eles. Este relacionamento é mostrado no *esquema global definitivo* da figura 3.5.

Reestruturação

Assim como na etapa de *Conversão de DTD*, um passo de *reestruturação* realiza ajustes manuais e automáticos no esquema global preliminar. Tais ajustes são necessários pois o esquema global agrega informações de diversos esquemas conceituais e é possível que alguns conceitos, relacionamentos e disjunções devam ser validados.

Para o esquema conceitual preliminar da figura 3.5, os seguintes ajustes são realizados:

- *validação de disjunções de relacionamentos*: o usuário pode modificar disjunções de relacionamentos para um conceito não-léxico de forma a adequá-los à intenção semântica do domínio. Tais modificações incluem união de disjunções e inclusão de relacionamento em uma disjunção. No esquema global preliminar, foi definido um relacionamento de herança *Publication-Manual* no

passo anterior. Considerando que *Manual* é uma especialização também disjunta de *Proceedings* e *Book*, o relacionamento *Publication-Manual* é incluído na disjunção que envolve *Proceedings* e *Book*;

- *remoção de relacionamentos redundantes*: considerando que *Manual* é agora uma especialização de *Publication*, o relacionamento de associação *Manual-Title* é redundante pois *Manual* herda este relacionamento de *Publication*. Este relacionamento redundante é então automaticamente removido. Como conseqüência, *Title* relaciona-se apenas com *Publication*. Logo, a cardinalidade da sua associação com *Publication* torna-se automaticamente obrigatória;

- *determinação de cardinalidades opcionais*: um conceito léxico c_l que relacionava-se apenas com um conceito não-léxico em um esquema conceitual pode estar se relacionando com vários conceitos não-léxicos c_{nl1}, ..., c_{nlx} no esquema global como conseqüência da unificação. Como cada ocorrência de c_l no nível global nunca se relaciona com todos os conceitos c_{nl1}, ..., c_{nlx} ao mesmo tempo nos esquemas conceituais locais, a cardinalidade da associação de c_l com estes conceitos é automaticamente definida como *opcional*. Um exemplo no esquema global preliminar é o conceito léxico *Year*. As cardinalidades das suas associações com *Proceedings* e *Manual* são obrigatórias pois *Year* relacionava-se apenas com estes conceitos em cada esquema conceitual local. No nível global, estas cardinalidades são então definidas como opcionais;

- *modificação de nomes de conceitos especiais*: conceitos especiais gerados automaticamente para o esquema global devem ter seus nomes modificados para nomes mais adequados ao domínio. Um exemplo ocorre para o conceito *FullAddressText* que é alterado para *Content*.

Concluídos estes ajustes, obtém-se o *esquema global definitivo* mostrado na figura 3.5.

Encerrada esta visão geral do processo de integração de BInXS, os capítulos 5 e 6 detalham as etapas deste processo. Antes de tal detalhamento, o capítulo seguinte apresenta o modelo canônico adotado e a estratégia de mapeamento de conceitos e relacionamentos do esquema global para elementos e atributos correspondentes nos esquemas XML.

4 Modelo Canônico e Mapeamentos

Este capítulo apresenta o modelo canônico adotado nesta tese para a representação dos esquemas conceituais derivados de DTDs e do esquema conceitual global. Este modelo é uma variante do modelo ORM/NIAM adequada à modelagem de esquemas XML. A especificação de esquemas neste modelo é feita através da linguagem *DAML+OIL*, uma recomendação da W3C para a definição de ontologias e esquemas de dados baseado nos padrões *RDF* e *RDF Schema*.

Este capítulo também mostra como são definidas as informações de mapeamento entre um esquema global no modelo canônico e um conjunto de DTDs correspondentes. Estas informações são vitais para a tradução de consultas formuladas com base no esquema conceitual global para as fontes XML. Informações de mapeamento são definidas através de expressões *XPath*, um padrão da W3C para o acesso a dados XML.

Um artigo submetido ao *XVII Simpósio Brasileiro de Banco de Dados (SBBD'2002)* apresenta o modelo canônico e a estratégia de mapeamento de um esquema conceitual global para esquemas XML para fins de tradução de consultas sobre um esquema global para fontes XML. Este artigo pode ser encontrado em `http://www.inf.ufrgs.br/~ronaldo/Publications`.

4.1 O Modelo Conceitual Canônico (MCC)

Modelos conceituais são uma abstração de alto nível para dados de um domínio de aplicação, uma vez que representam de forma direta um conjunto de entidades do mundo real e seus relacionamentos. Comparados com modelos de dados lógicos, apresentam um maior poder de expressão pois não estão vinculados a nenhuma estrutura de dados (tabela, árvore, etc) particular [BAT 92]. Esta característica garante que a intenção semântica dos dados do domínio seja compreendida mais facilmente.

O alto poder de expressão de um esquema conceitual permite que diversos esquemas XML possam ser derivados a partir dele. Um exemplo é mostrado na figura 4.1.

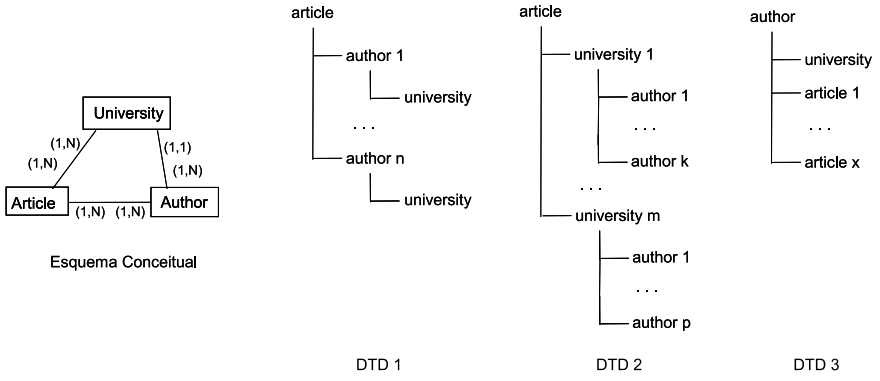


FIGURA 4.1 – Um esquema conceitual e algumas DTDs derivadas deste esquema

O esquema conceitual exemplo modela parcialmente um domínio bibliográfico, mantendo relacionamentos entre os conceitos *University*, *Author* e *Article*. Este esquema conceitual pode representar diversos esquemas hierárquicos correspondentes a DTDs, como mostra a figura 4.1. A DTD1 é derivada a partir do acesso a entidade *Article*, seguindo o relacionamento com *Author* e posteriormente o relacionamento com *University* (caminho *Article*→*Author*→*University*). A DTD2 é derivada do caminho *Article*→*University*→*Author* e a DTD3 é derivada a partir de dois caminhos que iniciam em *Author*: *Author*→*University* e *Author*→*Article*.

A possibilidade de um esquema conceitual abstrair a representação de diversas DTDs é interessante no contexto de um sistema de gerenciamento de dados heterogêneos e semi-estruturados pois tal esquema pode atuar como um esquema global resultante da integração semântica destas DTDs. Este esquema evita a necessidade de se conhecer as representações hierárquicas de cada DTD quando da formulação de uma consulta a várias fontes XML. Em função disto, um modelo conceitual foi escolhido como modelo canônico para BInXS.

O *Modelo Conceitual Canônico* (MCC) adotado é uma variante do modelo conceitual ORM/NIAM (*Object with Role Model/Natural Language Information Analysis Method*)[1] [HAL 98]. ORM é um modelo baseado em conceitos e relacionamentos semânticos entre conceitos. Um exemplo de modelagem em ORM é mostrado na figura 4.2 (a). Dois tipos de conceitos são definidos neste modelo: não-léxicos e léxicos. Um *conceito não-léxico* (círculo contínuo) modela informação que é composta por outras informações, como por exemplo, *Publication*, que possui *Title* e *Author*. Um *conceito léxico* (círculo tracejado) modela informação que possui um conteúdo diretamente associado, como *Title* e *Name*. Uma *enumeração* de valores permitidos pode estar associado a um conceito léxico, como é o caso do conceito *Publisher*.

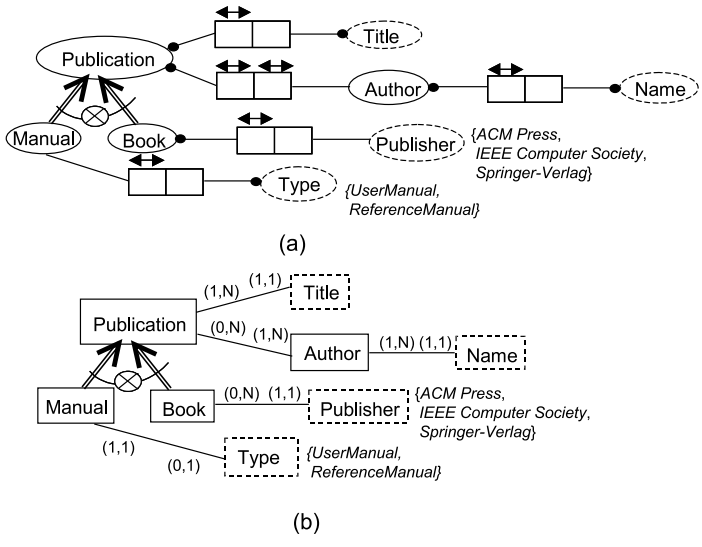


FIGURA 4.2 – Exemplo de modelagem em ORM (a) e no MCC (b)

Dois tipos de relacionamentos são suportados: *herança* e *associação*. Um exemplo de herança é o relacionamento entre *Publication* e *Manual*. Um exemplo de associação ocorre entre *Publication* e *Author*, denotando que uma publicação possui

[1]Para fins de simplificação, este modelo é chamado de ORM de agora em diante.

obrigatoriamente um ou mais autores e que um autor pode ter uma ou mais publicações. Relacionamentos podem ser mutuamente exclusivos (simbolizado por uma curva com um "X"que passa pelos relacionamentos), como no caso de *Publication* que ou é um *Manual* ou é um *Book*.

O MCC é uma variante do ORM pois adota convenções gráficas do modelo ER (*Entidade-Relacionamento*) [BAT 92] e lida apenas com relacionamentos binários. Tais convenções são utilizadas para facilitar a compreensão dos esquemas canônicos, considerando a popularidade do modelo ER no projeto conceitual de banco de dados. A figura 4.2 (b) representa a mesma modelagem da figura 4.2 (a) no MCC. No MCC, conceitos são agora representados por retângulos - analogia com entidades do ER -, relacionamentos de associação são linhas conectando conceitos com restrições de cardinalidade direta e inversa - notação de par de cardinalidades do ER - e um relacionamento de herança segue a notação do ORM.

Os modelos ORM e ER apresentam o mesmo poder de expressão [HAL 98]. Porém, cabe salientar que o modelo ORM foi escolhido como modelo canônico ao invés do modelo ER por apresentar um número menor de construtores de modelagem. No contexto de integração de esquemas, quanto menos construtores de modelagem um modelo canônico apresenta, menor é a possibilidade de conflitos de construtores quando da integração de dois ou mais esquemas. O ORM lida apenas com conceitos e relacionamentos. No ER existe ainda a noção de atributo, o que resulta em um maior número de possibilidades de modelagem do domínio, considerando que um fato do mundo real pode ser representado como uma entidade ou como um atributo. No ORM, esta *dicotomia entidade-atributo* é evitada pois um fato do mundo real é representado sempre através de um conceito. Além disso, o ER restringe as possibilidades de modelagem conceitual de um esquema XML pois um elemento E do tipo `#PCDATA` não pode ser modelado como um atributo em um esquema ER porque um atributo é uma propriedade exclusiva de uma entidade e E pode ser sub-elemento de mais de um elemento em uma DTD.

4.1.1 Definição Formal de um Esquema MCC

Esta seção apresenta a definição formal dos construtores do MCC. Estas definições são utilizadas posteriormente pelas etapas do processo de integração de BInXS.

Esquema MCC. *Um esquema MCC é uma 5-upla e = <N, NL, L, R, D>, onde N é o nome do esquema, NL é o conjunto de conceitos não-léxicos, L é o conjunto de conceitos léxicos, R é o conjunto de relacionamentos entre dois conceitos e D é o conjunto de conjuntos de relacionamentos disjuntos.*

Conceito Não-Léxico. *Um conceito não-léxico nl ∈ NL é uma 1-upla nl = <n>, onde nl.n é o nome do conceito.*

Conceito Léxico. *Um conceito léxico l ∈ L é uma 3-upla l = <n, t, [e]>, onde l.n é o nome do conceito, l.t é o seu tipo de dado, com l.t ∈ {string, integer, float, character, date, time, datetime}, e l.e é um conjunto de valores opcional permitido para o conceito. Considera-se l.e uma enumeração de valores V, tal que ∀ v ∈ V (domínio(v) = le.t)*[2].

Relacionamento. *Um relacionamento r ∈ R é uma 2-upla r = <c_1, c_2>, onde r.c_1 e r.c_2 são os nomes dos conceitos c_1 e c_2, com c_1 ∈ NL e c_2 ∈ NL ∪ L.*

[2]Assume-se *dominio(d)* como sendo a função que retorna o domínio de um dado *d*.

Relacionamento de Herança. *Um relacionamento de herança r_h* ***é um*** *relacionamento onde $r_h.c_1$ é o nome do conceito genérico C_g e $r_h.c_2$ é o nome do conceito especializado C_e, com C_g, $C_e \in NL$.*

Seja H o conjunto de relacionamentos de herança.

Relacionamento de Associação. *Um relacionamento de associação r_a* ***é um*** *relacionamento definido como $r_a = <c_1, c_2, c_d, c_i, [n], [p_1], [p_2]>$, onde $r_a.c_d$ é a cardinalidade direta (de $r_a.c_1$ para $r_a.c_2$) do relacionamento, $r_a.c_i$ é a cardinalidade inversa (de $r_a.c_2$ para $r_a.c_1$) do relacionamento, n é o nome opcional do relacionamento*[3]*, p_1 é o nome opcional do papel assumido pelo conceito $r_a.c_1$ e p_2 é o nome opcional do papel assumido pelo conceito $r_a.c_2$*[4]*.*

Seja A o conjunto de relacionamentos de associação.

Disjunção. *Uma disjunção d é uma 1-upla $d = <R_d>$, onde R_d é o conjunto de conjuntos de relacionamentos disjuntos. Considera-se $R_d = \{S_R\}$, onde S_R é um conjunto de relacionamentos disjuntos, tal que:*

1. $|S_R| > 1$;
2. $\forall\ r_i, r_j \in S_R\ (r_i, r_j \in R \wedge ((r_i.c_1 = r_j.c_1) \vee (r_i.c_1 = r_j.c_2) \vee (r_i.c_2 = r_j.c_1 \wedge r_i \notin H) \vee (r_i.c_2 = r_j.c_2 \wedge r_i, r_j \notin H)))$.

Seja D o conjunto de disjunções.

Na definição de *Disjunção*, a segunda restrição aplicada sobre S_R indica que seus relacionamentos devem ter um conceito em comum c, podendo c ser qualquer um dos conceitos de um relacionamento de associação ou o conceito genérico de um relacionamento de herança. Esta restrição está intimamente relacionada a forma como a hierarquia de um elemento é definida em XML. Se um elemento E suporta representações alternativas (*escolhas*) na forma $E\ (e_1|e_2|...|e_n)$, estas representações são modeladas no MCC como relacionamentos disjuntos de um conceito E com os conceitos e_1, e_2, ..., e_n. Um componente $e_i \in \{e_1, e_2, ..., e_n\}$ corresponde a um conceito associado ou especializado de E.

4.2 Mapeamento de um Esquema MCC Global para DTDs

Um esquema global é um esquema MCC resultante da integração semântica de diversas DTDs. Portanto, cada conceito e relacionamento deste esquema deve ter informações de mapeamento para elementos/atributos e relacionamentos nas DTDs que possuem correspondência. Uma informação de mapeamento para uma DTD deve ser capaz de localizar sem ambiguidade um tipo de elemento ou atributo[5]. Como uma DTD é um grafo hierárquico, uma *expressão de caminho* que parta

[3]Relacionamentos nomeados são necessários quando existe mais de um relacionamento de associação entre dois conceitos ou auto-relacionamentos.

[4]Relacionamentos com papéis podem ser necessários em relacionamentos para uma melhor interpretação semântica.

[5]Um determinado elemento/atributo pode ocorrer em diversos pontos da hierarquia do esquema de uma DTD, em diferentes contextos, como por exemplo, *cidade* do autor e *cidade* da conferência. Uma *busca não ambígua* é aquela que identifica as ocorrências deste elemento/atributo dentro do contexto desejado.

de um dos elementos raiz da DTD até o elemento/atributo desejado é suficiente para localizá-lo de forma precisa. Prova disto é que expressões de caminho são o princípio básico das pesquisas empregadas por linguagens de consulta para dados XML [XQU 2002].

Na literatura de integração de esquemas de banco de dados existem duas abordagens para a definição de informações de mapeamento de esquemas globais para esquemas locais: catálogos e visões [ELM 99]. Um *catálogo* é um esquema de dados sobre mapeamentos. Uma *visão* descreve, através de especificações de consulta, como conceitos de esquemas locais correspondentes a um conceito do esquema global são recuperados. Nesta tese, optou-se pela segunda opção. A justificativa para tal escolha é que a tradução de consultas para as fontes locais é mais simples, uma vez que visões indicam explicitamente os predicados de seleção a serem aplicados aos dados locais. Além disso, a manutenção dos mapeamentos tem uma complexidade menor se eventualmente ocorre uma mudança nos esquemas locais. Isto porque apenas as visões associadas ao esquema local modificado precisam ser alteradas.

Nesta tese, visões são *expressões de caminho XPath* associadas a conceitos e relacionamentos do esquema global. Estas expressões definem mapeamentos para os esquemas XML nos quais o conceito ou relacionamento tem correspondência. *XPath* foi escolhido por ser um padrão da W3C para pesquisa a dados XML [XML 2001]. Além disso, esta escolha permite que uma expressão *XPath* para uma fonte XML (consulta local) seja construída de forma simples, basicamente através da substituição de conceitos e relacionamentos especificados em uma consulta global por expressões *XPath* correspondentes[6]. As abordagens de tradução adotadas pelos trabalhos relacionados são em geral mais complexas pois exigem uma análise das especificações de visões para a reescrita da consulta global.

Expressões *XPath* de mapeamento são geradas automaticamente durante o processo de integração. A figura 4.3 mostra o mapeamento de alguns conceitos e relacionamentos de um esquema global para algumas DTDs de um domínio bibliográfico. A forma como estes mapeamentos são definidos é apresentada nas seções seguintes.

4.2.1 Mapeamento de Conceitos

Uma informação de mapeamento associada a um conceito deve indicar diretamente como o elemento ou atributo correspondente a este conceito é alcançado em uma DTD a partir de um elemento raiz. O *mapeamento de um conceito* do esquema global para uma DTD é definido como uma *expressão de caminho absoluto XPath*, ou seja, um caminho que indica a hierarquia de elementos que deve ser percorrida a partir da raiz da DTD até o elemento ou atributo desejado[7]. Exemplos deste tipo de mapeamento ocorrem para os conceitos *Author*, *Style* e *Publication*.

Author corresponde, na DTD B, ao sub-elemento *Writer* do elemento raiz *Book*. Seu mapeamento é então definido como `/Book/Writer`. *Style*, na DTD A, corresponde ao atributo *Category* do elemento *Book*, sendo definido o seu mapeamento como `/Publication/Book/@Category`[8]. O mapeamento de *Publication* mostra que, quando um conceito é uma generalização de outros conceitos, este tem

[6]Um exemplo de tradução de consulta é mostrado na seção 4.3.

[7]Uma expressão de caminho absoluto em XPath inicia com o símbolo "/".

[8]O símbolo "@"indica um nome de atributo.

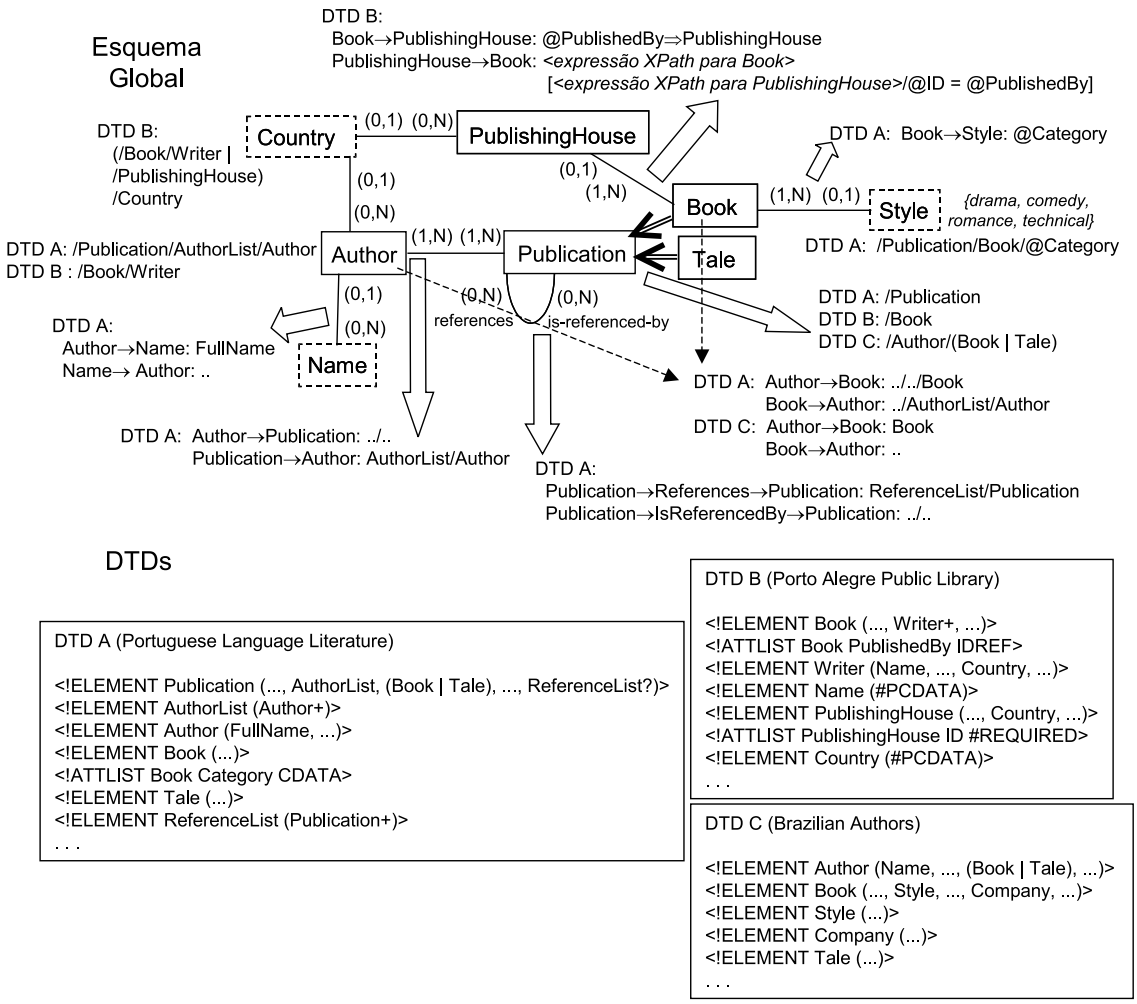


FIGURA 4.3 – Exemplos de mapeamento de um esquema global para DTDs

correspondência com os elementos relativos a todos os seus conceitos especializados, nos esquemas XML onde ele não está definido. Isto ocorre na DTD B, onde *Publication* é mapeado para *Book* (`/Book`), e na DTD C, no qual é mapeado para *Book* e *Tale* (`/Author/(Book | Tale)`), pois livros e contos são publicações.

Um conceito pode eventualmente apresentar mais de um caminho absoluto de um elemento raíz até o seu elemento/atributo correspondente. Quando isto ocorre, todos estas alternativas de caminho devem ser declaradas na expressão *XPath*. Um exemplo é o conceito *Country*, que na DTD B indica tanto o país onde reside um autor como o país onde se localiza uma editora.

4.2.2 Mapeamento de Relacionamentos

Uma informação de mapeamento associada a um relacionamento indica como alcançar um conceito relacionado a outro conceito através de um caminho existente entre seus elementos ou atributos correspondentes na DTD. O *mapeamento de um relacionamento* do esquema global é definido como uma *expressão de caminho relativo XPath*. Esta expressão indica como alcançar um elemento/atributo correspondente a um conceito destino a partir de um elemento/atributo correspondente

a um conceito origem em uma DTD (são expressões relativas à origem). O mapeamento é definido em ambos os sentidos do relacionamento para facilitar a tradução de consultas globais, que podem percorrer o esquema global em qualquer direção.

Exemplos de mapeamento de relacionamentos são mostrados para *Author-Book*, *Book-PublishingHouse* e um auto-relacionamento em *Publication*. O caso *Author-Book* serve para ressaltar a seguinte observação: quando um conceito C_e é especialização de um conceito C_G, mapeamentos são definidos também para relacionamentos entre C_e e cada conceito $C_i \in \{C_1, ..., C_n\}$ relacionado a C_G. Isto porque C_G pode não ter relacionamento com algum C_i em uma DTD, mas C_e sim. Por isso, mapeamentos entre *Book* e *Author* devem ser previstos porque *Author* se relaciona com *Publication* e *Book* é especialização de *Publication*.

O relacionamento *Author*→*Book*, para a DTD C, é definido pela expressão relativa `Book`. Isto porque *Book* é filho direto de *Author*. No sentido *Author*←*Book*, tem-se a expressão `".."`, pois *Author* é pai direto de *Book*. Um relacionamento entre dois conceitos nem sempre é direto, como no caso da DTD A, onde existe o elemento *AuthorList* no caminho entre *Book* e *Author*. Este elemento não foi representado no esquema global por não ser semanticamente relevante para o domínio, mas deve ser considerado no nível do mapeamento.

Auto-relacionamentos seguem a mesma abordagem de mapeamento relativo, devendo-se respeitar os sentidos corretos dos relacionamentos nas expressões de caminho, sejam eles indicados por papéis ou não. O conceito *Publication* apresenta um relacionamento que indica as publicações referidas por uma publicação e vice-versa. Neste relacionamento, o fato de uma publicação ter referências para outras publicações é expresso pelo papel *references*. No mapeamento para a DTD A, este papel é um relacionamento que parte de *Publication* e segue pelo elemento filho *ReferenceList* até as publicações referidas. O papel *is-referenced-by* é mapeado de forma inversa, ou seja, parte de elementos *Publication* que são filhos do elemento *ReferenceList* até alcançar o elemento *Publication* que o referencia (sobe dois níveis na hierarquia da DTD).

O relacionamento *Book-PublishingHouse* é representado na DTD B através de atributos do tipo ID para *PublishingHouse* (identificador) e IDREF para *Book* (referência a um identificador). No sentido *Book*→*PublishingHouse*, o mapeamento é simples, sendo definido através da indicação do atributo de referência *PublishedBy* seguido da intenção de se buscar um elemento do tipo *PublishingHouse* referido por ele (uso do operador *XPath* '⇒'). No sentido oposto, o mapeamento é mais sofisticado, utilizando uma expressão *XPath* com predicado (definido entre colchetes) para a comparação entre valores de referências e de identificadores. Este mapeamento é uma exceção à regra de definição de caminhos relativos para relacionamentos, uma vez que utiliza um algoritmo para a sua definição. Este algoritmo é exemplificado para os elementos *Book* e *PublishingHouse* da DTD B:

1. Deseja-se alcançar um elemento de referência *R* (*Book*) a partir de um elemento com identificador *I* (*PublishingHouse*). Então, especifica-se inicialmente o caminho absoluto *XPath* até *I* e seu atributo identificador (<*expressão XPath para PublishingHouse*>/*@ID*). Este caminho pode conter predicados de seleção que restringem as instâncias de *I*. Isto depende da consulta feita sobre o esquema global. Logo, este é um caminho dinâmico, definido em tempo de execução da consulta;

2. Insere-se este caminho *XPath* em um predicado que testa a igualdade do

atributo identificador com o atributo de referência (tem-se *[<expressão XPath para PublishingHouse>/@ID = @PublishedBy]*);

3. Inclui-se, antes do predicado, o caminho absoluto *XPath* até *R* (tem-se *<expressão XPath para Book>[<expressão XPath para PublishingHouse>/@ID = @PublishedBy]*). Este caminho também é dinâmico e depende das restrições sobre as instâncias de *R* impostas pela consulta.

A justificativa para esta exceção em termos de mapeamento decorre da própria filosofia da linguagem *XPath*. *XPath* trabalha com a noção de *contexto corrente único*, ou seja, a noção de um apontador que navega no documento XML conforme a sua expressão de consulta é avaliada. Para este caso de mapeamento de relacionamento inverso, deseja-se, a partir de um elemento I, buscar um elemento R que faça referência a ele, comparar a igualdade dos atributos identificador e de referência e retornar R caso a comparação seja verdadeira. Porém, *XPath* não possui um operador específico para referência inversa nem permite manter dois contextos correntes (dois apontadores) ao mesmo tempo para realizar a comparação[9]. Logo, a solução foi definir um algoritmo que modifica a expressão *XPath* de busca inicial de I de modo a tornar R o contexto corrente, seguido de um predicado que inclui a expressão de busca de I mais a comparação dos seus respectivos atributos.

4.3 Exemplo de Tradução de Consulta

Como já salientado, a escolha por expressões de mapeamento *XPath* se justifica pela facilidade de tradução de consultas para fontes XML. Esta tradução ocorre através da montagem de uma expressão de consulta local no próprio padrão *XPath*, com base na identificação de conceitos e relacionamentos do esquema global descritos em uma consulta global. Esta vantagem é mostrada no exemplo da figura 4.4.

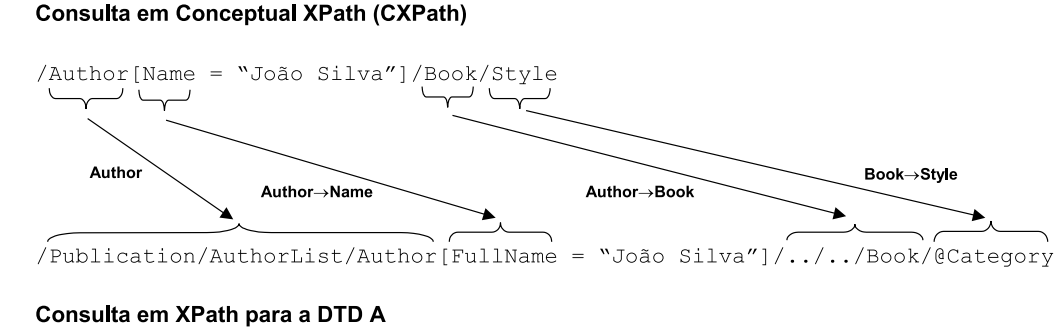


FIGURA 4.4 – Exemplo de tradução de uma consulta global (*CXPath*) para uma DTD

Uma consulta formulada sobre o esquema global é expressa na linguagem *CXPath*. *CXPath* é uma extensão de *XPath* para consultas a modelos conceituais. *CXPath* assume um esquema MCC como sendo um grafo onde conceitos são nodos e relacionamentos são arestas. Consultas podem partir de qualquer nodo e percorrer

[9]Tal situação exigiria a declaração de variáveis de consulta, que não existem em *XPath*.

qualquer caminho no grafo. A tradução de uma consulta *CXPath* para *XPath* significa adequar uma navegação em grafo a uma navegação em árvore, que é peculiar a esquemas XML. Detalhes sobre *CXPath* e o processo de tradução de consultas estão fora do escopo desta tese, sendo foco de uma dissertação de mestrado em andamento no *Instituto de Informática* da *UFRGS*.

Considerando o esquema global da figura 4.3, suponha uma consulta que deseja buscar os *estilos dos livros do autor João Silva*. Esta consulta deve ser traduzida para a DTD A. A tradução inicia pelo conceito de partida, que é *Author*. Este conceito corresponde a um elemento de mesmo nome na DTD A, que é alcançado através do caminho `/Publication/AuthorList/Author`. Na seqüência, tem-se um predicado que se refere ao conceito *Name* associado a *Author*. Deve-se, então, definir o relacionamento local *Author*→*Name*, que é *FullName*. Concluido o predicado, está-se no contexto do conceito *Author* novamente e o caminho da consulta segue para o conceito *Book*. Investiga-se então o relacionamento *Author*→*Book* na DTD A, que corresponde a `../../Book`. A partir de *Book* deseja-se recuperar o conceito relacionado *Style*, resultando na busca do mapeamento do relacionamento *Book*→*Style*, que é `@Category`. No final, tem-se uma expressão local de consulta *XPath* válida para a DTD A.

Apesar deste ser um exemplo simples de consulta, constata-se que o processo de tradução de consultas globais em locais não é complexo pois os mapeamentos de relacionamentos entre conceitos do esquema global já são por si só expressões de caminho *XPath* locais que necessitam basicamente ser substituidas em uma consulta global para se definir um caminho válido na DTD. A estrutura da consulta global (expressões de caminhos e predicados) não necessita ser alterada.

4.4 A Linguagem de Especificação de Esquemas MCC

A linguagem *DAML+OIL* é utilizada nesta tese para a especificação de esquemas MCC. *DAML+OIL* é uma linguagem de marcação semântica para definição de recursos na *Web*, sendo uma extensão dos padrões *RDF* e *RDF Schema* da W3C [RES 2002, DAM 2001].

Como o próprio nome sugere, *DAML+OIL* é uma combinação das linguagens para modelos conceituais DAML-ONT [DAM 2002] e OIL [OIL 2001] mais os tipos de dados de *XML Schema* [XML 2002][10]. Tal combinação resulta em uma linguagem de marcação para modelos conceituais com uma semântica bem definida e clara. Pelo fato de apresentar uma sintaxe XML, um esquema *DAML+OIL* pode ser exportado para diversas aplicações *Web* que fazem acesso a arquitetura de mediação proposta por esta tese. A escolha de *DAML+OIL* se deve ainda ao fato desta ser a única linguagem considerada até o presente momento pela W3C para ser a base de uma linguagem de modelagem conceitual para aplicações de *Web* semântica [SEM 2002].

DAML+OIL utiliza a noção de *classe* para descrever entidades de um esquema conceitual. Uma classe possui *propriedades* que descrevem seus atributos e seus relacionamentos com outras classes. Uma informação em DAML+OIL pode ser descrita em diversos níveis, desde o nível de metadados até o nível de instâncias. Nesta tese, utilizam-se especificações DAML+OIL para descrever três níveis de in-

[10] Ambas as linguagens têm por objetivo a especificação de *ontologias*. Uma ontologia no contexto destas linguagens é similar a um modelo conceitual.

formação de um esquema MCC: nível de metadados, nível de esquema e nível de dados. A figura 4.5 ilustra estes níveis de especificação.

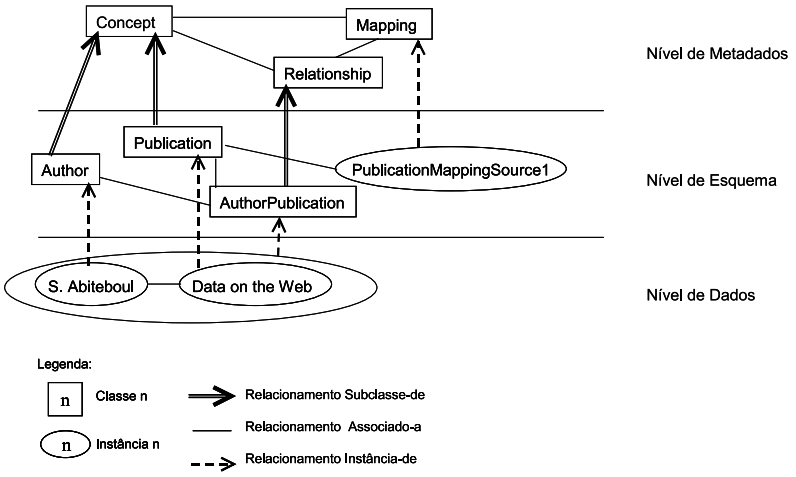


FIGURA 4.5 – Níveis de especificação de um esquema MCC

O *nível de metadados* é composto pelas classes de *construtores do MCC*, que apresentam propriedades básicas para conceitos léxicos e não-léxicos e seus relacionamentos. *Concept* and *Mapping* são exemplos destas classes na figura 4.5, descrevendo propriedades de conceitos e de mapeamentos de conceitos e relacionamentos, respectivamente.

O nível de metadados tem uma especificação fixa em DAML+OIL (vide especificação completa no anexo A). Esta especificação está de acordo com o esquema de metadados mostrado na figura 4.6. Este esquema modela no próprio MCC as informações referentes aos seus construtores: *conceitos* e seus tipos de *relacionamentos*. Vale ressaltar que o conceito *Name* associado a *Concept* (*Name* identifica um conceito léxico ou não-léxico) é representado pelo atributo predefinido *ID* de uma classe DAML+OIL.

O *nível de esquema* é composto pelas classes que descrevem os conceitos e relacionamentos de um *esquema MCC do mundo real*, como por exemplo, *Author*, *Publication* e o relacionamento entre eles. Este nível mantém também *instâncias de mapeamento* associadas a estes conceitos e relacionamentos. Instâncias de mapeamento são descritas neste nível pois são dados que não existem nas fontes XML, servindo apenas como informação de suporte para o mediador realizar o processamento de consultas.

Cabe ressaltar que relacionamentos entre conceitos de um esquema MCC são especificados como classes por dois motivos: eles apresentam propriedades (podem ter nome ou papéis) e podem ser disjuntos. Disjunção em DAML+OIL é uma característica que se aplica a classes.

Para exemplificar uma especificação DAML+OIL no nível de esquema, considera-se o esquema MCC da figura 4.3. *Author*, por exemplo, é um conceito não-léxico, sendo representado como uma classe DAML+OIL que é subclasse da classe de metadados *NonLexicalConcept*. Informações relativas a conceitos relacionados e mapeamentos são declaradas em uma tag `<subclassOf>`. Estas restrições dizem respeito, respectivamente, as propriedades *RelatedConcepts* e *ConceptMappings* de um con-

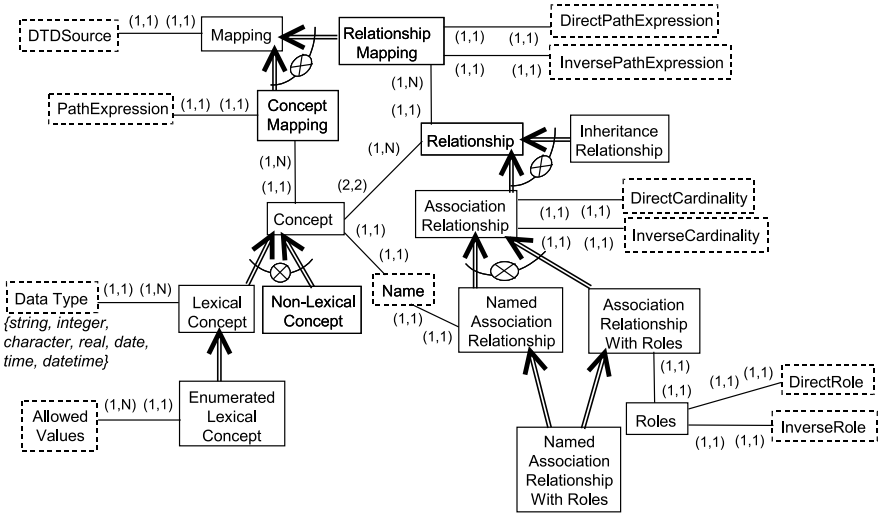


FIGURA 4.6 – Esquema de metadados de um esquema MCC

ceito.

```
<Class ID = "Author">
  <subClassOf resource = "NonLexicalConcept"/>
  <subClassOf>
    <restriction>
      <onProperty resource = "#RelatedConcepts">
        <toClass>
           <UnionOf parseType = "collection">
              <Thing about = "#AuthorName"/>
              <Thing about = "#AuthorCountry"/>
              <Thing about = "#AuthorPublication"/>
            </UnionOf>
        </toClass>
      </onProperty>
      <onProperty resource = "#ConceptMappings">
        <toClass>
           <UnionOf parseType = "collection">
              <Thing about = "#AuthorPortugueseLanguageLiterature"/>
              <Thing about = "#AuthorPortoAlegrePublicLibrary"/>
              <Thing about = "#AuthorBrazilianAuthors"/>
           </UnionOf>
        </toClass>
      </onProperty>
    </restriction>
  </subclassOf>
<\Class>
```

Country é um conceito léxico associado a *Author*. Para conceitos léxicos, informações sobre conceitos relacionados, tipo de dado e mapeamentos devem ser declaradas. No caso de *Country*, seu tipo de dado é *string*. Informações de restrição de valor, se existirem, são declaradas na propriedade *ValueConstraints*.

```
<Class ID = "Country">
  <subclassOf resource = "#LexicalConcept"/>
  <subclassOf>
    <restriction hasValue = "string"/>
      <onProperty resource = "#DataType"/>
    </restriction>
    <restriction>
      <onProperty resource = "#RelatedConcepts">
        <toClass>
           <UnionOf parseType = "collection">
```

```
            <Thing about = "#AuthorCountry"/>
            <Thing about = "#PublishingHouseCountry"/>
          </UnionOf>
        </toClass>
      </onProperty>
      <onProperty resource = "#ConceptMappings">
        <toClass resource = "#CountryPortoAlegrePublicLibrary"/>
      </onProperty>
      <onProperty resource = "#ValueConstraints">
        <toClass resource = "#CountryBrazilianAuthors"/>
      </onProperty>
    </restriction>
  </subclass>
</Class>
```

Como já salientado, relacionamentos são também declarados como classes DAML+OIL. O exemplo a seguir ilustra a declaração do relacionamento de associação entre *Author* e *Publication*. Assume-se nomes de relacionamentos como sendo a concatenação dos nomes dos conceitos envolvidos. Na especificação deste tipo de relacionamento, restrições relativas à identificação dos conceitos relacionados (propriedades *SourceConcept* e *TargetConcept*), cardinalidades (propriedades *DirectCardinality* e *InverseCardinality*) e informações de mapeamento (propriedade *RelationshipMappings*) devem ser declaradas.

```
<Class ID = "AuthorPublication">
  <subclassOf resource = "#AssociationRelationship"/>
  <subclassOf>
    <restriction toClass = "#Author">
      <onProperty resource = "#SourceConcept"/>
    </restriction>
    <restriction toClass = "#Publication">
      <onProperty resource = "#TargetConcept"/>
    </restriction>
    <restriction hasValue = "(1,N)">
      <onProperty resource = "#DirectCardinality"/>
      <onProperty resource = "#InverseCardinality"/>
    </restriction>
    <restriction>
      <onProperty resource = "#RelationshipMappings">
        <toClass>
          <UnionOf parseType = "collection">
            <toClass resource = "#AuthorPublicationPortugueseLanguageLiterature"/>
            <toClass resource = "#AuthorPublicationPortoAlegrePublicLibrary"/>
            <toClass resource = "#AuthorPublicationBrazilianAuthors"/>
          </UnionOf>
        </toClass>
      </onProperty>
    </restriction>
  </subclassOf>
</Class>
```

Relacionamentos de herança são especificados como subclasse de *Inheritance-Relationship*, apresentando apenas propriedades relativas à identificação de conceitos e informações de mapeamento. Disjunção de relacionamentos é indicada através da tag `<disjointWith>` na definição da classe de relacionamento. No trecho de especificação DAML+OIL a seguir tem-se dois relacionamentos de herança disjuntos entre *Publication-Book* e *Publication-Poem*.

```
<Class ID = "PublicationBook">
  <subclassOf resource = "#InheritanceRelationship"/>
  <disjointWith resource = "#PublicationPoem"/>
  <restriction toClass = "#Publication">
      <onProperty resource = "#SourceConcept"/>
    </restriction>
    <restriction toClass = "#Book">
```

```
      <onProperty resource = "#TargetConcept"/>
    </restriction>
  <restriction>
      <onProperty resource = "#RelationshipMappings">
        <toClass resource = "#PublicationBookPortugueseLanguageLiterature"/>
      </onProperty>
  </restriction>
</Class>

<Class ID = "PublicationPoem">
  <subclassOf resource = "#InheritanceRelationship"/>
  <disjointWith resource = "#PublicationBook"/>
  <restriction toClass = "#Publication">
      <onProperty resource = "#SourceConcept"/>
    </restriction>
    <restriction toClass = "#Poem">
      <onProperty resource = "#TargetConcept"/>
    </restriction>
  <restriction>
      <onProperty resource = "#RelationshipMappings">
        <toClass resource = "#PublicationPoemPortugueseLanguageLiterature"/>
      </onProperty>
  </restriction>
</Class>
```

Para ilustrar a especificação de informações de mapeamento, dois exemplos são mostrados a seguir. Eles mostram o mapeamento do conceito *Author* para a DTD2 e do relacionamento entre *Author* e *Publication* para a DTD1, respectivamente[11].

```
 <ConceptMapping ID = "#AuthorPortoAlegrePublicLibrary">
   <Source> http://www.poa.rs.gov/publicLibrary/DTD </Source>
   <PathExpression> /Book/Writer </PathExpression>
  </PathExpressionMapping>

 <RelationshipMapping ID = "#AuthorPublicationPortugueseLanguageLiterature">
   <Source> http://www.literatura_portuguesa.com.br/XML/schema </Source>
   <DirectMapping> ../AuthorList/.. </DirectMapping>
   <InverseMapping> /AuthorList/Author </InverseMapping>
 </RelationshipMapping>
```

O último nível de especificação de um esquema MCC, o *nível de dados*, é composto pelas instâncias de conceitos de esquemas MCC e seus relacionamentos. Este nível não é tratado nesta tese, pois diz respeito a especificação de resultados de consultas a fontes XML.

A especificação no nível de esquema é gerada automaticamente pelo processo de integração com base no esquema global resultante. Nos capítulos posteriores, é mostrado como uma informação de mapeamento é gerada, a cada vez que um conceito é criado na conversão de uma DTD para um esquema MCC, ou eventualmente atualizada, quando da integração de esquemas MCC. As etapas de *Conversão da DTD* e *Integração Semântica* são os dois capítulos a seguir.

[11] As URLs apresentadas nos exemplos são fictícias.

5 Conversão da DTD

Este capítulo descreve a primeira etapa do processo de integração de esquemas de BInXS. Esta etapa realiza a conversão de um esquema XML definido através de uma DTD em um esquema conceitual no MCC. Esta etapa é composta por um processo baseado em um conjunto de regras de conversão. Estas regras consideram diversos aspectos como o modelo da DTD, heurísticas relacionadas a interpretações semânticas sobre os seus dados, análise de instâncias XML e o auxílio de um usuário especialista.

Uma versão preliminar das regras de conversão e uma visão geral do processo foi publicado nos anais do *20th International Conference on Conceptual Modeling (ER'2001)* [MEL 2001a].

5.1 Uma Terminologia para a DTD

Esta seção define uma terminologia para o modelo de uma DTD. Esta terminologia é utilizada na definição das regras de conversão.

Elemento. Um elemento possui um *nome* e um modelo de conteúdo. Um *modelo de conteúdo* define o que está contido entre as *tags* que delimitam um elemento. Opcionalmente, um elemento pode ter um ou mais atributos.

Atributo. Um atributo possui um *nome*, um *tipo de dado* e, opcionalmente, restrições associadas a valores permitidos (*enumeração* ou *valor fixo*) e ao fato do atributo ser *obrigatório/opcional*. Um atributo do tipo `ID` é um identificador para o elemento. Um atributo do tipo `IDREF(S)` ou `xml:link href` atua como uma referência ao(s) identificador(es) de elemento(s).

Elemento Vazio. Um elemento E é dito vazio quando seu modelo de conteúdo é definido pela cláusula `EMPTY` (E não apresenta conteúdo).

Elemento Complexo. Um elemento E é dito complexo quando seu modelo de conteúdo se enquadra em uma das seguintes alternativas:

- Definido por um conjunto de elementos componentes;
- Definido pela cláusula `ANY` (E permite qualquer elemento da DTD como seu componente único);
- Definido pela cláusula `EMPTY` e E possui atributos;
- Definido pela cláusula `#PCDATA` (E é composto por texto) e E possui atributos;
- Misto, ou seja, inclui texto (cláusula(s) `#PCDATA`) e um conjunto de elementos componentes.

Elemento Composto. Um elemento composto **é um** elemento complexo cujo modelo de conteúdo não é definido pelas cláusulas `ANY` ou `EMPTY`. Um elemento composto tem um modelo de conteúdo definido por um dentre dois construtores sintáticos: seqüência e escolha. Uma *seqüência* $(e_1, e_2, \ldots, e_n)$ define n elementos

componentes ordenados, com $n \geq 1$[1]. Uma *escolha* ($\mathtt{e}_1|\mathtt{e}_2|\ldots|\mathtt{e}_m$) define m alternativas de elementos componentes, $m > 1$. Os operadores de expressões regulares '?', '*' e '+' denotam o número de ocorrências permitidas de um elemento componente, indicando 0 ou 1 ocorrência, 0 a n ocorrências e 1 a n ocorrências, respectivamente.

Elemento Composto Opcional. Um elemento composto opcional **é um** elemento composto cujo modelo de conteúdo é seguido pelo operador regular '?', como por exemplo, $\mathtt{E(e}_1\mathtt{, e}_2\mathtt{, \ldots, e}_n\mathtt{)?}$.

Elemento Composto com Repetição. Um elemento composto com repetição **é um** elemento composto cujo modelo de conteúdo é seguido pelo operador regular '*' ou '+', como por exemplo, $\mathtt{E(e}_1|\mathtt{e}_2|\ldots|\mathtt{e}_m\mathtt{)*}$.

Elemento Componente Nomeado. Um elemento componente nomeado é um nome de elemento declarado no modelo de conteúdo de um elemento composto.

Elemento Componente Texto. Um elemento componente texto é uma cláusula `#PCDATA` declarada no modelo de conteúdo de um elemento composto.

Elemento Componente. Um elemento componente é um elemento componente nomeado ou um elemento componente texto.

Elemento Componente com repetição. Um elemento componente com repetição **é um** elemento componente que está declarado mais de uma vez no modelo de conteúdo de um elemento composto, como por exemplo o elemento x na seqüência ($\mathtt{e}_1\mathtt{, e}_2\mathtt{, \ldots, x, }e_n\mathtt{, x}$).

Elemento Simples. Um elemento E é dito simples quando seu modelo de conteúdo é definido pela cláusula `#PCDATA` e E não possui atributos.

5.2 Processo de Conversão

A etapa de conversão da DTD executa o mapeamento de uma DTD para um esquema no MCC. Tal tarefa é complexa e não pode ser realizada de forma totalmente automática, pois está-se convertendo um esquema lógico em um esquema conceitual. Como um esquema conceitual é semanticamente mais rico que um esquema lógico, a intervenção do usuário é requerida em alguns momentos para confirmar certas inferências semânticas.

Esta etapa é fundamental para o processo de integração de BInXS pois é necessário que a intenção semântica de cada DTD esteja correta para que a etapa seguinte do processo possa determinar correspondências válidas entre conceitos a fim de realizar a unificação de esquemas. Assim sendo, considera-se esta etapa a mais importante do processo de integração de BInXS.

A figura 5.1 ilustra o processo de conversão de uma DTD, que ocorre em três passos: pré-processamento, aplicação de regras de conversão e reestruturação do esquema conceitual preliminar.

O passo de *pré-processamento* tem por objetivo a produção de uma DTD livre de detalhes técnicos de estruturação que não contribuem para o entendimento da semântica dos seus dados. Tais detalhes estão relacionados com a remoção de

[1] A ordem dos sub-elementos de um elemento não é modelada no MCC por ser considerada pouco relevante para fins de consulta.

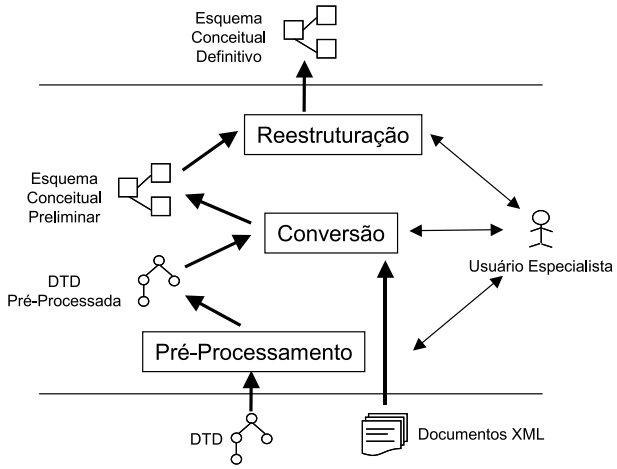


FIGURA 5.1 – Processo de conversão da DTD

estruturas aninhadas de elementos compostos e de elementos semanticamente desnecessários, dentre outras tarefas. Este passo recebe uma DTD e produz uma DTD "filtrada"com relação a estes detalhes técnicos, que é chamada de *DTD pré-processada*. Este passo é detalhado na seção 5.4.

O passo de *Conversão* é o núcleo desta etapa. Ele recebe uma DTD pré-processada e produz um esquema conceitual preliminar através da aplicação de regras que traduzem elementos e atributos em conceitos do MCC. A definição do esquema conceitual não leva em conta somente a estrutura da DTD mas também o conteúdo de documentos XML que estão de acordo com a DTD, heurísticas e questionamentos ao usuário especialista. Este passo é detalhado na seção 5.5.

O passo de *Reestruturação* realiza simplificações no esquema conceitual preliminar e alguns ajustes manuais. Ao final, um esquema conceitual definitivo para a DTD é produzido. Este passo é detalhado na seção 5.6.

As próximas seções detalham estes três passos, utilizando três DTDs exemplo descritas na seção a seguir.

5.3 DTDs Exemplo

Três DTDs fictícias para um domínio bibliográfico são apresentadas a seguir. Estas DTDs têm caráter didático, servindo apenas para exemplificar as regras e procedimentos do processo de integração. Um estudo de caso real é mostrado no capítulo 6.

A DTD1 diz respeito a um documento XML com a literatura de preferência de um certo leitor. A DTD2 considera o conjunto de periódicos publicado por uma editora de nome *ComputerScience*, ou seja, artigos em anais de conferências ou revistas na área de Ciências da Computação. A DTD3 esquematiza publicações científicas de autores, representando o *curriculum vitae* de cada um deles.

```
<!----- DTD 1: Livros Preferidos de um Leitor  ----->

<!ELEMENT MyFavouriteBooks (Book+, Best-Referenced)>
<!ELEMENT Book (WriterList, Year, PublishingHouse?, (Technical | Fiction))>
```

```
<!ATTLIST Book Title CDATA #REQUIRED>
<!ELEMENT WriterList (Writer)+>
<!ELEMENT Writer (Name, EMail, EMail?)>
<!ATTLIST Writer Style CDATA (science | arts | romance | horror | drama
          | comedy)>
          PreferenceRating CDATA>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT PublishingHouse (#PCDATA)>
<!ELEMENT Technical EMPTY>
<!ELEMENT Fiction EMPTY>
<!ELEMENT Best-Referenced ANY>

<! ----- DTD 2: Acervo de Periódicos da Editora ComputerScience ----->

<!ENTITY PublicationType "Proceedings | Journal">

<!ELEMENT ComputerScienceIncLiterature (Publication+)>
<!ELEMENT Publication (Title, Year, (%PublicationType;))*>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Proceedings (Forum, Paper+, Schedule)>
<!ELEMENT Forum (#PCDATA)>
<!ATTLIST Forum ID CDATA    venue CDATA #REQUIRED>
<!ELEMENT Paper (Title, Author+, ReferencedPapers?)>
<!ELEMENT ReferencedPapers(Paper*)>
<!ELEMENT Author (Name, Address?, (University+ | Enterprise), Office?)>
<!ELEMENT University (#PCDATA)>
<!ELEMENT Enterprise (#PCDATA)>
<!ELEMENT Office (#PCDATA)>
<!ELEMENT Address ((Street, Number)?, City, Country)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Schedule (#PCDATA | Session | Talk)*>
<!ELEMENT Session (#PCDATA)>
<!ELEMENT Journal (Name, Volume, Number, Paper+)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Volume (#PCDATA)>

<!----- DTD 3: Curriculum Vitae de Publicações Científicas ----->

<!ELEMENT MyCurriculumVitae (PersonalData, Publication+)>
<!ELEMENT PersonalData (Name, Sex, Address, Address?, (University |
ResearchInstitute | Business | Office), Occupation)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Sex (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT University (#PCDATA)>
<!ELEMENT ResearchInstitute (#PCDATA)>
<!ELEMENT Business (#PCDATA)>
<!ELEMENT Office (#PCDATA)>
<!ELEMENT Occupation (Teacher | Researcher | Student | Developer
| (Teacher, Student))>
<!ELEMENT Teacher (ResearchInterest+, Project*)>
<!ATTLIST Teacher rank CDATA (BsC, MsC, PhD) SSN ID #REQUIRED>
<!ELEMENT ResearchInterest (#PCDATA)>
```

```
<!ELEMENT Project (#PCDATA)>
<!ELEMENT Researcher (Project, Function)+>
<!ELEMENT Function (#PCDATA)>
<!ELEMENT Student (Course)>
<!ATTLIST Student level CDATA (BsC, MsC, PhD) advisor IDREF>
<!ELEMENT Developer (Function, Platform+)>
<!ELEMENT Platform (#PCDATA)>
<!ELEMENT Publication (Book | Proceedings)>
<!ELEMENT Book (Title, Year, Pages, Publisher)>
<!ATTLIST Book ISBN ID #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Pages (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Article (#PCDATA)>
<!ELEMENT Proceedings (Conference, Year, Article)>
<!ELEMENT Conference (#PCDATA)>
```

5.4 Passo 1: Pré-Processamento

O passo de pré-processamento realiza cinco tarefas, nesta ordem: (i) substituição de entidades, (ii) remoção de elementos componentes e atributos "sem semântica", (iii) remoção de estruturas aninhadas, (iv) tratamento de *elementos compostos opcionais* ou *com repetição* e (v) alteração de nomes de elementos e atributos. As tarefas (ii) e (v) exigem a intervenção do usuário. As próximas seções explicam estas tarefas.

Assume-se que uma DTD de entrada para este passo seja *completa*, ou seja, não contenha referências a definições externas de elementos e atributos que devam ser buscadas em outras fontes de dados na *Web*. Se uma DTD não for completa, ela deve ser modificada para se tornar completa, incluindo estas definições externas.

5.4.1 Substituição de Entidades

Uma DTD pode conter definições de entidades através de cláusulas `<!ENTITY ...>`. Uma *entidade* define total ou parcialmente o modelo de conteúdo de um elemento, sendo em geral utilizada na definição de diversos elementos. A substituição do conteúdo das entidades em todos os elementos que fazem referência a elas é necessária para tornar completo o modelo de conteúdo destes elementos.

Um exemplo de substituição de entidade ocorre na DTD2. O conteúdo da entidade `PublicationType` é substituído pela sua referência no elemento `Publication`, como mostrado a seguir.

```
<!ENTITY PublicationType "Proceedings | Journal">
<!ELEMENT Publication (Title, Year, (%PublicationType;))*>
                              ⇓
<!ELEMENT Publication (Title, Year, (Proceedings | Journal))*>
```

5.4.2 Remoção de Elementos e Atributos Semanticamente Desnecessários

Em uma DTD podem existir elementos que servem apenas para estruturar partes do modelo de conteúdo de um *elemento composto*, mas que do ponto de vista semântico são irrelevantes, ou seja, não são informações necessárias para o domínio em questão. Um exemplo é o elemento componente `WriterList` do elemento `Book` na DTD1. Este elemento serve apenas para encapsular a lista de autores de um livro, sendo, na realidade, um elemento intermediário no relacionamento entre livro e autor. Outros exemplos são os elementos `PersonalData` de `MyCurriculumVitae` na DTD3, que estrutura separadamente dados pessoais de um autor, `MyFavouriteBooks` na DTD1, que agrega uma lista de livros e `Computer ScienceIncLiterature` na DTD2, que mantém uma lista de publicações.

Para evitar que tais elementos sejam considerados no esquema conceitual, é interessante que eles sejam removidos. Esta remoção é uma tarefa manual pois apenas o usuário especialista tem condições de identificá-los. Um elemento E selecionado para remoção tem o seu modelo de conteúdo incluído em todos os elementos que fazem referência a ele. Eventuais ajustes de cardinalidade como conseqüência da inclusão do modelo de conteúdo de E em outro elemento são resolvidos através de uma função chamada *Card*, definida na próxima seção.

A seguir é exemplificada a remoção do elemento `WriterList` de `Book` na DTD1.

```
<!ELEMENT Book (WriterList, Year, PublishingHouse, (Technical | Fiction))>
                        <!ELEMENT WriterList (Writer)+>
                          ⇓ remoção do elemento WriterList
<!ELEMENT Book ((Writer)+, Year, PublishingHouse, (Technical | Fiction))>
```

`WriterList` apresenta cardinalidade (1,1) em `Book` e cardinalidade (1,N) no seu modelo de conteúdo. A cardinalidade resultante da sua inclusão em `Book` é (1,N).

Outros elementos removidos foram `PersonalData` na DTD3, `MyFavouriteBooks` na DTD1 e `ComputerScienceIncLiterature` na DTD2.

Um elemento só é removido se não tiver atributos. Porém, atributos considerados semanticamente irrelevantes também podem ser removidos. Um atributo removido da DTD1 é *PreferenceRating*, que mantém o grau de preferência do autor da DTD por um certo escritor.

Um elemento composto E que possui apenas um elemento componente, como `WriterList` na DTD1, é forte candidato a ser removido pois mantém separadamente apenas um sub-elemento que poderia estar definido diretamente nos elementos compostos que têm `E` como sub-elemento. Quando E tem mais de um elemento componente, pode-se apenas modificar o seu nome ao invés de introduzir o seu modelo de conteúdo em um elemento E' onde ele é sub-elemento, quando as propriedades de E não forem condizentes com propriedades de E'. Este não foi o caso para `PersonalData` em relação a `MyCurriculumVitae` na DTD3, mas pode se aplicar a outras DTDs. A tarefa de renomeação é explicada na seção 5.4.5.

Outro ponto a salientar é que esta tarefa introduz estruturas aninhadas em elementos compostos, que juntamente com outras já predefinidas na DTD, devem

ser removidas (`Book` recebeu um aninhamento (`Writer)+` na DTD1, por exemplo). A próxima seção explica esta situação.

5.4.3 Remoção de Estruturas Aninhadas

Um *elemento composto* E pode ter um componente que seja uma estrutura aninhada na forma de seqüência ou escolha, ou seja, um grupo G de elementos componentes. Para uma conversão adequada de elementos da DTD pelas regras do passo de *Conversão*, é necessário que E possua apenas *elementos componentes nomeados* ou *elementos componentes texto*. Desta forma, é possível determinar precisamente a cardinalidade do relacionamento entre os conceitos correspondentes à E e seus componentes no esquema conceitual. Deve-se, então, remover estas estruturas aninhadas.

Quando se remove uma estrutura aninhada G, deve-se verificar se os elementos componentes de G podem ser declarados como componentes de E. Isto nem sempre é possível pois existem casos em que uma restrição de cardinalidade aplicada à G não pode ser aplicada aos seus elementos componentes individualmente em E. Um destes casos é exemplificado a seguir para um elemento composto `Author`, que mantém uma estrutura aninhada `(street, number, city, country)?` referente a dados de seu endereço. A definição de `Author` em (i) não é equivalente à definição de `Author` em (ii) porque dados do endereço de um autor não podem existir parcialmente, como permite a definição (ii). Elas devem ou existir completamente ou não existir.

```
<!ELEMENT Author(..., (street, number, city, country)?, ...)> (i)
```

$$\neq$$

```
<!ELEMENT Author(..., street?, number?, city?, country?, ...)> (ii)
```

Para tratar tais casos, faz-se necessário um procedimento de *normalização* da DTD nos moldes de um processo de normalização para bancos de dados relacionais [ELM 2000]. Esta normalização deve manter os modelos de conteúdo de todos os elementos da DTD na primeira forma normal, ou seja, considerar G como sendo um novo elemento composto E' na DTD e definir E' como sendo um elemento componente de E. E' é chamado de *elemento virtual* pois é derivado a partir da definição de um elemento definido na DTD.

A remoção de estruturas aninhadas é regida por duas regras que tratam modelos de conteúdo de E na forma de seqüência ou escolha. Ambas as regras, apresentadas a seguir, utilizam uma regra auxiliar de unificação de operadores regulares (regra *Card*) que determina a cardinalidade geral dos elementos componentes de G.

Regra Card (Unificação de Operadores Regulares). *Dadas duas cardinalidades representadas pelos operadores regulares op_1 e op_2, podendo ambas serem vazias (-), a cardinalidade resultante é determinada de acordo com a seguinte tabela:*

op_1 \ op_2	-	?	*	+
-	-	?	*	+
?	?	?	*	*
*	*	*	*	*
+	+	*	*	+

Regra NormSeq (Normalização de uma Seqüência). *Dado um elemento composto* `E(`e_1`, ..., `G_i`<`c_G`>, ..., `e_n`)` *tal que* `$G_i$` = `(`g_1`<`c_1`>, ..., `g_m`<`c_m`>)`, `m` $\geq$ 1, *é a i-ésima estrutura aninhada no seu modelo de conteúdo,* `<`c_G`>` *é a cardinalidade de* `$G_i$` *e* `<`c_j`>` *é a cardinalidade do j-ésimo componente de* `$G_i$`. *Para tanto, valem as seguintes transformações:*

$$G_i = \begin{cases} (g_1 < c_1 >) < c_G > \\ \Rightarrow E(e_1, ..., g_1 < Card(c_1, c_G) >, ..., e_n) & (i) \\ (g_1 < c_1 >, ..., g_m < c_m >) < c_G > \\ \Rightarrow \begin{cases} E(e_1, ..., g_1 < c_1 >, ..., g_m < c_m >, ..., e_n), & \text{se } <c_G> \in \{\text{-}\} \quad (ii) \\ E(e_1, ..., EGroup_i < c_G >, ..., e_n) \\ EGroup_i(g_1 < c_1 >, ..., g_m < c_m >), & \text{se } <c_G> \in \{?,*,+\} \quad (iii) \end{cases} \\ (g_1 < c_1 > | ... | g_m < c_m >) < c_G > \\ \Rightarrow \begin{cases} E(e_1, ..., g_1 < Card(c_1, c_G) > | ... | g_m < Card(c_m, c_G) >, ..., e_n), & \text{se } <c_G> \in \{\text{-},?\} \, (iv) \\ E(e_1, ..., g_1*, ..., g_m*, ..., e_n), & \text{se } <c_G> \in \{*,+\} \, (v) \end{cases} \end{cases}$$

Regra NormEsc (Normalização de uma Escolha). *Dado um elemento com componentes* `E(`e_1 | ... | `$G_i$`<`$c_G$`>` | ... | $e_n$`)` *tal que* `G_i` = `(`$g_1$`<`$c_1$`>, ..., `$g_m$`<`$c_m$`>)`, `m` $\geq$ 1, *é a i-ésima estrutura aninhada no seu modelo de conteúdo,* `<`$c_G$`>` *é a cardinalidade de* `G_i` *e* `<`$c_j$`>` *é a cardinalidade do j-ésimo componente de* `G_i`. *Para tanto, valem as seguintes transformações:*

$$G_i = \begin{cases} (g_1 < c_1 >) < c_G > \\ \Rightarrow E(e_1 | ... | g_1 < Card(c_1, c_G) > | ... | e_n) & (i) \\ (g_1 < c_1 >, ..., g_m < c_m >) < c_G > \\ \Rightarrow E(e_1 | ... | EGroup_i < c_G > | ... | e_n) & (ii) \\ \quad EGroup_i(g_1 < c_1 >, ..., g_m < c_m >) \\ (g_1 < c_1 > | ... | g_m < c_m >) < c_G > \\ \Rightarrow \begin{cases} E(e_1 | ... | g_1 < Card(c_1, c_G) > | ... | g_m < Card(c_m, c_G) > | ... | e_n), & \text{se } <c_G> \in \{\text{-},?\} \; (iii) \\ E(e_1 | ... | EGroup_i < c_G > | ... | e_n) \\ EGroup_i(g_1 < c_1 >, ..., g_m < c_m >), & \text{se } <c_G> \in \{*,+\} (iv) \end{cases} \end{cases}$$

Conforme as definições das regras, elementos virtuais criados apresentam um nome formado pela concatenação do nome do elemento a que pertencem (`E`) mais a palavra *Group*, seguido do número de ordem da estrutura aninhada no modelo de conteúdo de `E`. O caso (i) da regra **NormSeq**, que lida com grupos de apenas um elemento, é aplicado para o elemento `Book` da DTD1:

```
<!ELEMENT Book ((Writer)+, Year, PublishingHouse, (Technical | Fiction))>
```

⇓ **NormSeq(i)**

```
<!ELEMENT Book (Writer+, Year, PublishingHouse, (Technical | Fiction))>
```

A aplicação do caso (iii) da regra *NormSeq* ocorre para o elemento `Address` da DTD2:

```
<!ELEMENT Address ((Street, Number)?, City, Country)>
```

⇓ **NormSeq(iii)**

```
<!ELEMENT Address (AddressGroup1?, City, Country)>
<!ELEMENT AddressGroup1 (Street, Number)>
```

Uma aplicação do caso (iv) da regra *NormSeq* ocorre para o elemento `Book` da DTD1, como mostrado a seguir.

```
<!ELEMENT Book (Writer+, Year, PublishingHouse, (Technical | Fiction))>
```

⇓ **NormSeq(iv)**

```
<!ELEMENT Book (Writer+, Year, PublishingHouse, Technical | Fiction)>
```

A transformação no modelo de conteúdo de `Book`, apesar de inválida na gramática de uma DTD, é aceita na DTD pré-processada, dando-se precedência para a avaliação do operador escolha (|).

O caso (v) da regra **NormSeq** é peculiar no sentido de que transforma uma escolha em uma seqüência. Uma seqüência teoricamente implica uma ordem de definição de elementos componentes, porém, assumindo que a noção de ordem não é levada em conta na modelagem conceitual da DTD, esta transformação é considerada válida.

A aplicação do caso (ii) da regra **NormEsc** é exemplificada para o elemento `Occupation` da DTD3 através da geração de um elemento virtual para o componente `(Teacher, Student)`.

```
<!ELEMENT Occupation (Teacher | Researcher | Student | (Teacher, Student))>
```

⇓ **NormEsc(ii)**

```
<!ELEMENT Occupation (Teacher | Researcher | Student | OccupationGroup1)>
<!ELEMENT OccupationGroup1 (Teacher, Student)>
```

5.4.4 Tratamento de Elementos Compostos Opcionais ou com Repetição

Elementos compostos opcionais ou com repetição também devem ser tratados pois existem casos nos quais novamente uma restrição de cardinalidade não pode ser aplicada a cada elemento componente individualmente. As regras a seguir definem este tratamento.

Regra SeqOpRep (Tratamento de Seqüências Opcionais ou com Repetição). *Dado um elemento composto* $E = (e_1\langle c_1\rangle, \ldots, e_n\langle c_n\rangle)\langle c_E\rangle$, $n \geq 1$, *tal que* $\langle c_E\rangle$ *é a cardinalidade de* E, $\langle c_E\rangle \in \{?, *, +\}$, *e* $\langle c_j\rangle$ *é a cardinalidade do j-ésimo componente de* E. *Para tanto, valem as seguintes transformações:*

- $E(e_1\langle c_1\rangle)\langle c_E\rangle \Rightarrow E(e_1\langle Card(c_1, c_E)\rangle)$ *(i)*
- $E(e_1\langle c_1\rangle, \ldots, e_n\langle c_n\rangle)\langle c_E\rangle \Rightarrow E(EGroup\langle c_E\rangle)$
  $EGroup(e_1\langle c_1\rangle, \ldots, e_n\langle c_n\rangle)$ *(ii)*

Regra EscOpRep (Tratamento de Escolhas Opcionais ou com Repetição). *Dado um elemento composto* $E = (e_1<c_1>| \ldots |e_n<c_n>)<c_E>$, $n > 1$, *tal que* $<c_E>$ *é a cardinalidade de* E, $<c_E> \in \{?, *, +\}$, *e* $<c_j>$ *é a cardinalidade do j-ésimo componente de* E. *Para tanto, valem as seguintes transformações:*

- $E(e_1<c_1>| \ldots |e_n<c_n>)? \Rightarrow E(e_1<Card(c_1, ?)> | \ldots | e_n<Card(c_n, ?)>)$ *(i)*
- $E(e_1<c_1>| \ldots |e_n<c_n>)* \Rightarrow E(e_1*, \ldots, e_n*)$ *(ii)*
- $E(e_1<c_1>| \ldots |e_n<c_1>)+ \Rightarrow E(EGroup+)$
  $EGroup(e_1<c_1>| \ldots |e_n<c_n>)$ *(iii)*

A regra **SeqOpRep** caso (ii) cria um elemento virtual correspondente ao modelo de conteúdo de `E` quando este modelo define mais de um elemento componente. Isto ocorre porque a restrição de cardinalidade se aplica sempre ao conjunto de componentes e não a cada um individualmente. Caso contrário (caso (i)), a cardinalidade externa é unificada à cardinalidade do próprio elemento componente.

Um exemplo de aplicação do caso (ii) ocorre na DTD3 para o elemento `Researcher`, como mostrado a seguir.

```
<!ELEMENT Researcher (Project, Function)+>
```

⇓ **SeqOpRep(ii)**

```
<!ELEMENT Researcher (ResearcherGroup1+)>
<!ELEMENT ResearcherGroup1 (Project, Function)>
```

Um pesquisador está associado a vários projetos, tendo uma função em cada um deles. Um elemento virtual é necessário para manter o relacionamento de pesquisador com pares <projeto, função>.

O caso (iii) da regra **EscOpRep** cria um elemento virtual porque a escolha como um todo deve ocorrer pelo menos uma vez. Não é possível aplicar o mesmo raciocínio do caso (ii) (gerando `E(`e_1`+, ..., `e_n`+)`) pois isto exigiria que todos os elementos componentes estejam relacionados a `E` pelo menos uma vez. Também não vale a transformação `E(`e_1{+,*} | ... | e_n{+,*})` pois apenas um elemento componente estaria relacionado a E uma/zero ou mais vezes.

Um exemplo da aplicação da regra **EscOpRep** caso (ii) ocorre na DTD2 para o elemento `Schedule`.

```
<!ELEMENT Schedule (#PCDATA | Session | Talk)*>
```

⇓ **EscOpRep(ii)**

```
<!ELEMENT Schedule ((#PCDATA)*, Session*, Talk*)>
```

5.4.5 Renomeação de Elementos e Atributos

Certos elementos e atributos de uma DTD apresentam nomes que não contribuem para a compreensão da intenção semântica ou, então, estão inseridos no contexto do domínio, mas agregados a particularidades do seu criador. Um exemplo é o elemento `MyCurriculumVitae` na DTD3, que se refere a informações de autores.

Para tanto, é necessária mais uma vez a intervenção do usuário especialista para a definição de nomes mais adequados. No caso de `MyCurriculumVitae`, vale a seguinte renomeação:

`MyCurriculumVitae` ⇒ Author

Nomes de elementos virtuais obrigatoriamente devem ser renomeados pelo usuário, pois seus nomes não carregam semântica. Para alguns destes elementos, criados na seção anterior, possíveis renomeações são:

`ResearcherGroup1` ⇒ Job (DTD3)
`AddressGroup1` ⇒ Home (DTD2)
`OccupationGroup1` ⇒ Teacher-Student (DTD3)

Convenciona-se que nomes compostos são separados por um hífen pelo usuário, como `Publishing-House` na DTD1.

Esta tarefa de renomeação é a última realizada pelo passo de pré-processamento. A ordem de execução das cinco tarefas deste passo é a ordem em que as mesmas foram apresentadas nas seções anteriores. Esta ordem é relevante pois evita que uma tarefa tenha de ser executada mais de uma vez. Por exemplo, a tarefa (ii) (remoção de elementos e atributos) pode gerar estruturas aninhadas. Portanto, a remoção de estruturas aninhadas (tarefa (iii)) deve ser feita posteriormente.

5.4.6 Informações de Mapeamento

As modificações realizadas em uma DTD pelas tarefas de pré-processamento, como remoção e renomeação de elementos, necessitam ser registradas, de modo que um mapeamento posteriormente gerado para conceitos do esquema global seja fiel a nomenclatura e a estrutura desta DTD original.

O esquema conceitual mostrado na figura 5.2 descreve os metadados de mapeamento que podem ser gerados na etapa de pré-processamento. Este esquema serve de base para a construção de um *catálogo temporário* a ser utilizado no passo seguinte de *Conversão* para a correta definição dos mapeamentos entre um esquema conceitual e a DTD original. Este catálogo pode ser, por exemplo, um conjunto de tabelas relacionais ou classes DAML+OIL.

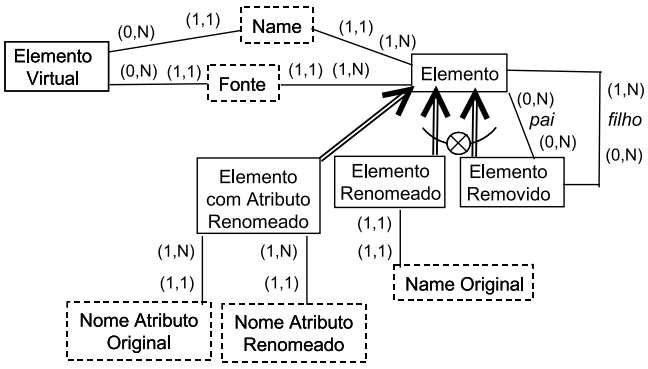


FIGURA 5.2 – Esquema conceitual de metadados de mapeamento temporários

Os dados deste catálogo registram as seguintes modificações na DTD original:

- *Elemento removido*: *nome* deste elemento em uma *fonte* (URL que localiza uma DTD, por exemplo) e *nomes* dos seus *elementos pais* e *filhos*. Desta forma é possível construir o caminho original que une dois elementos em uma DTD, considerando elementos que não estão representados no esquema conceitual.

Caso algum pai ou filho seja também um elemento removido, o caminho original pode ser reconstruído recursivamente através de vários acessos a estes dados;

- *Elemento renomeado*: *nome original* de um elemento em uma *fonte* para um *nome* de elemento na DTD pré-processada;
- *Atributo renomeado*: *nome original* de um atributo em uma *fonte* e *novo nome*, para um *nome* de elemento na DTD pré-processada;
- *Elemento virtual*: *nome* de um elemento virtual gerado para uma *fonte*. O cadastro de um elemento E como virtual permite inferir que os filhos de E são na verdade filhos do elemento que é pai de `E` na DTD original.

As consultas a este catálogo são explicadas na seção 5.5.4, quando da geração de informações de mapeamento para os conceitos e relacionamentos do esquema conceitual preliminar criados pelo passo de *Conversão*.

5.4.7 Aplicação do Passo de Pré-Processamento sobre as DTDs Exemplo

A aplicação do passo de pré-processamento sobre as DTDs exemplo resulta nas DTDs pré-processadas apresentadas a seguir, com destaque para os elementos e atributos que foram modificados ou criados.

```
<!----- DTD 1 ----->

<!ELEMENT Book (Writer+, Year, Publishing-House?, Technical | Fiction)>
<!ATTLIST Book Title CDATA #REQUIRED>
<!ELEMENT Writer (Name, EMail, EMail?)>
<!ATTLIST Writer Style CDATA (science | arts | romance | horror | drama
| comedy)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Publishing-House (#PCDATA)>
<!ELEMENT Technical EMPTY>
<!ELEMENT Fiction EMPTY>
<!ELEMENT Best-Referenced ANY>

<! ----- DTD 2 ----->

<!ELEMENT Publication (Title, Year, Proceedings | Journal)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Proceedings (Forum, Paper+, Schedule)>
<!ELEMENT Forum (#PCDATA)>
<!ATTLIST Forum ID CDATA    venue CDATA #REQUIRED>
<!ELEMENT Paper (Title, Author+, Paper*)>
<!ELEMENT Author (Name, Address?, University+ | Enterprise, Office?)>
<!ELEMENT University (#PCDATA)>
<!ELEMENT Enterprise (#PCDATA)>
<!ELEMENT Office (#PCDATA)>
<!ELEMENT Address (Home?, City, Country)>
<!ELEMENT Home (Street, Number)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
```

```
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Schedule ((#PCDATA)*, Session*, Talk*)>
<!ELEMENT Session (#PCDATA)>
<!ELEMENT Journal (Name, Volume, Number, Paper+)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Volume (#PCDATA)>

<!----- DTD 3 ----->
<!ELEMENT Author (Name, Sex, Address, Address?, University |
Research-Institute | Business | Office), Occupation, Publication+)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT Sex (#PCDATA)>
<!ELEMENT University (#PCDATA)>
<!ELEMENT Research-Institute (#PCDATA)>
<!ELEMENT Business (#PCDATA)>
<!ELEMENT Occupation (Teacher | Researcher | Student |
Developer | Teacher-Student)>
<!ELEMENT Teacher-Student (Teacher, Student)>
<!ELEMENT Teacher (ResearchInterest+, Project*)>
<!ATTLIST Teacher rank CDATA (BsC, MsC, PhD) SSN ID #REQUIRED>
<!ELEMENT ResearchInterest (#PCDATA)>
<!ELEMENT Project (#PCDATA)>
<!ELEMENT Researcher (Job+)>
<!ELEMENT Job (Project, Function)>
<!ELEMENT Function (#PCDATA)>
<!ELEMENT Student (Course)>
<!ATTLIST Student level CDATA (BsC, MsC, PhD)  advisor IDREF>
<!ELEMENT Developer (Function, Platform+)>
<!ELEMENT Platform (#PCDATA)>
<!ELEMENT Publication (Book | Proceedings)>
<!ELEMENT Book (Title, Year, Pages, Publisher)>
<!ATTLIST Book ISBN ID #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Pages (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Proceedings (Conference, Year, Article)>
<!ELEMENT Conference (#PCDATA)>
<!ELEMENT Article (#PCDATA)>
```

5.5 Passo 2: Conversão

O passo de conversão traduz uma DTD pré-processada para um esquema conceitual preliminar no MCC. Este passo é centrado em um conjunto de *regras de conversão*. Estas regras, apresentadas na seqüência, consideram a estrutura da DTD, heurísticas, análise de documentos XML e intervenção do usuário especialista.

5.5.1 Regras Auxiliares de Cardinalidade

As cardinalidades diretas dos relacionamentos de associação são determinadas através de regras auxiliares de cardinalidade. Estas regras são definidas a seguir.

Regra Card$_D$ (Cardinalidade Direta). *Dada a cardinalidade <c_i> de um elemento componente, a cardinalidade direta $Card_D$ é dada por:*

$$Card_D = \begin{cases} (0,1), & se\ c_i = \text{'?'}; \\ (0,\text{N}), & se\ c_i = \text{'*'}; \\ (1,\text{N}), & se\ c_i = \text{'+'}; \\ (1,1), & caso\ contrário. \end{cases}$$

Regra Card$_R$ (Cardinalidade Direta com Repetição). *Dado um conjunto de cardinalidades C_c = {<c_1>, ..., <c_n>} de um elemento componente que se repete n vezes, a cardinalidade direta com repetição $Card_R$ é dada por:*

- $Card_R.mínima = \sum_n^{i=1} Card_D(<c_i>).mínima$;
- $Card_R.máxima =$

$$\begin{cases} \text{N}, & se\ \exists <c_i> \in C_c\ (Card_D(<c_i>).máxima = N); \\ \sum_n^{i=1} Card_D(<c_i>).máxima, & caso\ contrário. \end{cases}$$

Regra Card$_U$ (Cardinalidade Direta Unificada). *Dado um conjunto de cardinalidades diretas C_c = {$Card_D1$, ..., $Card_Dn$}, a cardinalidade direta unificada ou geral $Card_U$ é dada por:*

- $Card_U.mínima = min(\text{Card}_D\text{1}.mínima, ..., \text{Card}_D\text{n}.mínima)$;
- $Card_U.máxima =$

$$\begin{cases} \text{N}, & se\ \exists\ \text{Card}_D\text{i} \in C_c\ (\text{Card}_D\text{i}.máxima = N); \\ max(\text{Card}_D\text{1}.máxima, ..., \text{Card}_D\text{n}.máxima), & caso\ contrário. \end{cases}$$

A regra $Card_R$ determina a cardinalidade direta de um único relacionamento de associação entre um *elemento composto* e um *elemento componente com repetição*. A regra $Card_U$ determina a cardinalidade mais ampla dentre um conjunto de cardinalidades, como por exemplo $Card_U$({(1,1), (2, N), (0,1)}) = (0,N). Ela é utilizada não só na etapa de *Conversão da DTD* mas também na etapa de *Integração Semântica* para a unificação de relacionamentos com afinidade.

5.5.2 Regras de Conversão

As regras de conversão tratam três tipos de definições em uma DTD: (i) definições de elementos, (ii) definições de atributos e (iii) definições de elementos componentes. As definições das regras consideram as definições dos construtores do MCC mostradas na seção 4.1.1.

Definições de Elementos

Duas são as regras que lidam com *definições de elementos*. A primeira diz que todo elemento complexo, por agregar sub-elementos ou atributos, deve ser definido como um conceito não-léxico.

Regra EC (Conversão de Elemento Complexo). *Um elemento complexo* `E` *gera* <`E.nome`> $\in$ *NL.*

Exemplos:

<!ELEMENT Best-Referenced ANY> $\Rightarrow$ <Best-Referenced> $\in$ *NL* (DTD1)
<!ELEMENT Forum (...)> $\Rightarrow$ <Forum> $\in$ *NL* (DTD2)
<!ELEMENT Author (...)> $\Rightarrow$ <Author> $\in$ *NL* (DTD3)
<!ELEMENT Publication (...)> $\Rightarrow$ <Publication> $\in$ *NL* (DTD3)

A segunda regra é responsável pela geração de conceitos léxicos.

Regra ES (Conversão de Elemento Simples). *Um elemento simples* `E` *gera* <`E.nome`, `tipoDado`> $\in$ *L, onde* `tipoDado` *é determinado através de análise de documentos XML ou definido como string por default.*

As tarefas que envolvem análise de documentos XML são detalhadas na seção 5.5.5. Caso esta análise não determine precisamente um tipo de dado para todos as instâncias do elemento, o tipo *string* é assumido.

Exemplos:

<!ELEMENT Publishing-House (#PCDATA)> $\Rightarrow$ <Publishing-House, string> $\in L$ (DTD1)
<!ELEMENT Year (#PCDATA)> $\Rightarrow$ <Year, integer> $\in L$ (DTD2)
<!ELEMENT Sex (#PCDATA)> $\Rightarrow$ <Sex, character> $\in L$ (DTD3)
<!ELEMENT Project (#PCDATA)> $\Rightarrow$ <Project, string> $\in L$ (DTD3)

Definições de Atributos

As regras que lidam com *definições de atributos* também são duas. A primeira é mais ampla, tratando a conversão de atributos de modo geral. A segunda é específica para atributos identificadores e de referência.

Regra AT (Conversão de Atributo). *Dados um elemento complexo* `E` *e um conjunto de atributos* `At` $= \{a_1, ..., a_n\}$ *pertencente à* `E`*, gera-se, para cada atributo* $a_i \in$ `At`*:*

- $\begin{cases} <a_i\texttt{.nome, tipoDado}> \in L, & \text{se } a_i \text{ não possui enumeração;} \\ <a_i\texttt{.nome, tipoDado, } a_i\texttt{.enumeração}> \in L, & \text{se } a_i \text{ possui enumeração.} \end{cases}$
- $\begin{cases} <\texttt{E.nome, } a_i\texttt{.nome, (1,1), } Card_I> \in A, & \text{se } a_i \text{ é seguido da cláusula } \texttt{\#REQUIRED;} \\ <\texttt{E.nome, } a_i\texttt{.nome, (0,1), } Card_I> \in A, & \text{caso contrário.} \end{cases}$

`tipoDado` *e a cardinalidade inversa* $Card_I$ *são determinados através de análise de documentos XML ou definidos como string e (1,N) por default, respectivamente.*

Exemplos:

<!ATTLIST Writer Style CDATA (science | arts | romance | horror | drama | comedy))>
$\Rightarrow$ <Style, string, {science, arts, romance, horror, drama, comedy}> $\in L$
<Writer, Style, (0,1), (1,N)> $\in A$ (DTD1)

<!ATTLIST Forum Venue CDATA #REQUIRED)> $\Rightarrow$
<Venue, string> $\in L$
<Forum, Venue, (1,1), (1,N)> $\in A$ (DTD2)

<!ATTLIST Book ISBN ID #REQUIRED)> $\Rightarrow$
<ISBN, string> $\in L$
<Book, ISBN, (1,1), (1,N)> $\in A$ (DTD3)

A regra **ATR**, definida a seguir, considera a geração de relacionamentos de associação para atributos de referência `IDREF(S)` ou `[xml:link] href` que se refiram sempre a um mesmo tipo de elemento E na DTD. Este relacionamento é definido entre o elemento que possui o atributo de referência e E. O relacionamento possui um nome, que é o nome do atributo. A verificação de referências para um mesmo tipo de elemento é feita através da análise de documentos XML.

Regra ATR (Conversão de Atributo de Referência). *Dados um elemento complexo* `E` *e um atributo* `at` *pertencente a* `E` *com tipo* `IDREF(S)` *ou* `[xml:link] href`*, tal que todas as referências de* `at` *indicam um elemento* `E'`*, gera-se:*

$$\begin{cases} <E.nome, E'.nome, (0,1), Card_I, at.nome> \in A, \\ \quad \textit{se } at \textit{ é do tipo } IDREF \textit{ ou } [xml:link]\ href; \\ <E.nome, E'.nome, (1,1), Card_I, at.nome> \in A, \\ \quad \textit{se } at \textit{ é do tipo } IDREF \textit{ ou } [xml:link]\ href \textit{ e seguido da cláusula } \#REQUIRED; \\ <E.nome, E'.nome, (0,N), Card_I, at.nome> \in A, \\ \quad \textit{se } at \textit{ é do tipo } IDREFS; \\ <E.nome, E'.nome, (1,N), Card_I, at.nome> \in A, \\ \quad \textit{se } at \textit{ é do tipo } IDREFS \textit{ e seguido da cláusula } \#REQUIRED. \end{cases}$$

Exemplo:

`<!ATTLIST Student ... advisor IDREF>` $\Rightarrow$
<Student, Teacher, (0,1), (1,N), advisor> $\in A$ (DTD3)

Este exemplo assume que todos os estudantes fazem referência a um professor através do atributo orientador (`advisor`).

Definições de Elementos Componentes

A última categoria de regras, apresentadas a seguir, são as que lidam com *definições de elementos componentes.*

Regra ECN (Conversão de Elemento Componente Nomeado). *Dados um elemento composto* $E = (\ldots, e_i<c_i>, \ldots)$ *e um elemento componente nomeado* $<e_i>$ *com cardinalidade* $<c_i>$*, gera-se* $<E.nome, e_i.nome, Card_D(<c_i>), Card_I> \in A$*, onde* $Card_I$ *é determinado através da análise de documentos XML ou assumido (1,N) por default.*

ECN é a regra básica de determinação de relacionamentos entre conceitos, utilizada nos casos onde regras mais especializadas (descritas na seqüência) não se aplicam. Ela gera um relacionamento de associação entre os conceitos correspondentes aos elementos composto e componente.

Exemplos:

`<!ELEMENT Book (Writer+, ...)>` $\Rightarrow$ <Book, Writer, (1,N), (1,N)> $\in A$ (DTD1)
`<!ELEMENT Paper (Title, ...)>` $\Rightarrow$ <Paper, Title, (1,1), (1,1)> $\in A$ (DTD2)
`<!ELEMENT Address(Home?, ...)>` $\Rightarrow$ `<Address, Home, (0,1), (1,1)>` $\in$ A (DTD2)

Regra ECT (Conversão de Elemento Componente Texto). *Dados um elemento composto* `E = (..., #PCDATA <c>, ...)` *e um elemento componente texto com uma cardinalidade* `<c>` *gera-se:*

- *<*`E.nome+"Text"`*, string> ∈ L;*
- *<*`E.nome`*,* `E.nome+"Text"`*, $Card_D$(<c>), (1,1)> ∈ A.*

A regra **ECT** gera um conceito especial referente ao componente `#PCDATA`, pois este é um dado passível de consulta. Seu nome *default* é a concatenação do nome do elemento composto E seguido do nome *"Text"*. Além disso, um relacionamento de associação é definido entre eles. A cardinalidade inversa é definida como (1,1) pois assume-se que uma ocorrência de texto é particular apenas de E.

Exemplos:

`<!ELEMENT Forum (#PCDATA)>` ⇒ <ForumText1, string> ∈ L
<Forum, ForumText, (1,1), (1,1)> ∈ A (DTD2)

`<!ELEMENT Schedule ((#PCDATA)*, ...)>` ⇒
<ScheduleText, string> ∈ L
<Schedule, ScheduleText, (0,N), (1,1)> ∈ A (DTD2)

Regra ECR (Conversão de Elemento Componente com Repetição). *Dados um elemento composto* `E = (..., e<`c_1`>, ..., e<`c_n`>)` *e um elemento componente* `e` *que se repete n vezes tal que <*c_1*>, ..., <*c_n*> são as cardinalidades das ocorrências de* `e`*, gera-se:*

- *<e.nome, string> ∈ L,* *se e é um elemento componente texto, sendo e.nome = E.nome+"Text";*

- *<*`E.nome`*,* `e.nome`*, $Card_R$(<c_1>, ..., <c_n>), $Card_I$> ∈ A,*
  se todas as n ocorrências de `e` *apresentam a mesma intenção semântica;*

- *{r}, r ∈ A, |{r}| = i ≤ n, sendo r' ∈ {r} =*
  <`E.nome`*,* `e.nome`*, $Card_R$(<c_f>, ..., <c_k>), $Card_I$, nome'> ou*
  <`E.nome`*,* `e.nome`*, $Card_R$(<c_f>, ..., <c_k>), $Card_I$, $papel_D$, $papel_I$> ou*
  <`E.nome`*,* `e.nome`*, $Card_R$(<c_f>, ..., <c_k>), $Card_I$, nome', $papel_D$, $papel_I$>,*
  para as i interpretações semânticas das ocorrências de `e`*.*

A regra **ECR** é uma regra semi-automática que exige a intervenção do usuário para definir as intenções semânticas de cada ocorrência de um elemento componente. Para cada grupo de ocorrências com a mesma intenção semântica é definido um relacionamento nomeado ou com papéis ou ambos, a critério do usuário. Se todas as ocorrências têm o mesmo significado, apenas um relacionamento de associação não-nomeado é definido.

Exemplos:

`<!ELEMENT Writer (Name, EMail, EMail?)>` ⇒ <Writer, EMail, (1,2), (1,1)> ∈ A (DTD1)

`<!ELEMENT Author (Name, Sex, Address, Address?, ...)>` ⇒
`<Author, Address, (1,1), (1,N), WorkAddress>` ∈ A
`<Author, Address, (0,1), (1,N), HomeAddress>` ∈ A (DTD3)

No exemplo para a DTD1, assume-se que ambas as ocorrências de `EMail` são meramente endereços eletrônicos para correspondência, podendo um autor ter até dois destes endereços. Já no exemplo para a DTD3, assume-se que cada ocorrência de endereço tem um significado diferente: a primeira diz respeito ao endereço profissional do autor e a segunda, opcional, é o seu endereço residencial. Neste caso, dois relacionamentos nomeados são requeridos para ressaltar estas interpretações semânticas.

Regra ECA (Conversão de Elemento Componente Nomeado Auto-Relacionado). *Dados um elemento composto* `E` $= (\ldots,$ `E`$<c_E>, \ldots)$ *e um elemento componente nomeado auto-relacionado* `E` *com uma cardinalidade* $<c_E>$*, gera-se:*

$$\begin{cases} <\texttt{E.nome}, \texttt{E.nome}, Card_D(<c_E>), Card_I, nome> \in A, \\ \quad \textit{se a intenção semântica é expressa por um nome;} \\ <\texttt{E.nome}, \texttt{E.nome}, Card_D(<c_E>), Card_I, papel_D, papel_I> \in A, \\ \quad \textit{se a intenção semântica é expressa por papéis;} \\ <\texttt{E.nome}, \texttt{E.nome}, Card_D(<c_E>), Card_I, nome, papel_D, papel_I> \in A, \\ \quad \textit{se a intenção semântica é expressa um nome e por papéis.} \end{cases}$$

A regra **ECA** é também semi-automática e trata de auto-relacionamentos. Como na modelagem conceitual tradicional de banco de dados, auto-relacionamentos necessitam que a sua intenção semântica esteja explícita para permitir a sua compreensão. Esta intenção semântica pode ser descrita através de um nome, da descrição dos papéis de cada conceito participante ou da descrição de ambos (nome e papéis). A definição destas intenções semânticas requer a intervenção do usuário.

Exemplo:

`<!ELEMENT Paper (..., Paper*)>` $\Rightarrow$
<Paper, Paper, (0,N), (0,N), references, referenced-by> $\in A$ (DTD2)

Neste exemplo, papéis definem a intenção semântica do relacionamento, indicando que um artigo tanto pode estabelecer uma referência como ser referido por outros artigos.

Regra EAny (Conversão de Elemento ANY). *Dado um elemento complexo* `E` *do tipo ANY, gera-se:*

- $\{r\}$, $r \in A$, *sendo* $r' =$ <`E.nome`, `E'.nome`, (1,1), (0,1)> $\in \{r\}$ *tal que* `E'` $\in NL \cup L$ *e* `E'` *não corresponde a um atributo da DTD;*
- $<\{r\}> \in D$.

A regra **EAny** considera os relacionamentos que um elemento `ANY` deve ter. Como um elemento deste tipo pode conter qualquer elemento da DTD como componente, são definidos relacionamentos para todos os conceitos do esquema MCC, exceto conceitos derivados de atributos. Como apenas um elemento pode ser componente de um elemento `ANY`, todos os relacionamentos definidos devem ser disjuntos. A cardinalidade direta dos relacionamentos é (1,1) pois sempre se tem uma única ocorrência de elemento componente. A cardinalidade inversa é opcional pois nem

todo elemento da DTD é componente de um elemento `ANY`.

Exemplo:

`<!ELEMENT Best-Referenced ANY>` $\Rightarrow$

r_1 = <Best-Referenced, Book, (1,1), (0,1)> $\in A$

. . .

r_n = <Best-Referenced, Best-Referenced, (1,1), (0,1)> $\in A$

<{r_1, ..., r_n}> $\in D$ (DTD1)

Neste exemplo, o elemento `ANY` mantém o elemento mais referido (livro, escritor, ...) nas preferências do leitor.

Esta regra apresenta o inconveniente de gerar muitos relacionamentos no esquema MCC. Outra alternativa de solução seria a geração de um conceito genérico para todos os elementos do esquema, com o qual o elemento `ANY` se relacionaria. Porém, estar-se-ia gerando um conceito artificial e diversos relacionamentos de herança no esquema. Apesar de não haver uma solução elegante para este caso, tem-se o atenuante de que na prática elementos *ANY* são bastante raros em DTDs. Isto porque tais elementos não apresentam conteúdo preciso, o que limita muito a sua usabilidade.

Regras Heurísticas

As duas próximas regras se valem de *heurísticas* que assumem certas interpretações semânticas sobre uma DTD. Uma delas verifica a existência de um relacionamento de herança entre um elemento composto e um componente. A outra verifica a existência de elementos componentes que se comportam como qualificações de um elemento composto.

As heurísticas são definidas primeiramente, seguidas das regras que as utilizam.

Heurística de Herança. *Se um relação <t_1 hiperônimo[2] t_2> pode ser inferido através de um Thesaurus para dois termos t_1 e t_2, então t_1 é uma generalização de t_2.*

Heurística de Qualificação. *Se `E'` é um elemento componente de um elemento `E` e `E'` está definido como um elemento vazio sem atributos, então `E'` é uma qualificação de `E`.*

Regra Her (Herança). *Dados um elemento composto `E = (..., ei<ci>, ...)` e um elemento componente nomeado `ei` com cardinalide <`ci`>, se a **Heurística de Herança** vale para <`E`, `ei`> e <`ci`> $\notin$ {*, +}, então gera-se <`E.nome`, `ei.nome`> $\in H$. Se `ei` é um elemento simples, então:*

- *converte-se `ei` $\in L$ para <`ei.nome`> $\in NL$;*
- *gera-se:*
  - *<`ei.nome+"Text"`, tipoDado> $\in L$;*
  - *<`ei.nome, ei.nome+"Text"`, (1,1), (1,1)> $\in A$.*

A regra **Her** se baseia na heurística de herança para determinar um relacionamento deste tipo entre um elemento composto e um elemento componente. Tal heurística se vale da consulta a um *thesaurus* para determinar se uma relação do tipo

[2] t_1 Hiperônimo t_2 significa "t_1 é um termo mais geral que t_2".

hiperônimo existe entre eles. Nesta tese faz-se acesso ao *WordNet*, que é atualmente um dos mais completos thesaurus da língua inglesa disponível na *Web* [WOR 2002]. Apesar de BInXS estar correntemente limitado à língua inglesa, muitos documentos e esquemas XML na *Web* encontram-se descritos nesta língua.

Na definição da regra **Her**, caso o elemento componente e_i seja um *conceito léxico*, ele se torna um conceito não-léxico e o seu conteúdo passa a ser representado por um conceito léxico especial de nome `e`$_i$`.nome+"Text"` associado a ele. O tipo de dado *tipoDado* de `e`$_i$`.nome+"Text"` é o tipo de dado que estava definido para e_i. Esta conversão de e_i é necessária pois um conceito especializado c_e em um relacionamento de herança é sempre não-lexico. Isto ocorre porque c_e pode herdar relacionamentos do conceito genérico, o que o torna um conceito composto por outras informações, ou seja, um conceito não-léxico[3].

Exemplos:

`<!ELEMENT Publication (..., | Journal)>` ⇒ <Publication, Journal> ∈ *H* (DTD2)
`<!ELEMENT Publication (Book | ...)>` ⇒ <Publication, Book> ∈ *H* (DTD3)

Regra Q (Qualificação). *Dados um elemento composto* `E = (..., e`$_i$`<c`$_i$`>, ..., e`$_k$`<c`$_k$`>, ...)` *e um conjunto de elementos componentes nomeados* $C = \{e_i<c_i>, ..., e_k<c_k>\}$ *com cardinalidades* $<c_i>, ..., <c_k>$, *respectivamente, se* $\forall\ e \in C$ (***Heurística de Qualificação*** *vale para* $<E, e>$), *então gera-se:*

- $<$`E.nome+"Type"`, *string,* $\{e_i, ..., e_k\}> \in L$;
- $<$`E.nome, E.nome+"Type"`, c_d, (1,N)$> \in A$, *onde* $c_d = Card_R(C_{seq} \cup Card_U(C_{esc_1}) \cup ... \cup Card_U(C_{esc_n}))$, *sendo* C_{seq} *o conjunto de cardinalidades dos elementos* $e_k \in C$ *que fazem parte da seqüência que define* `E`, *caso exista, e* C_{esc_1}, ..., C_{esc_n} *os conjuntos de cardinalidades dos elementos* $e_k \in C$ *envolvidos em n escolhas,* $n \geq 0$, *definidas em* `E`.

A regra **Q** leva em conta a heurística de qualificação. A justificativa para esta heurística é que um componente sem conteúdo e_i atua como uma característica diretamente vinculada ao seu elemento composto E, ou seja, é uma qualificação sua. Isto porque apenas o seu nome é declarado no conteúdo de E.

Uma vez que estes componentes são considerados qualificações de E, a regra **Q** os considera como uma enumeração de valores permitidos para um conceito léxico enumerado especial associado a E. O nome *default* para este conceito é a concatenação do nome do elemento composto com "Type". A enumeração associada a este conceito envolve os nomes destes elementos componentes e seu tipo é *string*.

Exemplo:

```
<!ELEMENT Book (..., Technical | Fiction)>
<!ELEMENT Technical (EMPTY)>
<!ELEMENT Fiction (EMPTY)> ⇒
```
<BookType, string, {Technical, Fiction}> ∈ *L*
<Book, BookType, (1,1), (1,1)> ∈ *A* (DTD1)

A cardinalidade direta do relacionamento de associação (c_d) gerado por esta regra considera o número mínimo e máximo de ocorrências de elementos compo-

[3]Na verdade, esta restrição evita a representação de um relacionamento entre dois elementos `#PCDATA` pois c_e pode herdar um relacionamento com outro conceito léxico. Tal relacionamento não existe em uma DTD.

nentes vazios presentes na *seqüência* definida no modelo de conteúdo de E, caso exista ($Card_R$). Para cada *escolha* eventualmente definida no modelo de conteúdo de E, vale a cardinalidade mais ampla dentre todos os seus elementos ($Card_U$). A cardinalidade inversa é assumida como (1,N).

Dado, por exemplo, um elemento *E = (*e_1*, ..., *e_i*?, ..., *e_m* | *e_n**, ..., *e_x* | *e_y*, ...)* e $C = \{e_1, e_i, e_m, e_n, e_x, e_y\}$, tem-se $C_{seq} = \{(1,1), (0,1)\}$ para e_1 e e_i; $C_{esc_1} = \{(1,1), (0,N)\}$ para e_m e e_n; e $C_{esc_2} = \{(1,1), (1,1)\}$ para e_x e e_y. Logo, $c_d = Card_R((1,1), (0,1), Card_U((1,1), (0,N)), Card_U((1,1), (1,1))) = Card_R((1,1), (0,1), (0,N), (1,1)) = (2,N)$.

Regra de Definição de Disjunção

A última regra verifica a necessidade de se gerar uma disjunção de relacionamentos, caso uma escolha esteja definida no modelo de conteúdo de um elemento composto.

Regra Disj (Disjunção). *Dado um elemento composto E = (..., e_i | ... | e_n, ...), gera-se uma disjunção d = <{r_i, ..., r_n}>, sendo r_i, ..., r_n os relacionamentos entre os conceitos correspondentes à E e os elementos e_i, ..., e_n no esquema conceitual.*

Exemplos:

`<!ELEMENT Author (..., University | Research-Institute | Business | Office, ...)>` ⇒
r_1 = <Author, University, (1,N), (1,N)> ∈ A
r_3 = <Author, Research-Institute, (1,1), (1,N)> ∈ A
r_2 = <Author, Business, (1,1), (1,N)> ∈ A
r_4 = <Author, Office, (1,1), (1,N)> ∈ A
<{r_1, r_2, r_3, r_4}> ∈ D (DTD3)

`<!ELEMENT Publication (Book | Proceedings)>` ⇒
r_x = <Publication, Book> ∈ H
r_y = <Publication, Proceedings> ∈ H
<{r_x, r_y}> ∈ D (DTD3)

5.5.3 Algoritmo de Conversão

As regras apresentadas na seção anterior são aplicadas no âmbito de um *algoritmo de conversão* que é executado sobre uma DTD pré-processada. Este algoritmo trabalha em duas etapas, cada etapa realizando uma varredura sobre a DTD. As regras que tratam de definições de elementos e atributos são aplicadas inicialmente para gerar os conceitos do esquema MCC. As demais regras são aplicadas num segundo momento para gerar os relacionamentos entre conceitos e eventualmente alguns conceitos especiais. As etapas do algoritmo, em alto nível, são as seguintes:

1. *Geração de conceitos*:

```
Se definição de elemento então
  aplica EC ou ES;
  C_E ← C_E ∪ {nome do elemento}
Senão /* definição de atributo de um elemento E */
  aplica ATR ou AT, nesta ordem;
  aplica EC sobre E.
```

2. *Geração de relacionamentos e conceitos especiais*:

```
Se elemento ANY então
  aplica EAny baseado no conjunto C_E
Senão
  aplica as seguintes regras, nesta ordem:
      (Q ou Her ou ECR ou ECA ou ECT ou ECN) e Disj.
```

Na etapa 1, **ATR** é verificada antes de **AT** porque **ATR** analisa um caso específico de declaração de atributos. Se **ATR** não se aplica, a regra **AT** é utilizada. A regra **ES** é relaxada neste algoritmo da seguinte forma: ela sempre gera um conceito léxico para um elemento `E` cujo modelo de conteúdo seja apenas `(#PCDATA)`. Caso seja descoberto posteriormente que `E` tenha atributos, a regra **EC** é executada para garantir uma representação não-léxica para `E`. Todos os nomes de conceitos derivados de elementos são mantidos em um conjunto C_E.

Na etapa 2, o conjunto C_E, definido na etapa anterior, facilita a definição de relacionamentos para um conceito correspondente a um elemento `E` do tipo `ANY`, ou seja, são estabelecidos relacionamentos somente entre `E` e os conceitos correspondentes a elementos $e \in C_E$.

Se um elemento `E` não for do tipo `ANY`, as demais regras de conversão de componentes são verificadas. A ordem de verificação é relevante pois as primeiras regras analisam casos específicos enquanto as últimas realizam conversões gerais de componentes. Elas também são exclusivas: no momento em que uma regra se verifica para um componente, as subseqüentes não precisam ser verificadas para ele. A única exceção é a regra **Disj**, que pode gerar disjunções para relacionamentos definidos pelas regras antecedentes.

5.5.4 Geração de Informação de Mapeamento

Toda vez que um conceito ou relacionamento é criado pelas regras de conversão, informações de mapeamento para a DTD original devem também ser criadas. A criação de tais informações faz parte do algoritmo de conversão e não foi apresentada na seção anterior para simplificar a sua explicação.

O algoritmo de conversão, enquanto percorre a DTD pré-processada, constrói uma estrutura semelhante a um grafo hierárquico *DOM*, que descreve a organização dos seus elementos com seus respectivos atributos e valores [4]. Com base no grafo DOM, são definidas expressões de mapeamento *XPath*.

Informações de mapeamento são criadas durante a conversão de elementos, atributos e elementos componentes, como apresentado a seguir.

Conversão de Elementos

A *conversão de elementos* é feita pelas regras **EC** e **ES**. Toda vez que um conceito é criado por uma destas regras é definido também uma expressão de caminho absoluto *XPath* a partir do nodo raiz até o elemento em questão, considerando todos os elementos já visitados no grafo DOM. A definição desta expressão *XPath* deve levar em conta a consulta ao *catálogo temporário* de informações de mapeamento (ver seção 5.4.6) para a verificação dos seguintes casos:

[4] DOM (Document Object Model) é um conjunto de APIs para processamento de dados XML que é parte do padrão XML da W3C. O modelo DOM organiza dados XML em um grafo hierárquico. Mais detalhes são encontrados em [DOC 2002].

Caso 1 (Elemento renomeado). um elemento pode ter sido renomeado na DTD pré-processada. Um exemplo é o elemento `Author`, que é raiz na DTD3. Supondo um catálogo relacional, tal informação pode ser consultada através da tabela mostrada a seguir.

Fonte	*Nome*	*NomeOriginal*
DTD3	Author	MyCurriculumVitae
...	...	...

Assim, a expressão de mapeamento para o conceito `Author` é `/MyCurriculum Vitae`. Conceitos gerados a partir de elementos que são filhos de `Author` se baseiam na expressão de mapeamento definida para o pai. O mapeamento do conceito `Occupation` é um exemplo: `/MyCurriculumVitae/Occupation`.

Caso 2 (Elemento removido). o mapeamento para um conceito E deve considerar a existência de um mais elementos removidos no caminho entre o seu elemento correspondente e o seu elemento pai na DTD original. Para tanto, uma consulta ao catálogo temporário deve ser feita a partir da informação de um elemento filho chamado E (caso exista) até alcançar, recursivamente, o seu elemento pai presente na DTD pré-processada. Um exemplo é o elemento `Writer` na DTD1, como mostra a tabela a seguir.

Fonte	*ElementoRemovido*	*ElementoPai*	*ElementoFilho*
DTD1	MyFavouriteBooks	-	Book
DTD1	WriterList	Book	Writer
...	...	...	...

Como o elemento `Book` também teve um elemento pai removido (`MyFavourite Books`), o mapeamento para o conceito `Writer` é: `/MyFavouriteBooks/Book/ WriterList/Writer`.

Caso 3 (Elemento virtual). um conceito não-léxico gerado pode corresponder a um elemento virtual *E*. Uma consulta ao catálogo deve verificar esta possibilidade. Em caso positivo, E não apresenta correspondência com elementos da DTD original. Mesmo assim, E encapsula um ou mais elementos de uma estrutura aninhada de um elemento composto *E'*. Portanto, seu mapeamento é dado pelo caminho absoluto até *E'* seguido de caminhos alternativos para todos os elementos da estrutura aninhada. Um exemplo é o elemento `Job` na DTD3, como indicado na tabela a seguir.

Fonte	*ElementoVirtual*
DTD3	Job
...	...

`Job` é um elemento componente do elemento original `Researcher` e encapsula uma seqüência de dois elementos: `Project` e `Position`. Logo, seu mapeamento é:

`/MyCurriculumVitae/Occupation/Researcher/(Project | Position)`.

Sobre elementos virtuais vale ainda a seguinte observação: quando um elemento E tem um ou mais elementos virtuais na sua expressão de caminho absoluto na DTD pré-processada, seu mapeamento para a DTD original é esta expressão de caminho sem a indicação destes elementos virtuais. Isto se justifica justamente

pelo fato destes elementos não existirem na DTD original. Um exemplo também na DTD3 ocorre para o mapeamento do conceito `Position`: `/MyCurriculumVitae/Occupation/Researcher/Position`. O elemento `Job`, que é pai do elemento `Position` na DTD3 pré-processada, não é considerado na expressão de mapeamento de `Position` pois é um elemento virtual.

Independente da consulta ao catálogo temporário para a verificação destes três casos, é possível que um conceito tenha mais de um caminho absoluto na DTD que conduz ao seu elemento correspondente. Isto ocorre quando um elemento é componente de mais de um elemento composto. Assim, caso um conceito já tenha uma expressão *XPath* de mapeamento definida para ele e outro caminho absoluto é encontrado, este caminho torna-se uma (outra) alternativa que é anexada a esta expressão. Um exemplo é o elemento `Year` na DTD3, cujos elementos que fazem referência a ele são `Book` e `Proceedings`. O mapeamento para o conceito `Year` é definido como:

```
(/MyCurriculumVitae/Publication/Book |
 /MyCurriculumVitae/Publication/Proceedings)/Year
```

Uma fatoração de sub-caminhos comuns é aplicada após a aplicação das regras de conversão para mapeamentos com estas características. Para o exemplo anterior, tal fatoração resulta em:

```
/MyCurriculumVitae/Publication/(Book | Proceedings)/Year
```

Conversão de Atributos

O mapeamento para um conceito léxico correspondente a um atributo *At* de um elemento *E* é uma expressão de caminho absoluto *XPath* até o atributo *At*. O **caso 1** do mapeamento de elementos vale também para atributos, para verificar se *At* foi renomeado.

O mapeamento para o relacionamento *E-At* é `@At.nome` no sentido *E*→*At* e "-" (vazio) no sentido *E*←*At*. O mapeamento é vazio no sentido inverso porque não existe deslocamento na hierarquia do documento XML quando se navega de um atributo para o elemento ao qual ele pertence.

Um exemplo é mostrado para o atributo `Style` do elemento `Writer` na DTD1:

Mapeamento do conceito `Style`:
`/MyFavouriteBooks/Book/WriterList/Writer/@Style`

Mapeamento do relacionamento `Writer-Style`:
Writer→Style: `@Style`
Writer←Style: -

A regra **ATR**, que lida com a conversão de atributos de referência e identificadores, gera mapeamentos dinâmicos, conforme explicado na seção 4.2.2. Um exemplo ocorre na DTD3 para o atributo identificador `SSN` de `Teacher` e o atributo de referência `advisor` de `Student`. Um mapeamento para o relacionamento `Student-Teacher`, sem considerar aspectos dinâmicos relativos a predicados aplicados sobre estes dois elementos, é:

Student→Teacher: `@advisor⇒Teacher`
Student←Teacher: `/MyCurriculumVitae/Occupation/Student`
`[/MyCurriculumVitae/Occupation/Teacher/@SSN = @advisor]`

Conversão de Elementos Componentes

A *conversão de elementos componentes* gera relacionamentos entre os conceitos assim como conceitos especiais. Portanto, mapeamentos para estes relacionamentos e conceitos especiais devem ser definidos. Diversos casos são apresentados a seguir, considerando consultas ao catálogo temporário de informações de mapeamento quando necessário.

Caso 1 (Mapeamento para um Elemento Componente Nomeado). Este mapeamento se aplica ao relacionamento entre os conceitos `E` e `E'` que correspondem ao caso `E(..., E', ...)`. Este caso é tratado pela regra **ECN**, ou pela regra **ECA** quando `E = E'`. Tem-se:

$$E \rightarrow E' = \begin{cases} \texttt{-}, & \textit{se } E \textit{ é elemento virtual;} \\ \texttt{[}e_i\texttt{/ ... /}e_n\texttt{/](E'.nomeOriginal | E'.nome)}, & \textit{caso contrário.} \end{cases}$$

$$E \leftarrow E' = \begin{cases} \texttt{-}, & \text{se } E \text{ é elemento virtual;} \\ \texttt{[../ .../]}(n \text{ vezes}) \texttt{ ..}, & \text{caso contrário.} \end{cases}$$

O mapeamento do relacionamento direto $E \rightarrow E'$ é vazio se E corresponde a um elemento virtual. Caso contrário, tem-se uma expressão de caminho que pode incluir elementos excluídos e_1, ..., e_n no caminho entre os elementos E e E', seguido ou do nome original de E' (caso ele tenha sido renomeado) ou do nome do próprio elemento E'. O mapeamento do relacionamento inverso segue o mesmo princípio: é vazio ou considera n níveis hierárquicos superiores relativos a elementos removidos no caminho inverso entre E' e E.

Exemplos:

`Writer-Name:`	`Writer→Name:`	`Name` (DTD1)
	`Writer←Name:`	`..`
`Book-Writer:`	`Book→Writer:`	`WriterList/Writer` (DTD1)
	`Book←Writer:`	`../..`
`Paper-Paper:`	`Paper→Paper:`	`Paper` (DTD2)
	`Paper←Paper:`	`..`
`Address-Home:`	`Address→Home:`	- (DTD2)
	`Address←Home:`	-

Caso 2 (Mapeamento para um Elemento Componente Nomeado com Repetição). Este tipo de mapeamento se aplica ao relacionamento entre os conceitos `E` e `E'` que correspondem ao caso `E(..., E', ..., E')` quando mais de um relacionamento é gerado entre `E` e `E'` (tratado pela regra **ECR**). Tem-se, para cada i-ésima ocorrência $\texttt{E'}_i \in \texttt{E}$:

$$E \rightarrow E'_i = \begin{cases} \texttt{-}, & \textit{se } E \textit{ é elemento virtual;} \\ \texttt{[}e_1\texttt{/.../}e_n\texttt{/](E'}_i\texttt{.nomeOriginal | E'}_i\texttt{.nome)[position() = i]}, & \\ & \textit{caso contrário.} \end{cases}$$

O mapeamento do relacionamento inverso $E \leftarrow E'_i$ é análogo ao mapeamento do caso 1 para $E \leftarrow E'$. O predicado `[position()= i]` de *XPath* busca um elemento

componente que se encontra na posição i do modelo de conteúdo de um elemento composto.

Podem ocorrer diversas alternativas de posição no mapeamento $E \rightarrow E'$ se $E \rightarrow E'$ representa o relacionamento de E com diversas ocorrências de E' que possuem a mesma intenção semântica. O predicado *XPath* terá então a forma `[position() = i | ... | position() = n]`.

Exemplo:

`Author HomeAddress Address:`

`Author→Address: Address[position() = 2]` (DTD3)

`Author←Address: ..`

Caso 3 (Mapeamento para um Elemento Componente Texto). Este tipo de mapeamento se aplica a um conceito E que corresponde ao caso `E(..., #PCDATA, ...)`, quando é gerado um conceito especial `EText` correspondente ao `#PCDATA`. Se um `#PCDATA` se repete como elemento componente em uma posição i, tem-se um conceito especial $\texttt{EText}_i$. O mapeamento do conceito especial `EText` é o mapeamento do conceito `E` seguido da expressão `"/text()"`. Se `EText` repete no modelo de conteúdo de `E`, o mapeamento para cada relacionamento `E-EText[i]` correspondente ao elemento definido na posição `i` é dado por:

$$E \rightarrow EText[i] = \begin{cases} \texttt{-}, & \textit{se } E \textit{ é elemento virtual;} \\ \texttt{[e}_1\texttt{/ ... /e}_n\texttt{/]text()[[position() = i]]}, & \textit{caso contrário.} \end{cases}$$

$$E \leftarrow EText[i] = \begin{cases} \texttt{[../ ... /](}n\texttt{ vezes)..}, & \textit{se existem n elementos removidos entre } E \textit{ e } EText[_i]; \\ \texttt{-}, & \textit{caso contrário.} \end{cases}$$

Assim como no *caso 2*, várias alternativas de posição podem ser definidas.

Um exemplo deste caso é o conceito `EText`, criado para o componente `#PCDATA` do elemento `Forum` na DTD2:

Mapeamento do conceito:

`/ComputerScienceIncLiterature/Publication/Proceedings/Forum/text()`

Mapeamento do relacionamento `Forum-ForumText`:

`Forum` → `ForumText`: text()

`Forum` ← `ForumText`: -

Caso 4 (Mapeamento de Relacionamentos de Herança). Um relacionamento de herança `E` ⇐ `E'` entre dois conceitos apresenta mapeamento *vazio* para os relacionamentos direto e inverso entre seus elementos correspondentes. Por outro lado, são definidos mapeamentos para os relacionamentos entre `E'` e todos os conceitos relacionados a `E`. Isto porque `E'` **é-um** `E` e pelo princípio de herança todos os conceitos relacionados a `E` estão relacionados a `E'`. Para a definição destes mapeamentos considera-se a análise do grafo DOM e a verificação dos casos de elementos virtuais, renomeados e removidos que possam existir no caminho [5].

Exemplos:

`Author-Book:` (DTD3)

[5] Estas verificações ocorrem também para os casos 5 e 6, descritos na seqüência.

```
    Author → Book:  Publication/Book
    Author ← Book:  ../..
Journal-Title: (DTD2)
    Journal → Title:  ../Title
    Journal ← Title:  ../Journal
```

Caso 5 (Mapeamento de uma Qualificação). Este tipo de mapeamento se aplica ao caso tratado pela regra **Q** que gera um conceito especial `EType` correspondente aos elementos componentes $\mathtt{e}_1$, ..., $\mathtt{e}_n$ de um elemento composto `E`. O mapeamento do conceito especial `EType` é o mapeamento do conceito `E` seguido de caminhos alternativos para todos os elementos componentes, na forma "`/(`$\mathtt{e}_1$ | ... | $\mathtt{e}_n$`)`". Isto porque `EType` abstrai um conjunto de elementos vazios. O mapeamento do relacionamento `E-EType` é dado por:

`E` → `EType` = `(`$\mathtt{e}_1$ | ... | $\mathtt{e}_n$`)` (*se uma consulta CXPath contém* `E/EType`)
ou
`E` → `EType` = `[local_name(*) =/!= "x"]` (*se uma consulta CXPath contém* `E[EType =/≠ "x"]`)
`E` ← `EType: ..`

O mapeamento do relacionamento `E` → `EType` possui duas alternativas. Estas alternativas são necessárias em função do tipo de consulta *CXPath* que pode ser formulada sobre o conceito *EType*. Se a consulta estabelece um caminho de `E` para `EType`, isto significa consultar todos os elementos vazios abstraídos. Se a consulta define um predicado sobre o valor de `EType` a partir de `E`, este predicado está-se referindo na verdade ao nome dos elementos vazios. Para tanto, deve ser utilizada para o mapeamento local a função *XPath local_name()*, que recupera o nome de um elemento (conteúdo de uma *tag*) em um documento XML. O símbolo '*' indica algum elemento filho do elemento corrente.

Exemplo:

Mapeamento do conceito `BookType`: (DTD1)
`/MyFavouriteBooks/Book/(Technical | Fiction)`

Mapeamento do relacionamento `Book-BookType`:
`Book` → `BookType`: (Technical | Fiction) ou
[local_name(*) =/!= ("Technical"/"Fiction")]
`Book` ← `BookType`: ..

Caso 6 (Mapeamento para um Relacionamento com Elemento ANY). Este tipo de mapeamento se aplica para o caso tratado pela regra **EAny** que gera relacionamentos de um conceito *E* correspondente a um elemento ANY com todos os conceitos $\mathtt{e}_1$, ..., $\mathtt{e}_n$ derivados de elementos da DTD. O mapeamento do relacionamento de cada $\mathtt{e}_i \in \{\mathtt{e}_1, \ldots, \mathtt{e}_n\}$ com `E` é dado por:

`E` → $\mathtt{e}_i$: $\mathtt{e}_i$
`E` ← $\mathtt{e}_i$: ..

Exemplo:

Mapeamento do relacionamento `Best-Referenced - Book`:
`Best-Referenced` → `Book`: Book
`Best-Referenced` ← `Book`: ..

5.5.5 Análise de Documentos XML

Um diferencial de BInXS em relação a trabalhos relacionados é a análise tanto de informações do esquema como dos dados presentes nos documentos XML. A análise de documentos XML contribui para um aumento do grau de automação do processo de integração de BInXS, reduzindo a intervenção do usuário na tomada de decisões inerente a este processo.

A análise de documentos XML complementa a análise de esquemas XML pois certas informações necessárias à geração do esquema conceitual para a DTD só são obtidas através da investigação de dados XML. Porém, esta análise não é sempre precisa: é possível que após a investigação de um conjunto de documentos XML não seja possível concluir, por exemplo, a cardinalidade inversa de um relacionamento entre conceitos. Nestes casos, *defaults* são assumidos e o usuário pode posteriormente alterar estes *defaults*. Mesmo assim, esta análise pode reduzir a intervenção manual.

A análise de documentos XML é requisitada para tentar inferir três informações no esquema conceitual: a *cardinalidade inversa* de relacionamentos de associação entre conceitos, o *tipo de dado* de conceitos léxicos e um relacionamento de associação derivado da investigação de atributos *identificadores* e *de referência*. Estas análises são detalhadas a seguir. Assume-se que o conjunto de documentos XML a ser analisado (quantos e quais documentos) é definido previamente pelo usuário, levando em conta a sua perícia para selecionar uma amostra de documentos que talvez seja mais conclusiva para as análises.

Análise de Cardinalidades Inversas de Relacionamentos

Um relacionamento de associação entre dois conceitos C_1 e C_2 no esquema conceitual pode corresponder ao relacionamento entre dois construtores XML E_1 e E_2, sendo E_1 um elemento composto e E_2 um elemento componente ou atributo. Nestes casos, uma análise dos documentos XML deve ser feita para verificar se uma *mesma* instância de E_2 é componente de mais de uma instância *diferente* de E_1. Em caso positivo, uma cardinalidade inversa (1,N) é assumida de forma definitiva. Em caso negativo, um *default* de mesmo valor é assumido ((1,N)), porém o usuário deve validá-lo posteriormente.

Instâncias XML, sejam elas ocorrências de elementos ou atributos, não possuem identificadores únicos predefinidos. Assim, assume-se que duas instâncias XML E_i e E_j são iguais se:

$$\begin{cases} \textit{possuem o mesmo valor}, & \textit{se } E_i, E_j \textit{ correspondem a um mesmo conceito } C \in L; \\ \textit{possuem a mesma organização}, & \textit{se } E_i, E_j \textit{ correspondem a um mesmo conceito } C \in NL. \end{cases}$$

No segundo caso, possuir a mesma *organização* significa que:

- todo sub-elemento ou atributo de E_i possui uma correspondência *um-para-um* com um sub-elemento ou atributo de E_j de mesmo nome e vice-versa;
- cada par correspondente de sub-elementos ou atributos $e_i \in E_i$, $e_j \in E_j$ deve,

por sua vez:

$$\begin{cases} \textit{possuir o mesmo valor}, & \textit{se } e_i,\ e_j \textit{ correspondem a um mesmo conceito } C' \in L; \\ \textit{possuir a mesma organização}, & \textit{se } e_i,\ e_j \textit{ correspondem a um mesmo conceito } C' \in NL. \end{cases}$$

No exemplo a seguir, tem-se `E1 (a)` = `E1 (b)` apesar da ordem de declaração dos seus sub-elementos ser diferente. Existe uma correspondência um-para-um entre todos os seus componentes (`at1`, `at2`, `X`, `Y`, `Y.A`, `Y.B`, `Z`) e iguais valores ou organizações para todos estes pares correspondentes.

```
<E1 at1 = "a" at2 = "b">
    <X>1</X>
    <Y>
        <A>2</A>
        <B>
            <C>3</C>
            <D>4</D>
        </B>
    </Y>
    <Z>5</Z>
</E1>
```

(a)

```
<E1 at2 = "b" at1 = "a">
     <Y>
          <B>
               <D>4</D>
               <C>3</C>
          </B>
          <A>2</A>
     </Y>
     <X>1</X>
     <Z>5</Z>
</E1>
```

(b)

```
<E1 at1 = "a" at2 = "b">
      <X>1</X>
      <Y>
          <A>7</A>
          <B>
              <D>9</D>
          </B>
      </Y>
      <W>12</W>
      <X>10</X>
</E1>
```

(c)

Tem-se ainda `E1 (a)` $\neq$ `E1 (c)` pois não existem correspondências um-para-um para todos os sub-elementos nem valores iguais para todos os sub-elementos correspondentes.

A análise da cardinalidade inversa de um tipo de elemento E_2 que é componente de um elemento E_1 em um documento XML ocorre da seguinte maneira:

1. Fixa-se uma primeira instância de E_2 (instância *i'*);
2. Busca-se uma outra instância de E_2 (instância *i"*) tal que *i'* = *i"* e *InstânciaPai(i')* $\neq$ *InstânciaPai(i")*;
   (a) Se encontrou *i"*, então a cardinalidade inversa definitiva é (1,N);
   (b) Se não encontrou *i"*, fixa-se uma próxima instância de E_2 como *i'* e repete o passo 2, se o final do documento não foi alcançado;
3. Se a cardinalidade inversa está imprecisa ao final da análise do documento, assume-se a cardinalidade inversa *default* (1,N) e notifica-se o usuário.

O mesmo raciocínio se aplica a um atributo E_2 de um elemento E_1. Apesar deste algoritmo de análise mencionar a investigação de apenas um documento XML, é possível que vários documentos XML possam ser investigados caso haja mais de um documento de entrada.

Identificação de Instâncias de Grupos

Um problema que ocorre na análise de cardinalidade inversa diz respeito a *identificação de instâncias de elementos virtuais* em um documento XML. Elementos virtuais são determinados na fase de pré-processamento. Logo, não estão definidos

na DTD original e os documentos XML baseados nesta DTD não possuem *tags* delimitadoras para eles. Para tanto, é necessário um procedimento que detecte instâncias de elementos virtuais (chamadas *instâncias de grupos* de sub-elementos) e os delimite temporariamente no documento XML, para que possam ser analisadas também as suas cardinalidades inversas.

Um *algoritmo de identificação de uma instância de grupo* é executado para cada elemento E_i da DTD original que possua uma estrutura aninhada (grupo). Este algoritmo executa da seguinte forma:

Para cada instância de E_i no documento XML:

1. *Pilha* de grupos sendo delimitados (*PG*) é definida como vazia;
2. Enquanto o final do conteúdo de E_i não for encontrado faça:
   (a) Analisa um elemento componente e_x:
      i. Se e_x é componente de um grupo que está associado a um elemento virtual na DTD pré-processada e este grupo não está na *PG*, então:
         - cria uma *tag* delimitadora *inicial* temporária em E_i, antes da ocorrência de e_x, para este grupo;
         - empilha este grupo na *PG*;
      ii. Senão, se e_x não é componente do grupo no topo da *PG*, então, enquanto e_x não for componente do grupo no topo da *PG* ou a *PG* não for vazia faça:
         - desempilha o grupo no topo da PG;
         - insere uma tag delimitadora final temporária para o grupo em E_i
3. Enquanto a *PG* não for vazia faça:
   - desempilha o grupo no topo da *PG*;
   - insere uma tag delimitadora final temporária para o grupo em E_i.

A definição de uma *tag* delimitadora temporária considera o nome do elemento E_i e uma seqüência numérica que leva em conta aninhamentos recursivos que possam ocorrer nos grupos. A figura 5.3 exemplifica esta definição, para um elemento A de uma DTD original.

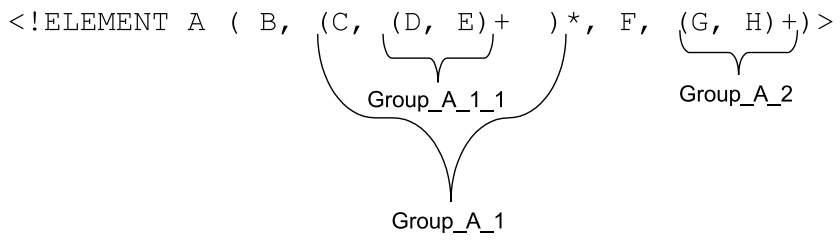


FIGURA 5.3 – Exemplo de definição de um grupo de elementos aninhados

Grupos diretamente associados ao elemento A são nomeados através da palavra *"Group"* seguido do nome do elemento (A) e de um número que corresponde a sua *posição* no modelo de conteúdo de A. Se um grupo, por sua vez, possui grupos aninhados, anexa-se mais um nível de numeração correspondente à posição interna dentro do grupo.

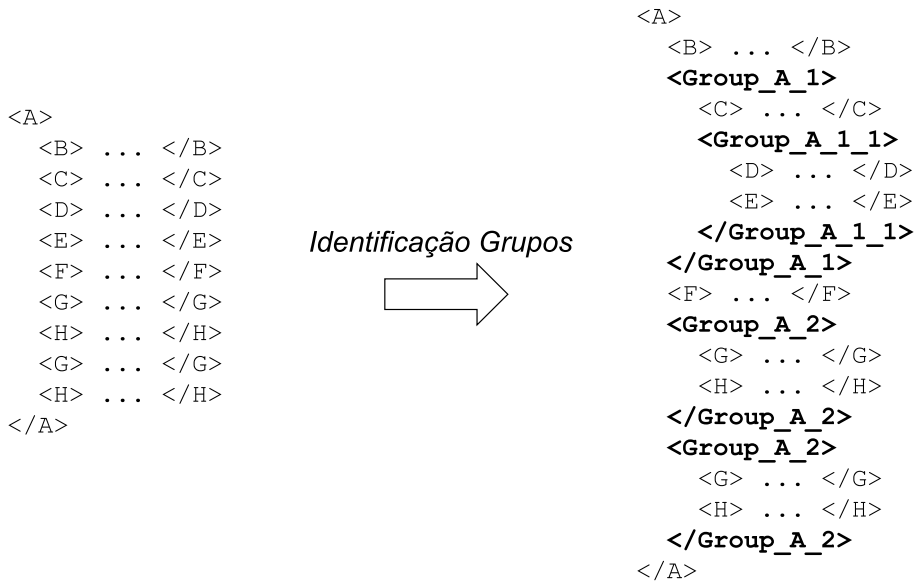


FIGURA 5.4 – Exemplo de aplicação do procedimento de identificação de grupos

Um exemplo de aplicação do algoritmo de identificação de grupos é mostrado na figura 5.4 para uma instância do elemento `A`. Esta instância está de acordo com o esquema do elemento `A` da figura 5.3.

A ocorrência do sub-elemento `C` indica a existência de um primeiro grupo para `A`, sendo definido como `Group_A_1`. O sub-elemento `D`, encontrado a seguir, identifica um sub-grupo do grupo `Group_A_1`, sendo definido como `Group_A_1_1`. Este sub-grupo termina após a ocorrência do sub-elemento `E`. As ocorrências posteriores dos sub-elementos *G* e *H* identificam ocorrências do grupo `Group_A_2`, delimitados como mostra a figura 5.4.

O algoritmo de identificação de grupos trata ainda especificações ambíguas de elementos, como por exemplo, a definição `<!ELEMENT A (B | C* | (C, D)+ | E)+>`. Para estes casos, a determinação de grupos leva em conta uma *regra* que tenta associar um sub-elemento ambíguo (`C`, para o exemplo) com a primeira definição válida no modelo de conteúdo do elemento. Supondo, por exemplo, uma instância:

`<A> ... <C> ... </C> <C> ... </C> <C> ... </C> <D> ... </D> ... </A>`,

a aplicação desta regra faz com que as duas primeiras ocorrências de `C` casem com a definição `C*` e a última ocorrência de `C` seja considerada membro do grupo `(C, D)+`.

Análise de Tipos de Dados de Conceitos Léxicos

Os conteúdos de elementos ou atributos correspondentes a conceitos léxicos devem também ser analisados na tentativa de se inferir um dentre os seguintes tipos de dados suportados pelo MCC: *string*, *integer*, *float*, *character*, *date*, *time* ou *datetime*.

Dado um certo elemento ou atributo *X*, o princípio desta análise é simples: determina-se o tipo de dado *td* da primeira instância de *X* no documento XML. Caso todas as demais instâncias de *X* possuam também este tipo, *td* é assumido como o tipo de dado de *X*. Caso algumas instâncias sejam do tipo *td* e outras de

um tipo mais genérico *td'*, *td'* é assumido. Caso contrário, *string* é o tipo de dado de *X*. Considera-se o tipo *string* um tipo mais genérico que *character*; *float* um tipo mais genérico que *integer*; e *datetime* um tipo mais genérico que *date* e *time*.

Um exemplo é mostrado para os elementos *X1* e *X2* considerando um documento XML analisado:

```
...
<X1> 24 </X1>
...
<X2> 538 </X2>
...
<X1> 3.5 </X1>
...
<X1> 65.43 </X1>
...
<X2> 1A2 </X2>
...
```

O tipo de dado definido para o conceito `X1` é *float* e o tipo de dado definido para o conceito `X2` é *string*. Quando o tipo determinado para uma instância for *string*, a análise termina e este tipo é assumido.

Análise de Atributos Identificadores e de Referência

A intenção de um atributo a_i de um elemento *E1* com tipo `IDREF(S)` ou `[xml:link] href` pode ser a referência para um atributo a_j de um elemento *E2* com tipo `ID`. Esta intenção deve ser analisada a fim de se determinar ou não um relacionamento de associação entre *E1* e *E2* com nome a_i.

O princípio desta análise é semelhante ao da análise de tipos de dados de conceitos léxicos: verifica-se o tipo de elemento *E2* referido pela primeira ocorrência do atributo a_i de *E1*. Caso todas as ocorrências de a_i façam referência a *E2* no conjunto de documentos XML analisados, estabelece-se um relacionamento de associação entre *E1* e *E2*.

Considerando o fragmento de documento XML a seguir:

```
...
<E2 aj = "1"> ... </E2>
...
<E1 ai = "1"> ... </E1>
...
<E2 aj = "2"> ... </E2>
...
<E1 ai = "1 2"> ... </E1>
...
<E1 ai = "2"> ... </E1>
...
```

define-se um relacionamento de associação *E1* $\stackrel{a_i}{-}$ *E2*. Vale salientar que a análise da cardinalidade inversa deste relacionamento procede de forma contrária a esta análise, ou seja, verifica-se se um tipo de elemento *E2* é referido por mais de uma ocorrência de um tipo de elemento *E1*.

Uma vez aplicado o passo de *Conversão*, obtém-se um *esquema conceitual preliminar* para a DTD em questão. A figura 5.5 mostra os esquemas conceituais preliminares para as DTDs exemplo.

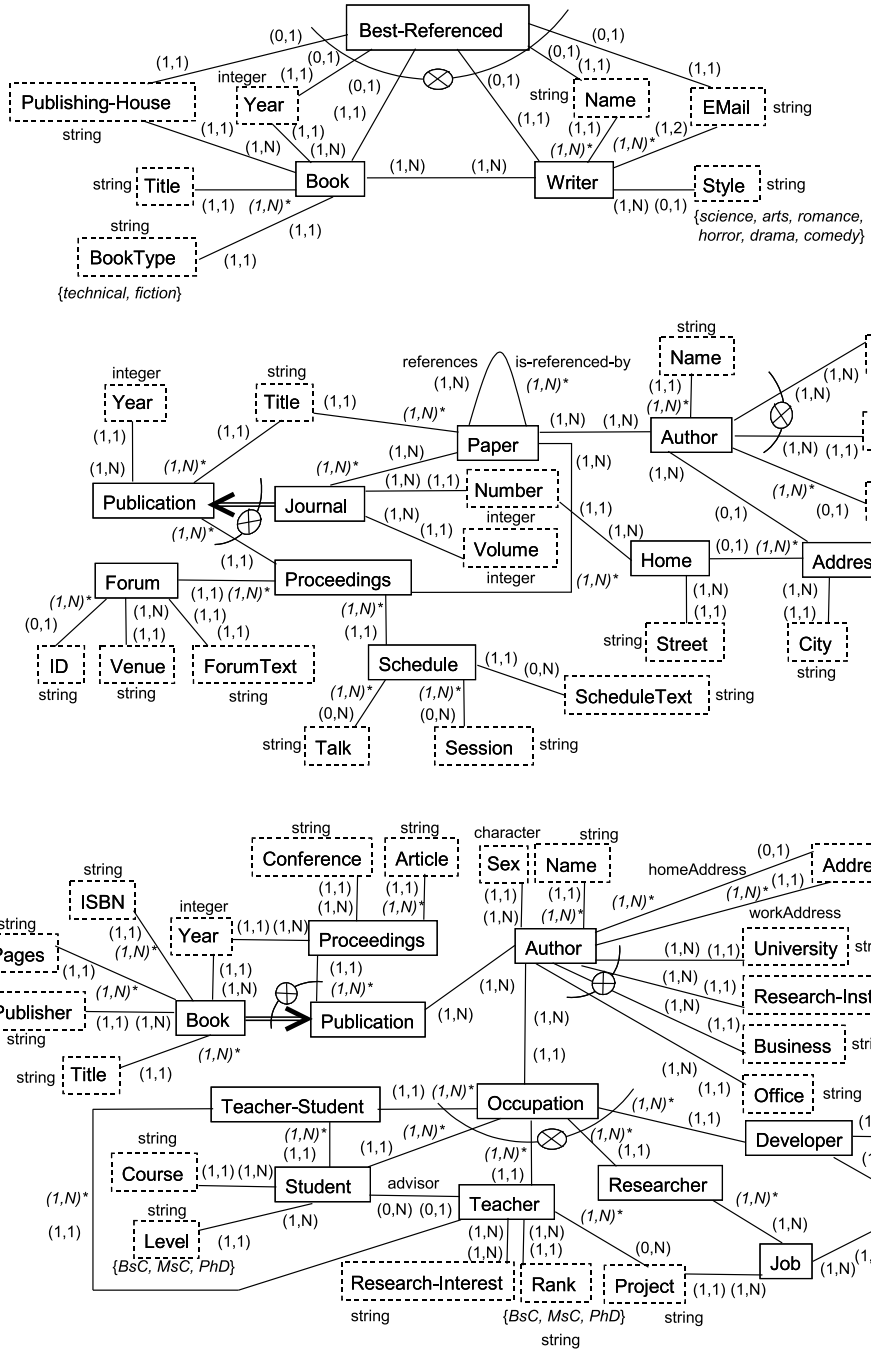


FIGURA 5.5 – Esquemas conceituais preliminares para as DTDs exemplo

5.6 Passo 3: Reestruturação

O último passo da etapa de *Conversão da DTD* é o passo de *Reestruturação*. Este passo realiza algumas atualizações automáticas e manuais no esquema conceitual preliminar a fim de produzir um esquema semanticamente correto e simplificado. Para tornar-se semanticamete correto, uma *validação manual* por parte do usuário especialista é feita a fim de corrigir certos *defaults* produzidos pelas regras de conversão ou pelos procedimentos de análise de documentos XML. A *simplificação do esquema* ocorre por conta de duas tarefas automáticas que reduzem o número de relacionamentos: generalização e remoção de redundâncias. Ambas as tarefas estão relacionadas com a existência de relacionamentos de herança no esquema conceitual. Estas atualizações são explicadas a seguir.

5.6.1 Validação do Usuário

Um esquema conceitual preliminar deve ser verificado pelo usuário especialista antes de se tornar um esquema definitivo. Isto porque certos *defaults* e inferências automáticas podem não ser condizentes com o domínio em questão. Os elementos do esquema conceitual que exigem validação são os seguintes:

- *Nomes de conceitos*: nomes *defaults* assumidos para conceitos especiais ou derivados de elementos virtuais são artificiais e devem ser atualizados pelo usuário. Exemplos de atualizações:
  - *BookType* $\Rightarrow$ *Type* (Esquema Preliminar 1);
  - *ForumText* $\Rightarrow$ *Name* (Esquema Preliminar 2);
  - *ScheduleText* $\Rightarrow$ *ScheduleDetails* (Esquema Preliminar 2);
- *Cardinalidades inversas*: cardinalidades inversas *defaults* devem ser validadas pelo usuário. Exemplos de validações para cardinalidade máxima 1 são as seguintes:
  - *EMail* $\xrightarrow{(1,N)}$ *Writer* $\Rightarrow$ (1,1) (Esquema Preliminar 1);
  - *Paper* $\xrightarrow{(0,N)}$ *Journal* $\Rightarrow$ (0,1) (Esquema Preliminar 2);
  - *Forum* $\xrightarrow{(1,N)}$ *Proceedings* $\Rightarrow$ (1,1) (Esquema Preliminar 2);
  - *Schedule* $\xrightarrow{(1,N)}$ *Proceedings* $\Rightarrow$ (1,1) (Esquema Preliminar 2);
  - *Article* $\xrightarrow{(1,N)}$ *Proceedings* $\Rightarrow$ (1,1) (Esquema Preliminar 3);
  - *Teacher* $\xrightarrow{(0,N)}$ *Teacher-Student* $\Rightarrow$ (0,1) (Esquema Preliminar 3);
- *Tipos de dados*: a atualização de tipos de dados pelo usuário ocorre quando a análise de documentos XML assume um tipo muito específico para um conceito, como por exemplo, uma informação do tipo inteiro que pode assumir valores reais. Nenhum exemplo deste tipo de atualização ocorre para os esquemas conceituais preliminares;
- *Tipos de relacionamentos*: um relacionamento de herança entre conceitos pode não ter sido detectado no passo de *Conversão* através da consulta ao Thesaurus, sendo definido como um relacionamento de associação. Da mesma forma,

um relacionamento de herança assumido para dois conceitos pode não corresponder à semântica desejada para uma dada DTD. Nestes casos, o usuário pode alterar o tipo de relacionamento. Um exemplo é o relacionamento entre os conceitos *Publication* e *Proceedings* nos esquemas preliminares 2 e 3. A consulta ao Thesaurus não inferiu uma relação do tipo hiperônimo entre eles. Porém, no contexto das DTDs correspondentes, *Publication* é um termo mais geral que *Proceedings*. Desta forma, os relacionamentos de associação entre eles nestes esquemas são substituidos por relacionamentos de herança. Da mesma forma, os relacionamentos disjuntos de *Occupation* com *Student*, *Teacher*, *Researcher*, *Developer* e *Teacher-Student* no esquema 3 denotam especializações de uma ocupação. *Teacher-Student* é ainda uma especialização de *Student* e *Teacher*.

As atualizações realizadas pelo usuário refletem-se nas definições de conceitos (nomes e tipos de dados) e relacionamentos (cardinalidades e tipos) de um esquema preliminar. A modificação do tipo de um relacionamento R afeta ainda as informações de mapeamento para R. Uma transformação *herança* $\rightarrow$ *associação* faz com que sejam geradas informações de mapeamento para os dois sentidos de R. Uma transformação *associação* $\rightarrow$ *herança* remove a informação de mapeamento de R e cria informações de mapeamento para os relacionamentos entre os conceitos associados ao conceito genérico e o conceito especializado.

Generalização de Conceitos

As validações recém-citadas são de caráter obrigatório, ou seja, o usuário deve considerá-las para garantir a corretude do esquema conceitual definitivo. Além dessas, uma validação opcional que o usuário pode realizar diz respeito a *generalização de conceitos*. Suponha, por exemplo, o esquema preliminar mostrado à esquerda da figura 5.6.

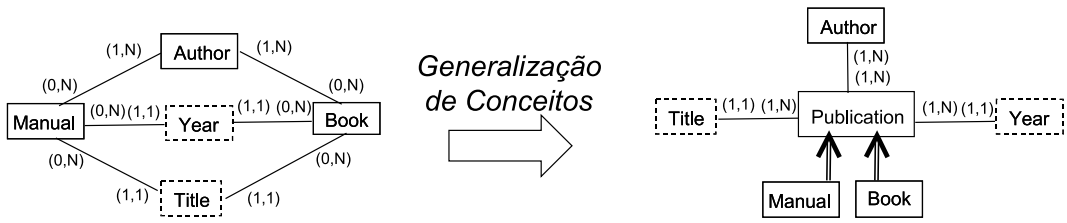


FIGURA 5.6 – Exemplo de generalização de conceito

Ambos os conceitos *Manual* e *Book* apresentam relacionamentos de mesmo tipo com os conceitos *Author*, *Title* e *Year*. Nesta situação, é possível definir um conceito *Publication* que generaliza estes relacionamentos comuns, reduzindo o número de relacionamentos do esquema. Esta generalização resulta no esquema modificado mostrado à direita da figura 5.6.

Uma generalização de conceitos só é permitida se os relacionamentos em comum são do mesmo tipo e apresentam as mesmas restrições de cardinalidade, papéis e nomes, no caso de relacionamentos de associação. Como o conceito genérico não está definido na DTD, seu mapeamento é a união de caminhos até os seus conceitos especializados, seguindo a estratégia de mapeamento definida na seção 4.2. O mapeamento para *Publication* seria `(.../Manual | .../Book)`.

A possibilidade de generalização de conceitos é deixada a cargo do usuário pois nem sempre uma generalização é significativa do ponto de vista conceitual. Um contra-exemplo seria a existência de conceitos *Person* e *Company* relacionados a conceitos *Name* e *Address*. Encontrar um conceito genérico para *Person* e *Company* pode não ser relevante.

5.6.2 Simplificação do Esquema

A *simplificação do esquema* tem por objetivo analisar relacionamentos de herança para reduzir o número de relacionamentos no esquema preliminar. Um conjunto de regras automáticas é responsável pelas tarefas de simplificação. Algumas definições de regras são descritas de maneira mais informal devido a sua complexidade.

Generalização de Relacionamentos

A tarefa de *generalização de relacionamentos* verifica se todos os conceitos especializados de um dado conceito genérico G apresentam relacionamentos de associação com um mesmo conceito X. Em caso positivo e dadas certas restrições, este relacionamento com X é generalizado, tornando-se um relacionamento exclusivo de G na hierarquia de herança. Esta tarefa é regida pela regra a seguir.

Regra GR (Generalização de Relacionamento). *Dados um conceito $G \in NL$ e um conjunto $C_E = \{E_1, ..., E_n\}$, $|C_E| > 1$, de conceitos especializados de G, tal que $\forall\ E_i \in C_E (\exists\ r_i \in A$ entre E_i e um conceito $X \in NL \cup L)$. Se todo r_i entre $E_i \in C_E$ e X possui os mesmos valores c_D e c_I de cardinalidade direta e inversa, o mesmo nome N para o relacionamento, se existir, e os mesmos nomes p_1 e p_2 de papéis direto e inverso, se existirem, então:*

1. *Gera-se $r' = <G, X, c_D, c_I, [N], [p_1], [p_2]> \in A$, com*

$$\begin{cases} C_I.mínima \leftarrow 0 & \text{se } X \text{ relaciona-se com um conceito } E_y \notin C_E, \\ C_I.mínima \leftarrow 1 & \text{caso contrário.} \end{cases}$$

2. *Remove-se r_i ($\forall\ r_i$ que associa $E_i \in C_E$ e X).*

Um exemplo de aplicação da regra **GR** ocorre no esquema preliminar 2, conforme mostra a figura 5.7.

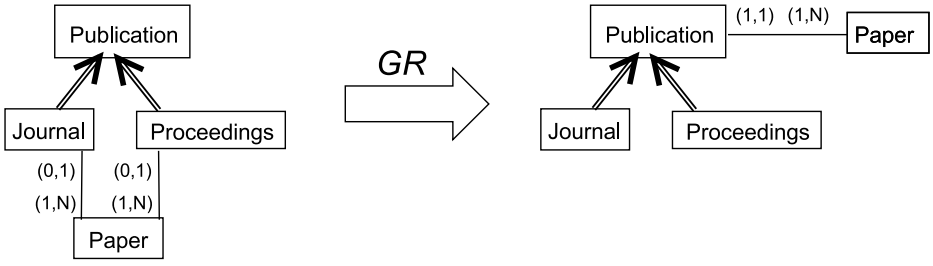


FIGURA 5.7 – Exemplo de generalização de relacionamento

O conceito `Paper` aparece relacionado a todas as especializações de `Publication`, tendo ambos os relacionamentos idênticos. Este relacionamento é então generalizado e os relacionamentos com `Proceedings` e `Journal` removidos.

De forma análoga, o conceito *Year* no esquema preliminar 3 passa a estar relacionado com *Publication* ao invés de *Book* e *Proceedings*.

A regra de generalização prevê ainda que o conceito destino X comum aos relacionamentos dos conceitos especializados E_1, ..., E_n pode corresponder na DTD a um elemento que é filho apenas dos elementos E_1, ..., E_n ou a um atributo definido apenas para E_1, ..., E_n. Se tal situação ocorre, X passa a ter uma associação *obrigatória* com o conceito G pois G generaliza todos os conceitos com os quais X está associado.

O mapeamento para um novo relacionamento generalizado é um caminho resultante da união dos caminhos referentes aos relacionamentos especializados até o conceito alvo. Para o exemplo da figura 5.7, os mapeamentos especializados para `Proceedings` e `Journal` são iguais a `Paper`, pois `Paper` é um elemento filho imediato de ambos. Logo, o mapeamento genérico deste relacionamento torna-se `(Proceedings | Journal)/Paper`. Os mapeamentos para os relacionamentos especializados são removidos.

Remoção de Relacionamentos Redundantes

A tarefa de *remoção de relacionamentos* verifica se relacionamentos de associação ou herança podem ser removidos do esquema conceitual sem alterar a sua semântica. Duas regras são responsáveis por esta tarefa, uma para cada tipo de relacionamento.

A *remoção de relacionamentos de associação* verifica se um relacionamento de associação idêntico ocorre tanto para um conceito genérico G como para um conceito especializado E. Por idêntico entende-se um relacionamento com as mesmas restrições de cardinalidade, além de nomes e papéis também idênticos, caso existam. Em caso positivo, o relacionamento é removido do conceito E, uma vez que ele o herda de G. A regra a seguir define formalmente esta tarefa.

Regra RRA (Remoção de Relacionamento de Associação). *Dados um conceito $G \in NL$ e um conceito $E \in NL \cup L$ especializado de G, tal que $\exists\ r_1 \in A$ entre G e um conceito $X \in NL \cup L$ e $\exists\ r_2 \in A$ entre E e um conceito X. Se r_1 e r_2 possuem os mesmos valores para cardinalidade direta e inversa, os mesmos nomes de relacionamentos, se existirem, e os mesmos nomes de papéis direto e inverso, se existirem, então remove-se r_2.*

Um exemplo de aplicação desta regra é mostrado na figura 5.8, supondo um esquema preliminar no qual um conceito genérico `Publication` e um conceito especializado `Book` possuem um relacionamento de associação com um conceito `Paper`. Neste caso, o relacionamento `Book-Paper` é removido uma vez que `Book` herda o relacionamento com `Paper` através de `Publication`.

O mapeamento do relacionamento redundante também é removido.

A tarefa de *remoção de relacionamentos de herança* verifica se um conceito especializado c_E apresenta um relacionamento de herança com um conceito genérico c_{G1} e um relacionamento de herança com um conceito genérico c_{G2}, sendo que c_{G2} é também especialização de c_{G1}. Neste caso, o relacionamento de herança entre c_{G1}

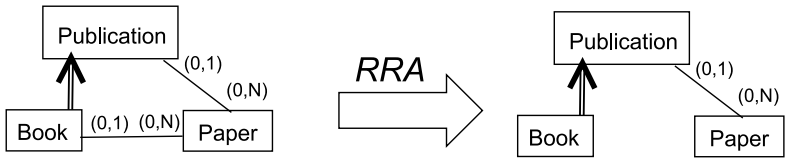


FIGURA 5.8 – Exemplo de remoção de um relacionamento de associação redundante

e c_E é redundante e deve ser removido. A regra a seguir define formalmente esta tarefa.

Regra RRH (Remoção de Relacionamento de Herança). *Dados os conceitos E, G_1, $G_2 \in NL$, se $\exists\ r_1 = <G_1.n,\ E.n> \in H$, $r_2 = <G_2.n,\ E.n> \in H$ e uma hierarquia de herança $<G_1.n,\ E_a.n> \in H$, $<E_a.n,\ E_b.n> \in H$, ..., $<E_n.n,\ G_2.n> \in H$ que defina G_2 como uma especialização de G_1, então remover r_1.*

Um exemplo de aplicação desta regra é mostrado na figura 5.9. No esquema preliminar 3, temos os seguintes relacionamentos de herança: `<Occupation, Teacher>`, `<Occupation, Teacher-Student>` e `<Teacher, Teacher-Student>`. O relacionamento `<Occupation, Teacher-Student>` é redundante pois o conceito *Teacher-Student* já é especialização de *Occupation* sendo especialização de *Teacher*. Este relacionamento é então removido do esquema.

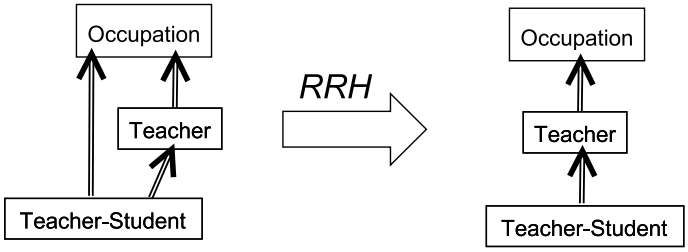


FIGURA 5.9 – Exemplo de remoção de um relacionamento de herança redundante

Os mapeamentos definidos para o conceito *Teacher-Student*, relativos ao seu relacionamento de herança com o conceito *Occupation*, mantêm-se válidos pois *Teacher-Student* continua ainda herdando relacionamentos de *Occupation*.

5.6.3 Relacionamentos Opcionais para Conceitos

Além das tarefas de simplificação do esquema, outra tarefa automática é a *definição de relacionamentos de associação opcionais* para conceitos que correspondem a elementos que são sub-elementos de mais de um elemento. Um exemplo da aplicação desta tarefa é mostrado na figura 5.10, para o esquema preliminar 2.

O conceito *Number* está relacionado com o conceito *Journal* (uma edição de periódico possui um número) e com o conceito *Home* (um número é parte de um endereço). Isto ocorre porque o elemento correspondente *number* é sub-elemento tanto do elemento *journal* como do elemento *home* na DTD2 pré-processada. Porém, uma instância de *number* em um documento XML ou é sub-elemento de *journal* ou

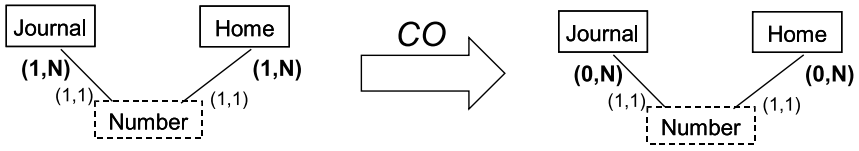


FIGURA 5.10 – Exemplo de definição de relacionamento opcional

é sub-elemento de *home*. Logo, as cardinalidades inversas dos relacionamentos de associação para o conceito *Number* deve ser opcional para refletir esta restrição.

Esta tarefa é regida pela regra **CO**, apresentada a seguir. Ela verifica a situação exemplificada anteriormente e define a cardinalidade mínima dos relacionamentos do conceito como sendo zero.

Regra CO (Cardinalidade Opcional). *Dado um conceito* $C \in NL \cup L$, *se* $\exists\ r_1,\ r_2 \in R\ (r_1.c_2 = r_2.c_2 = C \wedge r_1.c_1 \neq r_2.c_1 \wedge r_1.c_1,\ r_2.c_1 \neq C \wedge r_1.c_1,\ r_2.c_1 \neg \Leftrightarrow E$ *(E é elemento complexo do tipo* `ANY`*)), então* $\forall\ r \in A\ (r.c_2 = C \wedge r.c_1 \neq C) \Rightarrow r.c_i.mímima \leftarrow 0$.

A regra **CO** analisa se existe mais de um relacionamento (associação ou herança) no qual um conceito C corresponde a um elemento componente na DTD (referido pelo atributo c_2). Em caso positivo, a cardinalidade mínima de todos os relacionamentos de associação nos quais C é elemento componente são definidas como opcionais. Não são considerados relacionamentos nos quais C é componente de si próprio (atributo $c_1 = C.n$) pois tais relacionamentos não indicam se C possui mais de um pai. Também não são considerados relacionamentos nos quais C corresponde a um elemento componente de um elemento E do tipo `ANY` pois um componente de E nunca é exclusivo dele em uma DTD.

5.6.4 Resultados Obtidos

Uma vez reorganizado manual e automaticamente, um esquema conceitual preliminar se torna um esquema definitivo para a DTD em questão. A validação manual é realizada antes porque o usuário pode definir novos relacionamentos de herança que serão posteriormente testados pelas regras automáticas de simplificação.

Os esquemas definitivos correspondentes aos esquemas preliminares 1, 2 e 3 são mostrados na figura 5.11.

Com a execução do passo de *Reestruturação*, a primeira etapa do processo de integração de BInXS é concluída. Os esquemas canônicos obtidos para as DTDs nesta etapa servem de entrada para a segunda e última etapa do processo, que é a *Integração Semântica*. Esta última etapa é apresentada no capítulo a seguir.

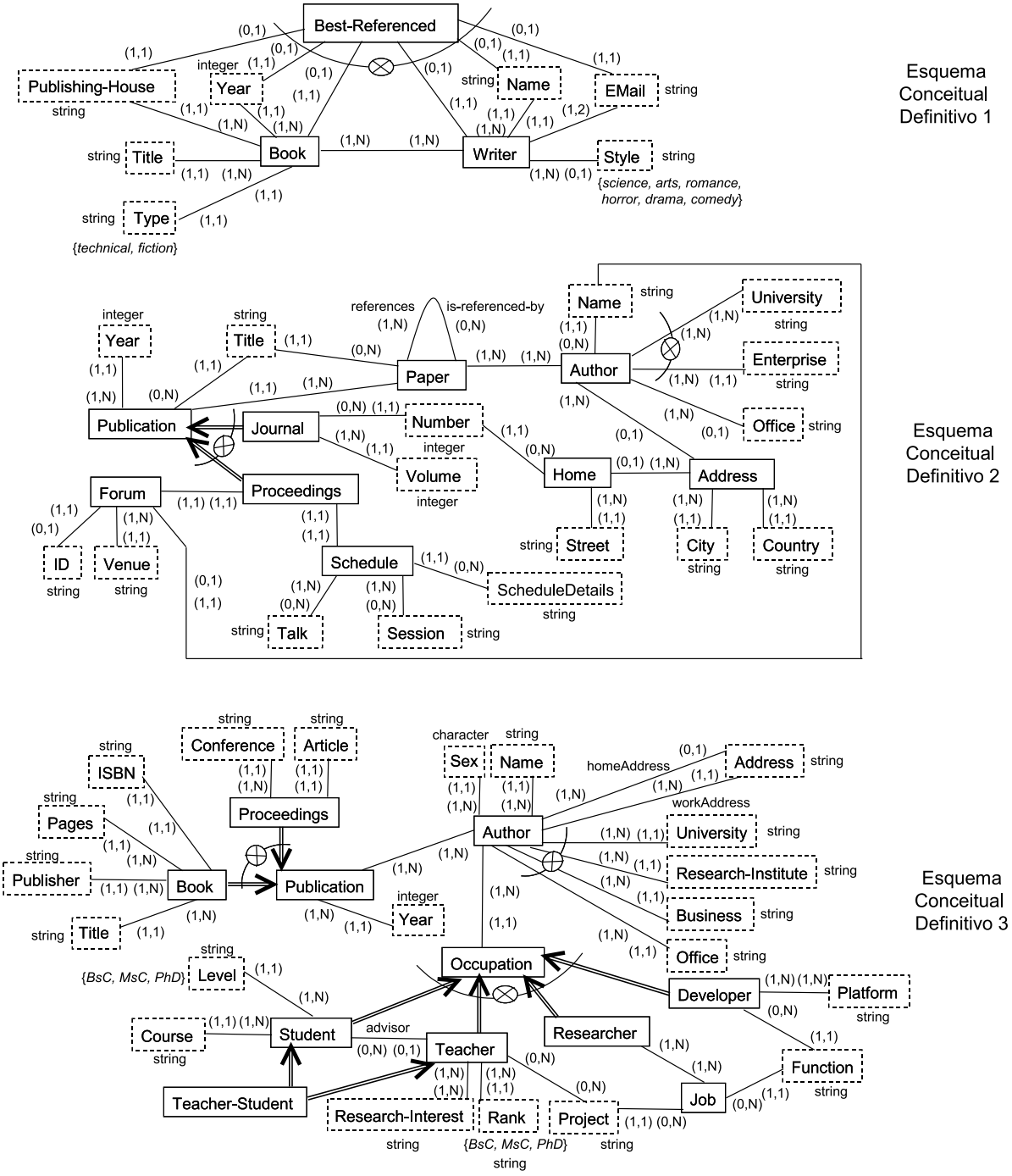


FIGURA 5.11 – Esquemas conceituais definitivos para as DTDs exemplo

6 Integração Semântica

Este capítulo descreve a segunda etapa do processo de integração de BInXS. Esta etapa realiza a integração semântica propriamente dita de esquemas conceituais obtidos a partir da conversão de DTDs. Esta etapa é composta por um processo que identifica equivalências semânticas entre conceitos de esquemas diferentes, aplica um conjunto de regras de unificação para tais conceitos para gerar um esquema global preliminar e realiza ajustes semi-automáticos neste esquema para transformá-lo em um esquema global definitivo para as fontes XML em questão.

Uma visão geral desta etapa está presente em um artigo publicado na edição de Março deste ano da revista *Information & Software Technology*, da *Elsevier Science* [MEL 2002]. Este artigo é resultado de um trabalho que foi parcialmente desenvolvido em um doutorado sanduíche realizado na Universidade de Milão, Itália, sob a supervisão da professora Silvana Castano.

Adota-se a seguinte terminologia neste capítulo: *esquema local* é um esquema conceitual representativo de uma DTD e *esquema global* é o esquema resultante da integração de esquemas locais.

6.1 Processo de Integração

A etapa de integração semântica executa um processo que gera um esquema global de referência para o acesso integrado a fontes XML. Este processo aplica as etapas tradicionais de integração de esquemas de banco de dados heterogêneos à integração de esquemas locais. Algumas particularidades deste processo são inerentes à abstração conceitual de dados XML: a integração de tipos diferentes de entidades (conceitos não-léxicos e léxicos), que representam a integração de informação textual e/ou estruturada em XML, e a integração de disjunções de relacionamentos. Estas particularidades são explicadas no decorrer deste capítulo.

Assim como a etapa de *conversão da DTD*, a etapa de *integração semântica* é semi-automática pois a intervenção do usuário especialista é necessária para confirmar se dois ou mais conceitos passíveis de integração apresentam realmente a mesma intenção semântica. Além do usuário como agente externo, utiliza-se uma ferramenta chamada *ARTEMIS* para o suporte à determinação de equivalências semânticas entre conceitos. ARTEMIS é uma ferramenta para determinação de visões globais de esquemas de dados heterogêneos desenvolvida pelo grupo de banco de dados da Universidade de Milão, Itália [CAS 2001]. Esta ferramenta foi cordialmente cedida pela professora Silvana Castano, líder do grupo, para ser aplicada nesta etapa.

A figura 6.1 ilustra o processo de integração semântica, que ocorre em quatro passos: agrupamento de sinônimos, unificação, inclusão de relações de herança e reestruturação. O passo de *agrupamento de sinônimos* define grupos (*clusters de afinidade*) de conceitos sinônimos advindos de esquemas locais diferentes. Estes clusters podem agrupar conceitos léxicos, conceitos não-léxicos ou um misto dos dois tipos de conceitos. Este passo é suportado pela ferramenta ARTEMIS e baseia-se na determinação e avaliação de *coeficentes de afinidade* entre conceitos para a construção dos clusters. Este passo é descrito na seção 6.3.

O passo de *unificação* é o núcleo desta etapa. Ele realiza a integração semântica

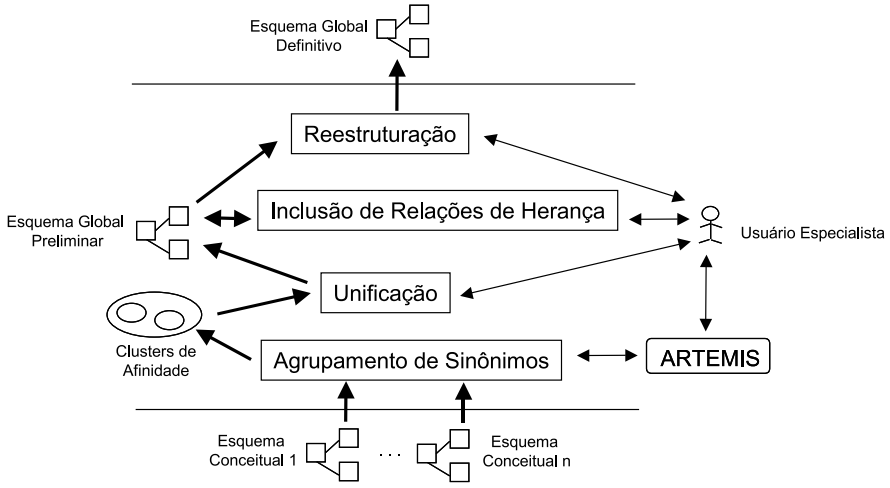


FIGURA 6.1 – Processo de integração semântica

propriamente dita de conceitos presentes em um mesmo cluster de afinidade, gerando um conceito global representativo do cluster e relacionamentos para outros conceitos globais ou clusters ainda não unificados. A intervenção do usuário pode ser solicitada para confirmar ou definir algumas correspondências semânticas entre conceitos. Um esquema global preliminar é obtido ao final deste passo, que é descrito na seção 6.4.

O passo de *inclusão de relações de herança* identifica novos relacionamentos de herança no esquema global preliminar com o auxílio de um Thesaurus. Estes relacionamentos, se confirmados pelo usuário especialista, são definidos no esquema global. Este passo é descrito na seção 6.5.

Por fim, o passo de *reestruturação* realiza ajustes automáticos e manuais no esquema global preliminar a fim de se obter um esquema global definitivo. Este passo é descrito na seção 6.6.

Os esquemas locais para as DTDs exemplo, apresentados no final do capítulo 5, servirão de exemplo nesta etapa. Inicialmente, é apresentada a ferramenta ARTEMIS e o seu método baseado em afinidade para a determinação de equivalências semânticas.

6.2 A Ferramenta ARTEMIS

ARTEMIS (*Analysis and Reconciliation Tool Environment for Multiple Information Systems*) é uma ferramenta semi-automática adequada à integração semântica de fontes de dados heterogêneos [CAS 2001]. Esta ferramenta é parte integrante do ambiente *MOMIS*, um dos trabalhos relacionados discutido no capítulo 2. ARTEMIS utiliza um modelo de referência para a representação canônica dos esquemas destas fontes, chamado ODL_I^3. ODL_I^3 dá suporte à representação de diversos esquemas de bancos de dados, como relacional e orientado a objetos, possuindo alguns operadores adicionais para a representação de esquemas semi-estruturados.

Uma especificação ODL_I^3 define um conjunto de *classes* de objetos. Cada classe possui um *nome*, um conjunto de *atributos* que descrevem suas propriedades ou relacionamentos com outras classes e um conjunto opcional de nomes de classes

que são suas *superclasses*. Uma classe deve ainda indicar o nome da *fonte de dados* a qual pertence.

Um exemplo de especificação de classe ODL_I^3 é a seguinte:

```
interface Journal : Publication (source semistructured schema2) {
  attribute set<Paper> Paper;
  attribute integer Number;
  attribute integer Volume;
};
```

Esta especificação indica que *Journal* é uma especialização de *Publication* e procede da fonte de dados *schema2*. *Journal* possui um relacionamento de associação com uma ou mais instâncias da classe *Paper* e dois atributos do tipo inteiro: *Number* e *Volume*.

O método baseado em afinidade adotado por ARTEMIS consiste das seguintes fases:

1. *Análise de classes*: nesta fase, o conceito de *afinidade* é utilizado para identificar equivalências semânticas entre conceitos. Um *coeficiente de afinidade global* é definido para cada par de classes, levando em conta seus nomes e estruturas. Este coeficiente mede o quanto duas classes denotam a mesma informação para o domínio em questão;

2. *Agrupamento de classes*: nesta fase, um *procedimento de agrupamento hierárquico* é utilizado para definir grupos de classes com afinidade. Clusters candidatos a se tornarem *clusters de afinidade* são sub-árvores selecionadas de uma *árvore de afinidade*. Nesta árvore existem nodos que identificam classes. Esta árvore é gerada pelo procedimento de agrupamento, com base nos coeficientes de afinidade global e um *threshold* predefinido pelo usuário;

3. *Construção de visões globais*: nesta fase, classes pertencentes a clusters de afinidade são unificados em uma visão global. Uma visão global é uma representação integrada destas classes e atua como uma referência para a consulta a fontes de dados.

As fases de ARTEMIS podem ser executadas de forma independente pelo usuário, sendo permitido a ele visualizar e modificar resultados intermediários gerados por cada uma delas. Nesta tese são utilizadas apenas as fases 1 e 2 de ARTEMIS pois está-se interessado na obtenção de um esquema global e não em um conjunto de visões. Estas fases são detalhadas a seguir.

6.2.1 Análise de Classes

A fase de *análise de classes* determina um *coeficiente de afinidade global* para cada par de classes. O cálculo deste coeficiente de afinidade leva em conta relacionamentos terminológicos entre dois termos t_1 e t_2 com o auxílio de Thesauri gerais e específicos do domínio[1]. Um termo se refere ao nome de uma classe ou atributo de classe. Os relacionamentos terminológicos considerados são *sinonímia* ($<t_1$ SYN

[1] ARTEMIS pode ser alimentado com Thesauri específicos definidos pelo usuário para um certo domínio.

t_2>) e *hipernímia* (<t_1 BT t_2>). Pesos σ_{SYN} e σ_{BT} são definidos pelo usuário para expressar o grau de relevância de cada um destes relacionamentos para a determinação do coeficiente de afinidade. Como a sinonímia é considerada um indicador mais preciso de afinidade, a restrição $\sigma_{SYN} \geq \sigma_{BT}$ vale sempre. Nesta fase, considera-se apenas afinidades entre pares de classes de esquemas locais diferentes pois não se vê sentido em unificar conceitos do mesmo esquema local.

A tabela 6.1 apresenta formalmente a função de afinidade e os coeficientes de afinidade utilizados para determinar o coeficiente de afinidade global. A *função de afinidade A(t, t')* define a afinidade entre dois termos com base no tipo de relacionamento terminológico que os conecta em um Thesaurus. Na verdade, ela retorna o valor máximo de afinidade encontrado nos k caminhos de comprimento $m \geq 1$ entre t e t' nos Thesauri consultados. $\sigma(t \rightarrow_i^m t') = \sigma_{1\Re}.\sigma_{2\Re}.\ldots.\sigma_{m\Re}$ é o peso de um i-ésimo caminho e $\sigma_{j\Re}$ denota o peso do j-ésimo relacionamento terminológico neste caminho.

TABELA 6.1 – Definições formais utilizadas por ARTEMIS

Nome	**Assinatura**	**Definição**
Função de Afinidade	$A(t, t')$	$max_{i=1..k}\ \{\sigma(t \rightarrow_i^m t')\}$, se $k > 1$; ou 0, c.c.
Coeficiente de Afinidade de Nome	$NA(c_j, c_h)$	$A(n(c_j), n(c_h))$, se $A(n(c_j), n(c_h)) \geq \alpha$; ou 0, c.c.
Coeficiente de Afinidade Estrutural	$SA(c_j, c_h)$	$2.\lvert\{\lceil a_l \rceil \in A_{jh}^{AC} \mid \lceil a_l \rceil^{wf}\}\rvert \ / \ \lvert A_{jh}\rvert$
Coeficiente de Afinidade Global	$GA(c_j, c_h)$	$\omega_{NA}.NA(c_j, c_h) + \omega_{SA}.SA(c_j, c_h)$, se $NA(c_j, c_h) \neq 0$; ou 0, c.c.

Suponha, por exemplo, que o usuário configurou $\sigma_{SYN} = 1$ e $\sigma_{BT} = 0{,}6$. Dado um Thesaurus com os seguintes relacionamentos:

Meeting BT *Event* BT *Symposium*
\BT | SYN
Conference

Tem-se: *A(Symposium, Conference)* = 1, *A(Meeting,Event)* = 0,6 e *A(Meeting,Conference)* = 0,36. No último exemplo, qualquer um dos caminhos entre *Meeting* e *Conference* produz o mesmo valor para a função de afinidade.

O *coeficiente de afinidade de nome* $NA(c_j, c_h)$ determina a afinidade de nomenclatura para duas classes c_j e c_h. O valor deste coeficiente coincide com o valor da função de afinidade avaliada sobre seus nomes $n(c_j)$ e $n(c_h)$, caso este valor seja maior ou igual a um threshold α definido pelo usuário. Valores elevados para α garantem que somente conceitos com alta afinidade de nome sejam considerados equivalentes.

O *coeficiente de afinidade estrutural* $SA(c_j, c_h)$ avalia o grau de correspondência semântica dos atributos das classes. Dois atributos a_t e a_q possuem *afinidade estrutural* se tiverem afinidade de nome ($NA(n(a_t), n(a_q)) \neq 0$) e uma opcional *compatibilidade de domínio*. ARTEMIS avalia a existência de compatibilidade de domínio de atributos da seguinte maneira:

- *Se a_t e a_q definem propriedades*: existe compatibilidade de domínio se seus tipos de dados são compatíveis, como por exemplo, *string* $\Leftrightarrow$ *string* e *integer*

$\Leftrightarrow$ *float*;

- *Se a_t e a_q definem relacionamentos*: existe compatibilidade de domínio se os nomes das classes referidas possuem afinidade de nome;

- *Se a_t define uma propriedade e a_q define um relacionamento*: existe compatibilidade de domínio se existe afinidade de nome entre $n(a_t)$ e o nome da classe referida por a_q.

De acordo com a tabela 6.1, o cálculo do coeficiente de afinidade estrutural está baseado na definição de grupos de afinidade $[a_l]$ para atributos de classes. Considerando A_{jh} o conjunto de atributos das classes c_j e c_h, um grupo de afinidade $[a_l]$ contém todos os atributos que possuem *afinidade estrutural* (definida anteriormente) com o atributo $a_l \in A_{jh}$. O conjunto de todos os grupos de afinidade é denotado por A_{jh}^{AC}. Para avaliar o coeficiente de afinidade estrutural, somente grupos de afinidade bem-formados são considerados. Um *grupo de afinidade bem-formado*, denotado por $[a_l]^{wf}$, contém no mínimo um atributo de c_j e um atributo de c_h. O valor do coeficiente de afinidade estrutural, mede o grau de sobreposição entre a estrutura das classes c_j e c_h, retornando um valor no intervalo [0,1].

Supondo, por exemplo, as seguintes classes ODL_I^3:

```
interface Author (source semistructured schema2) {
  attribute string Name;
  attribute set<Paper> Paper;
  attribute Address Address;
};

interface Author (source semistructured schema3) {
  attribute string Name;
  attribute string Sex;
  attribute string Address;
};
```

Tem-se: $|A_{jh}| = 6$ e dois grupos de afinidade bem-formados: `[Name]` e `[Address]`. Logo, *SA(Author (schema2), Author (schema3))* = 2.2/6 = 0,66.

Por fim, o *coeficiente de afinidade global GA(c_j, c_h)* define o grau de afinidade geral entre duas classes, sendo a soma dos coeficientes de afinidade de nome e estrutural. Os pesos ω_{NA} e ω_{SA} fornecem as relevâncias desejadas para cada tipo de afinidade, como mostra a tabela 6.1. Valem as seguintes restrições: w_{NA}, $w_{SA} \in$ [0,1] e $w_{NA} + w_{SA} = 1$.

Considerando as duas classes exemplificadas anteriormente e pesos[2] $w_{NA} = 0{,}7$ e $w_{SA} = 0{,}3$, tem-se *GA(Author (schema2), Author (schema3))* = 0,7.1 + 0,3.0,66 = 0,898 $\cong$ 0,9.

6.2.2 Agrupamento de Classes

A fase de *agrupamento de classes* aplica uma técnica de agrupamento dita *acumulativa*, ou seja, uma série de sucessivas unificações de classes em clusters. Esta

[2]Relevância maior é dada para a afinidade de nome pois esta é um indicador mais preciso de equivalência semântica.

técnica é implementada através de um *procedimento de agrupamento hierárquico* que produz uma hierarquia de elementos unificados chamada *árvore de afinidade*. A figura 6.2 mostra parte da árvore de afinidade definida por ARTEMIS para os conceitos dos esquemas locais exemplo.

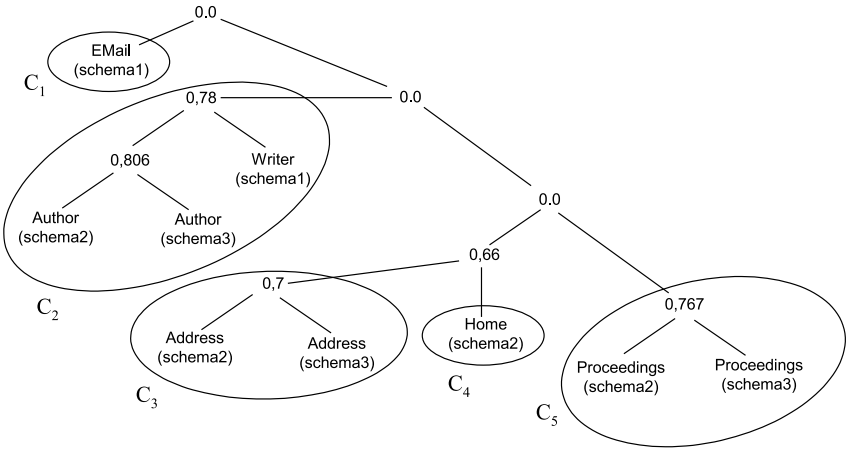


FIGURA 6.2 – Exemplo de uma árvore de afinidade

O procedimento de agrupamento recebe como entrada um conjunto S de classes e os coeficientes de afinidade global (GAs) definidos para pares de classes. Para cada classe $c_j \in S$ é definido inicialmente um cluster $C(c_j)$. O GA entre dois clusters $C(c_j)$ e $C(c_h)$ é dado por $GA(C(c_j), C(c_h))$. Com base em uma matriz M que armazena os GAs de todos os pares de clusters, o procedimento de agrupamento iterativamente seleciona o GA com maior valor em M e realiza a unificação dos seus clusters correspondentes $C(c_j)$ e $C(c_h)$, gerando um cluster $C(c_j+c_h)$. O GA de $C(c_j+c_h)$ com relação a um outro cluster $C(c_r)$ em M será o maior valor dentre $GA(C(c_j), C(c_r))$ e $GA(C(c_h), C(c_r))$. A matriz M é atualizada a cada iteração, mantendo inclusive os clusters gerados durante o procedimento.

Para exemplificar uma iteração deste procedimento, suponha que dois clusters a serem unificados sejam $C_i = \{Address\ (2)\}$ e $C_j = \{Address\ (3)\}$, com $GA(C_i, C_j) = 0{,}7$. Substitui-se os dois clusters individuais pelo cluster $C_{un} = \{Address\ (2), Address\ (3)\}$ em M e cria-se uma sub-árvore com nodos folha $Address\ (2)$ e $Address\ (3)$ e um nodo raíz $0{,}7$. A matriz M é então atualizada com as novas afinidades de C_{un} em relação aos outros clusters. Para um cluster $C_k = \{Home\ (2)\}$ em M, com $\mathrm{GA}(C_i, C_k) = 0{,}66$ e $\mathrm{GA}(C_j, C_k) = 0{,}61$, tem-se $\mathrm{GA}(C_k, C_{un}) = \mathrm{MAX}(\mathrm{GA}(C_i, C_k), \mathrm{GA}(C_j, C_k)) = 0{,}66$.

Como comentado anteriormente, a árvore de afinidade resultante possui classes nos nodos folhas e valores de GAs nos nodos intermediários que correspondem ao GA de dois clusters unificados (duas sub-árvores), como mostra a figura 6.2. Para definir os clusters candidatos a clusters de afinidade, cada GA g é comparado com um threshold de afinidade $\varphi \in (0{,}1]$ definido pelo usuário. Se $g \geq \varphi$, então a sub-árvore que tem g como raíz se torna um cluster candidato.

Considerando, por exemplo, um *threshold* $\varphi = 0{,}7$ e a árvore de afinidade da figura 6.2, os clusters C_1 a C_5 são definidos como clusters candidatos. Estes clusters contêm conceitos referentes a *EMail*, *Author*, *Address*, *Home* e *Proceedings*, respectivamente.

Dada esta visão geral da ferramenta ARTEMIS, as próximas seções apresentam os passos da etapa de integração semântica.

6.3 Passo 1: Agrupamento de Sinônimos

O passo de *agrupamento de sinônimos* é responsável pela determinação de equivalências semânticas entre conceitos de esquemas locais diferentes, ou seja, é responsável pela descoberta de conceitos semanticamente sinônimos. As duas primeiras fases da ferramenta ARTEMIS são empregadas para a determinação destas equivalências. As seguintes tarefas são executadas neste passo:

1. Configuração de ARTEMIS;
2. Conversão de conceitos dos esquemas de entrada em classes ODL_I^3;
3. Execução das duas primeiras fases de ARTEMIS e obtenção dos resultados.

A configuração de ARTEMIS é feita pelo usuário e prepara a ferramenta para a correta identificação de conceitos semanticamente sinônimos. Esta configuração procede da seguinte forma:

- deseja-se analisar apenas relacionamentos terminológicos de *sinonímia*, para que se possa proceder a unificação de conceitos. Assim, configura-se $\sigma_{SYN} = 1$ e $\sigma_{BT} = 0$;
- deseja-se dar mais ênfase à *afinidade de nome* pois esta afinidade é um indicador mais preciso de equivalência semântica. Assim, configura-se $\omega_{NA} = 0{,}7$ e $\omega_{SA} = 0{,}3$;
- desejam-se clusters compostos de conceitos com alta afinidade, para que a unificação possa ser a mais precisa possível. Experimentos com o uso de ARTEMIS sugerem um threshold $\varphi = 0{,}7$ como sendo o mais adequado para a definição de clusters de afinidade [CAS 2001].

Na seqüência, conceitos léxicos e não-léxicos são convertidos para classes ODL_I^3. Exemplos de conversões são mostrados na figura 6.3. Um conceito torna-se uma classe cujo nome é o nome do conceito, seguido do nome da fonte XML da qual deriva. Se o conceito for não-léxico, seus atributos são referências a todos os conceitos com os quais ele possui relacionamento. Um exemplo na figura 6.3 é a conversão do conceito *Writer* do esquema conceitual 1. Se o conceito for léxico, o conteúdo da sua classe ODL_I^3 é um atributo fixo de nome `content` cujo tipo corresponde ao tipo de dado do conceito. A conversão do conceito *Nome* é exemplificada na figura 6.3. Se o conceito léxico tiver uma enumeração, o atributo fixo é do tipo enumeração (`enum`) e os valores permitidos são declarados. Na figura 6.3, a conversão do conceito *Style* é um exemplo.

A conversão de conceitos léxicos para classe ODL_I^3 não considera os seus relacionamentos com conceitos não-léxicos. Enfatiza-se, assim, apenas a afinidade de nome e de tipo de conteúdo do conceito, que são as características fundamentais de um conceito léxico. Relacionamentos são considerados parte da estrutura de um conceito não-léxico, que é, por definição, um conceito composto por outros conceitos.

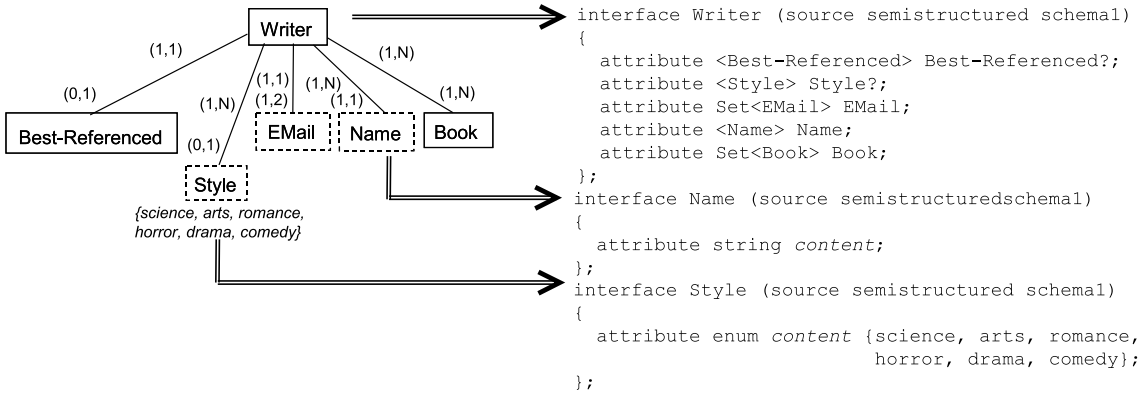


FIGURA 6.3 – Exemplo de conversão de conceitos para classes ODL_I^3

Uma vez feita a conversão para classes ODL_I^3, os esquemas locais são analisados e clusterizados por ARTEMIS. Uma descrição (arquivo) de todos os clusters de afinidade criados, com os nomes de conceitos participantes em cada cluster, é obtida como resultado.

Os resultados intermediários das fases de ARTEMIS podem ser investigados e modificados pelo usuário pois uma decisão automática em termos de afinidade de dados pode não corresponder precisamente à semântica do domínio.

Para os esquemas locais exemplo, a execução de ARTEMIS gera os clusters de afinidade mostrados a seguir. Os números associados aos conceitos indicam o esquema de procedência e o valor associado ao cluster é o seu coeficiente de afinidade. Este coeficiente é assumido como *zero* quando o conceito está sozinho em um cluster ou igual ou superior ao threshold φ, caso contrário.

```
C1 = {pages (3)} (0.0)                C2 = {venue (2)} (0.0)                C3 = {home (2)} (0.0)

C4 = {proceedings (2),                C5 = {number (2)} (0.0)               C6 = {sex (3)} (0.0)
      proceedings (3)} (0.767)

C7 = {writer (1),                     C8 = {conference (3)} (0.0)           C9 = {session (2)} (0.0)
      author (2),
      author (3)} (0.78)

C10 = {name (1), name (2),            C11 = {article (3)} (0.0)             C12 = {forum (2)} (0.0)
       name (3), title (1),
       title (2), title (3)} (0.86)

C13 = {publisher (3),                 C14 = {function (3)} (0.0)            C15 = {type (2)} (0.0)
       publishing_house (1),
       business (3)} (0.86)

C16 = {project (3),                   C17 = {researcher (3)} (0.0)          C18 = {e_mail (2)} (0.0)
       enterprise (2)} (0.86)

C19 = {year (1), year (2),            C20 = {best_referenced (1)} (0.0)     C21 = {course (3)} (0.0)
       year (3)} (1.0)

C22 = {university (2),                C23 = {country (3)} (0.0)             C24 = {level (3)} (0.0)
       university (3)} (1.0)

C25 = {office (2), office (3)} (1.0)  C26 = {schedule (2)} (0.0)            C27 = {city (3)} (0.0)

C28 = {platform (3)} (0.0)            C29 = {schedule_details (2)} (0.0)    C30 = {style (1)} (0.0)

C31 = {research_institute (3)} (0.0)  C32 = {occupation (3)} (0.0)          C33 = {paper (2)} (0.0)
```

```
C34 = {teacher (3)} (0.0)          C35 = {developer (3)} (0.0)          C36 = {job (3)} (0.0)

C37 = {street (2)} (0.0)           C38 = {volume (2), publication (3),  C39 = {rank (3)} (0.0)
                                          publication (2), book (1),
C40 = {student (3)} (0.0)                 book (3)} (0.731)

C41 = {research_interest} (0.0)    C42 = {journal (2)} (0.0)            C43 = {talk (2)} (0.0)

C44 = {address (2),                C45 = {teacher_student (3)} (0.0)    C46 = {ID (2)} (0.0)
       address (3)} (0.7)
```

Dos 46 clusters de afinidade gerados, 40 estão adequados ao domínio ($\cong 89\%$). Os clusters incorretos são os de número 10, 11, 13, 16, 33 e 38. Estes clusters são corrigidos pelo usuário da seguinte forma:

- *cluster 10*: separa os conceitos deste cluster (*nome* e *title*) em dois clusters: um para os conceitos *nome* e outro para os conceitos *title*, pois eles não são considerados sinônimos para o domínio;
- *clusters 11 e 33*: agrupa os conceitos destes dois clusters (*article* e *paper*) em um único cluster pois eles são considerados sinônimos para o domínio;
- *cluster 13*: separa os conceitos deste cluster em dois clusters: um para *publisher* e *publishing_house* (informações sobre editoras), e outra para *business* (informações sobre empresas);
- *cluster 16*: separa os conceitos deste cluster (*project* e *enterprise*) em dois clusters pois estes conceitos não são sinônimos para o domínio (informações sobre projetos de pesquisa e empresas, respectivamente);
- *cluster 38*: separa os conceitos deste cluster em três clusters: um para os conceitos *publication*, outro para os conceitos *book* (*book* não é considerado sinônimo de *publication* e sim uma especialização dela) e outro para *volume* (não é sinônimo para *publication* pois mantém o número do volume de um *journal*).

O conjunto de clusters de afinidade devidamente validado pelo usuário serve de entrada para o passo de *Unificação*.

6.4 Passo 2: Unificação

O passo de *Unificação* é o passo principal da etapa de integração semântica. Ele gera semi-automaticamente os conceitos e relacionamentos de um *esquema global preliminar* através da unificação de conceitos mantidos em um mesmo cluster de afinidade.

Este passo é baseado em um conjunto de *regras semi-automáticas de unificação* que resolvem conflitos de heterogeneidade entre conceitos. Estas regras são aplicadas no contexto de três *casos de unificação*. Cada caso trata um determinado tipo de cluster gerado no passo anterior. Estes casos são os seguintes:

1. **Caso LxL** (*Unificação Léxica*): unifica um cluster composto apenas por conceitos léxicos (*cluster léxico*), gerando um conceito global léxico. Regras são aplicadas para resolver conflitos de nome e de tipo de dado;

2. **Caso NLxNL** (*Unificação Não-Léxica*): unifica um cluster composto apenas por conceitos não-léxicos (*cluster não-léxico*), gerando um conceito global não-léxico. Regras são aplicadas para resolver conflitos de nome, de relacionamentos e de disjunções de relacionamentos;

3. **Caso NLxL** (*Unificação Mista*): unifica um cluster composto por conceitos léxicos e não-léxicos (*cluster misto*), gerando um conceito global não-léxico C_{un} representativo dos conceitos não-léxicos do cluster. Regras são aplicadas para resolver conflitos de heterogeneidade de nome e de relacionamentos. Os conceitos léxicos presentes no cluster são mapeados para conceitos globais léxicos relacionados a C_{un}. Se tal mapeamento não for possível, um conceito global léxico é gerado para representá-los.

O passo de unificação é realizado por um algoritmo que recebe como entrada um *conjunto de clusters de afinidade* C_c e especificações de esquemas canônicos, gerando como saída um *esquema global preliminar*. A execução deste algoritmo é a seguinte:

1. Gera uma esquema global inicial *EG = <N, NL, L, R, D>*, onde *N* é o nome do esquema global fornecido pelo usuário, *NL* é o conjunto de conceitos globais não-léxicos, *L* é o conjunto de conceitos globais léxicos, *R* é o conjunto de relacionamentos entre conceitos $\in$ *NL* $\cup$ *L* e *D* é o conjunto de disjunções definidas para relacionamentos em *R*;

2. Para cada cluster léxico $C_l \in C_c$ executa *LxL*(C_l);

3. Para cada cluster não-léxico $C_{nl} \in C_c$ ou misto $C_m \in C_c$ executa *NLxNL*(C_{nl}) ou *NLxL*(C_m), respectivamente.

Este algoritmo gera uma especificação de esquema global preliminar inicialmente vazia e vai definindo conceitos, relacionamentos e disjunções para este esquema através da execução dos passos 2 e 3. Os três casos de unificação são tratados pelos procedimentos *LxL*, *NLxNL* e *NLxL*, respectivamente.

Clusters léxicos são unificados primeiro porque *simplificam* a unificação de clusters não-léxicos e mistos. Isto porque a unificação de conceitos não-léxicos e mistos envolve a unificação ou união de seus relacionamentos. A unificação de relacionamentos analisa os conceitos destino dos relacionamentos (muitos são conceitos léxicos) para determinar a chamada *afinidade de relacionamentos* (ver seção 6.4.3 para mais detalhes). Esta unificação é facilitada se estes conceitos destino estiverem já unificados em conceitos globais ao invés de investigar se os conceitos léxicos destino pertencem a um mesmo cluster para fins de determinação desta afinidade.

Na unificação de clusters não-léxicos ou mistos, a preferência é pela unificação de clusters que possuam conceitos que são *generalizações* de outros conceitos. Isto porque os relacionamentos de um conceito genérico influenciam a unificação de relacionamentos de um conceito especializado. Dentre os clusters com conceitos genéricos, dá-se prioridade para o cluster com o conceito genérico cuja hierarquia de conceitos especializados apresenta o *maior número de níveis* pois o seu grau de influência é maior.

O tratamento de cada caso de unificação é descrito nas seções a seguir, após a apresentação da regra de unificação de nome.

6.4.1 Unificação de Nomenclaturas

Todos os tipos de cluster a serem unificados mantêm um conjunto de conceitos que podem apresentar nomes diferentes. Como apenas um conceito global é gerado como resultado da unificação de um cluster, uma regra para a unificação de nomenclaturas de conceitos deve ser adotada. Esta regra é definida a seguir.

Regra UnNome (Unificação de Nome). *Dado um conjunto de nomes de conceitos* $N = \{n_1, ..., n_m\}$, *o nome unificado* $\overline{n}$ *para* N *é dado por:*

$$\overline{n} = \begin{cases} n_i, & se\ \exists\ n_i \in N\ (|n_i| = MAX(\{|n_1|, ..., |n_m|\}) \wedge |n_i| > 1); \\ definido\ manualmente, & caso\ contrário,\ com\ \overline{n} \in N \cup SYN(N). \end{cases}$$

A regra **UnNome** diz que o nome de um conceito global é o nome que ocorre o maior número de vezes em um conjunto de nomes de conceitos. Caso não exista um nome com maioria, o nome é definido pelo usuário, podendo ser um dos nomes do conjunto ou um sinônimo adequado a todos eles.

Exemplos:

```
{style (1)} ⇒ style;
{title (1), title (2), title (3)} ⇒ title;
{writer (1), author (2), author (3)} ⇒ author;
{enterprise (2), business (3)} ⇒ enterprise.
```

6.4.2 Unificação Léxica

A *unificação léxica* (procedimento *LxL*) trata da unificação de um cluster de afinidade composto apenas por conceitos léxicos. Ela corresponde a unificação de informação textual presente em fontes XML. Para este tipo de unificação, é gerado um *conceito global léxico* representativo de todos os conceitos do cluster.

Como cada conceito léxico possui um *nome*, um *tipo de dado* e opcionalmente, uma *enumeração* de valores permitidos, conflitos de heterogeneidade relacionados a estas características devem ser resolvidos. A regra **UnNome** é utilizada para resolver o conflito de nome. Adicionalmente, duas regras são definidas a seguir: a *regra de unificação de tipo de dado* e a *regra de unificação de enumeração.*

Regra UnTipo (Unificação de Tipo de Dado). *Dado um conjunto de tipos de dados* $TD = \{td_1, ..., td_n\}$, *o tipo de dado unificado* $\overline{td}$ *para* TD *é dado por:*

$$\overline{td} = \begin{cases} td_i \in TD, & se\ \forall\ td_j \in TD\ (td_i \Leftrightarrow td_j \wedge \neg\ (td_i \subset td_j)); \\ string, & caso\ contrário. \end{cases}$$

A regra **UnTipo** diz que o tipo de dado unificado para um conjunto TD de tipos de dados é o tipo mais genérico td_i dentre todos os tipos em TD (td_i não está contido em um outro tipo mais genérico), se todos os pares de tipos são compatíveis entre si ($td_i \Leftrightarrow td_j$). Caso não exista tal compatibilidade, o tipo *string* é assumido como tipo unificado.

Exemplos genéricos:

```
{date, date} ⇒ date;
{integer, float, integer} ⇒ float;
```

{character, time, integer} ⇒ string.

Regra UnEnum (Unificação de Enumeração). *Dado um conjunto de conceitos léxicos* $L = \{l_1, ..., l_n\}$ *que podem ter enumerações* $e_1, ..., e_n$ *associadas, a enumeração unificada* $\overline{en}$ *para* L *é dada por:*

$$\overline{en} = \begin{cases} e_1 \cup ... \cup e_n, \text{ com validação manual}, & \text{se } \forall\ l_i \in L\ (\exists\ e_i); \\ \emptyset, & \text{caso contrário.} \end{cases}$$

A regra **UnEnum** diz que, caso todos os conceitos léxicos de um cluster possuam enumerações associadas, a enumeração unificada será a união destas enumerações. Uma validação manual é requerida para eliminar valores sinônimos redundantes. Caso um ou mais conceitos não possuam enumeração, não é definida uma enumeração unificada pois a mesma não é representativa de todos os conceitos do cluster.

Exemplos genéricos:

L = {l1 {a, b, c}, l2 {a, c, e}, l3 {b, x, y}} ⇒ $\overline{en}$ = {a, b, c, e, y} (supondo <a SYN x>);
L = {l1 {a, b, c}, l2, l3} ⇒ $\overline{en}$ = ∅.

Com base nestas regras, é mostrado a seguir o resultado das unificações léxicas para os clusters de afinidade gerados no passo anterior que apresentam mais de um conceito. Clusters com apenas um conceito produzem um conceito semelhante no esquema global.

{<Publishing-House, string> (1), <Publisher, string> (3)} ⇒ <Publisher, string>;
{<Year, integer> (1), <Year, integer> (2), <Year, integer> (3)} ⇒ <Year, integer>;
{<Title, string> (1), <Title, string> (2), <Title, string> (3)} ⇒ <Title, string>;
{<Nome, string> (1), <Nome, string> (2), <Nome, string> (3)} ⇒ <Nome, string>;
{<University, string> (2), <University, string> (3)} ⇒ <University, string>;
{<Enterprise, string> (2), <Business, string> (3)} ⇒ <Enterprise, string>;
{<Office, string> (2), <Office, string> (3)} ⇒ <Office, string>.

6.4.3 Unificação Não-Léxica

A *unificação não-léxica* (procedimento *NLxNL*) trata da integração de um cluster de afinidade composto apenas por conceitos não-léxicos. Ela corresponde à unificação de informação estruturada presente nas fontes XML. Para este tipo de unificação, é gerado um *conceito global não-léxico* representativo de todos os conceitos do cluster.

Um conceito não-léxico possui um *nome* e participa de *relacionamentos*, sendo que alguns destes relacionamentos podem ser *disjuntos*. Para tanto, conflitos de heterogeneidade relacionados a estas características devem ser resolvidos.

A unificação de nomes é feita pela regra **UnNome**. A unificação de relacionamentos e o tratamento de disjunções de relacionamentos são detalhados a seguir.

Unificação de Relacionamentos

A unificação de relacionamentos é realizada *iterativamente* sobre pares de conceitos do cluster, ou seja, um par c_i e c_j é substituido por um conceito unificado c_{un} resultante da *união* dos relacionamentos de c_i e c_j. Este processo se repete até que c_{un} seja o único conceito presente no cluster, resultando no conceito global.

Na unificação iterativa dos relacionamentos de um par de conceitos c_i e c_j, é possível que um relacionamento de c_i possua *afinidade* com um relacionamento de c_j. A definição de relacionamentos com afinidade é dada a seguir.

(Relacionamentos com Afinidade). *Um relacionamento r_i de um conceito c_i possui afinidade com um relacionamento r_j de um conceito c_j se:*

- *r_i, $r_j \in A$ e associam c_i e c_j com o mesmo conceito global ou com conceitos pertencentes a um mesmo cluster de afinidade;*
- *r_i, $r_j \in H$, sendo c_i e c_j ambos conceitos genéricos de um conceito global ou dos conceitos pertencentes a um mesmo cluster de afinidade.*

Exemplos:

Exemplo 1: <Book, Publishing-House, (1,1), (1,N)> $\in A$ (1) e
<Book, Publisher, (1,1), (1,N)> $\in A$ (3)

Exemplo 2: <Publication, Proceedings> $\in H$ (2) e
<Publication, Proceedings> $\in H$ (3)

No primeiro exemplo, um cluster de afinidade mantém os conceitos não-léxicos *Book* dos esquemas locais 1 e 3. Ambos possuem relacionamentos de associação com os conceitos *Publishing-House* e *Publisher* que se encontram em um mesmo cluster de afinidade. No segundo exemplo, tem-se um cluster de afinidade que mantém os conceitos *Publication* dos esquemas locais 2 e 3, ocorrendo afinidade entre os relacionamentos de herança destes conceitos para os conceitos afins *Proceedings*.

Um relacionamento de associação r_i de c_i pode ter afinidade com *mais de um* relacionamento de c_j. Em geral, tal situação ocorre quando c_j possui mais de um relacionamento com um mesmo conceito, sendo estes relacionamentos nomeados ou com papéis. Nestes casos, o usuário ou elege um relacionamento de c_j para ser unificado com r_i ou assume que r_i não possui afinidade com relacionamentos de c_j. Esta última decisão ocorre quando a intenção semântica de r_i não corresponde à intenção semântica dos relacionamentos de c_j. Então, r_i passa a ser um relacionamento no nível global e o usuário deve definir um nome ou papéis para ele a fim de evitar conflitos com os relacionamentos de c_j no momento da unificação.

Um exemplo para esta situação de afinidade *um-para-muitos* ocorre no cluster que mantém os conceitos *Author* dos esquemas 2 e 3 e seus relacionamentos com os conceitos *Address*. A afinidade de relacionamentos é definida da seguinte forma:

<Author, Address, (0,1), (1,N)> $\in A$ (2) e

$$\begin{cases} \text{<Author, Address, (1,1), (1,N), workAddress>} \in A\ (3) \\ \text{<Author, Address, (0,1), (1,N), homeAddress>} \in A\ (3) \end{cases}$$

$$\Downarrow$$

<Author, Address, (0,1), (1,N)> (2) $\equiv$ <Author, Address, (1,1), (1,N), workAddress> (3)

O usuário, neste exemplo, elegeu o relacionamento *workAddress* do esquema 3 como sendo o relacionamento afim ao relacionamento do esquema 2, assumindo que ambos dizem respeito ao endereço profissional do autor.

Quando um relacionamento r_i de c_i *não possui* afinidade com um relacionamento de c_j (ou vice-versa), r_i é definido como um relacionamento *opcional* de c_{un},

se $r_i \in A$. Isto ocorre porque r_i não está definido em todos os esquemas locais que estão sendo integrados. Porém, antes de se definir r_i como opcional, é necessário verificar se ele não possui afinidade com um relacionamento r_g definido para um *conceito genérico* c_g de c_j. Em caso positivo, r_i fica como relacionamento obrigatório de c_{un} se for obrigatório para c_i e r_g for obrigatório também para c_g.

Um exemplo que ilustra esta situação ocorre para o cluster que mantém os conceitos *Article* do esquema local 3 e *Paper* do esquema local 2:

r_i = <Proceedings, Article, (1,1), (1,1)> $\in A$ (3) e
r_g = <Publication, Paper, (1,N), (1,1)> $\in A$ (2) e
$\exists$ <Publication, Proceedings> $\in H$ (2)

Paper não possui relacionamento com *Proceedings* no esquema local 2, mas *Article* sim. Logo, o relacionamento r_i do conceito global referente ao cluster {*Paper* (2), *Article* (3)} com o conceito global *Proceedings* deveria ser opcional. Porém, *Paper* relaciona-se de forma *obrigatória* com o conceito genérico de *Proceedings*, que é *Publication* (r_g). Como *Article* também se relaciona de forma obrigatória com *Proceedings* , r_i passa a ser um relacionamento obrigatório.

Dois relacionamentos com afinidade são tratados pela *regra de unificação de relacionamentos*, apresentada a seguir.

Regra UnRel (Unificação de Relacionamentos). *Dados dois relacionamentos r_i,r_j $\in R$ pertencentes a conceitos c_i e c_j, respectivamente, com afinidade em relação a conceitos de um cluster C_x, o relacionamento unificado $\overline{r} \in R$ é dado por:*

$$\overline{r} = \begin{cases} <c_{un}.n,\ c_{C_x},\ Card_U(c_i\rightarrow,\ c_j\rightarrow),\ Card_U(c_i\leftarrow,\ c_j\leftarrow)> \in A, & \text{se } r_i, r_j \in A; \quad (i) \\ \text{definido manualmente,} & \text{se } r_i, r_j \in A \text{ e } r_i \text{ e/ou } r_j \text{ tem nome ou papéis;} \quad (ii) \\ <c_{un}.n,\ c_{C_x}> \in H, & \text{se } r_i, r_j \in H. \quad (iii) \end{cases}$$

A regra **UnRel** considera c_{C_x} o nome do conceito global correspondente ao cluster de afinidade C_x, sendo C_x o cluster que mantém os conceitos relacionados a c_i e c_j. Caso C_x não tenha sido ainda unificado, c_{C_x} é deixado temporariamente indefinido, devendo $\overline{r}$ ser atualizado posteriormente com este nome.

No caso (i), $c_i\rightarrow$ denota a cardinalidade do relacionamento que parte de c_i e $c_i\leftarrow$ denota a cardinalidade do relacionamento que chega a c_i. O mesmo vale para $c_j\rightarrow$ e $c_j\leftarrow$. Cardinalidades correspondentes são unificadas através da regra $Card_U$.

Quando r_i e/ou r_j possuem nome ou papéis (caso (ii)), não é possível determinar automaticamente o resultado da unificação. Isto porque não se pode precisar se ambos apresentam a mesma intenção semântica ou não. Assim, a intervenção do usuário é necessária para decidir entre:

- gerar um relacionamento unificado $\overline{r}$ com uma definição adequada de nome ou papéis, caso r_i e r_j possuam a mesma intenção semântica; ou
- manter ambos r_i e r_j associados ao conceito unificado c_{un}, caso contrário. O usuário deve definir nomes ou papéis adequados para ambos, a fim de explicitar a intenção semântica de cada um.

A unificação de dois relacionamentos de associação que não possuem nome nem papéis (caso (i)) pode ser relaxado na regra **UnRel** quando os dois relacionamentos não possuem a mesma intenção semântica. Um exemplo seria a existência de dois relacionamentos *autor-cidade*, sendo o primeiro a cidade que o autor reside e o segundo a cidade que o autor trabalha. Nesta situação, o usuário pode desfazer o relacionamento $\overline{r}$ e definir dois relacionamentos de associação globais com nomes e/ou papéis adequados a cada intenção semântica.

Exemplos de unificação de relacionamentos:

Exemplo1: `<Book, Publishing-House, (0,1), (1,N)>` $\in A$ (1) e
`<Book, Publisher, (1,1), (1,N)>` $\in A$ (3)
⇓
`<Book, Publisher, (0,1), (1,N)>`

Exemplo2: `<Author, Address, (0,1), (1,N)>` $\in A$ (2) e
`<Author, Address, (1,1), (1,N), workAddress>` $\in A$ (3)
⇓
`<Author, Address, (0,1), (1,N), workAddress>`

Exemplo3: `<Publication, Proceedings>` $\in H$ (2) e
`<Publication, Proceedings>` $\in H$ (3)
⇓
`<Publication, Proceedings>`

No primeiro exemplo (caso(i)), tem-se dois relacionamentos de associação com afinidade que não possuem nome nem papéis. Ambos pertencem a conceitos *Book*, associando-os com conceitos relacionados a editoras. Considerando que ambos possuam a mesma intenção semântica, um relacionamento de associação entre o conceito global *Book* e o conceito global *Publisher* é gerado.

No segundo exemplo (caso(ii)), tem-se também dois relacionamentos de associação com afinidade, sendo que um deles é nomeado. O usuário assume que ambos os relacionamentos apresentam a mesma intenção semântica e um relacionamento de associação unificado nomeado com as devidas cardinalidades generalizadas é gerado. O nome *workAddress* é mantido. O terceiro exemplo (caso(iii)) mostra dois relacionamentos de herança com afinidade, resultando em um relacionamento de herança unificado.

Uma observação importante sobre a unificação de relacionamentos diz respeito ao *conceito destino* de r_i e r_j. O cluster C_x que mantém estes conceitos destino pode já ter sido unificado. Se este cluster é não-léxico, um relacionamento $\overline{r'}$ do conceito global c_{un} correspondente a C_x já foi definido com o cluster C_y que mantém os conceitos origem de r_i e r_j (c_i e c_j). Desta forma, o relacionamento unificado $\overline{r}$ não precisa ser criado pois seria redundante com $\overline{r'}$. A única tarefa a ser feita neste caso é definir (atualizar) as cardinalidades de $\overline{r'}$ como sendo as cardinalidades mais gerais dentre $\overline{r'}$ e $\text{Card}_U(c_i\rightarrow, c_j\rightarrow)$, $\text{Card}_U(c_i\leftarrow, c_j\leftarrow)$.

Tratamento de Disjunções

Ambos os conceitos não-léxicos c_i e c_j podem apresentar disjunções definidas para os seus relacionamentos. Estas disjunções devem ser analisadas para fins de determinação de restrições sobre os relacionamentos do conceito unificado c_{un}.

Um *algoritmo* é proposto nesta tese para o tratamento de disjunções. Este algoritmo analisa cada disjunção definida para os relacionamentos de c_i e c_j. Esta

análise se baseia na existência ou não de afinidades entre os relacionamentos envolvidos em uma disjunção d_i para decidir se e como d_i será definida em termos de relacionamentos de c_{un}. O funcionamento deste algoritmo é explicado a seguir.

Suponha uma disjunção d_i definida para um conjunto de relacionamentos $C_{R_i} = \{r_{i1}, ..., r_{in}\}$ do conceito c_i. Esta disjunção é tratada por um dos seguintes casos:

- **Caso 1:** se *nenhum* ou apenas *um* relacionamento de c_j possui afinidade com os relacionamentos de C_{R_i}, então d_i é definida para os relacionamentos de c_{un} correspondentes aos relacionamentos de C_{R_i}. Este caso considera d_i de c_i para os relacionamentos de c_{un} pois não existem dois ou mais relacionamentos afins em c_j que possam determinar um conflito de disjunção com os relacionamentos de d_i;

- **Caso 2:** se existe um *conjunto* de relacionamentos $C_{R_j} = \{r_{j1}, ..., r_{jm}\}$ de c_j que possuem afinidade com relacionamentos de C_{R_i}, $|C_{R_j}| > 1$, então d_i não é definida em c_{un}. Ao invés disto, cada relacionamento $r_{jh} \in C_{R_j}$ é analisado da seguinte forma:

  - Se r_{jh} não é disjunto de algum outro relacionamento em C_{R_j}, então o relacionamento correspondente a r_{jh} torna-se *opcional* para c_{un} se $r_{jh} \in A$;
  - Uma disjunção d_{un} é considerada para os relacionamentos de c_{un}. Esta disjunção compreende:
    * o relacionamento global correspondente a r_{jh};
    * todos os relacionamentos globais correspondentes a relacionamentos de C_{R_j} que são disjuntos de r_{jh};
    * todos os relacionamentos globais correspondentes a relacionamentos de C_{R_i} que não possuem afinidade com relacionamentos de C_{R_j}.

    A disjunção d_{un} só é realmente definida no esquema global se:
    * não for redundante com outra disjunção definida para os relacionamentos de c_{un};
    * não estiver contida em outra disjunção d'_{un} definida para os relacionamentos de c_{un};
    * não conter apenas r_{jh};

    Se d_{un} contém uma outra disjunção $d"_{un}$ já definida para os relacionamentos de c_{un}, então $d"_{un}$ é removida.

Este mesmo tratamento é aplicado para as disjunções definidas para os relacionamentos do conceito c_j.

O **caso 2** analisa a possiblidade de conflito de disjunção para todos os relacionamentos afins de c_j. O fato de um relacionamento $r_{jh} \in C_{R_j}$ não ser disjunto de outro relacionamento no mesmo conjunto significa que existe um conflito de disjunção com os relacionamentos afins em C_{R_i}, uma vez que todos os relacionamentos em C_{R_i} são disjuntos. Para evitar tal conflito, d_i é desconsiderada e define-se o relacionamento correspondente a r_{jh} em c_{un} como *opcional*. Esta opcionalidade evita que r_{jh} ocorra sempre simultaneamente a outro relacionamento de C_{R_j} em c_{un}, visto que os relacionamentos afins a C_{R_j} em C_{R_i} são todos disjuntos.

Ainda no **caso 2**, apenas disjunções que sejam válidas para ambos os relacionamentos afins de c_i e c_j são definidas para os relacionamentos em c_{un}. Uma *disjunção válida* é aquela que considera r_{jh}, outros relacionamentos de C_{R_j} que sejam disjuntos de r_{jh} e relacionamentos de C_{R_i} que não tenham relacionamento afim em C_{R_j}. Estes relacionamentos de C_{R_i}, se existirem, devem ser disjuntos de r_{jh} para respeitarem a disjunção d_i.

Um exemplo de tratamento de disjunções é mostrado na figura 6.4. Tal tratamento é parte do processo de unificação dos conceitos não-léxicos *Author* dos esquemas locais 2 e 3.

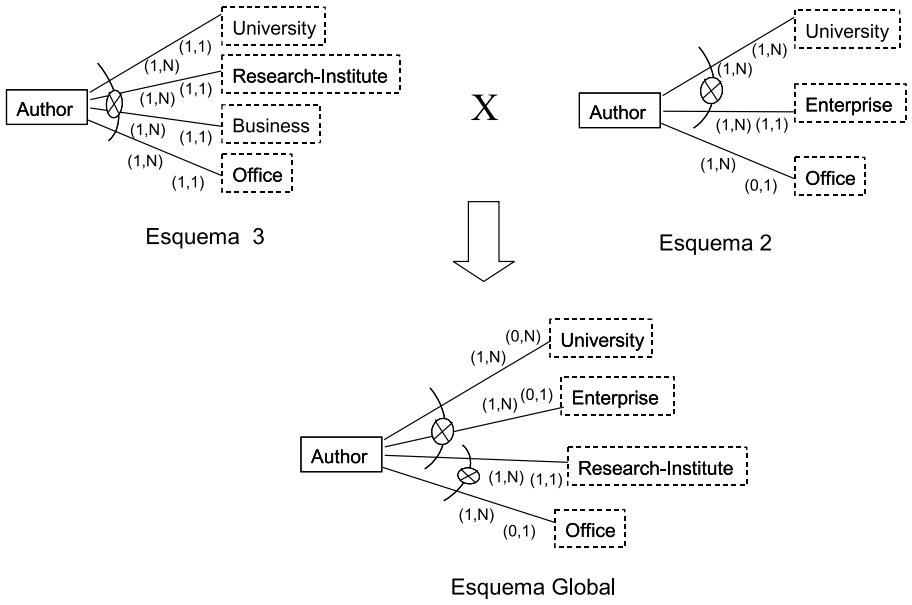


FIGURA 6.4 – Exemplo de tratamento de disjunções

Neste exemplo, o resultado final já é obtido quando se analisa a disjunção formada pelo conjunto de relacionamentos $C_{R_i} = \{$*Author-University*, *Author-Research-Institute*, *Author-Business*, *Author-Office*$\}$ no esquema 3. Para tal disjunção tem-se o conjunto de relacionamentos afins $C_{R_j} = \{$*Author-University*, *Author-Enterprise*, *Author-Office*$\}$ de *Author* no esquema 2. Como $|C_{R_j}| > 1$, recai-se no caso 2 e todos os relacionamentos de C_{R_j} são analisados. Começando por *Author-University*, constata-se que ele não é disjunto de *Author-Office*. O relacionamento *Author-University* é definido como opcional para *Author* no esquema global e define-se uma disjunção d_1 composta por este relacionamento mais *Author-Enterprise* (disjunto dele no esquema 3) mais *Author-Research-Institute* (disjunto dele no esquema 2). Na seqüência, a análise de *Author-Enterprise* o torna opcional no esquema global e produz uma disjunção semelhante a d_1, sendo portanto desconsiderada. A análise de *Author-Office* constata que ele não é disjunto dos demais relacionamentos em C_{R_j}. Ele então é definido como opcional no esquema global e uma disjunção d_2 que o inclui mais o relacionamento *Author-Research-Institute* (disjunto dele no esquema 3) é definida.

A análise que considera C_{R_i} o conjunto dos relacionamentos disjuntos do esquema 2 produzirá disjunções que serão cobertas por d_1 e d_2, não alterando o resultado final. As disjunções d_1 e d_2 respeitam as disjunções nos esquemas locais 2 e

3.

Exemplo de Unificação Não-Léxica

Uma vez apresentado o tratamento de relacionamentos e disjunções, um exemplo de unificação não-léxica é mostrado na figura 6.5. Este exemplo diz respeito a unificação do cluster que mantém os conceitos *Publication* presentes nos esquemas locais 2 e 3.

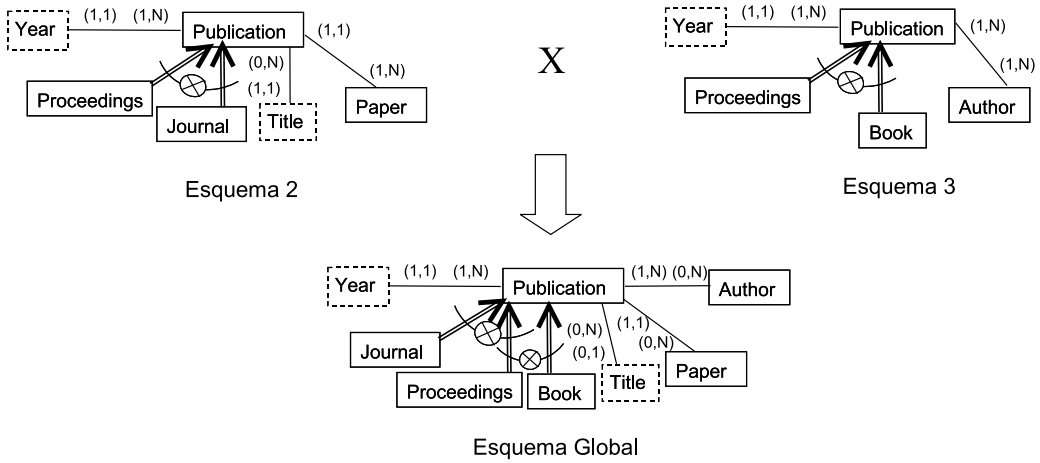


FIGURA 6.5 – Exemplo de unificação de conceitos não-léxicos

A regra **UnNome** define *Publication* como sendo o nome do conceito no esquema global. No tratamento de relacionamentos, dois casos de relacionamentos com afinidade são encontrados: *Publication-Year* e *Publication-Proceedings*. O primeiro deles é unificado em um relacionamento de associação global *Publication-Year* e o segundo gera um relacionamento de herança global para *Publication-Proceedings*. Os demais relacionamentos são considerados relacionamentos do conceito global *Publication*, sendo definidos como opcionais os relacionamentos que forem de associação.

Existem duas disjunções nos esquemas 2 e 3, que possuem apenas um conceito em comum. O caso 1 do tratamento de disjuções se aplica a ambas, sendo ambas consideradas para o conceito global.

A unificação dos demais clusters não-léxicos para os esquemas locais exemplo segue o mesmo princípio do exemplo recém-mostrado. Estas unificações não são apresentadas para não tornar extensa esta seção. O esquema global exemplo obtido é mostrado na seção 6.4.5 e ilustra a unificação de todos os clusters não-léxicos.

6.4.4 Unificação Mista

A *unificação mista* (procedimento *NLxL*) trata da integração de um cluster de afinidade composto por conceitos não-léxicos e léxicos. Ela corresponde a unificação de informação textual e estruturada presente nas fontes XML. Para este tipo de unificação, é gerado um *conceito global não-léxico* representativo de todos os conceitos do cluster. O princípio da unificação mista é dar preferência a uma representação global não-léxica para o cluster misto considerando que conceitos não-léxicos detalham a organização de uma informação.

A unificação de um cluster misto C_M é realizada da seguinte forma: a unificação de nomes de todos os conceitos em C_M é regida pela regra **UnNome**. Em seguida, procede-se a *unificação não-léxica* de todos os conceitos não-léxicos de C_M. Um conceito global não-léxico c_{un} é gerado. Após, para cada conceito léxico $c_L \in C_M$ e dado um conjunto de conceitos léxicos globais S_L alcançados a partir de c_{un}, o usuário decide por uma dentre as seguintes alternativas:

- *mapeia c_L para um conceito léxico $c'_L \in S_L$,* *se $\exists\ c'_L \in S_L$ cujo conteúdo tem a mesma intenção semântica de c_L;* *(i)*
- *define c_L como um conceito*
  - *global*
    - *que é uma alternativa não-léxica de especialização para c_{un},* *se $\exists\ S'_L \subseteq S_L, |S'_L| \geq 1$, tal que o conteúdo ou conjunto de conteúdos dos conceitos $c''_L \in S'_L$ tem a mesma intenção semântica de c_L; (ii)*
    - *associado opcionalmente a c_{un},* *se $\neg\ \exists\ c'_L \in S_L$ cujo conteúdo tem a mesma intenção semântica de c_L. (iii)*

A unificação mista é realizada em duas fases. Primeiro, determina-se um conceito não-léxico global c_{un} para todos os conceitos não-léxicos de C_M. Após, é definido um mapeamento de cada conceito léxico c_L de C_M para um conceito léxico global com correspondência semântica que esteja relacionado direta ou indiretamente a c_{un}, se existir (mapeamento *um-para-um* - caso (i)). Se não existir tal conceito, c_L torna-se um conceito léxico global associado opcionalmente a c_{un} (mapemamento inexistente - caso (iii)).

Uma outra possibilidade ocorre quando o conteúdo de c_L corresponde a união de conteúdos de vários conceitos globais (*Address* $\equiv$ *Street* $\cup$ *Number* $\cup$ *City*, por exemplo) ou a união de vários conteúdos de um ou mais conceitos globais (value(*Author*) $\equiv$ {value(Name)}, supondo que um conceito *Author* mantém uma lista de nomes de autores)(mapeamento *um-para-muitos* - caso (ii)). Neste caso, c_L é definido também como um conceito léxico global, porém, torna-se uma alternativa especializada (disjunção) não-léxica de c_{un} em relação aos outros relacionamentos de c_{un}[3]. Isto ocorre porque c_L conflitua com a representação de outros conceitos léxicos relacionados a c_{un}. Estes outros relacionamentos de c_{un} tornam-se relacionamentos de um conceito virtual c_V que é definido como uma especialização de c_L.

Dois exemplos de unificação mista são apresentados a seguir. A figura 6.6 mostra o primeiro exemplo, que diz respeito ao cluster composto pelos conceitos *Paper* do esquema local 2 (não-léxico) e *Article* do esquema local 3 (léxico).

Paper é o único conceito não-léxico do cluster. Logo, ele se torna o conceito global representativo do cluster. Supondo que as ocorrências do conceito *Article* sejam *títulos* de artigos, tem-se uma correspondência semântica de conteúdo com o conceito léxico global *Title* associado a *Paper* (caso (i)). Desta forma, *Title* representa tanto títulos de artigos para a DTD3 como o conteúdo léxico de artigos para a DTD2. Na figura 6.6 são mostrados estes mapeamentos.

[3] c_L deve se tornar um conceito não-léxico pois um conceito especializado é sempre não-léxico (ver regra **H** na seção 5.5.2).

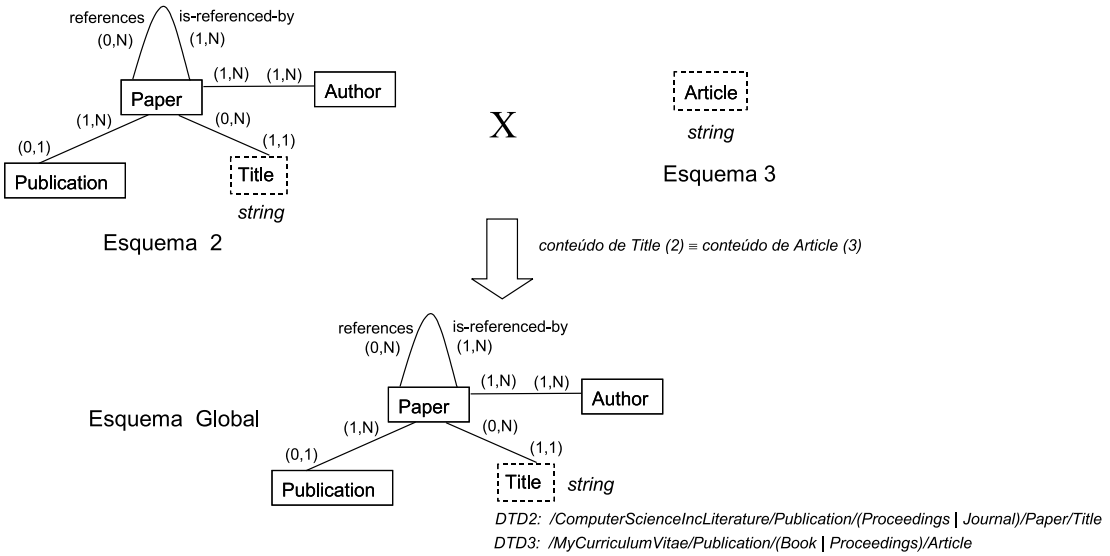


FIGURA 6.6 – Exemplo de unificação mista (I)

A figura 6.7 mostra um segundo exemplo de unificação mista e diz respeito ao cluster composto pelos conceitos *Address* dos esquemas locais 2 (não-léxico) e 3 (léxico).

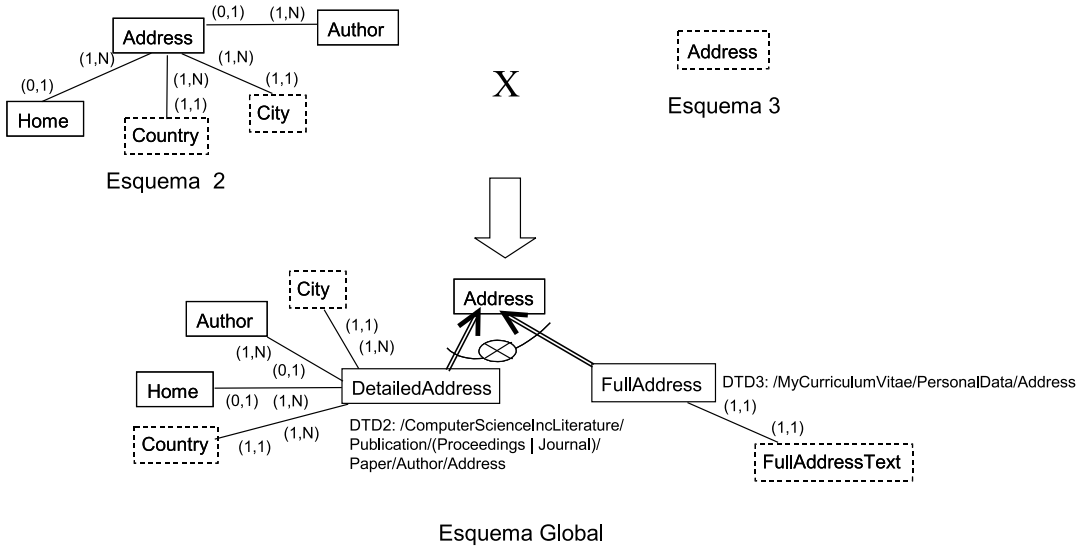


FIGURA 6.7 – Exemplo de unificação mista (II)

Neste exemplo, o usuário decide pelo caso (ii) de unificação pois supõe-se que o conteúdo do conceito *Address* no esquema local 3 é um endereço completo que corresponde a união dos conceitos *Street*, *Number*, *City* e *Country* no esquema local 2. Assim sendo, o usuário representa o conceito léxico *Address* do esquema local 3 como uma alternativa especializada não-léxica do conceito global *Address* chamada *FullAddress*. O conceito *FullAddress* torna-se disjunto do conceito virtual *DetailedAddress* que encapsula os demais relacionamentos de *Address*. O conteúdo

de *FullAddress* é representado pelo conceito léxico associado *FullAddressText*. Os mapeamentos dos conceitos são mostrados na figura 6.7.

Um exemplo para o caso (iii) é mostrado no estudo de caso descrito na seção 6.7 para a unificação do conceito *Journal*.

Vale ressaltar que a representação global de um conceito léxico c_L de um cluster misto pode ser postergada pelo usuário caso nem todos os clusters mistos de conceitos relacionados a c_{un} tenham sido unificados. Isto ocorre quando o usuário não encontra um mapeamento um-para-um para c_L. Pode ser que c_L corresponda a um conceito léxico relacionado a um conceito cujo cluster misto ainda não foi unificado.

6.4.5 Exemplo de Esquema Global Preliminar

Considerando os três esquemas locais exemplo, a unificação destes esquemas gera o esquema global preliminar mostrado na figura 6.8.

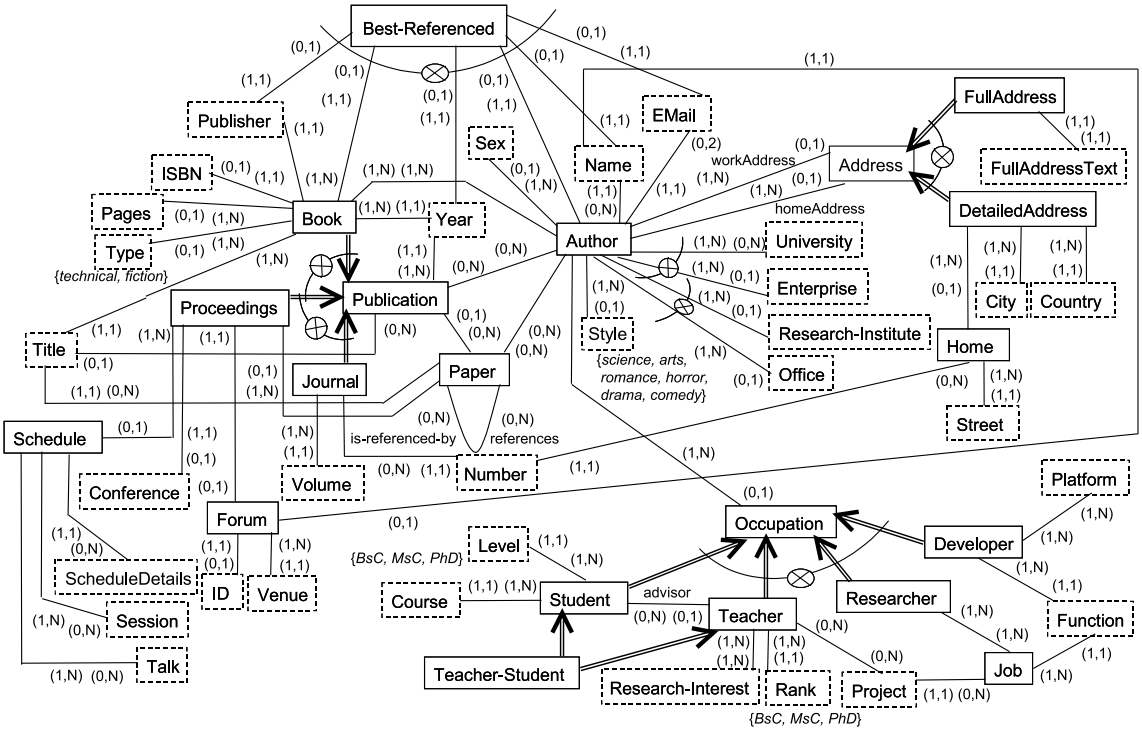


FIGURA 6.8 – Esquema global preliminar exemplo

Os mapeamentos de cada conceito para as fontes XML e os tipos de dados dos conceitos léxicos não são apresentados para facilitar a compreensão do esquema global.

Uma vez criado o esquema global preliminar, iniciam-se as tarefas de ajuste semântico deste esquema. A primeira delas analisa e define novos relacionamentos de herança, sendo realizada pelo passo de *inclusão de relações de herança*, explicado a seguir.

6.5 Passo 3: Inclusão de Relações de Herança

Novos relacionamentos de herança podem existir entre conceitos do esquema global preliminar, uma vez que este esquema é o resultado da integração de conceitos provenientes de diversos esquemas XML. Um conceito c_x, por exemplo, definido nos esquemas XML E_1, ..., E_f, pode ser uma especialização ou generalização de um conceito c_y definido nos esquemas XML E_k, ..., E_m. Como estes conceitos têm origens em conjuntos de esquemas XML disjuntos, estas relações de herança não foram identificadas anteriormente.

A definição de novos relacionamentos de herança no esquema global é interessante pois permite a formulação de consultas a diversos níveis de uma hierarquia de herança. Estas consultas podem ser relevantes para o domínio em questão. Assim sendo, o processo de integração semântica inclui este passo semi-automático de *inclusão de relações de heranca*.

Duas tarefas são realizadas neste passo: a *identificação* de novos relacionamentos de herança entre conceitos do esquema global e a *definição* destes relacionamentos no esquema, caso o usuário confirme que os mesmos são válidos para o domínio em questão.

Este passo realiza novamente a consulta a um Thesaurus (*Wordnet*) para a identificação de relacionamentos de hipernímia (BT) entre pares de nomes de conceitos do esquema global. Pares de conceitos que já possuem relacionamento de herança não necessitam ser verificados.

Após a descoberta de relacionamentos de herança, o usuário especialista deve determinar quais deles serão considerados no esquema global preliminar. Esta validação manual é necessária pois um Thesaurus mantém relacionamentos terminológicos para diversos contextos nos quais um termo pode se enquadrar. Alguns destes contextos não são relevantes para o domínio em questão.

Considerando o esquema global da figura 6.8, um exemplo de relacionamento irrelevante retornado pelo Thesaurus é `<country BT venue>` pois o conceito *country* é encarado no esquema global como um componente do endereço de um autor e não como uma generalização de um local de encontro de um fórum. Já um exemplo de relacionamento considerado relevante é `<forum BT conference>` pois os conceitos *forum* e *conference* dizem respeito ao contexto de encontros científicos dos quais são produzidos *proceedings*.

Relacionamentos de hipernímia retornados por um Thesaurus e considerados relevantes pelo usuário geram relacionamentos de herança no esquema global preliminar. O usuário pode inclusive definir outros relacionamentos de herança, além daqueles advindos do Thesaurus. A definição de tais relacionamentos é regida pela regra **IncRH**, apresentada a seguir.

Regra IncRH (Inclusão de Relacionamentos de Herança). *Dados dois conceitos c_g, $c_e \in NL \cup L$ tal que $<c_g.n\ BT\ c_e.n>$ é um relacionamento relevante, se c_g, $c_e \in NL$ e todos os conceitos associados a c_g estão também associados a c_e com as mesmas restrições de cardinalidade, então gera-se $<c_g, c_e> \in H$. Caso contrário:*

1. *gera-se $C = <n> \in NL$, sendo $C.n$ definido pelo usuário, com $<C.n\ BT\ c_g.n>$ e $<C.n\ BT\ c_e.n>$;*
2. *aplica-se a regra **Her** para $C.n$ e $c_g.n$;*
3. *aplica-se a regra **Her** para $C.n$ e $c_e.n$;*
4. *gera-se $\{<C.n, c_g.n>, <C.n, c_e.n>\} \in D$.*

A regra **IncRH** define um relacionamento direto de herança entre dois conceitos c_g e c_e do esquema global somente se *todos* os conceitos com o qual c_g está associado também estão associados a c_e e ambos são conceitos não-léxicos. Assim, a *semântica do relacionamento de herança* mantém-se válida, ou seja, todos os relacionamentos de associação de c_g são também relacionamentos de associação de c_e. Esta ação cria relacionamentos redundantes em c_e, que serão tratados no passo de *Reestruturação* do esquema global.

Caso c_g possua conceitos associados diferentes de c_e, define-se um conceito C como uma generalização disjunta de c_g e c_e. A existência da disjunção indica que C generaliza conceitos especializados alternativos provenientes de esquemas locais diferentes. A regra **Her**, definida na etapa de *Conversão da DTD* (seção 5.5.2), é utilizada para definir os relacionamentos de herança de C com c_g e c_e, tratando os casos em que c_g e/ou c_e sejam conceitos léxicos.

Um exemplo da aplicação deste segundo caso ocorre para os conceitos *Forum* e *Conference* do esquema global preliminar exemplo, considerando o relacionamento relevante <`Forum BT Conference`>. Este exemplo é mostrado na figura 6.9.

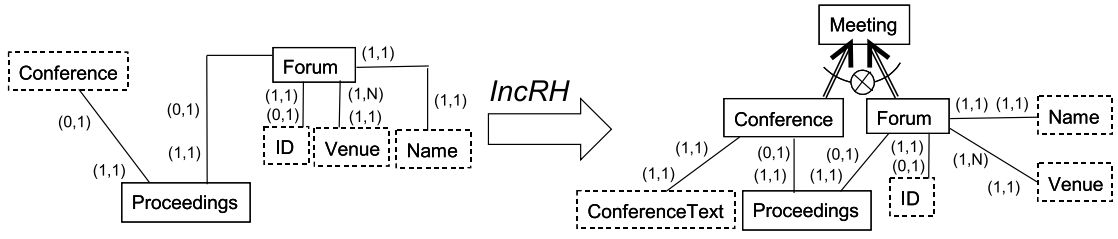


FIGURA 6.9 – Exemplo de inclusão de relacionamento de herança (I)

Forum é um conceito mais geral que *Conference* no domínio. Porém, nem todos os conceitos associados a *Forum*, como *ID* e *Venue*, estão associados a *Conference*. Portanto, não se pode definir *Conference* como uma especialização direta de *Forum* pois *Conference* herdaria relacionamentos que não possui nos esquemas locais em que está definido. Neste caso, define-se então um conceito genérico comum a ambos chamado *Meeting*. O nome deste conceito é escolhido pelo usuário dentre um conjunto de nomes de termos mais gerais que *Forum* e *Conference* disponibilizado pelo Thesaurus. Segundo a regra **Her**, *Conference* torna-se um conceito não-léxico associado ao conceito léxico especial *ConferenceText* que representa o seu conteúdo.

A vantagem da definição desta hierarquia de herança é a consulta integrada a todos os fóruns científicos de publicações presentes nas fontes XML, representado pelo conceito *Meeting*.

Um conceito como *Meeting*, criado como uma generalização de outros conceitos do esquema global, não tem correspondência direta nos esquemas XML. Logo, seu mapeamento para um esquema XML é uma expressão *XPath* com alternativas para todos os elementos que correspondam aos seus conceitos especializados. O mapeamento de *Meeting* para o *esquema conceitual 2*, por exemplo, é dado por `/MyCurriculumVitae/Publication/Proceedings/Conference` pois apenas o conceito *Conference* tem representatividade neste esquema.

Um exemplo fictício para o primeiro caso da regra **IncRH** é mostrado na figura 6.10. Este exemplo considera a existência de um relacionamento terminológico `<Publication BT Manual>` em um Thesaurus.

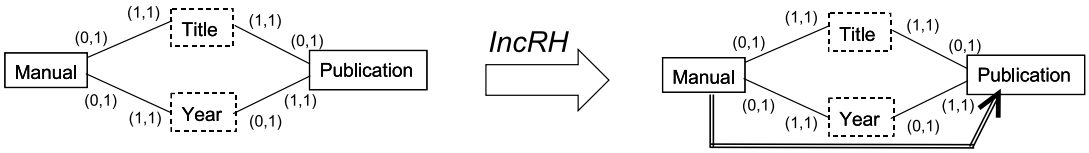


FIGURA 6.10 – Exemplo de inclusão de relacionamento de herança (II)

Manual é considerado uma especialização de *Publication* relevante pelo usuário. Como todos os relacionamentos de associação de *Publication* são comuns a *Manual*, define-se um relacionamento direto de herança entre eles. Os relacionamentos redundantes de associação de *Manual* serão removidos através de uma validação automática efetuada no passo de *Reestruturação*.

Para este caso da regra **IncRH**, o mapeamento do conceito especializado (*Manual*, no exemplo) e de seus relacionamentos permanecem inalterados. O mapeamento do conceito *Publication* considera adicionalmente o mapeamento para o conceito *Manual* em todos os esquemas XML em que *Manual* possui representatividade.

Para o esquema global preliminar exemplo, apenas o relacionamento de herança mostrado na figura 6.9 é efetivado.

Como explicado anteriormente, a geração de relacionamentos de herança provoca alterações no esquema global preliminar que pode exigir validações. As validações finais a serem realizadas no esquema global são detalhadas no passo de *Reestruturação*, explicado a seguir.

6.6 Passo 4: Reestruturação

O passo de *Reestruturação* valida um esquema global preliminar a fim de produzir um esquema global definitivo para um conjunto de esquemas locais. Similar ao passo de *Reestruturação* da etapa de *Conversão da DTD*, este passo realiza ajustes para garantir uma interpretação semântica correta do esquema global. Validações manuais e automáticas são executadas.

6.6.1 Validações Manuais

Na etapa de *Conversão da DTD*, o usuário especialista necessita realizar diversas validações no esquema conceitual preliminar gerado pelo passo de *Conversão*, como ajustes de cardinalidade de relacionamento, tipos de dados e semântica de relacionamentos. Isto ocorre porque uma DTD é uma gramática para estruturação hierárquica de dados XML e não um modelo semântico. Já neste passo, o número de validações manuais para fins de correção semântica do domínio é menor pois o esquema conceitual global respeita as intenções semânticas de cada esquema local.

As únicas validações a serem consideradas pelo usuário referem-se a disjunções de relacionamentos e definição de conceitos genéricos relevantes.

Tratamento de Disjunções

Após a execução dos passos de *Unificação* e *Inclusão de Relações de Herança*, é possível que a atualização de disjunções de relacionamentos seja necessária. Apenas o usuário pode realizar tal tarefa pois somente ele é capaz de identificar se disjunções de relacionamentos de um conceito não-léxico c_{un} devem ser unidas ou definidas, por exemplo, conforme a intenção semântica do domínio. Tal situação é passível de ocorrência uma vez que c_{un} agrega relacionamentos de conceitos não-léxicos provenientes de vários esquemas locais.

Três tipos de atualizações de disjunções de relacionamentos de c_{un} podem ser realizadas pelo usuário:

- *União*: dado um conjunto de disjunções $d_1 = \{r_1, r_2\}$, $d_2 = \{r_2, r_3, r_4\}$, ..., $d_n = \{r_4, r_x\}$, a união destas disjunções produz uma disjunção $d_u = \{r_1, r_2, r_3, r_4, ..., r_x\}$;
- *Inclusão*: dada uma disjunção $d = \{r_1, r_2, ..., r_i\}$ e um relacionamento r_n de c_{un}, a inclusão de r_n em d resulta em $d = \{r_1, r_2, ..., r_i, r_n\}$;
- *Definição*: dado um conjunto de relacionamentos $r_1, r_2, ..., r_n$ de c_{un} que devem ser disjuntos, produz-se uma disjunção $d = \{r_1, r_2, ..., r_n\}$.

Um exemplo de *união* de disjunções ocorre no esquema global preliminar para os relacionamentos do conceito *Publication*. A figura 6.11 mostra este exemplo.

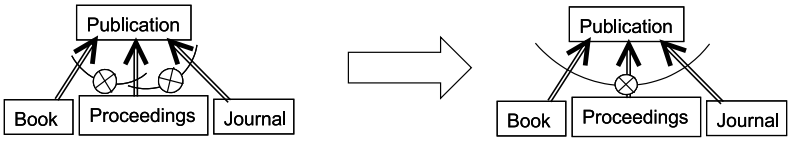


FIGURA 6.11 – Exemplo de união de disjunções

As disjunções definidas para os relacionamentos *Book*, *Proceedings* e *Proceedings*, *Journal* são provenientes de esquemas XML diferentes. Como elas são especializações disjuntas de uma publicação, o usuário as unifica em uma única disjunção.

Considerando novamente o exemplo fictício mostrado na figura 6.10, onde um relacionamento de herança foi definido de um conceito *Publication* para um conceito *Manual*, um exemplo de *inclusão* de relacionamento em uma disjunção é ilustrado

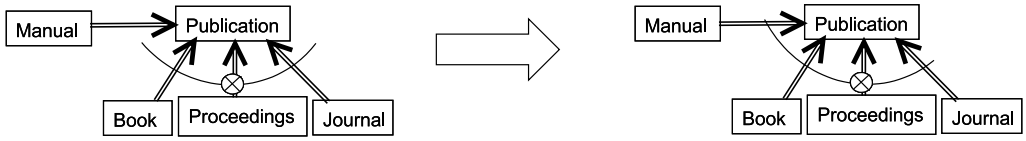


FIGURA 6.12 – Exemplo de inclusão de relacionamento em uma disjunção

na figura 6.12, considerando que *Manual* também é uma especialização disjunta de *Publication*.

Um outro exemplo hipotético, mostrado na figura 6.13, ilustra uma *definição* de disjunção de relacionamentos.

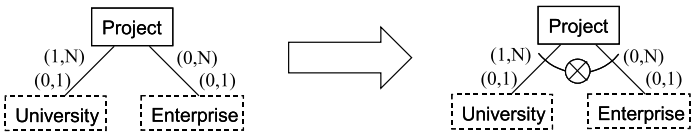


FIGURA 6.13 – Exemplo de definição de disjunção

Supõe-se um conceito *Project* como sendo o resultado da unificação de um conceito referente a projetos de pesquisa de uma Universidade (*University*) em um esquema local e a projetos de pesquisa de uma empresa (*Enterprise*) em outro esquema local. Como uma ocorrência de projeto é sempre ou um projeto universitário ou de empresa, uma disjunção é definida para garantir esta restrição.

Generalização de Conceitos

Da mesma forma que um conceito genérico pode ser definido pelo usuário em um esquema conceitual gerado para uma DTD (ver seção 5.6.1), um conceito genérico que seja relevante para fins de consulta sobre o domínio em questão pode ser definido no esquema global.

Supondo que o conceito *Publication* não estivesse definido em nenhum esquema local, mas os conceitos *Book* e *Journal* sim, em esquemas locais diferentes, existiriam apenas os conceitos representativos de *Book* e *Journal* no esquema global. Se estes conceitos apresentassem relacionamentos com conceitos comuns (*Year* e *Author*, por exemplo), seria interessante definir um conceito *Publication* como uma generalização de ambos. Os relacionamentos comuns tornar-se-iam relacionamentos de *Publication*. Esta validação é semelhante à descrita no exemplo da figura 5.6 (seção 5.6.1) quando da *reestruturação* do esquema conceitual no passo de *Conversão da DTD*.

Modificação de Nomes de Conceitos Especiais

Conceitos especiais podem ser gerados durante os passos de *Unificação* e *Inclusão de Relações de Herança*. Um exemplo foi a geração do conceito léxico *ConferenceText*, definido para representar o conteúdo do conceito *Conference*. O usuário deve fornecer um nome mais adequado para estes conceitos. No caso do conceito *ConferenceText*, ele passa a se chamar *Nome* pois representa o nome de uma conferência. Como *Nome* é um conceito já existente e com o mesmo tipo de dado, apenas um relacionamento entre *Conference* e *Nome* é mantido no esquema global. Caso

houvesse conflito de tipo de dado, o usuário deveria fornecer um nome alternativo ao conceito especial.

Outro exemplo ocorre para o conceito *FullAddressText*, que é renomeado para *Content* pois representa o conteúdo de um endereço.

6.6.2 Validações Automáticas

As validações automáticas são similares às realizadas no passo de *Reestruturação* da etapa de *Conversão da DTD* (ver seção 5.6). Estas validações visam a simplificação do esquema global e o ajuste de cardinalidades de relacionamentos.

Definição de Relacionamentos de Associação Opcionais

A *definição de relacionamentos de associação opcionais* no esquema global segue o mesmo princípio adotado na *reestruturação* do esquema conceitual derivado de uma DTD (ver seção 5.6.3). A motivação para esta validação é a seguinte: um conceito global c_x está associado de forma obrigatória a um conceito global c_y de acordo com um esquema local E_1. Esta associação é obrigatória pois c_x relaciona-se apenas com c_y em E_1. Porém, é possível que c_x esteja associado a outro conceito global c_w de acordo com um esquema local E_2 e c_y não tem conceito correspondente em E_2. Portanto, uma associação obrigatória entre c_x e c_y no esquema global não é válida para uma instância de c_x no esquema E_2. Esta associação deve, portanto, tornar-se opcional.

A regra **COG** é responsável por esta validação. Esta regra é definida a seguir.

Regra COG (Cardinalidade Opcional Global). *Dados os conceitos c_x, c_y, $c_w \in NL \cup L$ tal que $\exists\ r_1 \in A$ entre c_x e c_y com cardinalidade mínima $c_{min} = 1$ no sentido $c_x \rightarrow c_y$ e r_1 é válido para um esquema local E_1. Se $\exists\ r_2 \in A$ entre c_x e c_w tal que r_2 é válido para um esquema local E_2 e c_y não tem conceito correspondente em E_2, então, $c_{min} \leftarrow 0$.*

Um exemplo desta validação ocorre no esquema global preliminar da figura 6.8 para a associação entre os conceitos *Title* e *Book*. *Title* possui um relacionamento obrigatório com *Book* pois *Title* é o único conceito associado a *Book* no *esquema local 1*. Porém, *Title* possui ainda uma associação com o conceito *Paper* pois este relacionamento ocorre no *esquema local 2*. Como *Book* não tem representatividade no *esquema local 2*, uma instância de *Title* no *esquema local 2* não possui este relacionamento. Logo, a cardinalidade da associação *Title*→*Book* torna-se opcional.

A regra **COH** é responsável pela geração de um relacionamento de associação opcional no esquema global. Esta regra é definida a seguir.

Regra COH (Cardinalidade Opcional Global devido a Herança). *Dados um conceito genérico $c_g \in NL$ e um conceito especializado $c_e \in NL$ tal que $\exists\ r_a \in A$ para c_g, se r_a não é válido para c_e um dado esquema local com o qual c_e tem correspondência, então r_a torna-se um relacionamento opcional para c_g.*

Esta regra analisa relacionamentos de herança entre dois conceitos. Um conceito genérico c_g pode ter um relacionamento de associação r_a que não é válido para um conceito especializado c_e em um dado esquema local. Neste caso, r_a deve se tornar opcional para c_g para não impor que r_a seja obrigatório para c_e.

Remoção de Relacionamentos Redundantes

A remoção de relacionamentos redundantes presentes no esquema global é regida pelas regras **RRA** e **RRH**, definidas no passo de *Reestruturação* da etapa de *Conversão da DTD* (seção 5.6.2). Elas são responsáveis, respectivamente, pela remoção de relacionamentos de associação e herança redundantes.

Relacionamentos redundantes podem ocorrer no esquema global preliminar como conseqüência da integração de conceitos envolvidos em relacionamentos de herança ou da inclusão de novos relacionamentos de herança. Por exemplo, um conceito c_e que possui uma associação com um conceito c_y pode se tornar uma especialização de um conceito c_g proveniente de outro esquema XML que também se relaciona com c_y. A associação entre c_e e c_y torna-se redundante pois c_x agora a herda de c_g. Esta situação ocorre no esquema global preliminar exemplo. Ela é mostrada na figura 6.14.

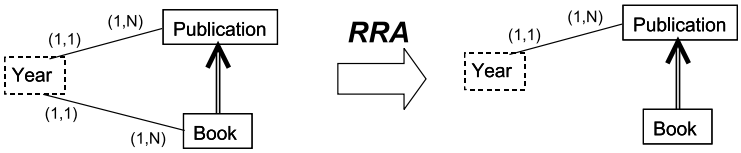


FIGURA 6.14 – Exemplo de remoção de relacionamento redundante

Neste caso, *Book* é o conceito c_e, *Year* é o conceito c_y e *Publication* é o conceito c_g. Cabe ressaltar que, caso a associação entre c_e e c_y tenha cardinalidades mais gerais que as cardinalidades da associação entre c_g e c_y, estas cardinalidades gerais migram para a associação c_g-c_y.

Generalização de Relacionamentos

A generalização de relacionamentos de associação presentes no esquema global é regida pela regra **GR** definida no passo de *Reestruturação* da etapa de *Conversão da DTD* (seção 5.6.2). Relacionamentos a serem generalizados podem ocorrer no esquema global preliminar como conseqüência da inclusão de relacionamentos de herança pelo passo anterior. Um exemplo é mostrado na figura 6.15 para o conceito *Meeting*, criado como generalização dos conceitos *Forum* e *Conference*.

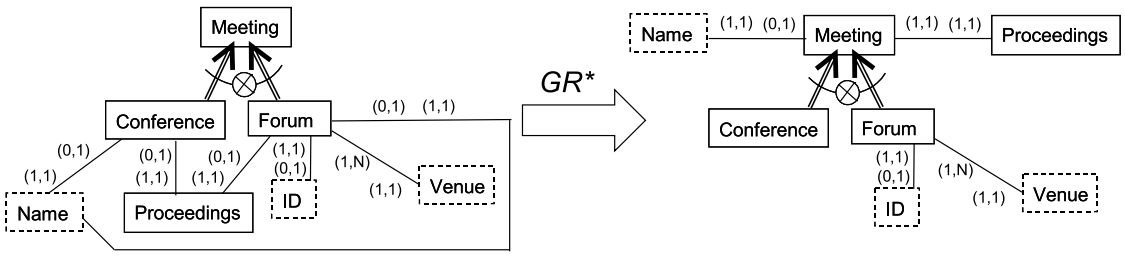


FIGURA 6.15 – Exemplo de generalização de relacionamentos

Neste exemplo, os relacionamentos com os conceitos *Proceedings* e *Name* são generalizados por estarem associados tanto com *Forum* quanto com *Conference* e

apresentarem as mesmas restrições de cardinalidade. Uma variante em relação a regra **GR** é a seguinte: se o conceito destino da associação tem relacionamento com algum dos conceitos especializados em todos os esquemas locais, ele se torna um relacionamento obrigatório com o conceito genérico. Tal situação ocorre neste exemplo para o conceito *Proceedings*.

6.6.3 Exemplo de Esquema Global Definitivo

Para o esquema global preliminar exemplo (figura 6.8), a execução do passo de *Reestruturação* produz o *esquema global definitivo* mostrado na figura 6.16.

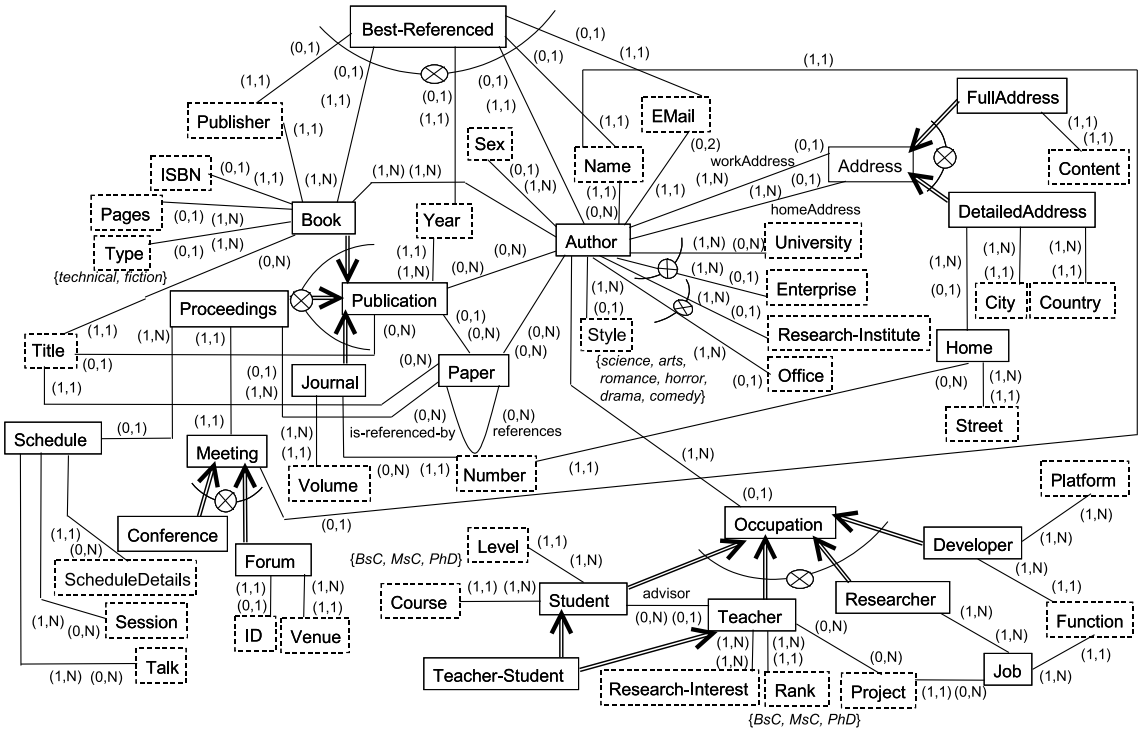


FIGURA 6.16 – Esquema global definitivo exemplo

Com a criação do esquema global definitivo, o processo de integração de esquemas XML proposto por BInXS é concluído. Este esquema global torna-se uma referência para a consulta integrada a dados de fontes XML que definem estes esquemas. Este esquema global não necessariamente é sempre absoluto pois novos esquemas XML neste mesmo domínio poderão ser integrados a ele através da reexecução do processo, sendo ele um dos esquemas de entrada.

Uma vez apresentado todo o processo de integração, um estudo de caso é sucintamente discutido antes do término deste capítulo.

6.7 Estudo de Caso

Um estudo de caso foi realizado para validar o processo de integração de BInXS. Este estudo de caso considerou três fontes XML para um domínio bibliográfico:

- *SIGMOD*: DTD para os periódicos do *SIGMOD Record* e um grande documento XML contendo exemplares de 1981 a 1999 [SIG 99];
- *DB&LP*: DTD para a bibliografia em Ciência da Computação criada pela Universidade de Trier, Alemanha. Um grande número de documentos XML organizados por categorias de publicação é também mantido [DBL 2002];
- *bibTeXML*: DTD para o formato de especificação de referências bibliográficas *BibTeX* utilizado pelo sistema de preparação de documentos *LaTeX*. Esta DTD e um documento XML exemplo foram desenvolvidos pela Faculdade de Ciências da Universidade de Amsterdam,Holanda [BIB 2002].

Os resultados deste estudo de caso encontram-se no anexo 2, onde são apresentadas as *DTDs pré-processadas*, os *esquemas conceituais definitivos* para cada DTD, os *clusters de afinidade* e o *esquema global definitivo*.

Uma análise realizada sobre os resultados do processo foi a seguinte:

1. Etapa de *Conversão da DTD*:
   - *DTD SIGMOD*: foi criado um *esquema definitivo* com 10 conceitos e 9 relacionamentos (19 componentes). Em relação ao *esquema preliminar*, foram atualizados 2 conceitos (alteração de nome) e 2 relacionamentos (mudança de cardinalidade). Para 19 componentes do esquema, 4 sofreram alteração(21%). Obteve-se então um *esquema preliminar* gerado automaticamente com 79% de precisão;
   - *DTD DB&LP*: *esquema definitivo* com 59 componentes (30 conceitos e 29 relacionamentos). O *esquema preliminar* teve 10 conceitos atualizados (alteração de nome, criação de conceito genérico e conceitos com relacionamentos generalizados) e 12 relacionamentos (alteração de cardinalidade e de tipo). Para 59 componentes, 22 sofreram alteração (37%). *Esquema preliminar* com 63% de precisão;
   - *DTD BibTeXML*: *esquema definitivo* com 373 componentes (52 conceitos e 321 relacionamentos). O *esquema preliminar* teve 18 conceitos atualizados (alteração de nome, criação de conceitos genéricos, conceitos com relacionamentos generalizados e um conceito com relacionamentos modificados) e 135 relacionamentos modificados (alteração de cardinalidade e de tipo e remoção). Para 373 componentes, 153 sofreram alteração (41%). *Esquema preliminar* com 59% de precisão.
2. Etapa de *Integração Semântica*:
   (a) *Agrupamento de Sinônimos*(ver anexo 2): para os *esquemas conceituais* correspondentes às três DTDs, foram gerados automaticamente 55 *clusters de afinidade*, sendo 3 incorretos (94,5% de corretude). Deveriam ter sido gerados 61 clusters corretos. Como foram gerados 52, obteve-se $\cong$ 85% de precisão;
   (b) *Unificação e Reestruturação*: os procedimentos automáticos executados nestas fases produziram praticamente o *esquema global definitivo*. As únicas ações manuais efetuadas foram (ver esquema global no anexo 2):

- *Unificação do conceito Author*: *Author* no esquema *SIGMOD* é não-léxico. Logo, uma representação global não-léxica foi criada automaticamente. *Author* no esquema *DP&LP* corresponde ao nome do autor. Logo, ele foi mapeado para o conceito *Nome* associado a *Author*. *Author* no esquema *BibTeXML* corresponde a uma lista de nomes de autores. Como não é possível uma correspondência *um-para-um* com o conceito *Nome*, uma representação alternativa (conceito *AuthorNames*) foi criada;
- *Unificação do conceito Journal*: Um conceito global não-léxico foi criado automaticamente, correspondente à representação no esquema *SIGMOD*. *Journal* nos esquemas *DP&LP* e *BibTeXML* mantém o título do periódico. Como não existe conceito global equivalente, o conceito *Journal_ Title* foi criado e associado a *Journal*;
- *União de disjunções do conceito Publication*: *Publication* nos esquemas *DP&LP* e *BibTeXML* possui uma disjunção para as suas especializações. Como no nível global todas as especializações de *Publication* continuam sendo disjuntas, estas disjunções foram unificadas em uma só disjunção.

Os resultados desta análise foram considerados positivos. Na etapa de *Conversão da DTD* obteve-se uma precisão automática mínima em torno de 60% na geração dos esquemas conceituais (esta precisão tende a ser menor quanto maior for o tamanho da DTD). Na etapa de *Integração Semântica* obteve-se uma boa precisão na determinação de equivalências semânticas e, conseqüentemente, uma unificação automática que exigiu pouca correção do usuário.

Obviamente, a boa precisão automática do processo de integração deve-se ao fato dos esquemas XML pertencerem ao mesmo domínio e seus elementos possuírem uma terminologia e propriedades com afinidade. Acredita-se que esta limitação não compromete a qualidade do processo pois quando se deseja integrar informações de esquema, geralmente estas informações pertencem ao mesmo domínio.

7 Conclusão

Esta tese apresenta uma abordagem para a *integração semântica de esquemas* de dados semi-estruturados chamada *BInXS* . Em particular, *esquemas XML* são considerados, dado que XML é um padrão atual para representação e intercâmbio de dados semi-estruturados entre aplicações e usuários na *Web*. O gerenciamento de dados semi-estruturados e XML é um tópico atual de pesquisa na comunidade de banco de dados. Este fato justifica a relevância do tema da tese.

Este capítulo apresenta as conclusões e alguns comparativos com respeito a BInXS. Todos estes pontos têm por objetivo salientar as suas contribuições e alguns trabalhos futuros mais significativos.

7.1 Técnicas de Integração de Esquemas

O capítulo 2 mostra uma *taxonomia de técnicas de integração* de esquemas que podem ser alternativas ou complementares, dependendo do caso. BInXS se enquadra nas seguintes técnicas:

1. O *processo de integração* adotado é *bottom-up*. Esta técnica é a mais adequada quando se deseja obter um esquema global como resultado. Este esquema, por ser construído a partir dos esquemas das fontes de dados, considera de forma direta a semântica destes esquemas;

2. O *processo de unificação* é *n-ário* no seu contexto geral pois permite vários esquemas como entrada para o processo. Reduz-se, com isso, o número de passos de unificação. Já no contexto específico de unificação de conceitos não-léxicos, utiliza-se uma técnica de unificação *binária* para reduzir a complexidade de resolução de conflitos e construção de conceitos globais. Esta técnica foi adotada para conceitos não-léxicos pois estes conceitos possuem uma representação complexa que envolve relacionamentos e disjunções;

3. Quanto a *resolução de conflitos*, consideram-se tanto os casos de conflitos de *nome* como de *estrutura*. Conflitos de nome resolvem problemas de *sinonímia* e conflitos de estrutura resolvem problemas de heterogeneidade de *construtores de modelagem* (conceito não-léxico vs. léxico), *propriedades* (tipos de dados e enumerações para conceitos léxicos) e *relacionamentos* (unificação de relacionamentos com afinidade). Conflitos de *identificação* e *comportamento* não se aplicam a esquemas XML;

4. Quanto a *determinação de equivalências semânticas*, consideram-se técnicas baseadas em *linguística* (nomenclatura) e em *restrições* (tipos de dados e enumerações para conceitos léxicos; relacionamentos e disjunções para conceitos não-léxicos). Utiliza-se ainda uma técnica baseada em *esquema* pois apenas informações sobre os esquemas a serem integrados são analisados para a determinação de equivalências semânticas. Técnicas baseadas em instâncias não são utilizadas devido a sua complexidade.

Para os itens 3 e 4, quanto mais técnicas são empregadas, maior é a qualidade e o grau de automação do processo de integração. BInXS suporta todas as técnicas

sugeridas, exceto a técnica de integração baseada em *instâncias* para o item 4. Tal técnica exige procedimentos complexos, como processamento de linguagem natural, a serem incluídos no processo de integração. A implementação destes procedimentos deve ser eficiente para não comprometer o desempenho do processo. A aplicação desta técnica pode ser considerada como trabalho futuro.

7.2 Contribuições

BInXS representa um diferencial em relação as abordagens de integração de esquemas tradicionais de bancos de dados pois se aplica a esquemas semi-estruturados XML. Em particular, ela traz as seguintes contribuições:

- *Um processo de conversão de uma DTD para uma representação conceitual canônica.* Este processo realiza uma análise detalhada do modelo de uma DTD a fim de obter um esquema conceitual correspondente que seja semanticamente preciso. A representação conceitual modela todos os tipos de elementos (compostos, `#PCDATA`, mistos, etc) e atributos; define relacionamentos semânticos de associação (com cardinalidades direta e inversa) e herança entre elementos e atributos; e modela representações alternativas através de disjunções de relacionamentos;

- *Um processo de integração semântica de esquemas XML.* Este processo considera a resolução de conflitos particulares a dados XML. Estes conflitos são inerentes a unificação de informações léxicas e não-léxicas com seus relacionamentos e disjunções associadas;

- *Uma estratégia para a definição de mapeamentos entre um esquema global e esquemas XML.* O padrão *XPath* para acesso a dados XML é utilizado para a definição de informações de mapeamento, facilitando o processo de tradução de uma consulta global para uma consulta sobre um esquema XML.

A seção 2.3 apresenta trabalhos relacionados a integração de esquemas semi-estruturados e um comparativo dos mesmos com relação a estratégias de integração de esquemas e modelo canônico adotado. O atendimento destes critérios por BInXS é mostrado nas tabelas 7.1, 7.2 e 7.3.

TABELA 7.1 – Comparação de BInXS com os trabalhos relacionados (I)

	TSIMMIS	Garlic	MIX	Xyleme	LSD	DIXSE	**BInXS**
Semi-Automático (A) ou Manual (M)	M	M	M	A	A	M	A
Análise de nomenclatura				√	√		√
Analise estrutural				√	√		√
Informações do esquema	√	√	√	√		√	√
Informações das instâncias					√		√
Esquema global(G) ou visões integradas (V) ou mapeamentos entre esquemas locais (M)	V	G	V	G	G	G	G
Top-down (T) ou Bottom-up (B)	B	B	B	T	T	B	B
Processo de Integração	n-ário	n-ário	n-ário	binário	binário	n-ário	n-ário
Modelo canônico	grafo	ODMG	DTD	DTD	DTD	conceitual	conceitual

TABELA 7.2 – Comparação de BInXS com os trabalhos relacionados (II)

	YAT	*McBrien*	*Vdovjak*	*Jensen*	*Lim*	**BInXS**
Semi-Automático (A) ou Manual (M)	M	M	M	M	A	A
Análise de nomenclatura					√	√
Analise estrutural						√
Informações do esquema	√	√	√	√	√	√
Informações das instâncias						√
Esquema global(G) ou visões integradas (V) ou mapeamentos entre esquemas locais (M)	M	M	G	G	G	G
Top-down (T) ou Bottom-up (B)	-	-	T	B	B	B
Processo de Integração	binário	binário	binário	?	binário	n-ário
Modelo canônico	árvore	grafo	conceitual	UML	grafo	conceitual

TABELA 7.3 – Comparação de BInXS com os trabalhos relacionados (III)

	CUPID	MOMIS	**BInXS**
Semi-Automático (A) ou Manual (M)	A	A	A
Análise de nomenclatura	√	√	√
Analise estrutural	√	√	√
Informações do esquema	√	√	√
Informações das instâncias			√
Esquema global(G) ou visões integradas (V) ou mapeamentos entre esquemas locais (M)	G	G	G
Top-down (T) ou Bottom-up (B)	B	B	B
Processo de Integração	binário	n-ário	n-ário
Modelo canônico	conceitual	conceitual	conceitual

Comparando com os trabalhos relacionados, verifica-se que o *processo de integração* proposto por BInXS é o mais completo com relação ao atendimento das estratégias de integração (linhas 2 a 5 das tabelas). Em particular, BInXS é a única abordagem que utiliza não só informação do esquema mas também de documentos XML (instâncias) para a integração de esquemas XML. Além disso, BInXS é a única abordagem conhecida que propõe um tratamento detalhado de conflitos para a alta heterogeneidade das representações semi-estruturadas, conforme salientado nas contribuições.

Com relação ao *modelo canônico* adotado, nenhum trabalho relacionado apresenta uma modelagem conceitual que considera todas as peculiaridades do modelo de uma DTD como BInXS. Uma categoria de trabalhos, como *YAT* e *TSIMMIS*, utilizam estruturas de dados em árvore ou grafo para representar uma DTD, mas não realizam nenhuma inferência sobre a semântica dos seus dados a fim de obter um esquema conceitual correspondente. Outra categoria, formada por *MOMIS CUPID*, *DIXSE* e o trabalho de *Vdokjak* , propõe representações conceituais para dados semi-estrturados. Porém, estas representações ou são limitadas ao nível de construtores de modelagem ou definem o esquema conceitual manualmente ou não analisam em detalhe o esquema de uma DTD para fins de modelagem conceitual.

MOMIS e *CUPID* são os trabalhos mais completos para integração de esquemas XML. Mesmo assim, BInXS supre as suas limitações, que são o suporte à integração de disjunções de relacoinamentos e a consideração de diversas alternativas

para a integração do caso NLxL.

A *estratégia de mapeamento* adotada permite a tradução direta de uma expressão de consulta global em *CXPath* para uma consulta *XPath* equivalente a ser executada em uma fonte XML. Abordagens de tradução encontradas em trabalhos relacionados executam atividades mais complexas de tradução, que envolve análise de visões individuais para conceitos globais ou de esquemas de mapeamento para a montagem da consulta local.

7.3 Trabalhos Futuros

BInXS apresenta algumas limitações que podem ser tratadas e pode também sofrer algumas extensões. Estas atividades constituem trabalhos futuros que são descritos a seguir:

- *integração de esquemas descritos em XML-Schema.* BInXS lida com esquemas XML descritos através de DTDs. A DTD foi o primeiro padrão para especificação de esquemas XML e grande parte dos documentos XML definem esquemas através de DTDs. Por isso, a escolha deste padrão pela tese. Porém, *XML-Schema* [XML 2002] já é uma recomendação da W3C para definição de esquemas XML e possui mais recursos de modelagem que uma DTD, como tipos de dados básicos e derivação de tipos. O seu uso tende a aumentar. Portanto, BInXS deve prever futuramente a conversão de um esquema em *XML-Schema* para uma representação conceitual. Esta conversão tende a ser mais simples que a conversão de uma DTD pelo fato de *XML-schema* ter recursos de modelagem adicionais que facilitam a determinação de tipos de dados para conceitos léxicos e relacionamentos de herança a partir de tipos derivados;

- *consulta a vários thesauri para a análise de relacionamentos terminológicos.* BInXS utiliza apenas o *Wordnet* para a consulta a relacionamentos de sinonímia e hipernímia por este ser um dos thesauri mais completos da língua inglesa disponível para acesso eletrônico. Porém, outros thesaurus poderiam ser também consultados para melhorar a qualidade da análise de terminologia do domínio. Thesaurus específicos para um domínio poderiam ser também criados para facilitar processos de integração de fontes de dados deste domínio. A ferramenta ARTEMIS permite a inclusão de thesaurus específicos para serem consultados durante a determinação de afinidades;

- *uso de técnicas baseadas em instâncias.* Como já salientado, a única técnica de integração que não é utilizada nesta tese para a determinação de equivalências semânticas é a *técnica baseada em instâncias.* BInXS pode ser estendida para suportar esta técnica, com o objetivo de aumentar o grau de automação do processo. Isto requer um estudo das técnicas existentes e a implementação daquela que for mais eficiente para a análise de documentos XML;

- *tratamento de restrições de integridade.* Além de informações de mapeamento, informações sobre *restrições de integridade* para conceitos léxicos em determinadas fontes XML poderiam ser catalogadas. Para o estudo de caso mostrado anteriormente, o conceito *Title* associado a *Journal* no esquema global, por

exemplo, poderia ter uma restrição *"Title = "Sigmod Record"* para a fonte *SIGMOD* pois apenas instâncias XML deste periódico são mantidas. Para um módulo processador de consultas, tais informações seriam úteis para a otimização de processamento: se um conceito léxico a ser pesquisado não apresentasse os valores desejados pela consulta em uma dada fonte XML, uma tradução desta consulta para esta fonte não seria realizada;

- *otimização dos algoritmos utilizados no processo de integração.* Os algoritmos utilizados no processo de integração não passaram por uma análise para medir a sua complexidade. Certos algoritmos, relacionados a análise de documentos XML e reestruturação de esquemas, lidam com um volume grande de dados e devem estar o mais otimizado possível para não comprometer o desempenho do processo de integração. Tal análise de complexidade e otimização deve ser realizada;

- *realização de mais estudos de caso para revisão do processo de integração.* Novos estudos de caso devem ser realizados para testar a aplicabilidade do processo de integração e eventualmente realizar manutenções nas suas regras e algoritmos. A *geração de relacionamentos de herança* no esquema global é um exemplo de tarefa que deve ser revista pois é possível que relacionamentos de herança gerados para pares de conceitos globais possam ser unificados se existir afinidade entre os conceitos genéricos definidos;

- *desenvolvimento de um protótipo completo para o processo de integração.* Atualmente existe um protótipo implementado em *Java* que realiza a conversão de uma DTD para um esquema conceitual. Este protótipo deve ser continuado para suportar todo o processo de integração. Uma *interface Web* deve ser também desenvolvida para tornar BInXS disponível na *Internet*.

7.4 Considerações Finais

Esta tese é uma proposta de solução para a problemática atual de integração de esquemas semi-estruturados e XML. Ela representa uma contribuição para a pesquisa e desenvolvimento de abordagens efetivas e detalhadas para a integração deste tipo de dado. Alguns trabalhos relacionados são encontrados na literatura, porém, são soluções incompletas pois não consideram todas as particularidades do modelo de uma DTD. Esta tese leva em conta estas particularidades, visando a definição de um esquema integrado que considera de forma mais precisa a intenção semântica dos esquemas XML.

Integrar dados semi-estruturados como XML significa tratar a alta heterogeneidade de representação existente nos seus esquemas de dados. Esta heterogeneidade exige que abordagens mais liberais de integração sejam adotadas para contemplar todas as representações definidas para uma mesma informação, evitando que o esquema de alguma fonte de dados fique sem representatividade no nível global. Esta tese define um processo *bottom-up* de integração de esquemas que vai ao encontro deste requisito, ou seja, considera a união de todas as representações de uma mesma informação em todas as fontes XML a serem integradas.

Além disso, esta tese adota um *modelo conceitual* para a definição de esquemas canônicos para os esquemas XML e do esquema global. Uma representação

conceitual abstrai detalhes de estruturação dos dados ao nível lógico (como a hierarquia de uma DTD), modelando uma informação XML apenas em função do seu nome e dos seus relacionamentos com outras informações. Tal representação facilita a determinação de equivalências entre informações no momento da integração e a definição de um esquema global que abstrai diversos esquemas XML.

A abordagem de integração proposta, por ser semi-automática, garante sempre o resultado desejado para o esquema global, graças ao acompanhamento de um usuário especialista. Espera-se que, com o avanço da tecnologia em Informática, seja possível cada vez mais automatizar o processo de integração semântica de esquemas de modo a se obter uma precisão automática sempre melhor do resultado.

Anexo 1 Definição das Classes de Metadados do MCC em DAML+OIL

```
<!------- Definição em DAML+OIL dos Metadados Referentes ao CCM ------->


<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:daml="http://www.w3.org/2001/10/daml+oil#"
xmlns="http://www.w3.org/2001/10/daml+oil#">

<!------- Definição dos tipos de dados de CCM permitidos para conceitos léxicos ------->

<simpleType name = "CCMDataTypes">
  <restriction base = "string">
    <enumeration value = "string"/>
    <enumeration value = "integer"/>
    <enumeration value = "character"/>
    <enumeration value = "float"/>
    <enumeration value = "date"/>
    <enumeration value = "time"/>
    <enumeration value = "datetime"/>
  </restriction>
</simpleType>

<!------- Definição de classes e propriedades ------->

<!------- Classes de Conceitos ------->

<Class ID = "Concept" />

<ObjectProperty ID = "RelatedConcepts">
  <domain resource = "#Concept"/>
  <range resource = "#Relationship"/>
</ObjectProperty>

<ObjectProperty ID = "ConceptMappings">
  <domain resource = "#Concept"/>
  <range resource = "#ConceptMapping"/>
</ObjectProperty>

<Class ID = "NonLexicalConcept">
  <subclassOf resource = "#Concept"/>
  <disjointWith resource = "#LexicalConcept"/>
</Class>

<Class ID = "LexicalConcept">
  <subclassOf resource = "#Concept"/>
  <disjointWith resource = "#NonLexicalConcept"/>
  <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#DataType"/>
    </restriction>
  </subclassOf>
</Class>

<ObjectProperty ID = "DataType">
  <domain resource = "#LexicalConcept"/>
  <range resource = "#CCMDataTypes"/>
</ObjectProperty>

<ObjectProperty ID = "ValueConstraints">
  <domain resource = "#LexicalConcept"/>
  <range resource = "#ValueConstraint"/>
</ObjectProperty>

<Class ID = "EnumeratedLexicalConcept">
  <subclassOf resource = "#LexicalConcept"/>
</Class>
```

```
<ObjectProperty ID = "AllowedValues">
  <domain resource = "#EumeratedLexicalConcept"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>


<!------- Classes de Mapeamentos ------->

<Class ID = "Mapping">
  <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#Source"/>
    </restriction>
  </subclass>
</Class>

<ObjectProperty ID = "Source">
  <domain resource = "#Mapping"/>
  <domain resource = "#ValueConstraint"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#URI"/> (confirmar tipo URI)
</ObjectProperty>

<Class ID = "ConceptMapping">
  <subclassOf resource = "#Mapping"/>
  <disjointWith resource = "#RelationshipMapping"/>
  <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#PathExpression"/>
    </restriction>
  </subclass>
</Class>

<ObjectProperty ID = "PathExpression">
  <domain resource = "#ConceptMapping"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

<Class ID = "RelationshipMapping">
  <subclassOf resource = "#Mapping"/>
  <disjointWith resource = "#ConceptMapping"/>
  <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#DirectMapping"/>
      <onProperty resource = "#InverseMapping"/>
    </restriction>
  </subclass>
</Class>

<ObjectProperty ID = "DirectMapping">
  <domain resource = "#RelationshipMapping"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

<ObjectProperty ID = "InverseMapping">
  <domain resource = "#RelationshipMapping"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>


<!------- Classes de Relacionamentos ------->

<Class ID = "Relationship">
  <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#SourceConcept"/>
      <onProperty resource = "#TargetConcept"/>
    </restriction>
  </subclassOf>
</Class>

<ObjectProperty ID = "SourceConcept">
  <domain resource = "#Relationship"/>
  <range resource = "#Concept"/>
```

```
  <inverseOf resource = "#RelatedConcepts"/>
</ObjectProperty>

<ObjectProperty ID = "TargetConcept">
  <domain resource = "#Relationship"/>
  <range resource = "#Concept"/>
  <inverseOf resource = "#RelatedConcepts"/>
</ObjectProperty>

<ObjectProperty ID = "RelationshipMappings">
  <domain resource = "#Relationship"/>
  <range resource = "#RelationshipMapping"/>
</ObjectProperty>


<Class ID = "InheritanceRelationship">
  <subclassOf resource = "#Relationship"/>
  <disjointWith resource = "#AssociationRelationship"/>
</Class>


<Class ID = "AssociationRelationship">
  <subclassOf resource = "#Relationship"/>
  <disjointWith resource = "#InheritanceRelationship"/>
   <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#DirectCardinality"/>
      <onProperty resource = "#InverseCardinality"/>
    </restriction>
  </subclassOf>
</Class>

<ObjectProperty ID = "DirectCardinality">
  <domain resource = "#AssociationRelationship"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

<ObjectProperty ID = "InverseCardinality">
  <domain resource = "#AssociationRelationship"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

<Class ID = "NamedAssociationRelationship">
  <subclassOf resource = "#AssociationRelationship"/>
  <disjointWith resource = "#AssociationRelationshipWithRoles"/>
   <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#Name"/>
    </restriction>
  </subclassOf>
</Class>

<ObjectProperty ID = "Name">
  <domain resource = "#NamedAssociationRelationship"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

<Class ID = "AssociationRelationshipWithRoles">
  <subclassOf resource = "#AssociationRelationship"/>
  <disjointWith resource = "#NamedAssociationRelationship"/>
   <subclassOf>
    <restriction cardinality = "1">
      <onProperty resource = "#SourceRole"/>
      <onProperty resource = "#TargetRole"/>
    </restriction>
  </subclassOf>
</Class>

<ObjectProperty ID = "SourceRole">
  <domain resource = "#AssociationRelationshipWithRoles"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>
```

```
<ObjectProperty ID = "TargetRole">
  <domain resource = "#AssociationRelationshipWithRoles"/>
  <range resource = "http://www.w3.org/2000/10/XMLSchema#string"/>
</ObjectProperty>

</rdf:RDF>
```

Anexo 2 Estudo de Caso

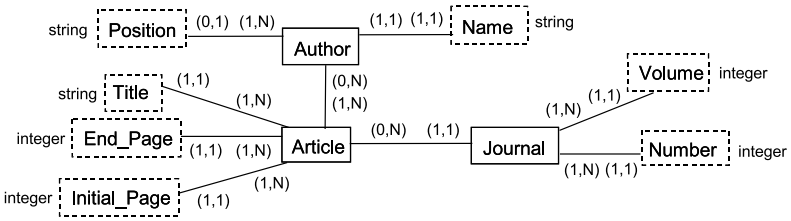
```
<!----------------- DTD: Sigmod Record ----------------->

<!ELEMENT journal (volume,number,article*) >
<!ELEMENT volume (#PCDATA)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT article (title,initial_page,end_page,author*) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT initial_page (#PCDATA)>
<!ELEMENT end_page (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author position CDATA #IMPLIED>
```

```
<!----------------------- DTD: DB&LP ----------------------->

<!ENTITY   % properties "author*, editor*, title*, book_title*,
               pages*, year*, address*, journal*, volume*, number*,
               month*, url*, electronic_edition*, cd_rom*, cite*,
               publisher*, note*, isbn*, series*, school*, chapter*" >

<!ELEMENT in_proceedings (%properties;)>
<!ELEMENT book (%properties;)>
<!ELEMENT proceedings (%properties;)>
<!ELEMENT article (%properties;)>
<!ELEMENT in_collection (%properties;)>
<!ELEMENT phd_thesis (%properties;)>
<!ELEMENT master_thesis (%properties;)>
<!ELEMENT home_page (%properties;)>

<!ELEMENT author             (#PCDATA)> <!ELEMENT editor  (#PCDATA)>
<!ELEMENT address            (#PCDATA)> <!ELEMENT title   (#PCDATA)>
<!ELEMENT book_title         (#PCDATA)> <!ELEMENT pages   (#PCDATA)>
<!ELEMENT year               (#PCDATA)> <!ELEMENT journal (#PCDATA)>
<!ELEMENT volume             (#PCDATA)> <!ELEMENT number  (#PCDATA)>
<!ELEMENT month              (#PCDATA)> <!ELEMENT url     (#PCDATA)>
<!ELEMENT electronic_edition (#PCDATA)> <!ELEMENT cd_rom  (#PCDATA)>
<!ELEMENT cite               (#PCDATA)> <!ELEMENT school  (#PCDATA)>
<!ELEMENT publisher          (#PCDATA)> <!ELEMENT note    (#PCDATA)>
<!ELEMENT chapter            (#PCDATA)> <!ELEMENT isbn    (#PCDATA)>
<!ELEMENT series  (#PCDATA)>
```

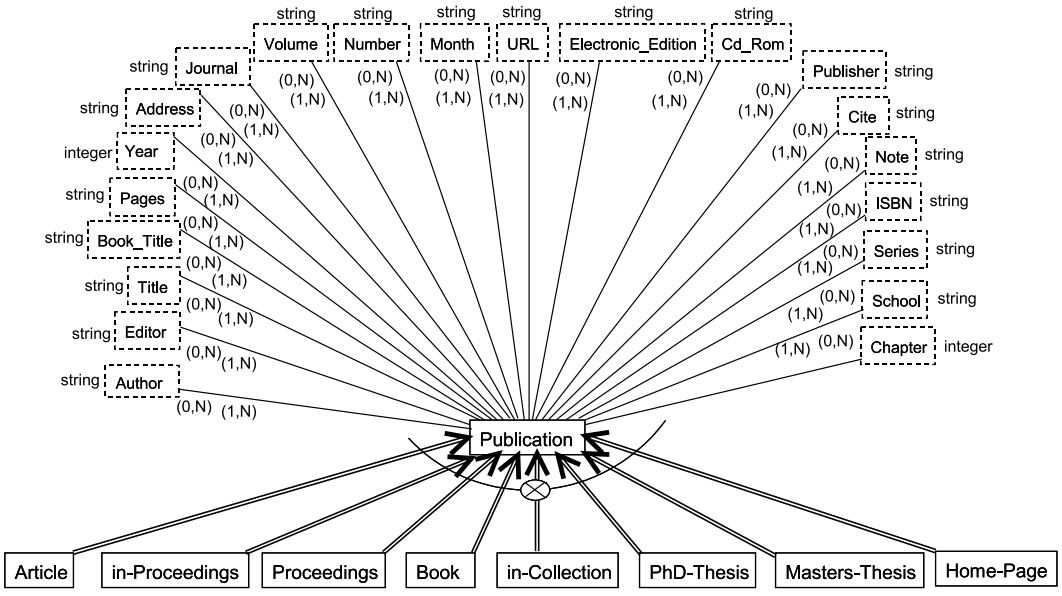
FIGURA B.1 – DTDs pré-processadas(I)

```
<!---------------------------------- DTD: BibTeXML ---------------------------------->

<!ELEMENT publication (article | book | booklet | manual | technical_report |
   masters_thesis | phd_thesis | in_book | in_collection | proceedings | in_proceedings
   | conference | unpublished | miscellaneous)>

<!ATTLIST publication id ID #REQUIRED>

<!ENTITY % commonProperties "abstract?, affiliation?, contents?, copyright?, isbn? |
   issn?, keywords?, language?, location?, price?, size?, url?, month?, category?" >

<!ELEMENT article (author, title, journal, year, volume?, number?, pages?, note?,
   %commonProperties;)>

<!ELEMENT book (author | editor, title, publisher, year, volume? | number?, series?,
   address?, edition?, note?, %commonProperties;)>

<!ELEMENT booklet (author?, title, how_published?, address?, year?, note?,
   %commonProperties;>

<!ELEMENT manual (author?, title, organization?, address?, edition?, year?, note?,
   %commonProperties;)>

<!ELEMENT technical_report (author, title, institution, year, type?, number?, address?,
   note?, %commonProperties;)>

<!ELEMENT masters_thesis (author, title, school, year, type?, address?, note?,
   %commonProperties;)>

<!ELEMENT phd_thesis (author, title, school, year, type?, address?, note?,
   %commonProperties;)>

<!ELEMENT in_book (author | editor, title, in_book_chapter | pages, publisher, year,
   volume? | number?, series?, type?, address?, edition?, note?, %commonProperties;)>
<!ELEMENT in_book_chapter(chapter, pages?)>

<!ELEMENT in_collection (author, title, book_title, publisher, year, editor?, volume? |
   number?, series?, type?, chapter?, pages?, address?, edition?, note?,
   %commonProperties;)>

<!ELEMENT proceedings (editor?, title, year, volume? | number?, series?, address?,
   organization?, publisher?, note?, %commonProperties;>

<!ELEMENT in_proceedings (author, title, book_title, year, editor?, volume? | number?,
   series?, pages?, address?, organization?, publisher?, note?, %commonProperties;)>

<!ELEMENT conference (author, title, book_title, year, editor?, volume? | number?,
   series?, pages?, address?, organization?, publisher?, note?, %commonProperties;)>

<!ELEMENT unpublished (author, title, note, year?, %commonProperties;)>

<!ELEMENT miscellaneous (author?, title?, how_published?, year?, note?,
   %commonProperties;)>

<!ELEMENT   address         (#PCDATA) > <!ELEMENT   author          (#PCDATA) >
<!ELEMENT   book_title      (#PCDATA) > <!ELEMENT   chapter         (#PCDATA) >
<!ELEMENT   edition         (#PCDATA) > <!ELEMENT   editor          (#PCDATA) >
<!ELEMENT   how_published   (#PCDATA) > <!ELEMENT   institution     (#PCDATA) >
<!ELEMENT   journal         (#PCDATA) > <!ELEMENT   month           (#PCDATA) >
<!ELEMENT   note            (#PCDATA) > <!ELEMENT   number          (#PCDATA) >
<!ELEMENT   organization    (#PCDATA) > <!ELEMENT   pages           (#PCDATA) >
<!ELEMENT   publisher       (#PCDATA) > <!ELEMENT   school          (#PCDATA) >
<!ELEMENT   series          (#PCDATA) > <!ELEMENT   title           (#PCDATA) >
<!ELEMENT   type            (#PCDATA) > <!ELEMENT   volume          (#PCDATA) >
<!ELEMENT   year            (#PCDATA) > <!ELEMENT   category        (#PCDATA) >
<!ELEMENT   abstract        (#PCDATA) > <!ELEMENT   affiliation     (#PCDATA) >
<!ELEMENT   contents        (#PCDATA) > <!ELEMENT   copyright       (#PCDATA) >
<!ELEMENT   isbn            (#PCDATA) > <!ELEMENT   issn            (#PCDATA) >
<!ELEMENT   keywords        (#PCDATA) > <!ELEMENT   language        (#PCDATA) >
<!ELEMENT   location        (#PCDATA) > <!ELEMENT   price           (#PCDATA) >
<!ELEMENT   size            (#PCDATA) > <!ELEMENT   url             (#PCDATA) >
```

FIGURA B.2 – DTDs pré-processadas(II)



SIGMOD



DB&LP

FIGURA B.3 – Esquemas conceituais para as DTDs(I)

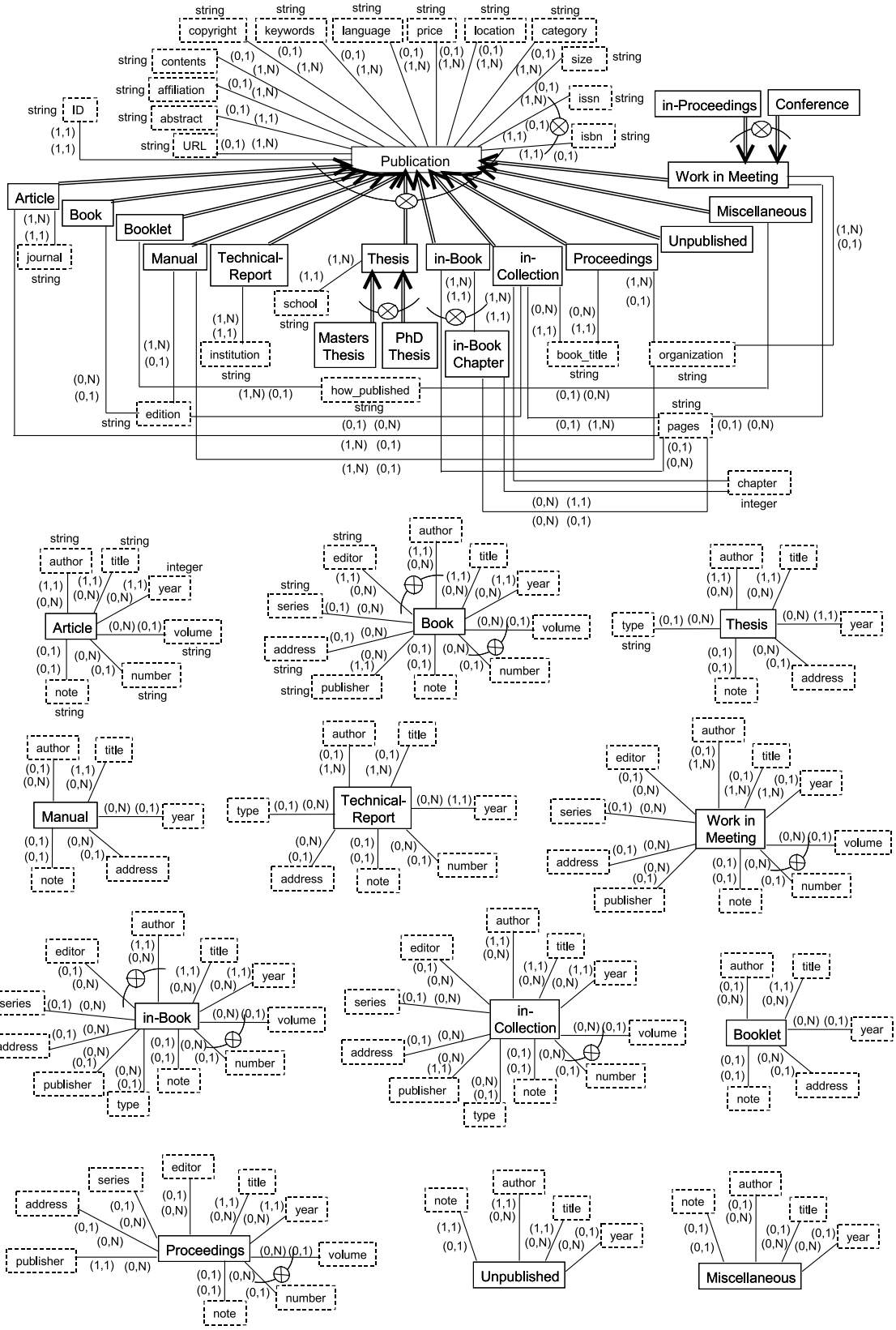


FIGURA B.4 – Esquemas conceituais para as DTDs(II)

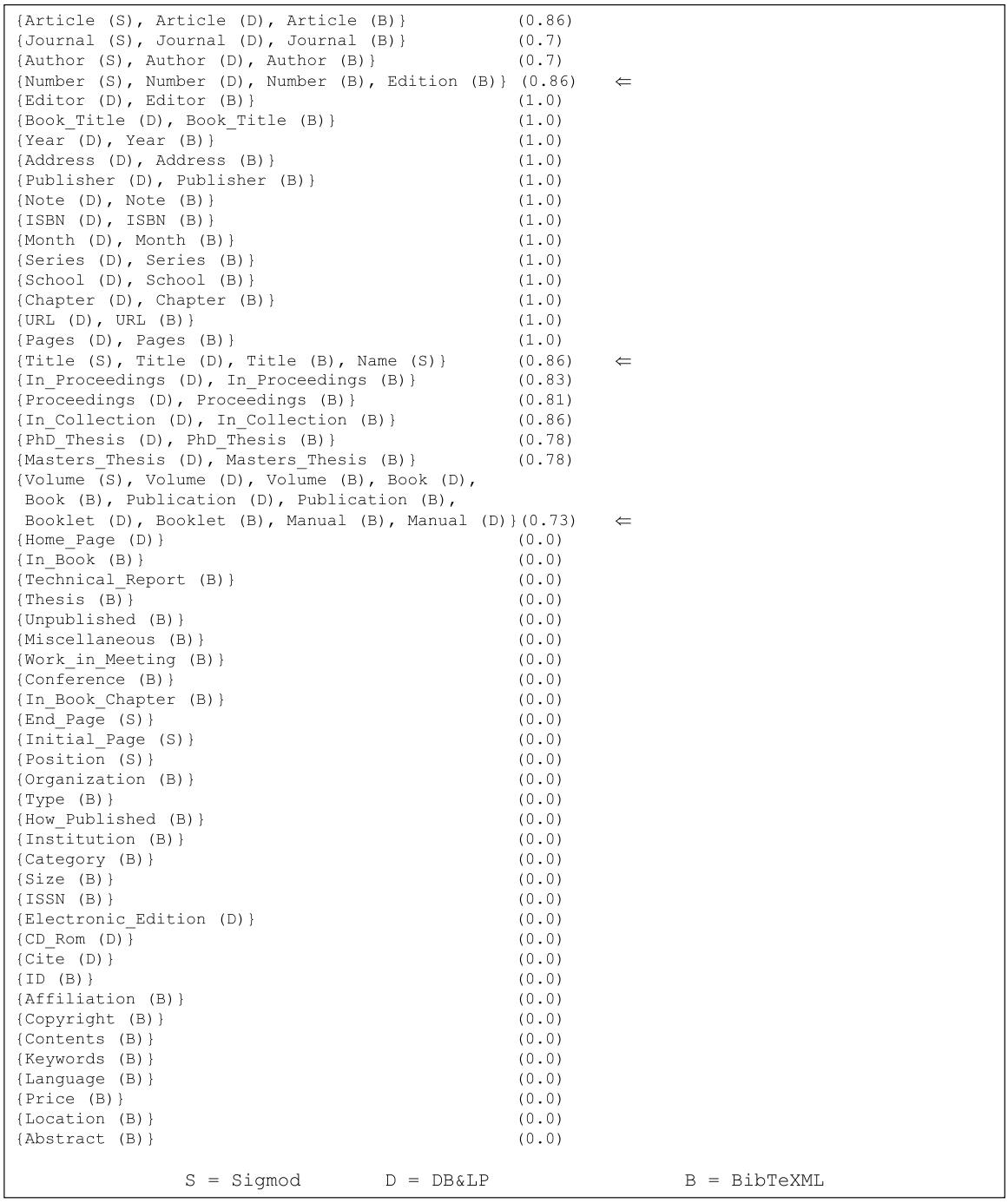


FIGURA B.5 – Clusters de afinidade gerados e indicação de clusters a serem alterados

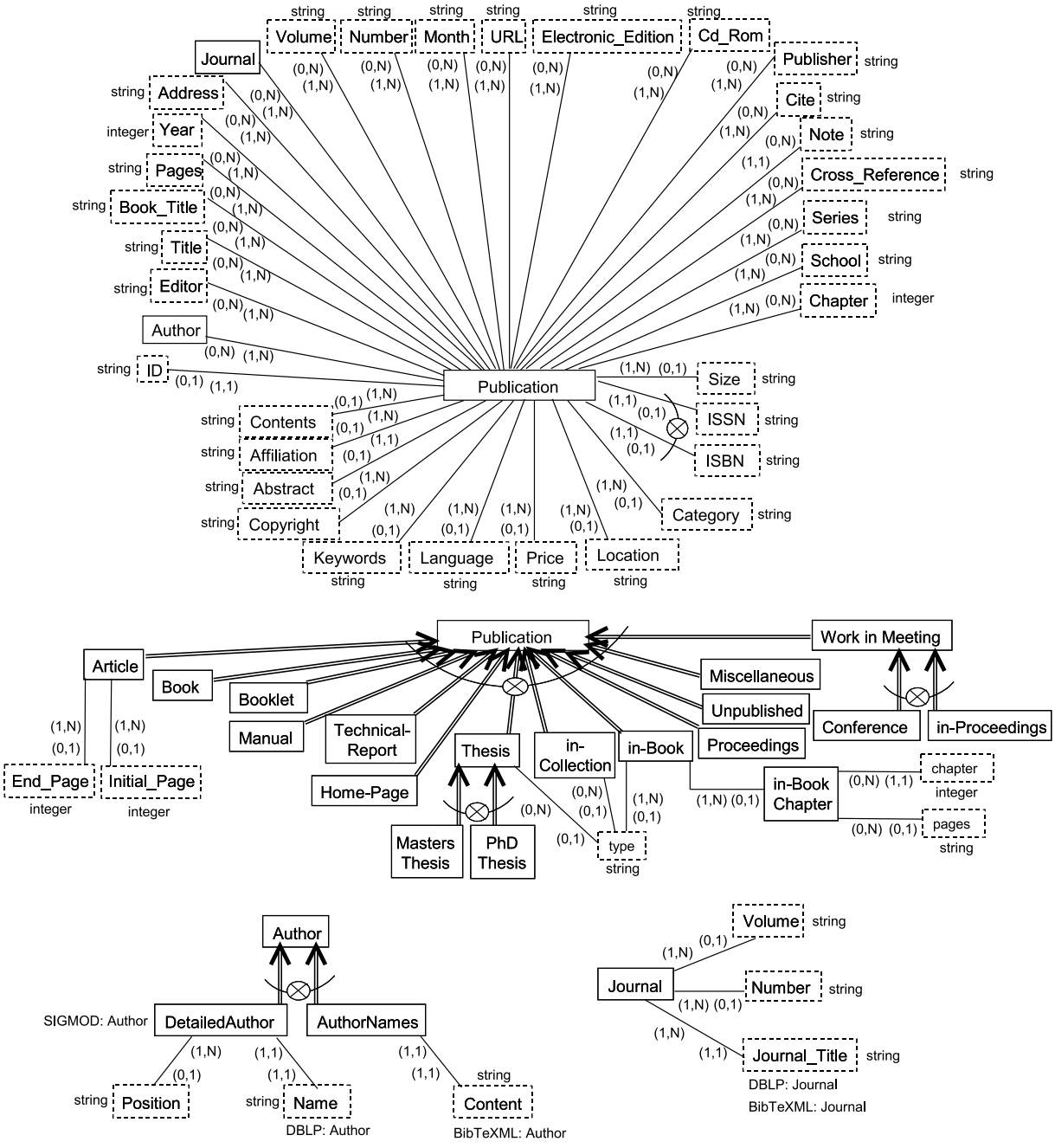


FIGURA B.6 – Esquema global

Bibliografia

[ABI 97] ABITEBOUL, S. Querying Semistructured Data. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, 1997, Delphi, Greece. **Proceedings...** [S.l.: s.n.], 1997. p.1–18.

[ABI 2000] ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. **Data on the Web:** From Relations to Semistructured Data and XML. San Francisco, California: Morgan Kaufmann, 2000.

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

[BAT 86] BATINI, C.; LANZERINI, M.; NAVATHE, S. B. A Comparative Analysis of Methodologies for Database Schema Integration. **ACM Computing Surveys**, New York, v.18, n.4, p.323–364, Dec. 1986.

[BEN 2001] BENEVENTANO, D. et al. The MOMIS approach to Information Integration. In: IEEE AND AAAI INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS, ICEIS, 2001, Setúbal, Portugal. **Proceedings...** [S.l.: s.n.], 2001.

[BER 2001] BERGAMASCHI, S. et al. Semantic Integration of Heterogeneous Information Sources. **Data & Knowledge Engineering**, Amsterdam, v.36, n.1, p.215–249, Mar. 2001.

[BIB 2002] BIBTEX as XML Markup. Disponível em: *<http://bibtexml.sourceforge.net/>*. Acesso em: jul. 2002.

[BRA 2000] BRADLEY, N. **The XML Companion**. 2nd ed. Harlow, USA: Addison-Wesley Longman, 2000. 435p.

[BUN 97] BUNEMAN, P. Semistructured Data. In: SIGMOD INTERNATIONAL SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS, PODS, 16., 1997, Tucson, Arizona, USA. **Proceedings...** [S.l.: s.n.], 1997. p.117–121.

[BUS 99] BUSSE, S. et al. **Federated Information Systems:** Concepts, Terminology and Architectures. Berlin: Universität Berlin, 1999. (Technical Report, 99-9).

[CAR 95] CAREY, M. et al. Towards Heterogeneous Multimedia Information Systems: the garlic approach. In: INTERNATIONAL WORKSHOP ON RESEARCH ISSUES IN DATA ENGINEERING, RIDE, 5.; DISTRIBUTED OBJECT MANAGEMENT, 1995. **Proceedings...** [S.l.: s.n.], 1995.

[CAS 2001] CASTANO, S.; ANTONELLIS, V.; VIMERCATI, S. C. Global Viewing of Heterogeneous Data Sources. **IEEE Transactions on**

Knowledge and Data Engineering, New York, v.13, n.2, p.277–297, Mar. 2001.

[CHA 94] CHAWATHE, S. et al. The TSIMMIS Project: integration of heterogeneous information sources. In: IPSJ CONFERENCE, 1994, Tokyo, Japan. **Proceedings...** [S.l.: s.n.], 1994. p.7–18.

[CLU 98] CLUET, S. et al. Your Mediators Need Data Conversion! **SIGMOD Records**, New York, v.27, n.2, June 1998. Trabalho apresentado na ACM SIGMOD International Conference on Management of Data, 1998.

[CXM 2002] CXML.org. Disponível em: *<http://www.cxml.org>*. Acesso em: jan. 2002.

[DAM 2002] DAML Language. Disponível em: *<http://www.daml.org>*. Acesso em: maio 2002.

[DAM 2001] DAML+OIL Reference Description. Disponível em: *<http://www.w3.org/TR/daml+oil-reference>*. Acesso em: dez. 2001.

[DBL 2002] DBLP Bibliography. Disponível em: *<http://www.informatik.uni-trier.de/ ley/db/>*. Acesso em: jul. 2002.

[DOA 2001] DOAN, A.; DOMINGOS, P.; HALEVY, A. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: ACM INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD, 2001., 2001, Santa Barbara, USA. **Proceedings...** [S.l.: s.n.], 2001. p.509–520.

[DOC 2002] DOCUMENT Object Model. Disponível em: *<http://www.w3.org/DOM>*. Acesso em: jun. 2002.

[EBi 2002] EBisXML. Disponível em: *<http://www.basda.org>*. Acesso em: maio 2002.

[ELM 99] ELMAGARMID, A.; RUSINKIEWICZ, M.; SHETH, A. **Management of Heterogeneous and Autonomous Database Systems**. San Francisco, California: Morgan Kaufmann, 1999. 413p.

[ELM 2000] ELMASRI, R. A.; NAVATHE, S. B. **Fundamentals of Database Systems.** 3rd ed. [S.l.]: Addison-Wesley, 2000. 960p.

[EXT 2002] EXTENSIBLE Markup Language - XML. Disponível em: *<http://www.w3.org/XML>*. Acesso em: jun. 2002.

[FLO 97] FLORESCU, D.; LEVY, A.; MENDELZON, A. Database Techniques for the World Wide Web: a survey. **SIGMOD Record**, New York, v.27, n.3, p.59–74, Mar. 1997.

[GAR 99] GARLIC Project. Disponível em: *<http://www.almaden.ibm.com/cs/garlic/homepage.html>*. Acesso em: oct. 1999.

[GCI 2002] GCI. Disponível em: *<http://www.globalcommerceinitiative.org>*. Acesso em: maio 2002.

[HAL 98] HALPHIN, T. **Object-Role Modeling (ORM/NIAM), Handbook on Architectures of Information Systems**. [S.l.]: Springer-Verlag, 1998. p.81–102.

[JEN 2001] JENSEN, M. R.; MOLLER, T. H.; PEDERSEN, T. Converting XML Data to UML Diagrams for Conceptual Data Integration. In: INTERNATIONAL WORKSHOP ON DATA INTEGRATION OVER THE WEB, DIWEB; CAISE, 1., 2001, Interlaken, Switzerland. **Proceedings...** [S.l.: s.n.], 2001.

[KIM 95] KIM, W. **Modern Database Systems:** the Object Model, Interoperability, and Beyond. New York: ACM Press, 1995.

[LIM 2001] LIM, S.; NG, Y. An Automated Integration Approach for Semistructured and Structured Data. In: INTERNATIONAL SYMPOSIUM ON COOPERATIVE DATABASE SYSTEMS FOR ADVANCED APPLICATIONS, CODAS, 3., 2001, Beijing, China. **Proceedings...** [S.l.]: IEEE, 2001. p.12–21.

[LIT 90] LITWIN, W.; MARK, L.; ROSSOPOULOS, N. Interoperability of Multiple Autonomous Databases. **ACM Computing Surveys**, New York, v.22, n.3, p.267–293, Sept. 1990.

[LUD 99] LUDäSCHER, B. et al. View Definition and DTD Inference for XML. In: WORKSHOP ON QUERY PROCESSING FOR SEMISTRUCTURED DATA AND NON-STANDARD DATA FORMATS, ICDT, 1999, Jerusalém, Israel. **Proceedings...** [S.l.: s.n.], 1999.

[MAD 2001] MADHAVAN, J.; BERNSTEIN, P.; RAHM, E. Generic Schema Matching with Cupid. In: CONFERENCE ON VERY LARGE DATA BASES, VLDB, 27., 2001, Rome, Italy. **Proceedings...** [S.l.]: Morgan Kaufmann, 2001. p.49–58.

[MCB 2001] MCBRIEN, P.; POULOVASSILIS, A. A Semantic Approach to Integrating XML and Structured Data Sources. In**:** CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAISE, 13., 2001, Interlaken, Switzerland. **Advanced Information Systems Engineering**:proceedings. Berlin: Springer-Verlag, 2001. p.330-345.

[MEL 2000] MELLO, R. S. **Aplicação de Ontologias a Bancos de Dados Semi-Estruturados**. 2000. Exame de Qualificação (Doutorado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MEL 2002] MELLO, R. S.; CASTANO, S.; HEUSER, C. A. A Method for The Unification of XML Schemata. **Information and Software Technology,** [S.l.], v.44, n.4, p.241–249, Mar. 2002.

[MEL 2000a] MELLO, R. S. et al. Dados Semi-Estruturados. In: SIMPóSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 15.; SIMPóSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 14., 2000, João Pessoa, Paraíba. **Mini-Cursos/Tutoriais...** João Pessoa: PUCRS, 2000. p.475–513.

[MEL 2000b] MELLO, R. S.; HEUSER, C. A. Aplicação de Ontologias a Dados Semi-Estruturados. In: CONFERENCIA LATINOAMERICANA DE INFORMÁTICA, CLEI, 26., 2000, Cidade do México, México. **Memorias...** México: Tec de Monterrey, 2000. 1 CD.

[MEL 2001] MELLO, R. S.; HEUSER, C. A. A Bottom-Up Approach for Integration of XML Sources. In: INTERNATIONAL WORKSHOP ON INFORMATION INTEGRATION ON THE WEB, WIIW, 2001, Rio de Janeiro, Brazil. **Proceedings...** Rio de Janeiro: UNIRIO, 2001. p.118–124.

[MEL 2001a] MELLO, R. S.; HEUSER, C. A. A Rule-Based Conversion of a DTD to a Conceptual Schema. In**:** INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 20., 2001, Yokohama, Japan. **Conceptual Modeling**:proceedings. Berlin: Springer-Verlag, 2001. p.133-148.

[OIL 2001] OIL - Ontology Inference Language. Disponível em: *<http://www.ontoknowledge.org/oil>*. Acesso em: out. 2001.

[OSZ 99] OSZU, M. T.; VALDURIEZ, P. **Distributed Databases:** Principles and Systems. [S.l.]: Prentice Hall, 1999.

[RAH 2001] RAHM, E.; BERNSTEIN, P. **On Matching Schemas Automatically**. [S.l.]: University of Leipzig, 2001. (Microsoft Research Technical Report, 17).

[RES 2002] RESOURCE Description Framework - RDF. Disponível em: *<http://www.w3.org/RDF>*. Acesso em: jun. 2002.

[REY 2001] REYNAUD, C.; SIROT, J.; VODISLAV, D. Semantic Integration of XML Heterogeneous Data Sources. In: INTERNATIONAL DATABASE ENGINEERING & APPLICATIONS SYMPOSIUM, IDEAS, 2001, Grenoble, France. **Proceedings...** Los Alamitos: IEEE, 2001. p.199–208.

[ROD 2001] RODRIGUEZ-GIANOLLI, P.; MYLOPOULOS, J. A Semantic Approach to XML-Based Data Integration. In**:** INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 20., 2001, Yokohama, Japan. **Conceptual Modeling**:proceedings. Berlin: Springer-Verlag, 2001. p.117-132.

[SEM 2002] SEMANTIC Web. Disponível em: *<http://www.w3.org/2001/sw>*. Acesso em: jun. 2002.

[SHE 90] SHETH, A. P.; LARSON, J. A. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. **ACM Computing Surveys**, New York, v.22, n.3, p.183–236, Sept. 1990.

[SIG 99] SIGMOD Record: xml version (v. 1.0). disponível em: *<http://www.acm.org/sigmod/record/xml>*. Acesso em: dez. 1999.

[TSI 98] TSIMMIS. Disponível em: *<http://www-db.stanford.edu/tsimmis/tsimmis.html>*. Acesso em: abr. 1998.

[VDO 2001] VDOVJAK, R.; HOUBEN, G. RDF-Based Architecture for Semantic Integration of Heterogeneous Information Sources. In: INTERNATIONAL WORKSHOP ON INFORMATION INTEGRATION ON THE WEB, WIIW, 2001, Rio de Janeiro, Brazil. **Proceedings...** Rio de Janeiro: UNIRIO, 2001. p.51–57.

[WIE 92] WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. **Computer**, New York, v.25, n.3, p.38–49, Mar. 1992.

[WIE 93] WIEDERHOLD, G. Inteligent Integration of Information. **SIGMOD Record**, New York, v.22, n.2, p.434–437, June 1993.

[WOR 2002] WORDNET 1.7.1. Disponível em: *<http://www.cogsci.princeton.edu/cgi-bin/webwn1.7.1>*. Acesso em: fev. 2002.

[WOR 2002a] WORLD Wide Web Consortium. Disponível em: *<http://www.w3.org>*. Acesso em: jul. 2002.

[XML 2001] XML Path Language. Disponível em: *<http://www.w3.org/TR/xpath>*. Acesso em: dez. 2001.

[XML 2002] XML Schema. Disponível em: *<http://www.w3.org/XML/Schema>*. Acesso em: jul. 2002.

[XQU 2002] XQUERY 1.0 and XPath 2.0 Data Model. Disponível em: *<http://www.w3.org/TR/query-datamodel>*. Acesso em: abr. 2002.