UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

XMAP: MAPEAMENTO E ARMAZENAMENTO DE DADOS XML EM BANCOS DE DADOS RELACIONAIS

DISSERTAÇÃO DE MESTRADO

Francisco Tiago Machado de Avelar

Santa Maria, RS, Brasil
2012

XMAP: MAPEAMENTO E ARMAZENAMENTO DE DADOS XML EM BANCOS DE DADOS RELACIONAIS

Francisco Tiago Machado de Avelar

Dissertação apresentada ao Programa de Pós-Graduação em Informática,
da Universidade Federal de Santa Maria (UFSM, RS),
como requisito parcial para obtenção do grau de
Mestre em Informática

Orientador: Prof[a]. Dr[a]. Deise de Brum Saccol

Santa Maria, RS, Brasil
2012

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

XMAP: MAPEAMENTO E ARMAZENAMENTO DE DADOS XML EM BANCOS DE DADOS RELACIONAIS

elaborada por
Francisco Tiago Machado de Avelar

como requisito parcial para obtenção do grau de
Mestre em Informática

COMISSÃO EXAMINADORA:

Dr^{a}. Deise de Brum Saccol
(Presidente/Orientador)

Dr. Ronaldo dos Santos Mello (UFSC)

Dr. Eduardo Kessler Piveta (UFSM)

Santa Maria, 30 de março de 2012.

AGRADECIMENTOS

Este trabalho não estaria finalizado caso não existisse um espaço para eu dedicar algumas palavras pessoais de gratidão. Afinal, o resultado alcançado não foi unicamente mérito próprio, apesar de ser o protagonista do encerramento desta etapa vivenciada.

Na realidade, a caminhada iniciou ao ingressar no mestrado, sendo que devo meu primeiro agradecimento a Deus por sempre me trazer força de espírito e ajuda nos momentos necessários. Acredito na existência divina, pois a ciência ainda não é capaz de explicar muitas das criações e acontecimentos da vida. Além disso, concentrados em nós mesmos, precisamos crer que estamos sendo ajudados para seguir adiante.

Deus criou a pessoa de modo abstrato, a partir do qual definiu cada integrante de maneira especializada e estruturada. Na minha família, tenho a minha mãe como sendo a instância principal, pois, diante de tantas funções que ela desempenhou, tratando meus erros e estipulando exceções, ela soube sempre me incentivar a crescer. Existe a minha namorada, instância muito especial a qual está incluída na minha linha de vida, pois foi através de muitas trocas de mensagens que veio a fazer composição de meu cenário presente e futuro. Não posso deixar de mencionar a minha orientadora, que, com grande receptividade, aceitou meu pedido de orientação quando precisei trocar de tema de pesquisa.

Minha expectativa é que esta nova etapa tenha ótimas realizações. Nos momentos de dificuldade, sei que há pessoas que me ajudarão. Por outro lado, nas realizações quero dividir o sentimento de alegria para que possam saber o quanto fazem parte do meu caminho.

Qualquer dificuldade que eu tenha tido na escrita deste trabalho, assim como deficiência em alguma abordagem para resolver algum problema, espero melhorar para ser um profissional melhor. Apesar das minhas limitações pessoais e técnicas, creio que seja importante o esforço que eu possa empregar na tentativa de resolver qualquer questão que me seja atribuída.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

XMAP: MAPEAMENTO E ARMAZENAMENTO DE DADOS XML EM BANCOS DE DADOS RELACIONAIS

AUTOR: FRANCISCO TIAGO MACHADO DE AVELAR
ORIENTADOR: DEISE DE BRUM SACCOL
Data e Local da Defesa: Santa Maria, 30 de março de 2012.

O gerenciamento de informação pode ser possibilitado através da materialização das fontes de dados de maneira persistente e acessível. Tal materialização pode ocorrer através do uso de um sistema gerenciador de banco de dados relacional. Diversos domínios de conhecimento usam de XML como padrão para o armazenamento e para o processamento de documentos. No entanto, documentos XML de um mesmo domínio de aplicação podem ter estruturas diferentes, tornando mais difícil o processo de mapeamento para um único esquema de banco de dados. Para solucionar esse problema, este trabalho assume que existe uma ontologia que descreve os documentos XML de entrada. Dessa forma, a ontologia é mapeada para um esquema relacional e posteriormente os arquivos XML originais descritos pela ontologia são armazenados no banco de dados. Considerando o contexto descrito, este trabalho propõe: a definição de regras de mapeamento de dados XML para o esquema relacional, previamente gerado; a criação de funções de transformação para a solução de conflitos estruturais e semânticos entre os dados XML de entrada; e a definição do mecanismo de inserção de dados XML em bancos de dados relacionais.

Palavras-chave: XML, banco de dados, mapeamento, armazenamento

ABSTRACT

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

XMAP: MAPPING AND STORAGE OF XML DATA IN RELATIONAL DATABASES

AUTOR: FRANCISCO TIAGO MACHADO DE AVELAR
ORIENTADOR: DEISE DE BRUM SACCOL
Data e Local da Defesa: Santa Maria, 30 de março de 2012.

Information management can be achieved with the materialization of data sources in a persistent and accessible way. This materialization can occur through the use of a relational database management system. Several knowledge domains make use of XML as standard for storing and processing of documents. However, XML documents from one application domain may have different structures, making more difficult the mapping process to a single database schema. As a solution of this problem, this work assumes that there is an ontology that describes the input XML documents. Thus, the ontology is mapped to a relational schema and the XML files described by the ontology are stored into the database. On considering this scenario, this work proposes: the definition of mapping rules from XML to relational schema, previously generated; the generation of transformation functions for solving structural and semantic conflicts between the input XML data; and the mechanism definition for the XML data insertion into the relational database.

Keywords: XML, database, mapping, storage.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

API	*Application Programming Interface*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CMap	*Concept Mapper*
DML	*Data Manipulation Language*
DTD	*Document Type Definition*
HSQLDB	*Hyper Structured Query Language Database*
INPE	Instituto Nacional de Pesquisas Espaciais
J2SE	*Java Platform, Standard Edition*
LAV	*Local-As-View*
OntoGen	*Ontology Generator*
OntoRel	*Ontology to Relational*
OWL	*Web Ontology Language*
QMap	*Query Mapping*
RSS	*Really Simple Syndication*
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDR	Sistema de Gerenciamento de Banco de Dados Relacional
W3C	*World Wide Web Consortium*
X2Rel	*XML to Relational*
XMap	*XML Mapping*
XML	*Extensible Markup Language*
XPath	*XML Path language*
XQuery	*XML Query Language*
XSD	*XML Schema Definition*
XSDM	*XML Schema Data Model*

SUMÁRIO

1 INTRODUÇÃO

A materialização de informações textuais em aplicações computacionais pode ser feita de maneira conveniente através de documentos XML (W3C, 2012a). O conteúdo é organizado de maneira hierárquica, sendo normalmente o esquema do documento associado a cada aplicação.

Em um ambiente heterogêneo, onde ocorre o intercâmbio de informação entre aplicações, o processamento de documentos XML distintos é uma realidade (CHAMBERLIN; ROBIE; FLORESCU, 2001). No contexto deste trabalho, entende-se por distintos aqueles documentos que possuem esquemas diferentes, embora o conteúdo possa ser idêntico ou semelhante, conforme mostrado na Figura 1.1.

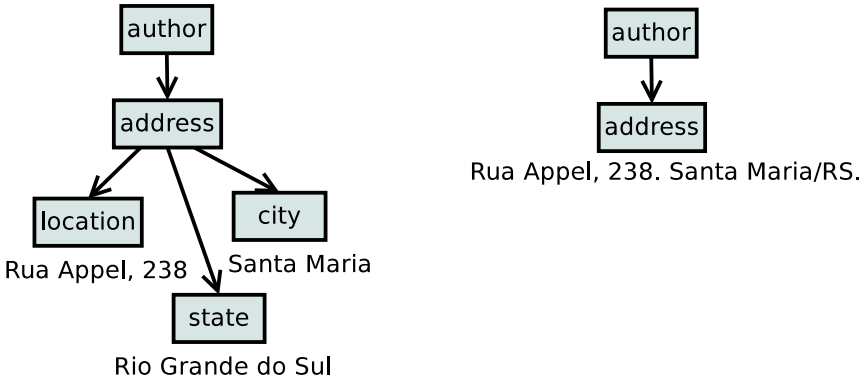

Figura 1.1: Representação gráfica de esquemas XML distintos contendo o mesmo conteúdo.

Em um cenário ideal, os documentos XML estariam definidos por um esquema único, possibilitando assim uma tradução direta para um conjunto de tabelas. Tal fato é factível no caso de bancos de dados cujos dados XML foram inicialmente gerados a partir de um esquema relacional, assim servindo como modelo de intercâmbio entre os formatos relacionais.

Em contrapartida, muitas aplicações não geram dados XML a partir de um banco de dados relacional. Cada documento XML resultante possui um esquema próprio, levando a uma variação na estrutura de documentos, inclusive entre documentos pertencentes a um mesmo domínio de aplicação. Conforme exemplificado na Figura 1.2, há dois esquemas XML diferentes para um mesmo domínio, sendo, nesse caso, dados de endereços de autores.

doc_A.xml

```
01 <author>
02   <address>
03     <location>Rua Appel, 238</location>
04     <city>Santa Maria</city>
05     <state>Rio Grande do Sul</city>
06   </address>
07 </author>
```

doc_B.xml

```
01 <author>
02   <address>Rua Appel, 238. Santa Maria/RS</address>
03 </author>
```

Figura 1.2: Equivalente textual dos documentos XML de exemplo referente à Figura 1.1.

Em uma busca pelo endereço de autores, uma consulta em particular deve ser definida para cada documento XML. Na ausência de um modelo global, a recuperação da informação a partir de fontes XML está relacionada diretamente à estrutura do documento. Analisando a Figura 1.3, existem duas consultas em XQuery (W3C, 2012c) para os documentos `doc_A.xml` e `doc_B.xml` ilustrados na Figura 1.2. Ambas objetivam extrair o endereço dos autores que moram em Santa Maria.

Consulta XQuery em doc_A.xml

```
01 xquery version "1.0";
02 <author>
03 {
04   for $local in doc("doc_A.xml")/author/address
05   where $local/city = "Santa Maria"
06   return
07     <address>{$local/location}</address>
08 }
09 </author>
```

Consulta XQuery em doc_B.xml

```
01 xquery version "1.0";
02 <author>
03 {
04   for $local in doc("doc_B.xml")/author/address
05   where contains($local/text(), "Santa Maria")
06   return $local
07 }
08 </author>
```

Resultado da consulta XQuery em doc_A.xml

```
01 <author>
02   <address>
03     <location>Rua Appel, 238</location>
04   </address>
05 </author>
```

Resultado da consulta XQuery em doc_B.xml

```
01 <author>
02   <address>Rua Appel, 238. Santa Maria/RS</address>
03 </author>
```

Figura 1.3: Consultas XQuery de acordo com os documentos XML da Figura 1.2

Em razão de cada documento XML possuir uma estrutura própria, uma consulta XQuery diferente deve ser definida para cada instância de dados. Existindo diversos documentos distintos, consultas individuais passam a ser inviáveis.

1.1 Contexto

Este trabalho está inserido no contexto do framework X2Rel, apresentado na seção 2.3, cujo objetivo é armazenar dados XML de documentos heterogêneos caracterizados pela presença de estruturas distintas. Consideram-se os documentos como pertencentes a um mesmo domínio de aplicação e os dados são armazenados em uma base de dados relacional descrita por um único esquema.

Um único esquema relacional é uma abordagem adequada para reduzir a complexidade computacional envolvida no processamento de diversos documentos XML distintos. Ao invés de interpretar diretamente documentos XML distintos, a estratégia é agrupá-los em um domínio comum de conhecimento. Esse agrupamento pode ser feito através do uso de ontologias (W3C, 2012b). Para resolver as diversidades estruturais e semânticas existentes entre documentos XML heterogêneos, este trabalho utiliza ontologias como representação do domínio de conhecimento. A ontologia, definida como um esquema global integrado, é resultante a partir da integração de esquemas que descrevem os documentos XML distintos relativos a um mesmo domínio de aplicação (MELLO, 2007).

No contexto deste trabalho, a ontologia é mapeada para uma coleção de tabelas e colunas de um banco de dados relacional. Esse mapeamento ocorre a partir de um conjunto pré-definido de regras de transformação (ANDRADE, 2010) e (SACCOL; PIVETA; ANDRADE, 2011). Uma vez que o esquema do banco de dados esteja criado, a etapa seguinte é a inserção dos dados XML originais nas tabelas relacionais, que é o foco desta dissertação. Especificamente, este trabalho propõe uma técnica para a inserção de dados XML em bancos de dados relacionais, considerando o ambiente do framework X2Rel. O armazenamento de dados XML estruturalmente heterogêneos em um banco de dados relacional é descrito por um único esquema lógico.

1.2 Objetivos e contribuições

O objetivo geral deste trabalho é propor um conjunto de mecanismos que possibilite o mapeamento e o armazenamento de dados XML em bancos de dados relacionais. Para isso, assume-se que documentos XML com estruturas diferentes sejam descritos por uma mesma ontologia, a qual representa os conceitos e relações de um certo domínio de aplicação.

O mapeamento do conteúdo de documentos XML para o armazenamento em um banco de dados relacional não é uma tarefa cuja execução seja direta. Em razão da existência de diversos esquemas XML pertencentes a aplicações heterogêneas, é necessário um mecanismo de mapeamento do conteúdo original em XML para o armazenamento em tabelas de banco de dados. Neste cenário, as principais contribuições deste trabalho são:

- a análise e a classificação de tipos de conflitos estruturais e semânticos em documentos XML;
- a definição de regras de mapeamento de dados XML heterogêneos para um único esquema relacional, com base nos conflitos definidos;
- a especificação de funções de transformação para eliminar conflitos estruturais e semânticos;
- a definição de uma estrutura de documento de mapeamento que descreve as equivalências de conceitos entre documentos XML, a ontologia e o esquema lógico relacional de bancos de dados;
- a implementação de um protótipo do mecanismo proposto;

1.3 Organização do texto

O texto é estruturado conforme segue:

- o Capítulo 2, **Fundamentação teórica**, apresenta as pesquisas envolvendo o armazenamento de dados XML em banco de dados relacionais. Além disso, será abordado o uso de ontologias e a fundamentação teórica que situa o escopo da dissertação;
- o Capítulo 3, **XMap: Mapeamento de Documentos XML para Banco de Dados Relacionais**, propõe a arquitetura do sistema proposto neste trabalho, além da classificação de conflitos existentes no mapeamento do conteúdo de documentos XML para armazenamento em banco de dados relacionais;
- o Capítulo 4, **Implementação**, aborda as tecnologias utilizadas na implementação do protótipo, bem como o diagrama de classes da implementação. Algoritmos para a resolução de conflitos são propostos de acordo com a classificação prévia;
- o Capítulo 5, **Experimentos e Avaliação de Resultados**, expõe a metodologia utilizada para a busca de resultados e a análise dos dados. Associam-se os documentos XML com a ontologia e o esquema lógico relacional para armazenamento em banco de dados;
- o Capítulo 6, **Conclusões**, apresenta as principais conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste trabalho, o armazenamento do conteúdo de documentos XML em banco de dados relacionais é a abordagem adotada para centralizar a busca da informação. O problema tratado nesta dissertação é parte do framework X2Rel (SACCOL; PIVETA; ANDRADE, 2011), um ambiente para o armazenamento e a consulta de arquivos XML com incompatibilidades estruturais e semânticas. Conceitos e tecnologias recorrentes são apresentados ao longo deste capítulo, além dos trabalhos relacionados ao tema de pesquisa.

2.1 Armazenamento de documentos XML em bancos de dados relacionais

O armazenamento da informação em um banco de dados relacional é determinada pela estrutura lógica relacional. A linguagem SQL, mais especificamente a correspondente subdivisão DDL, é o padrão de expressão para a criação do esquema relacional de banco de dados.

Para armazenamento em banco de dados, o conteúdo de interesse deve possuir um tipo específico. Em banco de dados relacional, tabelas são constituídas por uma ou por mais colunas; estas, por sua vez, armazenam dados de um determinado tipo. Como exemplo, a data de nascimento de uma pessoa poderia ser armazenada de maneira mais adequada em uma coluna tipo `date`, assim como a altura, em metros, corresponderia a um dado do tipo numérico `float` ou `double`.

Uma possibilidade de armazenamento de arquivos XML em bancos de dados relacionais é utilizar uma forma não-estruturada através de campos do tipo CLOB. Trata-se de um formato de dados que corresponde ao armazenamento de grandes cadeias de caracteres. Vantagens em armazenar o conteúdo XML em formato CLOB estão na inexistência de pré-processamento e também na preservação do conteúdo XML original. Em contrapartida, o uso de campos CLOB possui a limitação de um tamanho máximo de armazenamento especificado no esquema relacional em SQL DDL (SCHAFFNER, 2012). Além disso, a recuperação de informaçôes específicas a partir de documentos XML em formato CLOB é limitada em razão de o tipo de dado relacional corresponder a uma cadeia contínua de caracteres.

Outra alternativa para armazenamento de documentos XML em bancos de dados é a fragmentação. O processo de fragmentação corresponde a extração dos dados relevantes da estrutura hierárquica de um documento XML, colocando assim tais dados em uma estrutura relacional plana. Os dados XML são armazenados em colunas relacionais, mas a fragmentação do documento XML faz com que os relacionamentos hierárquicos entre os dados sejam perdidos (JONGE, 2012).

Ao invés de armazenar documentos XML de maneira não-estruturada ou fragmentada, algu-

mas abordagens propõem preservar a estrutura lógica desses documentos XML, possibilitando o armazenamento estruturado dos documento. Como proposto em (KHAN; RAO, 2001), o armazenamento da informação XML para o formato relacional ocorre de maneira automatizada e transparente ao usuário. O mapeamento do esquema XML para o relacional é proposto por (MARTINS; LAENDER, 2005), porém não utiliza uma representação global das diferenciações estruturais e semânticas entre documentos XML diversificados.

Bancos de dados com suporte a XML mapeiam os arquivos para um banco de dados relacional, aceitando documentos XML como entrada e podendo retornar o resultado de uma consulta no mesmo padrão. Uma solução empregada em bancos de dados comerciais é um tipo especial para tratamento de dados XML em uma estrutura relacional. Mais especificamente, o tipo `XMLType` (ORACLE, 2012) oferece mecanismos para criar, extrair e indexar dados XML. Campos do tipo `XMLType` podem ser utilizados com instruções SQL, reunindo assim as funcionalidades de banco de dados relacionais com os recursos de XML, podendo a informação ser recuperada com expressões de caminho XPath combinados com SQL.

Em uma abordagem dedicada, bancos de dados XML nativos (TAMINO, 2012) usam tecnologias relacionadas com o padrão XML de maneira exclusiva, ou seja, sem a adoção de recursos e linguagens de abordagens relacionais, como o SQL, por exemplo. Contudo, soluções que oferecem suporte ao XML, assim como abordagens dedicadas, consideram a estrutura individual de cada documento XML. Uma consulta em uma base de dados muito diversificada pode resultar na recuperação incompleta da informação.

De maneira geral, técnicas para armazenamento de documentos XML em banco de dados relacionais procuram adaptar o esquema hierárquico do padrão XML para a estrutura relacional. A criação de estruturas em banco de dados relacionais para o armazenamento de documentos XML acrescenta maior complexidade ao sistema gerenciador. Tendencialmente, armazenar documentos XML em tipos complexos irá requerer um processo de conversão do esquema XML para o relacional. Além disso, a criação de estruturas auxiliares de armazenamento tornará a recuperação da informação mais complexa, podendo resultar em maior tempo de processamento de uma requisição.

2.2 Ontologias

Em computação, o propósito de uma ontologia é estabelecer um entendimento comum e compartilhado de um domínio, que pode ser comunicado entre pessoas e sistemas computacionais heterogêneos e distribuídos, provendo um vocabulário de termos e relações com os quais um domínio pode ser modelado (STUDER et al., 1999).

Gruber define ontologias como uma especificação explícita de uma conceituação (GRUBER, 1993). Uma conceituação é uma abstração simplificada da realidade que se deseja repre-

sentar, isso é, um conjunto de objetos, restrições, relacionamentos e entidades que se assumem necessárias em alguma área de aplicação.

A conceituação de Gruber foi modificada por Borst, definindo ontologias como uma especificação formal de uma conceituação compartilhada (BORST, 1997). Esta definição enfatiza o fato que deve haver um acordo na conceituação do que é especificado.

Para descrever uma ontologia, Maedche propõe uma estrutura (O) composta da quíntupla: $O = C, R, Hc, rel, Ao$, onde (MAEDCHE; STAAB, 2002):

- C e R são dois conjuntos disjuntos formados por conceitos e relacionamentos, respectivamente;
- Hc representa a taxonomia da ontologia, isso é, a hierarquia dos conceitos e dos relacionamentos;
- *rel* representa os conceitos não taxonômicos;
- *Ao* representa o conjunto de axiomas da ontologia.

As ontologias vêm sendo utilizadas para descrever artefatos com diferentes níveis de estruturas. Estes níveis variam desde simples taxonomias, esquemas para metadados até teorias lógicas. Normalmente, elas são expressas em linguagens lógicas para que possam ser consistentes o suficiente para a extração do conhecimento (NOLL, 2007).

Para definir e manipular ontologias, sugere-se a utilização de linguagens que ofereçam suporte estruturas para representação do conhecimento. Esta representação é realizada através da descrição formal de um conjunto de termos sobre um domínio específico. Em razão disso, a linguagem OWL foi recomendada pela W3C como uma linguagem de manipulação de ontologias. Sendo definida sobre XML, seu diferencial é a capacidade de processamento semântico através de inferência. Um documento OWL é definido conforme a apresentação dos conceitos a seguir (W3C, 2012b):

- Classe: conjunto de instâncias com características comuns. A partir da definição de classe, algumas ocorrências em uma ontologia podem ser:
  - Superclasse: classe genérica que reúne atributos comuns entre instâncias. As subclasses herdam atributos das superclasses;
  - Relacionamento: forma de interação entre classes, podendo ser associação ou herança;
  - Disjunção: forma de relacionamento entre classes em que não há compartilhamento de atributos.

- Propriedades: modo de estabelecer relacionamento de classes entre si e de classes para seus tipos de dados. Classes podem apresentar as seguintes propriedades:
  - Tipo (*datatype property*): identifica os valores primitivos das instâncias, como `integer`, `float`, `string`, `boolean`, etc.
  - Objeto (*object property*): representa o vínculo de duas instâncias, isto é, seus relacionamentos.
  - Inversa (*inverseOf*): representa um relacionamento bidirecional. Adicionando valores a uma propriedade, consequentemente, se adicionam valores a uma segunda.
  - Transitiva (*transitive property*): se a instância *x* está relaciona com a instância *y*, e a instância *y* está relacionada à instância *z*, então *x* está relacionado com *z*. Usado principalmente em relações "parte-de".
- Indivíduos: são instâncias de classes em um domínio. As propriedades podem ser utilizadas para relacionar indivíduos entre si.

Ontologias em formato OWL são utilizadas neste trabalho com o propósito de abstrair as diferenças estruturais e semânticas de documentos XML pertencentes a um mesmo domínio ou contexto. Desse modo, um conceito, reunindo a definição de classe e propriedade, é a terminologia adotada no framework X2Rel. Como exemplo, considerando os documentos XML da Figura 1.2, os elementos dos arquivos XML passam a ser conceitos em uma ontologia. Como resultado, o conceito `address` está relacionado com os conceitos `author`, `location`, `city` e `state`. A Figura 2.1 ilustra a ontologia em questão.

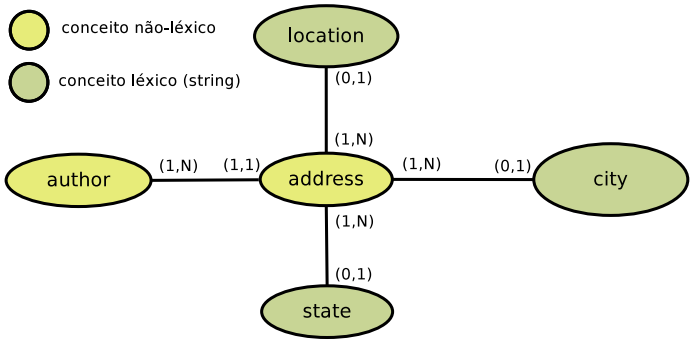

Figura 2.1: Exemplo gráfico de ontologia para os documentos XML ilustrados na Figura 1.2. A definição de conceito não-léxico e conceito léxico pode ser encontrado na seção 2.5.

Elementos e atributos em documentos XML são considerados conceitos na ontologia OWL. A definição de domínio ou contexto relativo a um conjunto de documentos XML deve ser feita previamente pelo usuário. Como exemplo, o domínio dos documentos XML da Figura 1.2 corresponde a currículos para a criação de uma ontologia. A tentativa de agrupar documentos XML de contextos diferentes pode resultar na criação de ontologias que venham a modelar documentos XML de maneira incompleta.

Maiores detalhes sobre ontologias em uma visão prática podem ser encontrados nas seções 3.1.1 e 5.2.

2.3 Framework X2Rel

Muitas aplicações armazenam dados no formato XML, podendo o armazenamento ser realizado através de banco de dados. Como uma das alternativas possíveis, define-se um conjunto de regras de transformação que mapeiam a estrutura do documento XML para uma coleção de relações cuja representação lógica pode ser determinada pela estrutura lógica relacional.

Todavia, documentos XML, mesmo pertencentes a um domínio de aplicação, podem ter estruturas distintas, tornando mais difícil o processo de mapeamento. Para resolver essa questão, pode-se gerar previamente um esquema integrado de dados que representa a estrutura individual de cada instância XML. A partir disso, mapeia-se o esquema integrado para o formato relacional e então os dados XML originais passam a ser armazenados em banco de dados.

O framework X2Rel é o ambiente que envolve as etapas de mapeamento e armazenamento do conteúdo de documentos para bancos de dados relacionais. X2Rel define o esquema integrado como global para um conjunto de instâncias XML pertencentes a um mesmo domínio, sendo representado como uma ontologia. Para evitar submeter diferentes consultas em XML em função da estrutura particular do documento, este trabalho propõe armazenar os arquivos XML em um banco de dados relacional. Com o mapeamento dos arquivos XML para a estrutura relacional, somente uma consulta SQL precisa ser submetida ao banco de dados. Os documentos XML são mapeados para tabelas e as consultas iniciais em XML são transformadas para o equivalente em SQL.

As funcionalidades de X2Rel são fornecidas por quatro componentes diferentes, conforme ilustrados na Figura 2.2 e descritos a seguir:

- OntoGen: responsável pela geração da ontologia, sendo a entrada um conjunto de arquivos XML e a saída a ontologia OWL (esquema integrado);
- OntoRel: responsável pelo mapeamento da ontologia para um esquema lógico relacional, sendo a entrada um arquivo da ontologia em OWL e a saída um *script* SQL com os comandos de criação de tabelas e relacionamentos;
- XMap: responsável pelo mapeamento e pelo armazenamento dos dados XML em bancos de dados relacionais, tendo como entrada os mesmos documentos XML utilizados na OntoGen e a saída o *script* SQL com os comandos de inserção de dados nas tabelas;
- QMap: responsável pelo mapeamento de consultas, sendo a entrada uma consulta em XQuery e a saída uma consulta em SQL.

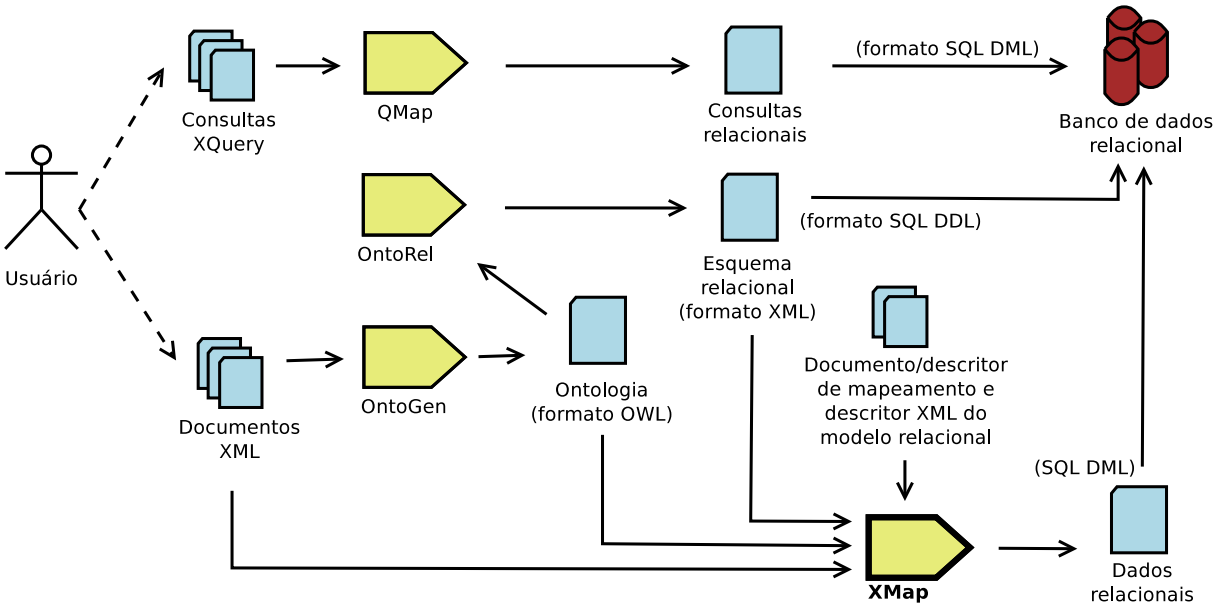

Figura 2.2: Arquitetura do framework X2Rel.

As etapas de transição para o armazenamento da informação presente no formato XML para a estrutura relacional, conforme ilustrado na Figura 2.2, delimita o escopo deste trabalho no contexto de X2Rel. Especificamente, a proposta deste trabalho destina-se em desenvolver o módulo XMap, o qual é responsável por mapear os dados no formato XML para dados relacionais de acordo com a estrutura relacional gerada pela ferramenta OntoRel. A partir de então, o armazenamento ocorre em um banco de dados tradicional, sendo que as consultas, antes especificadas para os dados em XML, agora passam a ser definidas como consultas em SQL.

Como parte integrante do framework X2Rel, a ferramenta OntoGen deve ser apresentada para entendimento do processo de mapeamento da informação em XML para armazenamento em banco de dados relacional. A seção 2.4 a seguir contém mais informações a respeito da OntoGen. O módulo XMap é apresentado de maneira mais detalhada no capítulo 3.

2.4 OntoGen

O objetivo da ferramenta OntoGen é a integração de esquemas XML, gerando ao final do processo uma ontologia que represente os esquemas integrados. Um conjunto de dois ou mais esquemas representados na linguagem XML Schema ou dois ou mais documentos XML, pertencentes a um mesmo domínio, faz parte da entrada de dados do sistema. Como saída, é gerada uma ontologia que representa a integração das instâncias XML de entrada, sendo definida através da linguagem OWL.

O processo de integração de esquemas da ferramenta OntoGen, conforme ilustrado na Figura 2.3, carrega inicialmente os documentos de entrada, que podem ser documentos XML ou esquemas XML. No caso de esquemas XML, ocorre o repasse diretamente para a camada de

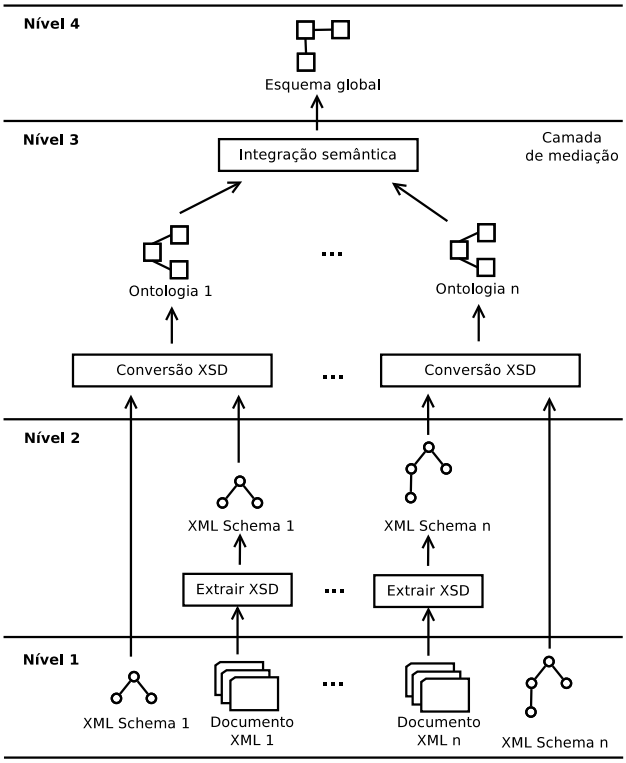

Figura 2.3: Arquitetura da ferramenta OntoGen (MELLO, 2007).

mediação, enquanto para os documentos XML é necessário uma etapa adicional, a qual consiste em extrair o esquema do documento XML de entrada baseando-se nos dados e na estrutura do documento. A extração do esquema do documento XML é feita automaticamente utilizando a biblioteca Castor (EXOLAB, 2012). No último nível está a ontologia global, a qual descreve os conceitos e relações dos documentos XML utilizados na entrada. Para obter a ontologia global, os esquemas locais devem passar pelas etapas de conversão do XML Schema e a integração semântica. Maiores detalhes sobre a concepção e implementação da OntoGen podem ser lidos em (SACCOL et al., 2008) e (MELLO, 2007).

Uma vez a ontologia representativa do domínio de conhecimento tenha sido definida, a tarefa seguinte passa a ser a criação da estrutura lógica relacional de banco de dados. A ferramenta OntoRel, apresentada na seção 2.5 a seguir, possui a finalidade de criar o esquema do banco de dados.

2.5 OntoRel

A ferramenta OntoRel é responsável por criar o esquema do banco de dados a partir de uma ontologia. Tendo a saída da ontologia no formato OWL criada pela OntoGen, a tarefa de OntoRel é, através de um conjunto de regras de mapeamento, transformar a ontologia para um conjunto de tabelas. Dessa forma, realiza-se o posterior armazenamento de documentos XML descritos pela ontologia em uma base de dados relacional, sendo esta tarefa atribuída à XMap.

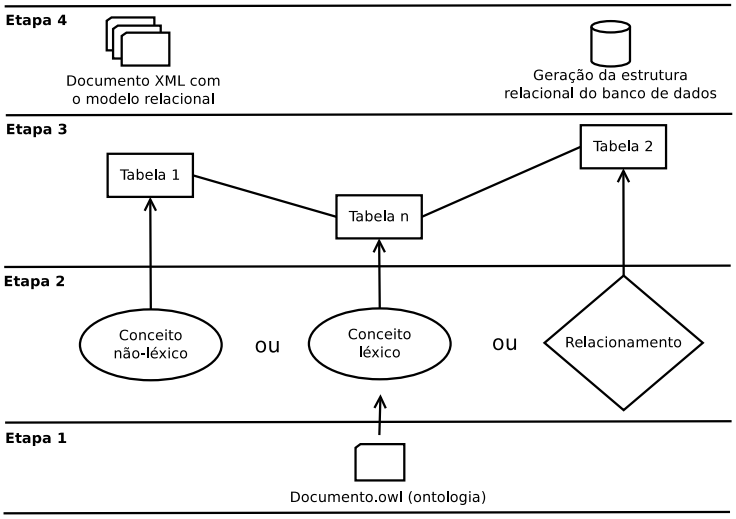

Figura 2.4: Arquitetura da ferramenta OntoRel (ANDRADE, 2010) (adaptado).

Como representado na Figura 2.4, ao receber a ontologia OWL de entrada, a OntoRel passa a criar objetos dos tipos especificados abaixo:

- Conceito não-léxico: refere-se a objetos sem representação direta em computador (objetos complexos). Por exemplo, o objeto artigo é composto de um ou mais autores;

- Conceito léxico: é um objeto diretamente representável em computador através de cadeias de bits (inteiros, cadeias de caracteres, etc.) Por exemplo, um autor de um artigo possui uma idade, sendo esta representada diretamente no computador como um número inteiro;

- Relacionamento: corresponde às cardinalidades direta e inversa entre os conceitos da ontologia. Utilizado para determinar as chaves primárias e estrangeiras, além de campos `null` e `not null`.

Na próxima etapa, aplicam-se as regras de transformação e armazenam-se as informações em objetos do tipo tabela. Na última etapa, tendo todas essas informações, é construído o arquivo XML de saída representativo da estrutura lógica relacional e o *script* SQL que pode ser editado e eventualmente executado direto no banco de dados. A concepção e a implementação da ferramenta OntoRel podem ser encontradas em (ANDRADE, 2010) e (SACCOL; PIVETA; ANDRADE, 2011).

De forma a contextualizar a pesquisa deste trabalho com as abordagens existentes na literatura, a seção 2.6 possui o propósito de avaliar os trabalhos relacionados sobre frameworks e sobre classificação e resolução de conflitos.

2.6 Trabalhos relacionados

Para armazenar documentos XML em um banco de dados relacional, pode-se utilizar uma abordagem baseada no mapeamento da estrutura do documento para a estrutura do banco de dados. No entanto, para adotar esta solução é desejável que os arquivos XML estejam de acordo com um esquema único. Desse modo, o esquema poderia ser traduzido para um conjunto de tabelas, usando algum dos métodos de mapeamento existentes (DAYEN, 2001), (ATAY et al., 2007) e (WU; HUANG, 2009). No entanto, essa situação não é o que normalmente ocorre; é comum encontrar arquivos XML relacionados a um esquema próprio ou até mesmo relacionados a nenhum esquema. Sendo assim, a integração de esquemas pode ser uma solução satisfatória, solução adotada no framework X2Rel.

Para resolver o problema de heterogeneidade, este trabalho apresenta uma abordagem baseada em ontologias, que pode ser utilizada para o aperfeiçoamento das técnicas tradicionais de mapeamento e de armazenamento de dados XML em um formato relacional. A finalidade do uso de ontologias é tornar explícito o conteúdo dos recursos, independente de como a informação é armazenada estruturalmente. Assim, essa abordagem acrescenta semântica aos arquivos, enquanto reduz a informação de estrutura desnecessária para mapear os dados XML para o formato relacional.

O mapeamento de arquivos XML para bancos de dados relacional não é uma questão nova, mas ainda há uma falta de algoritmos adequados para transformar conceitos de ontologias em um esquema relacional. Embora haja algumas técnicas conhecidas para mapear XML para a estrutura relacional, apenas algumas delas são baseadas no mapeamento da ontologia que descreve os documentos. No contexto deste trabalho, a diversidade de estrutura de arquivos XML não permite usar técnicas de mapeamento que se baseiam na estrutura individual dos documentos.

De modo a estabelecer uma visão geral dos trabalhos relacionados, a seção 2.6.1 apresenta as pesquisas à nível do framework X2Rel. A seção 2.6.2 aponta os trabalhos realizados na classificação e na resolução de conflitos decorrentes do processo de mapeamento de dados XML para a estrutura relacional.

2.6.1 Frameworks para armazenamento de conteúdo XML em banco de dados relacionais

Aplicações devem desenvolver mecanismos de integração de dados que permitam consultas em um conjunto de dados XML, como na Internet.

Um framework XML em (POGGI; ABITEBOUL, 2005) aborda a integração de dados XML fundamentado em (i) um esquema global especificado por DTD e por um conjunto de restrições XML definidos em (FAN; LIBKIN, 2002); (ii) o esquema das fontes de dados definido em DTD; e (iii) um conjunto de mapeamentos LAV especificados por uma linguagem prefixada

de seleção inspirado em (ABITEBOUL; SEGOUFIN; VIANU, 2006). Além disso, uma função de identificação é definida para identificar dados originados de diferentes fontes.

A abordagem LAV consiste em caraterizar o conteúdo das fontes de informação em termos do esquema global. No objetivo de retornar o resultado de uma consulta realizada no esquema global, o sistema precisa da especificação de relacionamentos entre as instâncias de dados e o esquema global em um processo de mapeamento. O esquema global atua como um indexador das instâncias de dados na recuperação da informação requisitada por uma consulta.

O modelo de dados orientado a objetos chamado XSDM (MADRIA; PASSI; BHOWMICK, 2008) utiliza uma arquitetura de três camadas para a integração de dados XML denominadas pré-integração, comparação e integração. Durante a pré-integração, os esquemas XML são convertidos no formato XSDM e, na comparação, conflitos são identificados. Finalmente, na integração, a resolução de conflitos é realizada.

A proposta XSDM apresenta um mecanismo na qual a consulta definida pelo usuário no esquema global é convertida em versões locais de consulta para serem executadas em esquemas locais. Os resultados são integrados antes de serem apresentados para o usuário. O esquema integrado forma a base para uma linguagem correta de consulta sobre um conjunto particular de documentos XML. Conhecer a estrutura global de todos os documentos XML facilita a validação das consultas em um conjunto particular de documentos XML.

2.6.2 Abordagens para o mapeamento e resolução de conflitos

As pesquisas que têm sido realizadas tratam o caso do mapeamento de XML para OWL, como em (BOHRING; AUER, 2005), (LEHTI; FANKHAUSER, 2004), (MELLO, 2007) e (BOZZI; MELLO, 2011), ao passo que a transformação de OWL para SQL foi abordada em (ASTROVA; KORDA; KALJA, 2007), (VYSNIAUSKAS; NEMURAITE, 2006) e (ANDRADE, 2010). A tradução direta de XML para SQL, como apresentando em (MARTINS; LAENDER, 2005), usa mapeamentos sem a adoção de um modelo global intermediário em OWL. Como consequência disso, resoluções de conflitos e replicação de instâncias, por exemplo, possuem soluções para um determinado conjunto de arquivos.

Dentre as abordagens clássicas de mapeamento de documentos XML para bancos de dados relacionais, destaca-se o mapeamento por aresta e *inlining* (FLORESCU; KOSSMANN, 1999). O mapeamento por aresta usa duas tabelas, sendo uma tabela chamada `Aresta` destinada ao armazenamento único de todos os documentos, e uma tabela `V` para armazenamento dos valores de cada tipo. De outro modo, o mapeamento *inlining* armazena o documento e os valores em uma única tabela, podendo haver duas variações: (i) uma coluna para cada tipo de valor e (ii) uma única coluna para todos os tipos de valores, sendo os valores convertidos para `string`.

(MELLO, 2002) propõe uma abordagem para a integração semântica de esquemas XML

relativos a um domínio de aplicação chamada BInXS. A abordagem adota um processo ascendente de integração, no qual o esquema global é definido para um conjunto de esquemas XML representados através do padrão DTD.

O processo de integração é baseado em um conjunto de regras e de algoritmos que realizam a conversão de cada DTD para um esquema canônico conceitual e a posterior integração semântica propriamente dita desses esquemas semânticos. O processo é semi-automático, pois considera uma eventual intervenção de um usuário especialista do domínio para validar ou confirmar alternativas de resultado produzidas automaticamente.

Em razão da complexidade no mecanismo de mapeamento, (LEGLER; NAUMANN, 2007) apresenta uma classificação de mapeamentos ao associar esquemas XML. Ao invés de utilizar DTD, (LEEO et al., 2002) classifica os conflitos na integração de esquemas XML e propõe um mecanismo de resolução utilizando XQuery. Além disso, o trabalho apresenta uma taxonomia de técnicas para resolver conflitos.

Para esquemas semi-estruturados, como documentos XML em geral, a abordagem utilizada em (KEDAD; XUE, 2005) realiza uma divisão em mapeamentos parciais com menor complexidade para então executar o mais abrangente. Tal técnica decompõe o documento em subárvores, assim conhecida pela estratégia de “dividir para conquistar”. Para resolver conflitos, realiza-se operações de junção representadas por um grafo não-direcionado acíclico juntamente com algoritmos de manipulação de tais estruturas de dados e de mapeamento.

O padrão XML pode ser utilizado como uma maneira de estruturar os dados para intercâmbio de informações entre sistemas. Em razão disso, (DO; RAHAYU; TORABI, 2011) propõem mapear os dados armazenados em banco de dados para documentos XML. O artigo classifica diversos conflitos em função da transformação dos dados, os quais foram de interesse ao longo desta dissertação.

2.6.3 Comparativo dos trabalhos relacionados

Diversas abordagens foram desenvolvidas para mapear conteúdo XML para bancos de dados relacionais. Conforme apresentado na seção 2.6.1, frameworks reúnem as etapas necessárias no processo de mapeamento. No esforço de resolver os conflitos existentes, a seção 2.6.2 apontou as pesquisas que classificam e propõem soluções para o problema.

Dentre as abordagens discutidas, a transformação direta do conteúdo XML para banco de dados considera as diferenças de cada documento XML. Nesse caso, o processo de criação do esquema lógico relacional terá maior complexidade e estará mais restrito a um conjunto de dados estruturalmente semelhantes.

De outra maneira, trabalhos usaram de ontologias para abstrair as diferenças entre documentos XML. Desse modo, foram propostas abordagens para transformar as fontes de dados

XML para uma ontologia representativa do domínio de conhecimento. Uma vez definida uma ontologia, outras abordagens propuseram converter uma ontologia para a criação da estrutura lógica de bancos de dados relacionais.

Os trabalhos relacionados apresentaram soluções para a criação da estrutura lógica relacional, seja diretamente a partir de documentos XML, seja com a adoção de ontologias. No caso particular de propostas que partem estritamente de ontologias, sem prévio conhecimento das fontes XML, a solução resultante envolve somente a criação da estrutura relacional. A saber, ontologias são um mecanismo de abstração da estrutura de documentos XML. Não se considera o conteúdo de cada documento XML, mas sim somente seus elementos e atributos.

Este trabalho busca usar a ontologia e o esquema lógico correspondente já existentes para realizar o mapeamento e a posterior inserção dos dados XML em um banco de dados relacional. Convém destacar que a abordagem descrita neste trabalho possibilita o armazenamento de dados XML estruturalmente heterogêneos em um banco de dados relacional descrito por um único esquema lógico. Dessa forma, a heterogeneidade dos documentos XML é abstraída para o usuário, que passa a enxergar a coleção de documentos como uma única base de dados descrita por uma mesma estrutura.

Além disso, este trabalho faz parte do ambiente X2Rel, um framework para armazenamento e consulta de documentos XML heterogêneos armazenados em um banco de dados relacional. Assim, ao mesmo tempo em que o framework considerado delimita o escopo do trabalho, também impõe restrições quanto a algumas decisões de projeto que precisam ser tomadas. Desta forma, este trabalho propõe um processo de mapeamento e armazenamento de documentos XML no banco de dados relacional previamente gerado. Para realizar tal mapeamento, uma série de conflitos estruturais e semânticos precisa ser solucionada. As etapas necessárias para o processo de migração dos dados XML para o formato relacional devem estar integradas em um ambiente único. No contexto deste trabalho, o framework X2Rel centraliza e integra essas etapas necessárias para possibilitar a migração.

2.7 Considerações finais

O uso do tipo de dados de cadeias de caracteres para armazenamento de documentos XML em banco de dados não representa uma solução adequada do ponto de vista da estrutura XML. O armazenamento e o processamento de documentos XML individuais em bancos de dados acrescenta maior complexidade no gerenciamento da informação. Campos específicos para armazenamento de documentos XML em bancos de dados relacionais, como `XMLType`, consideram a estrutura particular de cada documento XML para recuperação da informação. Desse modo, em razão da ausência de um modelo global como abstração das diferenças entre documentos XML heterogêneos, uma única consulta na base de dados pode não ser suficiente na busca da informação.

De maneira a estabelecer uma abstração da diversidade estrutural de documentos XML, a ontologia, definida na linguagem OWL, foi a alternativa encontrada por ser um solução compatível com XML. O usuário seleciona um conjunto de documentos XML pertencentes a um mesmo domínio de aplicação para que o componente OntoGen do framework X2Rel gere a ontologia correspondente. De maneira sistemática, o usuário seleciona um conjunto de documentos XML como pertencentes a um contexto como entrada de dados para a OntoGen compor a ontologia representativa.

O framework X2Rel é o ambiente integrante das etapas de mapeamento e armazenamento do conteúdo XML em banco de dados. Considerando a arquitetura de X2Rel, o escopo deste trabalho foi definido pelo módulo XMap. Essencialmente, a tarefa de XMap é processar os documentos XML utilizados para a criação da ontologia para armazenar o conteúdo em bancos de dados relacionais. Para isso, XMap utiliza a estrutura relacional criada pela ferramenta OntoRel a partir da ontologia. Espera-se, como artefato de saída do componente XMap, um *script* SQL com uma coleção de comandos de inserção de dados no banco de dados relacional. Para tal, conflitos precisam ser resolvidos de maneira a solucionar a heterogeneidade dos documentos de entrada.

Por fim, os trabalhos relacionados e comparados com esta dissertação apresentam as abordagens estabelecidas para resolver os problemas existentes no processo de armazenamento do conteúdo XML em bancos de dados relacionais.

3 XMAP: MAPEAMENTO DE DOCUMENTOS XML PARA BANCOS DE DADOS RELACIONAIS

Os capítulos anteriores abordaram a existência de documentos XML distintos e a justificativa de utilizar um modelo global para a busca otimizada de informação. Para isso, foram apresentados os conceitos e tecnologias decorrentes do processo de mapeamento e armazenamento do conteúdo XML para bancos de dados relacionais. Partindo-se da arquitetura do framework X2Rel, afirma-se que a proposta deste trabalho está em desenvolver o módulo XMap para mapear dados dispostos em XML para bancos de dados relacionais.

3.1 Visão geral

Sistemas gerenciadores de banco de dados relacionais oferecem recursos amplamente utilizados e testados. A proposta de armazenar documentos XML em uma única representação lógica facilita o acesso aos dados por parte do usuário, de modo que diferentes representações estruturais possam estar integradas em um mesmo esquema lógico.

A definição do domínio de conhecimento em uma ontologia será utilizada como a descrição conceitual de um domínio de conhecimento. Trata-se de uma abordagem para mapear instâncias de dados XML para o esquema lógico relacional de SGBDR. Para tanto, o módulo OntoGen cria a ontologia OWL a partir de um conjunto de arquivos XML. Posteriormente, o módulo OntoRel gera o esquema lógico com instruções em DDL para o banco de dados relacional.

Por outro lado, uma vez tendo mapeado e armazenado os dados das instâncias XML pertencentes a um domínio para um banco de dados, o processo de sincronização de dados será assumido como existente. Em outras palavras, se o banco de dados relacional sofrer modificações, tais como inclusão, alteração ou exclusão de registros, a devida propagação das alterações para as fontes originais XML assume-se que seja realizada de forma automática. Da mesma forma, se alguma alteração nas fontes de dados XML for realizada posteriormente ao mapeamento para o banco de dados, será considerado que a propagação das alterações será efetuada de modo automático no esquema lógico relacional. Esse mecanismo, conhecido como atualização de visões em banco de dados, é tema de um trabalho de dissertação de pós-graduação à parte, inclusive tendo sido assunto de pesquisa na tese de doutorado (BRAGANHOLO, 2004).

Em razão de XMap usar o esquema lógico relacional gerado por OntoRel, o qual, por sua vez, usa a ontologia OWL criada por OntoGen, as seções 3.1.1 e 3.1.2 apresentam um exemplo prático para a compreensão do funcionamento de ambas as ferramentas. A partir de então, XMap será mais facilmente entendido juntamente com os demais componentes de X2Rel.

3.1.1 Funcionamento da ferramenta OntoGen

De modo a exemplificar os arquivos XML de entrada necessários para a execução da OntoGen, a Figura 3.1 ilustra três arquivos XML que possuem dados a respeito de publicações de autores e instituições.

Apesar de `Doc_A.xml`, `Doc_B.xml` e `Doc_C.xml` pertencerem a um mesmo domínio estabelecido, o qual corresponde a dados de publicações em eventos, percebem-se diferenças estruturais entre os arquivos. Algumas dessas diferenças são descritas a seguir:

- A raiz do documento `A` e do documento `B`, ambos na linha 2 da Figura 3.1, corresponde ao elemento `<conferences>`, ao passo que no documento `C`, linha 2, trata-se de `<papers>`. Sob outro aspecto, nos dois primeiros documentos, um `<paper>` (linha 7 no documento `A` e linha 8 no documento `B`) está em um `<conference>` (linha 3 nos documentos `A` e `B`). No terceiro documento, um `<conference>` (linha 5) está em um `<paper>` (linha 3);
- O documento `B`, em relação aos demais, possui duas propriedades distintas para `<conference>` (linha 3), os quais são `<city>` (linha 5) e `<state>` (linha 6);
- Nos documentos `A` e `C`, ambos na linha 10, o conceito autor é denominado `<author>` e no documento `B`, também na linha 10, esse mesmo conceito é denominado `<writer>`;
- No documento `A`, linha 10, o conceito autor é um conceito léxico e nos demais esquemas esse conceito é não-léxico (linha 10 do documento `B`), com o elemento `<writer>`, e na linha 10 do documento `C`);
- O conceito não-léxico `<institution>` está presente nos documentos `A` e `B`, linha 12 e 14, respectivamente, mas não está presente no documento `C`.

Para armazenar o conteúdo dos arquivos `Doc_A.xml`, `Doc_B.xml` e `Doc_C.xml` em um banco de dados relacional, é necessário existir um esquema global, de modo a reunir as distinções estruturais entre os documentos. Assumindo os arquivos de exemplo como entrada para o processamento pela ferramenta OntoGen, a Figura 3.2 ilustra a ontologia OWL resultante.

A ontologia global resultante reúne os atributos presentes nos arquivos XML representados na Figura 3.1. Em razão de alguns elementos não estarem presentes em todos os documentos, eles passam a ter cardinalidade mínima igual a zero. Tal fato ocorre com os campos `city` e `state`, por exemplo, pois esses fazem parte do documento `B`. Por outro lado, o campo `title`, seja este de `conference` ou de `paper`, é comum a todos os documentos XML, assim possuindo cardinalidade mínima igual a um. Além disso, conforme anteriormente constatado, nos documentos `A` e `C`, o conceito autor é denominado `<author>`. No esquema `B`, no entanto, este mesmo conceito é denominado `<writer>`.

Doc_A.xml

```
01 <?xml version="1.0" encoding="utf-8" ?>
02 <conferences>
03   <conference>
04     <title>XX SBBD</title>
05     <day>01/01/2005</day>
06     <papers>
07       <paper>
08         <title>Temporal Versioned Constraint Language</title>
09           <authors>
10             <author>Robson Mendes</author>  ▪▪▪
11           </authors>
12           <institution>
13             <name>Instituto de Informática</name>
14             <city>Porto Alegre</city>
15             <state>RS</state>
16           </institution>
17       </paper> ▪▪▪
18     </papers>
19   </conference> ▪▪▪
20 </conferences>
```

Doc_B.xml

```
01 <?xml version="1.0" encoding="utf-8" ?>
02 <conferences>
03   <conference>
04     <title>XX Simposio Brasileiro de Banco de Dados</title>
05     <city>Belo Horizonte</city>
06     <state>MG</state>
07     <papers>
08       <paper>
09         <title>TVCL - Temporal Versioned Constraint Language</title>
10         <writer code="1">
11           <name>Edson Marques</name>
12           <e_mail>emarques@inf.ufsm.br</e_mail>
13         </writer>
14         <institution>
15           <name>UFSM</name>
16           <department>Instituto de Informática</department>
17         </institution>
18       </paper> ▪▪▪
19   </conference> ▪▪▪
20 </conferences>
```

Doc_C.xml

```
01 <?xml version="1.0" encoding="utf-8" ?>
02 <papers>
03   <paper>
04     <title>TVCL</title>
05     <conference>
06       <title>Simposio Brasileiro de BD</title>
07       <day>01/07/2011</day>
08     </conference>
09     <authors>
10       <author>
11         <code>2</code>
12         <name>Carlos Souza</name>
13         <e_mail>csouza@inf.ufsm.br</e_mail>
14         <age>28</age>
15       </author> ▪▪▪
16     </authors>
17   </paper> ▪▪▪
18 </papers>
```

Figura 3.1: Documentos XML referentes à publicação de artigo em evento. O sinal de reticências indica que entidades repetem-se estruturalmente no documento XML.

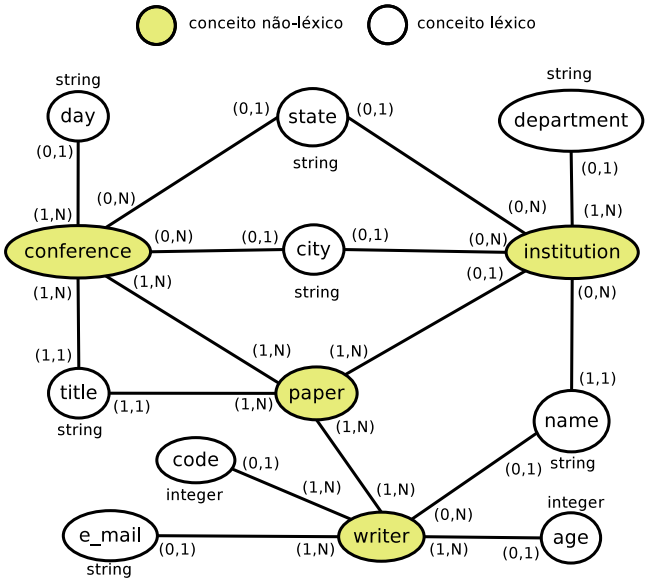

Figura 3.2: Ontologia OWL global gerada pela ferramenta OntoGen tendo como entrada os arquivos XML da Figura 3.1.

Analisando a ontologia resultante, a nomenclatura que passou a ser utilizada foi `writer`, tanto para designar `<author>`, como para `<writer>` nos documentos XML. Tal fato justifica-se

de acordo com a função de afinidade da OntoGen utilizada para agrupar termos sinônimos. Se os nomes dos conceitos forem iguais, ou forem sinônimos, segundo um dicionário de sinônimos, esses dois conceitos serão considerados semanticamente iguais e a função de afinidade retornará valor igual a um. Caso contrário, a função de afinidade retornará zero.

A proposta de adoção de ontologias, a fundamentação teórica e a implementação do módulo OntoGen do framework X2Rel foi tema de trabalho proposto por (MELLO, 2007) e (SACCOL et al., 2008). O WordNet (WORDNET, 2011) foi o dicionário de sinônimos utilizado pela OntoGen para a determinação de termos equivalentes.

3.1.2 Funcionamento da ferramenta OntoRel

Após o processamento dos arquivos XML de entrada pela ferramenta OntoGen, o resultado é um arquivo textual contendo a ontologia no formato OWL (Figura 3.2). A OntoRel utilizará esse arquivo como entrada para criar o esquema lógico relacional ilustrado na Figura 3.3.

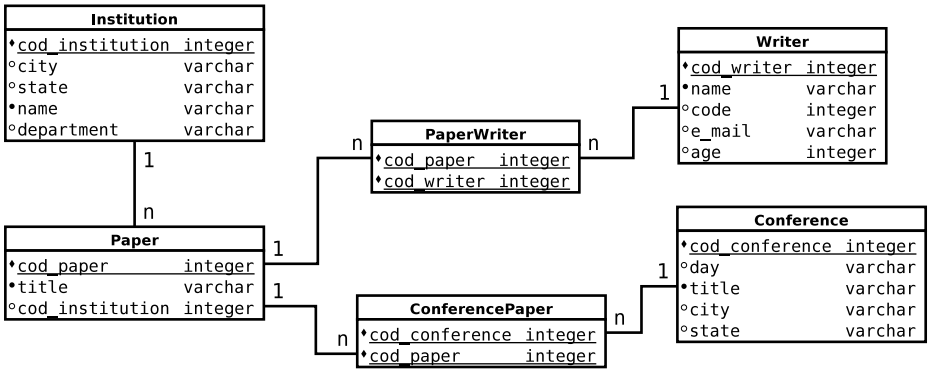

Figura 3.3: Representação gráfica do esquema lógico relacional criado pela ferramenta OntoRel.

Analisando a ontologia com o esquema lógico relacional, pode-se constatar que os conceitos não-léxicos passaram a ser tabelas no banco de dados. As tabelas `ConferencePaper` e `PaperWriter` são resultado da proposta de geração de esquema relacional de OntoRel. Se a cardinalidade for N:N, observando-se a ontologia da Figura 3.2, é criada uma nova tabela de acordo com o relacionamento, a qual possui como nome a concatenação dos dois conceitos da relação. A tabela resultante contém uma chave primária constituída pela coluna com o mesmo nome e tipo da chave primária da tabela de origem e pela coluna com o mesmo nome e tipo da chave primária da tabela de destino (ANDRADE, 2010). Os conceitos léxicos são atributos de entidades. Os campos cuja cardinalidade mínima é igual a um, naturalmente passam a ser diferente de nulo no banco de dados, pois são elementos comuns às instâncias XML.

A ferramenta OntoRel gera como saída o *script* SQL de criação das tabelas em banco de dados relacional. Além disso, a OntoRel cria um documento XML representando o esquema lógico relacional. Em especial, a versão em XML é de interesse para XMap para facilitar o

processo de mapeamento para o armazenamento do conteúdo XML em bancos de dados relacionais. A adoção do documento XML do esquema lógico relacional ao invés da ontologia justifica-se no mapeamento dos documentos XML para banco de dados relacionais. A ontologia abstrai as diferenciações de documentos XML. Por outro lado, o esquema XML gerado pela ferramenta OntoRel representa o esquema de armazenamento em banco de dados determinado pela ontologia. O Apêndice A contém o esquema XML criado pela ferramenta OntoRel conforme o esquema lógico da Figura 3.3.

A proposta de adoção de ontologia para a criação do esquema lógico relacional de banco de dados, a fundamentação teórica e a implementação do módulo OntoRel do framework X2Rel foi tema de trabalho à parte (ANDRADE, 2010) e (SACCOL; PIVETA; ANDRADE, 2011).

3.2 Arquitetura proposta

Partindo da arquitetura do framework X2Rel, o módulo XMap passa a ser o foco de interesse, como ilustrado na Figura 3.4. Considerando os documentos XML, a correspondente ontologia de abstração e o respectivo esquema lógico relacional, a tarefa de XMap é gerar o *script* de inserção do conteúdo XML em linguagem SQL DML para execução em um banco de dados relacional. Para isso, é necessário o uso de um documento de mapeamento e do esquema XML do esquema relacional. O documento de mapeamento será apresentado mais adiante na Seção 3.4, sendo criado de maneira automatizada por CMap (NEGRINI, 2011).

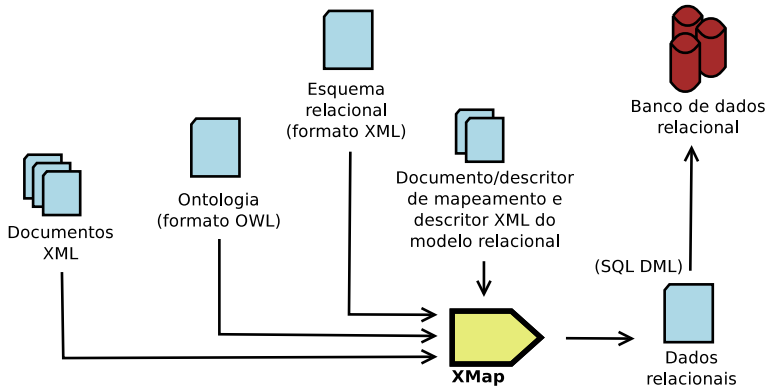

Figura 3.4: Módulo XMap situado na arquitetura do framework X2Rel.

De maneira a propor uma abordagem para o armazenamento de instâncias XML em banco de dados relacionais, a arquitetura do sistema deve solucionar os principais conflitos decorrentes da transformação de modelos. De acordo com a Figura 3.5, que ilustra a arquitetura do componente XMap, o primeiro nível (carregamento dos dados) corresponde a entrada dos arquivos necessários para o mapeamento dos dados XML para o banco de dados relacional, além do próprio conjunto de instâncias de dados XML. Os arquivos auxiliares são fundamentais no processo de mapeamento, sendo a função descrita a seguir:

- Documento XML do banco de dados: arquivo XML do esquema lógico relacional de banco de dados criado pela ferramenta OntoRel;
- Descritor do documento XML do banco de dados: arquivo auxiliar criado pelo programador em razão da biblioteca Castor utilizada em XMap.
- Documento de mapeamento: arquivo XML de mapeamento representando a integração das instâncias de dados XML, da correspondente ontologia criada pela ferramenta OntoGen e, consecutivamente, do esquema lógico relacional criado pela ferramenta OntoRel. O documento de mapeamento será apresentado com mais detalhes na Seção 3.4.
- Descritor do documento de mapeamento: arquivo auxiliar no mecanismo de mapeamento em razão da biblioteca Castor.

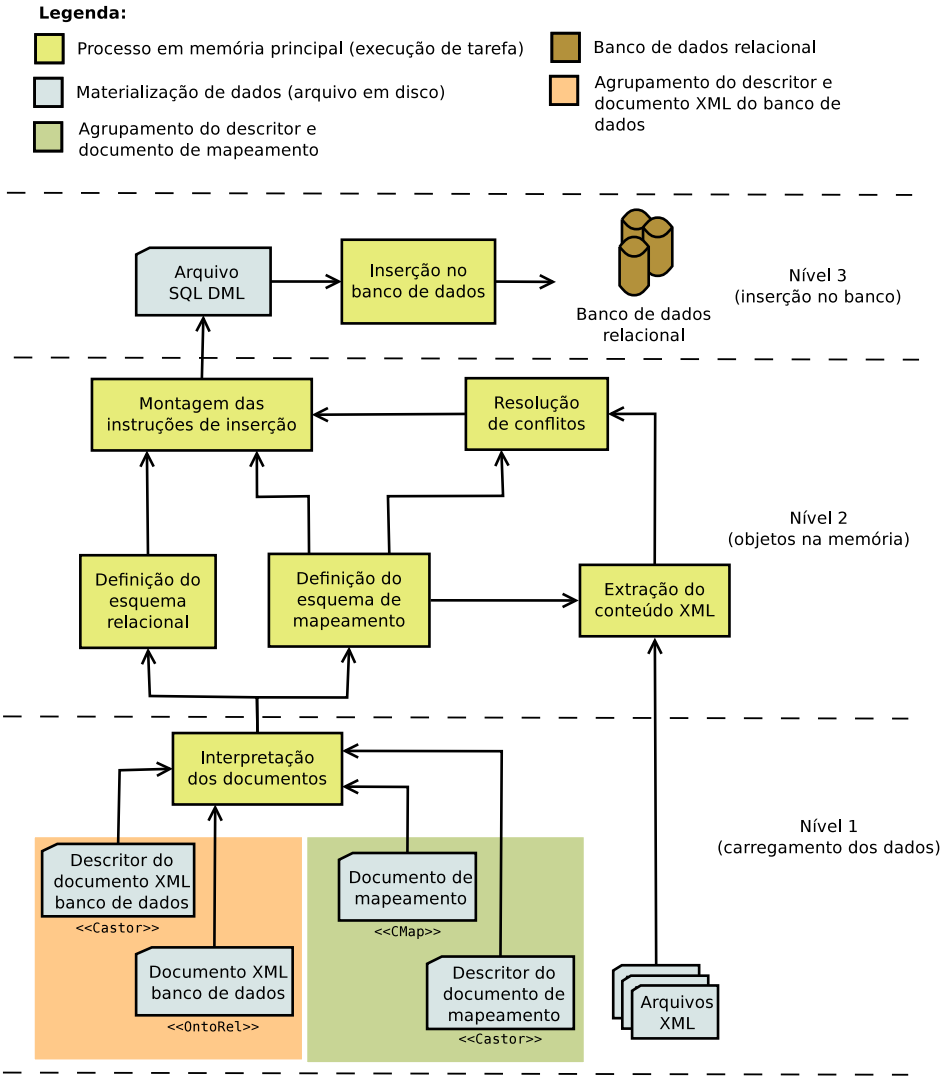

Figura 3.5: Arquitetura proposta para o módulo XMap.

O processo referente à interpretação dos documentos, ainda no primeiro nível da Figura 3.5, possui a tarefa de verificar os caminhos de armazenamento do documento de mapeamento

e do esquema XML do banco de dados, assim como os respectivos descritores. A partir disso, o resultado é a interpretação dos descritores para carregar em memória principal o documento de mapeamento e o esquema XML do banco de dados, na forma de objetos[1].

A criação do documento de mapeamento foi proposto de maneira semiautomática (NEGRINI, 2011). Entretanto, a concepção teórica e estrutural do documento de mapeamento associando os documentos XML, ontologia e o esquema relacional é uma contribuição deste trabalho. Nesta abordagem, foram utilizados regras e garantias dos processos de unificação e transformação feitos pela OntoGen e OntoRel para gerar e documentar as equivalências entre esquemas. Como resultado, o documento de mapeamento em XML é utilizado por XMap, juntamente com o descritor XML, como documento de apoio no processo de mapeamento do conteúdo de arquivos XML para armazenamento em bancos de dados relacionais. Além disso, o esquema lógico relacional escrito em XML criado pela ferramenta OntoRel também faz parte da etapa inicial de carregamento dos dados, pois descreve a organização do esquema relacional cujo interesse é armazenar o conteúdo dos arquivos XML.

Ainda observando a Figura 3.5, o segundo nível (objetos em memória) possui os processos necessários para o mapeamento do conteúdo XML presente na memória. Como se trata de um nível de execução de tarefas, cada processo foi discriminado como segue:

- A definição do esquema relacional possui a funcionalidade de acesso ao documento XML do esquema relacional por meio de métodos de encapsulamento. A entrada corresponde ao esquema XML do banco de dados como objeto em memória. A saída são as entidades e os respectivos atributos, necessários para a montagem das instruções de inserção de conteúdo em banco de dados.

- A definição do esquema de mapeamento é responsável por acessar as estruturas de mapeamento, como os conceitos da ontologia, as fontes de dados XML, as entidades e atributos do esquema relacional. Ela recebe como entrada o documento de mapeamento como objetos em memória. A saída são os caminhos XPath que apontam para o conteúdo das fontes de dados XML, os conceitos da ontologia e as entidades com os respectivos campos.

- A extração do conteúdo XML possui a finalidade de interpretar os caminhos XPath para busca de conteúdo nos documentos XML. Ela recebe como entrada os caminhos XPath na forma de objetos do documento de mapeamento, sendo a saída o conteúdo XML apontado por cada caminho associado às fontes de dados.

- A resolução de conflitos é responsável pelo tratamento dos conflitos que podem ocorrer no mapeamento do conteúdo XML para armazenamento em banco de dados. Ela possui

[1] Paradigma de orientação a objetos.

como entrada as estruturas do documento de mapeamento e o conteúdo XML apontado pelos caminhos XPath. A saída é a organização do conteúdo XML após a detecção e a solução dos conflitos detectados.

- A montagem das instruções de inserção possui o propósito de criar as instruções de inserção em linguagem SQL, respeitando as regras referenciais de banco de dados. Ela associa, como entrada, o esquema lógico relacional, as definições do documento de mapeamento e o conteúdo das instâncias XML após a resolução de conflitos. A saída é o *script* de inserção do conteúdo XML no banco de dados relacional.

O terceiro nível (inserção no banco) é responsável por inserir os dados no banco de dados relacional. As instruções de inserção em banco de dados, definidas em arquivo, é carregada em memória no processo de inserção no banco de dados. A partir de então, a inserção propriamente dita do conteúdo das instâncias XML iniciais é realizada em uma base de dados relacional.

3.3 Categorização de conflitos

O processo de mapeamento para armazenar o conteúdo de documentos XML no banco de dados utiliza um esquema global de organização, sendo definido pelo esquema lógico relacional originado a partir da ontologia. Conflitos podem ocorrer para adequar o conteúdo XML para a base centralizada de dados. Desse modo, uma classificação dos conflitos existentes é uma maneira conveniente para a proposta de solução em particular.

Os principais conflitos que podem ocorrer na transformação dos dados XML para o esquema relacional são descritos em (DO; RAHAYU; TORABI, 2011) e foram adaptados para as particularidades deste trabalho, sendo descritos nas seções a seguir. Os algoritmos propostos para a resolução dos conflitos são apresentados a partir da Seção 3.5.

3.3.1 Conflitos de nomenclatura

Há dois tipos de conflitos de nomenclatura, sendo eles do tipo homonímia e sinonímia, conforme descritos a seguir. Para exemplificar os conflitos, são utilizados os documentos XML da Figura 3.1. De modo a facilitar a leitura, esses documentos são apresentados de maneira simplificada na Figura 3.6.

Partindo da visualização de documentos XML, o conflito de homonímia ocorre quando um mesmo nome de elemento é utilizado para dois ou mais conceitos diferentes. Analisando-se os documentos XML da Figura 3.6, deve-se interpretar o fragmento dos documentos `Doc_B.xml` e `Doc_C.xml`. No documento `B`, o elemento `<name>` na linha 2 refere-se ao nome de um `<writer>`. No documento `C`, `<name>` na linha 3 é o nome de um `<author>`. Em contrapartida,

Doc_A.xml (fragmento)

```
01   <institution>
02     <name>Instituto de Informática</name>
03     <city>Porto Alegre</city>
04     <state>RS</state>
05   </institution>
```

Doc_C.xml (fragmento)

```
01 <author>
02   <code>2</code>
03   <name>Carlos Souza</name>
04   <e_mail>csouza@inf.ufsm.br</e_mail>
05   <age>28</age>
06 </author>
```

Doc_B.xml (fragmento)

```
01 <writer code="1">
02   <name>Edson Marques</name>
03   <e_mail>emarques@inf.ufsm.br</e_mail>
04 </writer>
05 <institution>
06   <name>UFSM</name>
07   <department>Instituto de Informática</department>
08 </institution>
```

Figura 3.6: Fragmento dos documentos XML da Figura 3.1.

`<name>` é o nome de `<institution>` na linha 2 do documento `A` e na linha 6 do documento `B`. A inserção no banco de dados, vide Figura 3.7, ocorre diretamente, pois a OntoRel cria o campo `name` tanto para a tabela `Institution` como para a tabela `Writer`.

Doc_A.xml

```
INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1263, "Robson Mendes", null, null, null);
INSERT INTO Institution(cod_institution, city, state, name, department) VALUES
(2052, "Porto Alegre", "RS", "Instituto de Informática", null);
```

Doc_B.xml

```
INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1374, "Edson Marques", 1, "emarques@inf.ufsm.br", null);
INSERT INTO Institution(cod_institution, city, state, name, department) VALUES
(2982, null, null, "UFSM", "Instituto de Informática");
```

Doc_C.xml

```
INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1107, "Carlos Souza", 2, "csouza@inf.ufsm.br", 28);
```

Figura 3.7: Exemplo de inserção no banco de dados para o caso de conflito de homonímia considerando os arquivos XML da Figura 3.1.

Outra possibilidade de conflito, partindo da visualização de documentos XML, o conflito de sinonímia ocorre quando o mesmo conceito é descrito por dois ou mais nomes. A partir da Figura 3.6, deve-se novamente interpretar os documentos `Doc_B.xml` e `Doc_C.xml`. Em razão de, nos documentos `A` e `C`, o conceito autor ser denominado `<author>` e no documento `B` ser denominado `<writer>`, um conflito de sinonímia está presente para o caso de conceitos não-léxicos. Conforme visto na ontologia global da Figura 3.2, a nomenclatura utilizada no banco de dados foi `Writer` seja para um `<author>` ou para um `<writer>` nos documentos XML. A adoção de `Writer` está relacionada ao mecanismo de OntoGen para o tratamento de conceitos sinônimos. Para os documentos `A` e `C`, a inserção dos dados sobre autor será feita na tabela `Writer`.

3.3.2 Conflitos de estrutura

Conflitos de estrutura estão relacionados a diferentes escolhas de construtores de linguagem dos conceitos. Há dois tipos de conflitos de estrutura, o de tipo e o de generalização, conforme

descritos a seguir. Para exemplificar os conflitos, são utilizados os documentos XML da Figura 3.1. De modo a facilitar a leitura, esses documentos serão apresentados de maneira simplificada na Figura 3.8.

Doc_B.xml (fragmento)

```
01 <writer code="1">
02   <name>Edson Marques</name>
03   <e_mail>emarques@inf.ufsm.br</e_mail>
04 </writer>
05 <institution>
06   <name>UFSM</name>
07   <department>Instituto de Informática</department>
08 </institution>
```

Doc_C.xml (fragmento)

```
01 <author>
02   <code>2</code>
03   <name>Carlos Souza</name>
04   <e_mail>csouza@inf.ufsm.br</e_mail>
05   <age>28</age>
06 </author>
```

Figura 3.8: Fragmento dos documentos XML da Figura 3.1.

Visualizando a estrutura dos documentos XML, o conflito de estrutura de tipo ocorre quando um mesmo conceito é representado por construtores diferentes. De acordo com a Figura 3.8, a linha 1 do documento `B` possui `code` como atributo do elemento `<writer>`. O documento `C` na linha 2 possui o elemento `<code>` pertencente ao ancestral `<author>`. Nesse caso, como se trata de informação de interesse para armazenamento no banco de dados, a solução foi transformar `code` do documento `B`, assim como `<code>` do documento `C`, no correspondente campo na tabela `Writer`. A inserção no banco de dados passa a ser a mesma como indicado na Figura 3.7.

O conflito de estrutura de generalização ocorre quando um elemento em um documento é uma união de outros elementos em outro documento. Como exemplo, considera-se o elemento `<name>` na linha 2 do documento `B` e na linha 3 do documento `C` (Figura 3.8). Nesse caso, tanto o nome quanto o sobrenome fazem parte do mesmo elemento `<name>`. Um conflito de generalização ocorre se no documento `C`, por exemplo, o nome e o sobrenome estivessem em elementos separados, como mostrado na Figura 3.9, e o documento `B` permanecesse inalterado.

Doc_C.xml (fragmento)

```
01 <?xml version="1.0" encoding="utf-8" ?>
02 <papers>
03   <paper>
... ...
09     <authors>
10       <author>
11         <code>2</code>
12         <names>
13           <name>Carlos</name>
14           <surname>Souza</surname>
15         </names>
16         <e_mail>csouza@inf.ufsm.br</e_mail>
17         <age>28</age>
18       </author>
19     </authors>
20   </paper>
21 </papers>
```

Figura 3.9: Exemplo de conflito de generalização para uma versão hipotética de `Doc_C.xml` da Figura 3.1.

Na modificação do documento `C`, inclui-se o elemento `<names>` para agrupar o nome e o sobrenome do autor, sendo destaque a partir da linha 12 até a linha 15 na Figura 3.9. Comparando

com o documento B original na linha 11, é verificado que o nome e o sobrenome estão somente em um elemento. Pode-se constatar que <names> é um conceito não-léxico para o documento C adaptado, ao passo que <name> é um conceito léxico para o documento B inalterado. Para a inserção no banco de dados, uma solução para resolver o conflito de generalização é concatenar o conteúdo do elemento <name> e <surname> nas linhas 13 e 14 do documento C adaptado da Figura 3.9. Dessa forma, o esquema lógico relacional permanece a mesma. A operação de concatenação em documentos XML pode ser feita utilizado-se a função concat de XPath. A Figura 3.10 exemplifica o uso de XPath juntamente com a inserção no banco de dados.

Doc_A.xml

INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1263, "Robson Mendes", null, null, null);

Doc_B.xml

INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1374, "Edson Marques", 1, "emarques@inf.ufsm.br", null);

Uso de XPath em Doc_C.xml hipotético

concat(/papers/paper/authors/names/name, ' ', /papers/paper/authors/author/names/surname)

Doc_C.xml

INSERT INTO Writer(cod_writer, name, code, e_mail, age) VALUES
(1107, "Carlos Souza", 2, "csouza@inf.ufsm.br", 28);

Figura 3.10: Inserção em banco de dados com a concatenação do conteúdo dos elementos <name> e <surname> de Doc_C.xml hipotético.

Convenientemente, utiliza-se XPath para percorrer arquivos XML para buscar informação. Funções pertencentes ao padrão XPath, como a concatenação de cadeias de caracteres com a função concat, é uma maneira de resolver o conflito de generalização.

Existe o caso implementado na ferramenta OntoGen, que é a criação de um conceito não-léxico names em razão do elemento complexo <names> do documento C adaptado na linha 12. Tal fato remete à criação de uma tabela Names pela OntoRel, conforme mostra a Figura 3.11.

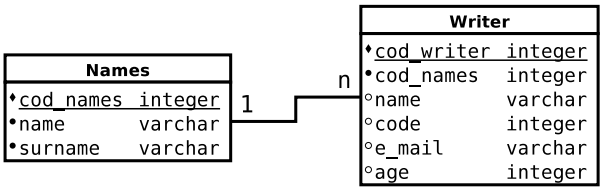

Figura 3.11: Alteração do esquema lógico relacional criada pela ferramenta OntoRel em razão da ontologia global gerada pela ferramenta OntoGen em função do Doc_C.xml adaptado da Figura 3.9.

Em razão da alteração do esquema lógico relacional devido a modificação do documento C, a inserção no banco de dados precisa ser reavaliada. Deve-se incluir na inserção a tabela Names conforme especificado na Figura 3.12. Devido à existência dos campos name e surname na tabela Names, é necessário separar em nome e sobrenome o conteúdo do elemento <author> do documento A na linha 10 e o conteúdo do elemento <name> do documento B na linha 11 ilustrados na Figura 3.1. O processo de separação de nome e sobrenome é feito com a função substring-before e substring-after de XPath, respectivamente.

Uso de XPath em Doc_A.xml
Para nome:
substring-before(/conferences/conference/papers/paper/authors/author, " ")
Para sobrenome:
substring-after(/conferences/conference/papers/papers/author, " ")

Doc_A.xml

INSERT INTO Names(cod_names, name, surname) VALUES
(328, "Robson", "Mendes");
INSERT INTO Writer(cod_writer, cod_names, name, code, e_mail, age) VALUES
(1263, 328, "Robson Mendes", null, null, null);

Uso de XPath em Doc_B.xml
Para nome:
substring-before(/conferences/conference/papers/paper/writer/name, " ")
Para sobrenome:
substring-after(/conferences/conference/papers/writer/name, " ")

Doc_B.xml

INSERT INTO Names(cod_names, name, surname) VALUES
(241, "Edson", "Marques");
INSERT INTO Writer(cod_writer, cod_names, name, code, email, age) VALUES
(1374, 241, "Edson Marques", 1, "emarques@inf.ufsm.br", null);

Uso de XPath em Doc_C.xml adaptado
concat(/papers/paper/authors/names/name, ' ', /papers/paper/authors/author/names/surname)

Doc_C.xml

INSERT INTO Names(cod_names, name, surname) VALUES
(252, "Carlos", "Souza");
INSERT INTO Writer(cod_writer, cod_names, name, code, email, age) VALUES
(1107, 252, "Carlos Souza", 2, "csouza@inf.ufsm.br", 28);

Figura 3.12: Inserção no banco de dados para a modificação no esquema lógica referencial da Figura 3.11.

Para o documento `C` adaptado da Figura 3.9, nome e sobrenome estão separados. Em razão de compatibilidade com o esquema lógico relacional referente à tabela `Writer`, a concatenação do conteúdo dos elementos `<name>` e `<surname>` é necessária para informar o nome completo do autor no campo `name`. Mesmo com a criação da tabela `Names`, o campo `name` da tabela `Writer` foi mantido para armazenar o nome completo de autor. A justificativa é manter compatível o armazenamento de conteúdo dos documentos cujos nomes de autor não estejam separados em nome e sobrenome.

3.3.3 Conflito de elemento inexistente

Uma vez tendo sido gerada a ontologia global OWL e o correspondente esquema lógico relacional, ao inserir o conteúdo dos arquivos XML iniciais no esquema relacional, pode surgir um conflito do tipo elemento inexistente. Tal situação acontece caso algum documento XML não possua um dado para armazenamento em banco de dados.

Para facilitar a visualização, a Figura 3.13 ilustra parcialmente a ontologia global representada graficamente na Figura 3.2 e o esquema lógico relacional da Figura 3.3. O documento `C` é replicado na Figura 3.13 de maneira integral conforme anteriormente ilustrado na Figura 3.1.

Verificando a Figura 3.13, que possui parte da ontologia global resultante dos arquivos XML de exemplo, a relação entre os conceitos `paper` e `institution` revela que um artigo possui, no máximo, uma instituição relacionada. Por consequência, o esquema lógico rela-

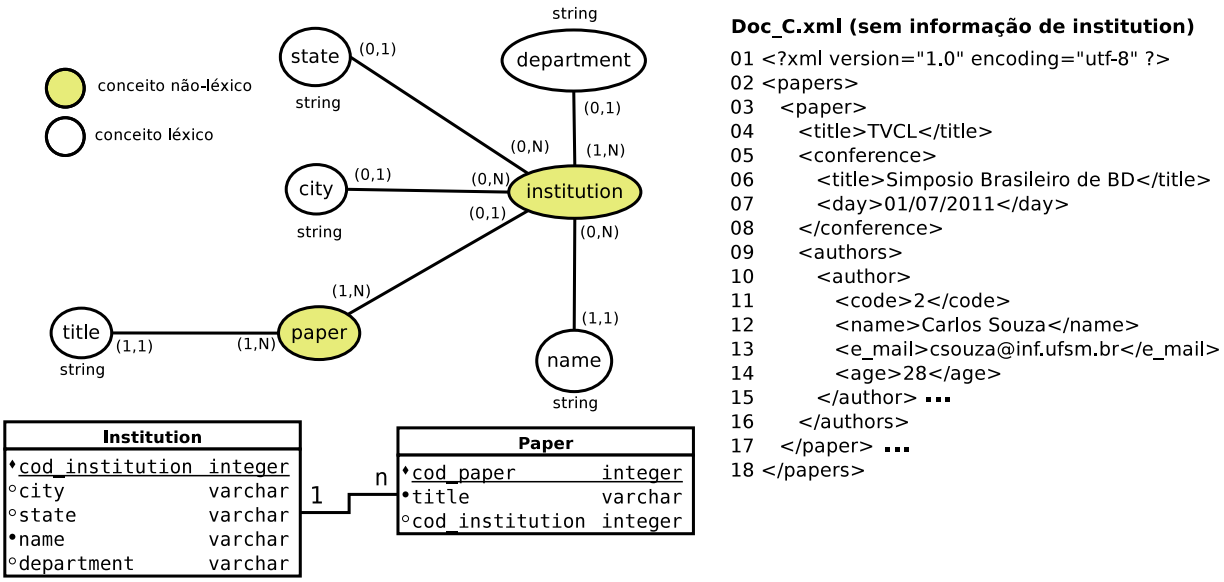

Figura 3.13: A ontologia global, o correspondente esquema lógico relacional e o documento `C` de exemplificação do conflito de elemento inexistente.

cional correspondente na Figura 3.13 apresenta uma relação 1:N entre as tabelas `Institution` e `Paper`.

O atributo `cod_institution` da tabela `Paper` é chave estrangeira da tabela `Institution`. Seguindo a Figura 3.13,ao tentar armazenar o conteúdo do documento `C`, haverá uma falha devido a ausência do elemento `<institution>`. Não há informação da instituição sobre a qual o artigo existe. Como solução, informações sobre `<paper>` são armazenadas na correspondente tabela, sendo que o campo `cod_institution` assumirá valor `null` na inserção no banco de dados.

3.3.4 Conflito de representação

Um conflito de representação pode ocorrer quando, em pelo menos um documento XML, um elemento possui o conteúdo representado de uma forma distinta do formato exigido para armazenamento em banco de dados. Mais especificamente, a informação existente em XML não é compatível com o tipo de dado correspondente no esquema relacional.

Ao interpretar os documentos XML para compor a ontologia, a ferramenta OntoGen estabelece regras de unificação de tipos, conforme exemplificado a seguir (MELLO, 2007). Ressalta-se que as regras são consideradas somente para conceitos léxicos.

- (date, date) $\rightarrow$ date
- (integer, float) $\rightarrow$ float
- (string, integer) $\rightarrow$ string

- (integer, date) $\rightarrow$ string

Interpretando as regras, é possível verificar que a unificação ocorre para o tipo mais abrangente de dados para evitar incompatibilidade de representação em computador. Em contrapartida, um exemplo a ser considerado é o conteúdo XML do elemento `<day>` na linha 7 de `Doc_C.xml` da Figura 3.13. Nesse caso, datas no formato `dd/mm/aaaa` são interpretadas como `strings` e, como consequência, o correspondente campo no esquema lógico relacional será do tipo `varchar`. De forma a armazenar o conteúdo do tipo data em questão, o formato deveria ser adaptado para a máscara `aaaa-mm-dd`, além do correspondente campo na tabela relacional ser alterado para o tipo `date` através da intervenção do usuário.

Através do documento de mapeamento, apresentado na Seção 3.4, a estratégia é analisar os dados do tipo `date` para verificar o formato presente no documento XML. A partir de então, o conflito de representação é resolvido modificando a representação da informação para o armazenamento correto em banco de dados.

Conforme mencionado na Seção 2.5, um conceito léxico é um objeto diretamente representável em computador através de cadeias de bits (inteiros, cadeias de caracteres, etc). No esquema lógico relacional, os atributos possuem tipos de dados associados, como `varchar`, `integer` e `date`, de maneira a armazenar o dado no computador de uma maneira representável. Em razão do conflito de representação envolver os tipos de dados representados em documentos XML para armazenamento em banco de dados relacionais, pode-se concluir que ocorrerá conflito de representação somente com conceitos léxicos.

3.4 Documento de mapeamento

Documentos XML, agrupados em um mesmo domínio de conhecimento, são interpretados pela ferramenta OntoGen para a criação da ontologia. A partir de então, a ontologia, como modelo global de abstração das diferenciações dos documentos XML, passa a ser utilizada pela ferramenta OntoRel para criação do esquema lógico relacional.

De modo a estabelecer uma equivalência entre os documentos XML, a ontologia global de representação do domínio de conhecimento e o esquema lógico relacional, convencionou-se o documento de mapeamento como artefato de representação das equivalências. Conforme ilustrado na Figura 3.14, verifica-se que `Doc_B.xml` possui o elemento `<writer>`, assim como `Doc_C.xml` possui o elemento `<author>` e os respectivos elementos internos.

A equivalência na ontologia dá-se pelo conceito não-léxico `writer`. Portanto, seja o elemento `<writer>` de `Doc_B.xml`, assim como o elemento `<author>` de `Doc_C.xml`, ambos são equivalentes ao conceito `writer` da ontologia em função do agrupamento de sinônimos da ferramenta OntoGen. Os elementos internos possuem equivalência na ontologia com conceitos léxicos de mesmo nome.

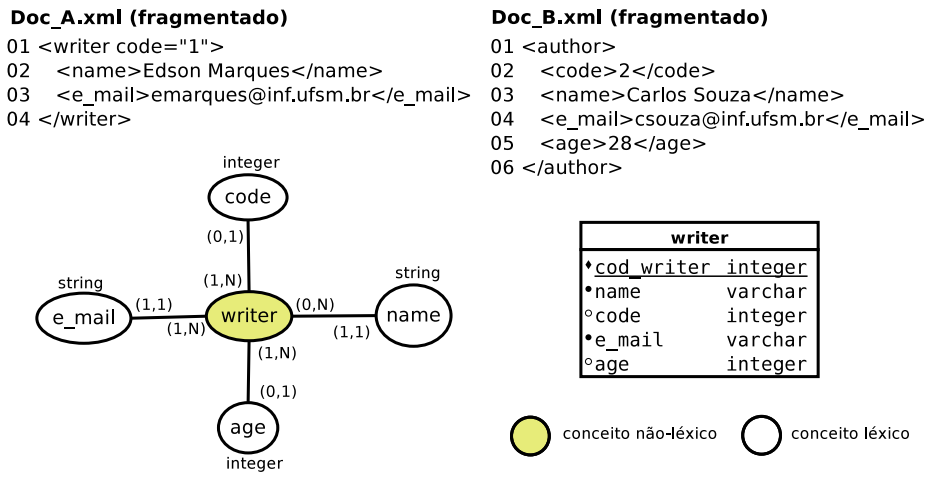

Figura 3.14: Equivalência entre documento XML, ontologia e o esquema lógica relacional. A versão dos documentos fragmentados está na Figura 3.1.

Em se tratando do esquema lógico relacional, a equivalência do conceito `writer` da ontologia corresponde a tabela `writer`. Os conceitos léxicos passaram a ser campos da tabela. De maneira análoga, o elemento `<writer>` de `Doc_B.xml` e o elemento `<author>` de `Doc_C.xml` correspondem a tabela `writer` no banco de dados relacional, assim como os respectivos elementos internos equivalem aos atributos da tabela. Desse modo, o conteúdo de `Doc_B.xml` e `Doc_C.xml` é mapeado para a tabela `writer`.

A concepção do documento de mapeamento como artefato necessário para o processo de mapeamento do conteúdo XML para o formato relacional é uma contribuição deste trabalho. Entretanto, a criação do documento de mapeamento de maneira automatizada foi implementada na ferramenta CMap (NEGRINI, 2011).

3.4.1 Geração de equivalências e inserção de dados

As equivalências no processo de mapeamento são caracterizadas pelas transformações ocorridas na estrutura original dos documentos XML durante os processos realizados pelos módulos OntoGen e OntoRel.

Para cada conceito da ontologia, será encontrado o conceito equivalente no modelo relacional, e realizada uma varredura nos esquemas extraídos dos documentos XML, buscando encontrar um ou mais conceitos equivalentes em cada esquema. Tal processo ocorrerá com o uso de um conjunto de assertivas de correspondência que definem a equivalência de conceitos (NEGRINI, 2011).

A geração de equivalências ocorre de maneiras distintas para conceitos léxicos e conceitos não-léxicos. Caso um conceito C_i presente na ontologia seja do tipo não-léxico, ele será mapeado apenas uma vez. Caso seja do tipo léxico, ele será mapeado uma vez para cada diferente relacionamento R que possua com um conceito não léxico. São apresentadas abaixo as asserti-

vas de correspondência criadas (NEGRINI, 2011).

- Definição 1: um conceito não-léxico C_i presente na ontologia será considerado equivalente a um conceito C_j de um documento XML, se C_i e C_j possuírem afinidade.
- Definição 2: um conceito não-léxico C_i presente na ontologia será considerado equivalente a um conceito C_j no modelo relacional se C_i e C_j possuírem afinidade, e se C_j representar uma tabela.
- Definição 3: Um conceito léxico C_i presente na ontologia e em um relacionamento R_i que possua também um conceito não léxico C_k, será considerado equivalente a todo conceito C_j de um documento XML em que C_i e C_j possuam afinidade, e se C_j possuir um relacionamento R_j que possua afinidade com o relacionamento R_i citado.
- Definição 4: um conceito léxico C_i presente na ontologia e em um relacionamento R_i que possua também um conceito não léxico C_k, será considerado equivalente a um conceito C_j do modelo relacional, se C_i e C_j possuírem afinidade, se C_j for uma coluna no modelo relacional, e se esta coluna pertencer a uma tabela equivalente ao conceito C_k citado.

Conforme ilustrado na ontologia da Figura 3.14, o conceito `writer` foi criado pela ferramenta Ontogen para associar os elementos `<writer>` e `<author>` referente aos documentos XML `B` e `C`. Nesse caso, no documento de mapeamento, deve haver a equivalência dos documentos XML contendo o elemento `<author>` relativo ao conceito `writer` estabelecido pela OntoGen. O resultado são as informações de `author` serem mapeadas para a tabela `writer` no banco de dados através da equivalência estabelecida. O agrupamento de elementos XML em conceitos sinônimos faz parte da implementação da ferramenta OntoGen.

Para criar as inserções em banco de dados relacional através da cláusula `INSERT` de SQL, foi utilizado o documento de mapeamento para associar a ontologia global com a busca da informação nas fontes de dados XML e para o devido armazenamento no esquema lógico relacional. De acordo com a arquitetura de XMap, o documento de mapeamento é um mecanismo essencial para a transformação de conteúdo de documentos XML para fins de armazenamento em banco de dados relacional.

3.4.2 Equivalências entre modelos de dados

Para registrar as transformações envolvidas no armazenamento da informação presente em documentos XML para o esquema relacional, o mapeamento descreve o conceito da ontologia em questão através do elemento `<concept>` na linha 3 da Figura 3.15. De acordo com o conceito existente na ontologia global, as fontes de dados XML são identificadas pelo atributo `id` do elemento `<source>`, ambos na linha 4. O conteúdo de cada fonte XML de interesse,

de acordo com o conceito envolvido, está apontado pelo caminho XPath presente no elemento `<xpath>` na linha 5. Caso uma fonte XML não tenha o conteúdo referente a um dado conceito da ontologia, o elemento `<xpath>` é vazio.

Figura 3.15: Arquivo de mapeamento parcial associando as fontes de dados XML com a ontologia global OWL e o esquema lógica relacional.

Em razão do conteúdo das fontes de dados XML serem de interesse para armazenamento em banco de dados relacional, o elemento `<relational>` na linha 13 indica como o dado será armazenado. Para conceitos não-léxicos, o armazenamento é feito em uma tabela, conforme consta no elemento `<type>` na linha 14, sendo que o nome da tabela no banco de dados está no elemento `<name>` na linha 15.

Quanto aos conceitos léxicos, o elemento `<relational>` conta com mais dois elementos internos, indicando que o armazenamento da informação é feito em um coluna de uma tabela no banco de dados relacional. A partir da linha 28, considera-se que o conceito `title` especificado na linha 18 é armazenado em um campo do tipo coluna (linha 29), cujo nome é `title` (linha 30), sendo que se trata da tabela `conference` (linha 31). Por ser um conceito léxico, há uma representação direta no computador, sendo que, neste caso, é uma `string` (linha 32).

O documento de mapeamento ilustrado (Figura 3.15), comparado com os exemplos de arquivos XML abordados ao longo deste capítulo, resulta em um arquivo de mapeamento extenso

envolvendo todos os conceitos presentes. No objetivo de facilitar a compreensão, somente os conceitos `conference` e `title` foram ilustrados. O exemplo completo pode ser visto no Apêndice D.

3.5 Algoritmos para a resolução de conflitos

O armazenamento do conteúdo das instâncias XML em banco de dados ocorre através do processo de mapeamento para o esquema lógico relacional previamente definido. Para isso, os documentos XML de interesse de um mesmo domínio são descritos por uma ontologia para que, a partir de então, possa ser criado o esquema lógico relacional. As ferramentas responsáveis por desempenharem essas tarefas foram apresentadas nas Seções 2.4 e 2.5, respectivamente.

O mapeamento considera o caminho XPath para apontar o conteúdo dos documentos XML para armazenamento em banco de dados. O módulo XMap do framework X2Rel associa as instâncias de dados XML pertencentes a um mesmo domínio com a ontologia em linguagem OWL e o esquema relacional previamente gerado em um documento de mapeamento, como proposto na Seção 3.4.

Conforme apresentado na Seção 3.3, existem conflitos que podem ocorrer no processo de mapeamento do conteúdo XML para armazenamento em banco de dados. O documento de mapeamento serve como um artefato na transformação de dados XML, além de auxiliar na resolução de conflitos. De acordo com a Figura 3.16, a montagem da instrução de inserção em linguagem SQL no banco de dados foi dividida em cabeçalho e em conjunto de tuplas, este chamado de contêiner.

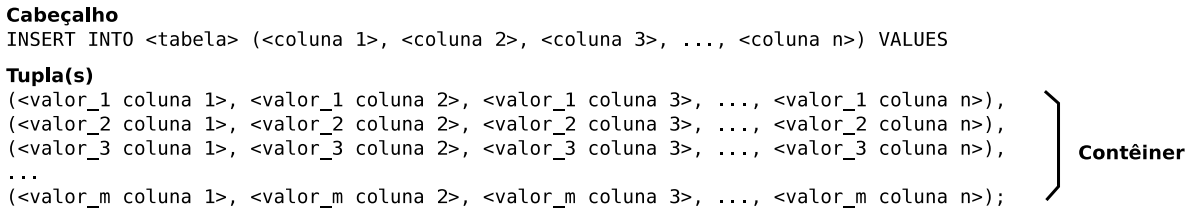

Figura 3.16: Esquema do cabeçalho criado para a instrução `INSERT` de SQL com o conjunto de tuplas correspondente.

De acordo com a Figura 3.16, o cabeçalho contém a instrução `INSERT` com o nome da tabela e as respectivas colunas. De maneira genérica, uma tabela pode ter até *n* colunas, sendo visto como os campos para a inserção de dados. O conjunto de tuplas, denominado contêiner, pode possuir até *m* tuplas, podendo serem consideradas também como linhas. De maneira que cada tupla do contêiner possa ser compatível com a instrução de inserção do cabeçalho, a quantidade de valores de uma tupla deve ser igual a quantidade de campos para inserção de dados.

Para a formação do cabeçalho, é necessário inicialmente o nome da tabela. O Algoritmo 1 possui a tarefa de buscar os conceitos não-léxicos, os quais correspondem a tabelas no esquema

relacional. O laço de repetição interpreta o documento de mapeamento na forma de objetos segundo o descritor da bibioteca Castor. Caso o teste da linha 2 seja verdadeiro, o `rótulo`, o qual corresponde ao nome da tabela, é adicionado a um `vetor` (linha 4).

Algoritmo 1 Agrupamento de conceitos não-léxicos.

Entrada: `mapeamento` {documento de mapeamento em memória}
Saída: `vetor` {agrupamento dos conceitos não-léxicos}

```
1: for all < conceito > em < mapeamento > do
2:    if conceito não-léxico then
3:       rótulo ← < nome > de < relacional > de conceito
4:       adicionar(rótulo, vetor)
5:    end if
6: end for
```

Uma vez preenchido o `vetor` com os nomes das tabelas após a busca dos conceitos não-léxicos, realizado pelo Algoritmo 1, a etapa seguinte é compor o restante do cabeçalho segundo a cláusula `INSERT` de SQL.

De acordo com o Algoritmo 2, responsável pela inserção do conteúdo XML em banco de dados, o laço mais externo percorre o `vetor` dinâmico criando o cabeçalho com o nome da tabela, na linha 2, além do contêiner de tuplas inicialmente vazio (linha 3). O laço interno novamente faz uma varredura no documento de mapeamento na forma de objetos. Contudo, a busca passa a ser por conceitos léxicos, pois o teste condicional da linha 5 faz com que o laço interno seja retomado imediatamente para um conceito não-léxico.

A estratégia de combinação entre os dois laços faz com que, para cada elemento do `vetor`, o qual corresponde ao rótulo da tabela no esquema relacional, seja verificado em todos os conceitos léxicos do documento de mapeamento a presença dos campos cujo nome da tabela seja igual ao `rótulo`. A condição de verificação está estabelecida na linha 11 do Algoritmo 2. Em caso verdadeiro, o determinado `campo` é adicionado ao cabeçalho para compor as colunas de inserção de dados na tabela do esquema relacional.

Por conseguinte, considerando o Algoritmo 2, as fontes de dados XML, presentes no elemento `<fonte>` do arquivo de mapeamento, são reunidas em um `vetorFonte` dinâmico e adicionados em uma estrutura de `conteúdo`, sendo realizado nas linhas 13 e 14, respectivamente. A adição do `conteúdo` ao contêiner, na linha 15, faz com que os registros estejam agrupados de maneira a reunir todas informações para inserção em banco de dados relacionais. Finalmente, a junção entre as duas estruturas que compõem a instrução `INSERT` passa a ser efetuada na linha 18. As diversas inserções de todas as tabelas fazem parte do `vetorInserção` adicionadas de acordo com a linha 19.

Tendo em vista os algoritmos principais de formação das estruturas de inserção do conteúdo em banco de dados relacionais, as Seções 3.5.1, 3.5.2, 3.5.3 e 3.5.4 detalham o funcionamento do algoritmo 3 proposto como solução para o conflito de nomenclatura, conflito de estrutura,

Algoritmo 2 Criação da estrutura de inserção do conteúdo XML em banco de dados.

Entrada: `vetor` {agrupamento dos conceitos não-léxicos}
Saída: `vetorInserção` {sequência de inserção no banco de dados}

```
1:  for all rótulo em vetor do
2:     cabeçalho ← criar(rótulo) {cria um cabeçalho com um rótulo}
3:     contêiner ← criar() {cria um contêiner vazio}
4:     for all < conceito > em < mapeamento > do
5:        if conceito não-léxico then
6:           continuar
7:        end if
8:        relacional ← < relacional > de conceito
9:        tabela ← < tabela > de relacional
10:       campo ← < nome > de relacional
11:       if tabela = rótulo then
12:          adicionar(campo, cabeçalho)
13:          vetorFonte ← arranjo < fonte > de < conceito >
14:          conteúdo ← criar(vetorFonte, relacional)
15:          adicionar(conteúdo, contêiner)
16:       end if
17:    end for
18:    inserção ← criar(cabeçalho, contêiner)
19:    adicionar(inserção, vetorInserção)
20: end for
```

conflito de elemento inexistente e conflito de representação, respectivamente.

Estruturas auxiliares utilizadas no algoritmo 3 são apresentadas do Apêndice B.

3.5.1 Algoritmo para conflito de nomenclatura

Cada fonte de dados XML está identificada pelo atributo `id` do elemento `fonte`, o qual corresponde ao nome de arquivo, segundo especificado no documento de mapeamento. A partir disso, o elemento `<expressão>` contém o caminho XPath que aponta para o conteúdo localizado no identificador `id`, sendo de interesse para armazenamento no banco de dados.

Ilustrado na Figura 3.17, o exemplo indica o caminho XPath para o elemento `<name>`. Observando o elemento-pai mais próximo, constata-se que `<name>` é utilizado para mais de um caso, assim resultando no conflito de homonímia.

Em razão de ser utilizado em diferentes casos, o questionamento ocorre sobre o conteúdo de qual `<name>` deve ser armazenado na correspondente entidade relacional. A partir do algoritmo do agrupamento de conceitos não-léxicos (Algoritmo 1) e, por conseguinte, da criação da estrutura de inserção do conteúdo XML em banco de dados (Algoritmo 2), a solução para o conflito de homonímia é proposta a partir do documento de mapeamento.

Sabendo que a estrutura de inserção no banco de dados é dividida em cabeçalho e em

Algoritmo 3 Algoritmo de resolução do conflito de nomenclatura, de estrutura, de elemento inexistente e de representação.

Entrada: `inserção` {estrutura de inserção (cabeçalho e tupla)}
Saída: `agrupamento` {agrupamento dos conceitos não-léxicos}

```
1:  contêiner ← devolverContêiner(inserção)
2:  conteúdo ← devolverConteúdo(contêiner)
3:  vetorFonte ← devolverVetor(conteúdo)
4:  relacional ← devolverRelacional(conteúdo)
5:  domínio ← devolverDomínio(relacional)
6:  for all fonte em vetorFonte do
7:     arquivo ← atributo id de fonte
8:     expressão ← expressão de fonte {caminho XPath}
9:     if expressão ≠ ∅ then
10:       valor ← ler(expressão, arquivo) {leitura do caminho XPath na instância XML}
11:       if compatível(valor, domínio) then
12:          adicionar(valor, vetorValores)
13:       else
14:          valorAjustado ← ajustar(valor, domínio)
15:          adicionar(valorAjustado, vetorValores)
16:       end if
17:    else
18:       adicionar(NULL, vetorValores)
19:    end if
20: end for
21: agrupamento ← montarTuplas(vetorValores)
```

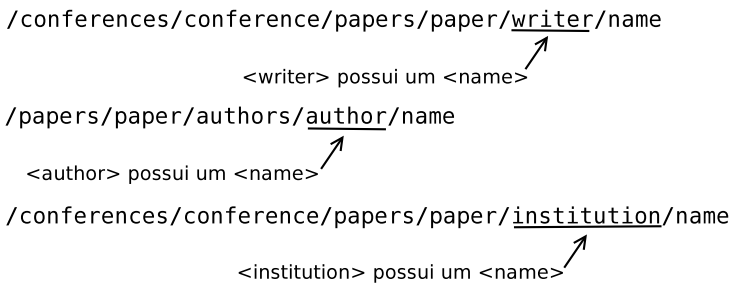

Figura 3.17: Exemplo de caminho XPath para o conflito de homonímia.

conjunto de tuplas (contêiner), o Algoritmo 3 possui a tarefa de extrair de cada fonte XML o conteúdo endereçado pelo caminho XPath de `<expressão>`. Para isso, busca-se o `contêiner` de cada `inserção` para que seja possível acessar a estrutura `conteúdo` e então as fontes de dados XML (linhas 1 a 3 do Algoritmo 3) através do documento de mapeamento na forma de objetos.

O laço de repetição percorre as fontes de dados XML na busca pelo identificador de arquivo XML e o caminho XPath (linhas 7 e 8). Se uma `expressão` é verificada como não-vazia (linha 9), o conteúdo XML apontado pela expressão XPath é lido no arquivo XML referenciado (linha 10). A verificação de compatibilidade (linha 11) refere-se ao conflito de representação

e será visto na seção 3.5.4. Havendo compatibilidade, a adição de conteúdo ocorre em em `vetorValores` dinâmico (linha 12). Se a expressão for vazia, o conteúdo tipo `NULL` é adicionado ao vetor. Considerando que `inserção` possui um `cabeçalho`, o nome da tabela e os campos já estão agrupados, conforme realizado pelos Algoritmos 1 e 2. Desse modo, a montagem de tuplas (linha 21 do Algoritmo 3) deverá produzir a instrução de inserção em banco de dados relacional associando de maneira correta o nome da tabela, as colunas e o conteúdo em um `agrupamento`. Essa estrutura contém o conteúdo XML agrupado na forma da estrutura de inserção do banco de dados, porém não possui ainda os valores de chaves primárias e sincronização de tuplas e valores nulos. Maiores detalhes sobre a criação da inserção dos dados serão vistos na Seção 3.6.

Para o conflito de sinonímia, o Algoritmo 3 é proposto como solução, independente da classificação léxica do conceito na ontologia. Nesse caso, cada termo pertencente ao conflito de sinonímia possuirá um caminho XPath distinto e será considerado como campo individual para montagem da instrução de inserção no banco de dados. Conforme a Figura 3.18, ambos os caminhos XPath possuem distinções de nomes sinônimos, embora o conteúdo seja mapeado na mesma tabela `writer` definida no documento de mapeamento.

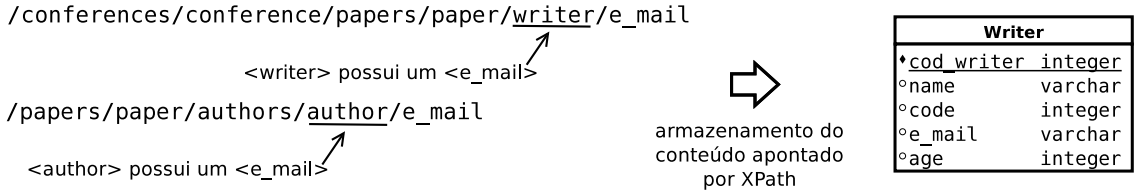

Figura 3.18: Exemplo de caminho XPath de mapeamento de conteúdo para o conflito de sinonímia.

3.5.2 Algoritmo para conflito de estrutura

A informação em documento XML é representada por um atributo de um elemento ou por um elemento propriamente dito. A ferramenta OntoGen define a ontologia OWL considerando ambos os casos como conceitos léxicos. Por consequência, a ferramenta OntoRel cria os correspondentes atributos para as entidades na estrutura lógica relacional.

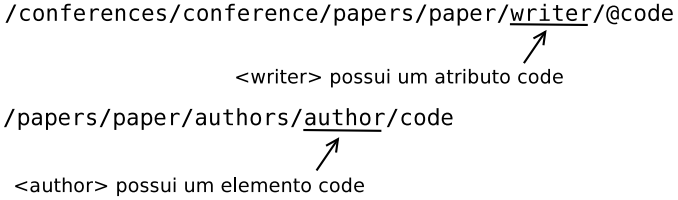

Figura 3.19: Exemplo de caminho XPath para atributo e elemento em XML.

A expressão XPath de cada fonte de dados XML possui a característica de endereçar o conteúdo XML seja na forma de atributo ou de elemento. Desse modo, há um caminho XPath

diferenciado para cada caso de representação, conforme exemplificado na Figura 3.19. Para a criação da estrutura de inserção em banco de dados, o Algoritmo 3 é apresentado como solução para o conflito de estrutura envolvendo o tipo de representação através da leitura e da interpretação do documento de mapeamento.

Em se tratando do conflito de estrutura de generalização, a ontologia OWL, criada pela ferramenta OntoGen, transforma elementos complexos de documentos XML em conceitos não-léxicos. Por conseguinte, o esquema lógico relacional definido pela ferramenta OntoRel converte conceitos não-léxicos e conceitos léxicos em tabelas e colunas, respectivamente.

O documento de mapeamento apresenta o caminho XPath para o conteúdo de cada instância XML no elemento `<expressão>`, associando com a ontologia e a estrutura lógica relacional. Como uma solução automatizada para o conflito de generalização, a proposta está na inserção no banco de dados de acordo com o Algoritmo 3 definido para o conflito de nomenclatura. Partindo da interpretação do usuário, o uso de funções XPath podem ser utilizados modificando as expressões XPath que apontam para o conteúdo a ser inserido no banco de dados.

3.5.3 Algoritmo para conflito de elemento inexistente

Quando ocorrer a ausência de conteúdo em um determinado documento XML, é necessário verificar o reflexo que pode ocorrer na organização do banco de dados relacional. O fato do elemento `<expressão>` de uma fonte de dados estar vazio, ou seja, sem um caminho XPath, indica um campo com valor nulo no banco de dados.

O conflito de elemento inexistente ocorre para conceitos não-léxicos. Caso um documento XML não possua um conteúdo a ser armazenado em banco de dados para o correspondente conceito da ontologia, o respectivo campo na estrutura relacional deve permitir atribuição de valor nulo. A partir da estrutura de inserção do banco de dados, o Algoritmo 3 realiza uma busca pelas fontes de dados cuja expressão seja vazia (bloco `else` na linha 17 do Algoritmo 3). A associação entre `tabela` e `campo` ocorre na leitura do documento XML da estrutura lógica relacional, resultado da ferramenta OntoRel. Notavelmente, `NULL` passa a ser inserido como valor para o campo em `vetorValores` (linha 18).

3.5.4 Algoritmo para conflito de representação

O conteúdo para armazenamento em banco de dados relacionais está apontado pelo caminho XPath presente no elemento `<expressão>` vinculado a cada fonte de dados XML, conforme especificado no documento de mapeamento (Seção 3.4). Ao extrair o conteúdo de cada instância XML interpretando o respectivo caminho XPath, é necessário verificar qual o tipo de dado correspondente na estrutura relacional.

Primeiramente, as estruturas de inserção de banco de dados são carregadas (linhas 1 a 5) para iniciar o laço de repetição. A diferença do algoritmo de resolução do conflito de representação está na interpretação do conteúdo extraído da fonte de dados XML. Verifica-se a compatibilidade de `valor` e `domínio` o qual pertence à estrutura lógica relacional (linha 11). O resultado é verdadeiro caso exista compatibilidade. Um exemplo pode ser a leitura da idade de uma pessoa em um documento XML. O domínio correspondente são números do tipo inteiro. Nesse caso, não ocorre conflito de representação, pois o valor está correto para o tipo de dado relacional correspondente.

Outro exemplo, o conteúdo XML contendo datas em formato diferente ao estipulado pela máscara `aaaa-mm-dd` será considerado como `string` pela ferramenta OntoGen, tipo de dados mais abrangente. No objetivo de armazenar o conteúdo de datas em formato mais específico em banco de dados, o campo correspondente na estrutura lógica relacional deverá ser alterado para o tipo `date` através da intervenção do usuário. O componente XMap irá considerar o tipo mais abrangente para inserção de dados de maneira padrão caso não haja uma ação do usuário. A partir de então, o ajuste de valor (linha 14) ocorreria na leitura do conteúdo data, sendo os valores ajustados do formato `dd/mm/aaaa` para a máscara `aaaa-mm-dd` e atribuído para `valorAjustado` para adição no `vetorValores` (linhas 14 e 15).

3.6 Criação das instruções de inserção dos dados

A busca da informação em formato XML ocorre a partir do sequenciamento determinado no documento de mapeamento. O conteúdo dos documentos XML, apontado por expressões XPath, é agrupado em cada conceito da ontologia e, a partir disso, é associado com a estrutura relacional.

3.6.1 Adaptação dos dados para inserção

Através da disposição no documento de mapeamento, o resultado é o sequenciamento do conteúdo XML em ordem vertical. Analisando o documento de mapeamento da Figura 3.4, verifica-se que, para cada conceito da ontologia, todo o conteúdo XML é recuperado para mapeamento na estrutura lógica relacional. Desse modo, para um determinado campo no banco de dados, os valores presentes nos documentos XML são listados sequencialmente. A Figura 3.20 apresenta a instrução `INSERT` da linguagem SQL e a disposição vertical do conteúdo XML do banco de dados.

Devido à construção do `INSERT` no banco de dados, é necessário organizar os valores recuperados dos documentos XML para compatibilidade com a instrução de inserção. A partir da disposição vertical do conteúdo XML determinado pelo documento de mapeamento, é necessário adaptar o sequenciamento para o formato horizontal. Conforme a cláusula `INSERT`,

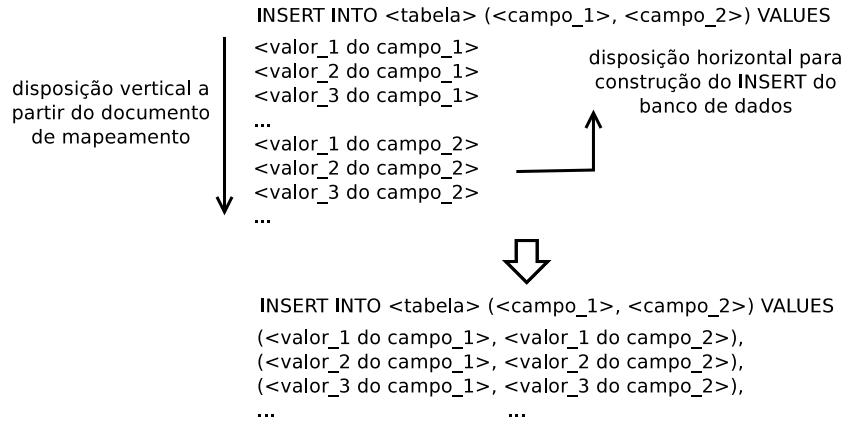

Figura 3.20: Adaptação da ordem sequencial do conteúdo XML determinado pelo documento de mapeamento para o formato da cláusula `INSERT` do banco de dados.

os campos são determinados sequencialmente, implicando que o conteúdo seja disposto na mesma ordem. A partir do documento de mapeamento, para cada campo no banco de dados, o conteúdo é integralmente sequenciado. Em contrapartida, para a inserção do banco de dados, para todos os campos, os valores correspondentes devem ser sequenciados vindo a compor uma ou mais tuplas.

Como proposta de adaptar o sequenciamento determinado pelo documento de mapeamento para a instrução de inserção no banco de dados, a estratégia foi agrupar o conteúdo em uma estrutura de dados que pudesse diferenciar o conteúdo relativo aos campos de inserção. Para isso, foi utilizado um `HashMap`[2] como solução para integrar campos e conteúdo de maneira mais adequada.

Considerando o documento de mapeamento, conceitos não-léxicos correspondem a tabelas no banco de dados. A partir disso, cada entidade no banco de dados possuirá um `HashMap` para organização do conteúdo em memória para montagem da instrução de inserção. O `HashMap` foi organizado possuindo a chave de recuperação de dado correspondente ao campo da tabela, e o conteúdo organizado em uma fila determinado como uma lista encadeada[3]. A Figura 3.21 ilustra o aspecto do `HashMap` para montagem da instrução de inserção.

Em se tratando de documentos XML estruturalmente distintos, existe a possibilidade de os campos possuírem conteúdos em quantidades distintas. Conforme ilustrado na Figura 3.21, a quantidade de valores definidos a partir dos documentos XML pode variar para cada campo de inserção em tabelas no banco de dados. A ilustração exemplifica, para os três campos indicados as respectivas quantidades com quatro, um e dois valores. Como consequência, ocorrerá o caso de inserção de conteúdo nulo para um determinado campo no banco de dados. A maneira para solucionar a variação da quantidade de conteúdo foi utilizar uma fila de inserção de dados. À medida que o documento de mapeamento é interpretado, o conteúdo XML apontando pelas

[2]Pacote `java.util.HashMap<Chave, Valor>`.
[3]Pacote `java.util.LinkedList<Elemento>`

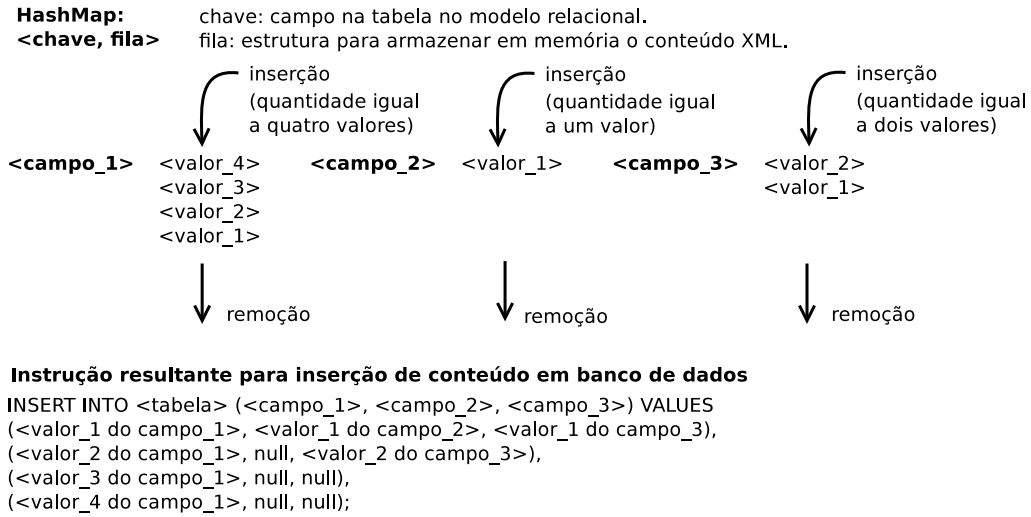

Figura 3.21: Estrutura para a criação da instrução de inserção do conteúdo XML para cada tabela do banco de dados.

expressões XPath é inserido na fila de acordo com o determinado campo da tabela.

Para a montagem da instrução de inserção no banco de dados, os valores dos campos são removidos da fila a partir da chave do `HashMap`. A instrução é montada até o último valor da fila ser removido. Caso alguma fila tenha tido todos os valores removidos, o conteúdo `null` será atribuído ao respectivo campo. Além disso, os próprios valores dos campos da tabela podem ser nulos, podendo ser enfileirados da mesma maneira que valores não-nulos.

3.6.2 Sequenciamento das instruções de inserção

A inserção do conteúdo XML em banco de dados deve considerar os relacionamentos entre as entidades na estrutura lógica relacional. Caso alguma tabela possua uma restrição de chave estrangeira, será necessário primeiramente inserir o conteúdo da tabela referenciada. A partir de então, o conteúdo da tabela contendo a chave estrangeira poderá ser inserido sequencialmente de maneira correta.

Assumindo que o cabeçalho e o contêiner de tuplas estão criados como a estrutura de inserção no banco de dados, é preciso definir uma ordem para montagem do *script* de inserção dos dados. Partindo da estrutura de inserção, sendo uma para cada entidade na estrutura relacional, o `vetorInserção`, conforme proposto no Algoritmo 2 (linha 19), é organizado para o sequenciamento de inserção de conteúdo em banco de dados.

De maneira simplificada, conforme ilustrado na Figura 3.22, toda a inserção sem dependência de chaves estrangeiras deverá ser inserida no começo do vetor dinâmico. Para a inserção que possua dependências de chaves estrangeiras, o reposicionamento ocorre ao final do vetor.

Analisando o sentido de percurso no vetor, pode-se perceber que a ordem de inserção inicia pelas entidades que não possuem dependência de chave estrangeira. Posteriormente, a inserção

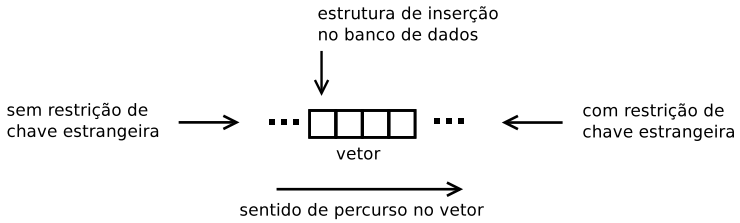

Figura 3.22: Estrutura de sequenciamento para inserção no banco de dados.

dependente de chave estrangeira é realizada com a garantia que as entidades referenciadas foram anteriormente sequenciadas na inserção.

3.7 Considerações finais

O entendimento das etapas da criação da ontologia a partir dos documentos XML e a correspondente criação do esquema lógico relacional possui importância para o contexto deste trabalho. Para isso, foram criados exemplos demonstrando a entrada de dados para a ferramenta OntoGen e foi analisada a ontologia resultante. Por sua vez, partindo da ontologia, foi demonstrado o funcionamento da ferramenta OntoRel responsável pela criação do esquema lógica relacional.

Uma vez apresentadas as etapas de criação da ontologia e do esquema de banco de dados, foi proposta a arquitetura do módulo XMap do framework X2Rel. Definida em camadas, a organização de XMap utiliza a biblioteca Castor para interpretação de arquivos XML necessários do processo de mapeamento do conteúdo XML.

Conflitos podem ocorrer no processo de mapeamento do conteúdo XML para banco de dados relacionais. Mais especificamente, a informação disposta em XML é armazenada no banco de dados definido pelo esquema lógico criado pela ferramenta OntoRel. De maneira a resolver os conflitos existentes, buscou-se classificar e descrever os conflitos possíveis.

Finalmente, como artefato criado para representar a equivalência entre os documentos XML, a ontologia e o esquema lógico relacional, o documento de mapeamento foi proposto como solução no processo de armazenamento de dados XML em banco de dados.

Algoritmos foram definidos para cada caso de conflito de maneira a propor uma solução teórica no processo de mapeamento do conteúdo XML para banco de dados relacionais. O documento de mapeamento é necessário para o funcionamento dos algoritmos em questão.

Devido ao sequenciamento determinado pelo documento de mapeamento para a busca da informação em documentos XML, uma adaptação foi feita para montar a instrução de inserção em banco de dados. Para isso, foi utilizada a estrutura de dados `HashMap`, cuja chave é o campo da tabela relacional que aponta para uma fila de dados. As tuplas são montadas acessando o `HashMap` de cada entidade e percorrendo as filas de inserção de dados indexadas pela chave

hash. O conteúdo da fila é removido a medida que as tuplas são montadas.

4 IMPLEMENTAÇÃO

A partir da concepção do módulo XMap da ferramenta X2Rel, o próximo passo é apresentar a maneira como o sistema foi desenvolvido. Tendo em vista a apresentação dos conflitos existentes no processo de mapeamento do conteúdo XML para banco de dados relacionais, a próxima etapa é apresentar as soluções para os tipos de conflitos.

4.1 Tecnologias utilizadas

O módulo XMap do framework X2Rel foi desenvolvido em Java (plataforma J2SE versão 1.6.0_25). Para interpretação do arquivo de mapeamento e do esquema XML da estrutura lógica relacional da ferramenta OntoRel, a biblioteca Castor (EXOLAB, 2012) foi utilizada para carregar os arquivos XML na memória principal.

Conforme consta na arquitetura de XMap, ilustrada na Figura 3.5, a função dos descritores presentes no primeiro nível, tanto para o documento XML do banco de dados, assim como para o documento de mapeamento, é estabelecer a maneira como o arquivo XML deve ser organizado em memória na forma de estruturas em Java. Trata-se de uma solução da biblioteca Castor para carregar um arquivo XML em objetos Java. O documento de mapeamento, apresentado na Seção 3.4, assim como a estrutura lógica relacional estruturado em formato XML, conforme mencionado na Seção 3.1.2, foram convencionados de maneira a possuir uma estrutura única. O uso de descritores pela biblioteca Castor é necessário para processar documentos XML em memória. O Apêndice C contém um exemplo dos descritores utilizados para o arquivo de mapeamento e para o esquema XML da estrutura relacional de banco de dados.

Para o processamento de caminhos XPath existentes no elemento `<expressão>` do documento de mapeamento ilustrado na Figura 3.15, foi utilizado o pacote `javax.xml.xpath` da API padrão Java. O processamento de documentos XML com expressões XPath é mais eficiente que utilizar métodos `getter`, pois, com expressões XPath, um `Element` pode ser selecionado sem iteração a partir de uma lista de nós. Desse modo, XPath revela-se ser mais eficiente na busca de conteúdo XML, pois uma lista de nós recuperada com um método `getter` requer iteração para recuperar o conteúdo de elementos.

A seguir, a Seção 4.2 apresenta o diagrama de classes do módulo XMap. A partir de então, é possível ter um melhor entendimento da organização do sistema.

4.2 Diagramas de classes

Juntamente com os descritores XML definidos conforme convenção estabelecida pela biblioteca Castor, o diagrama de classes deve ser definido de maneira a apresentar a implementação do módulo XMap. Através de Castor, a estrutura XML pode ser carregada em memória e manipulada através de chamadas de métodos usuais em Java.

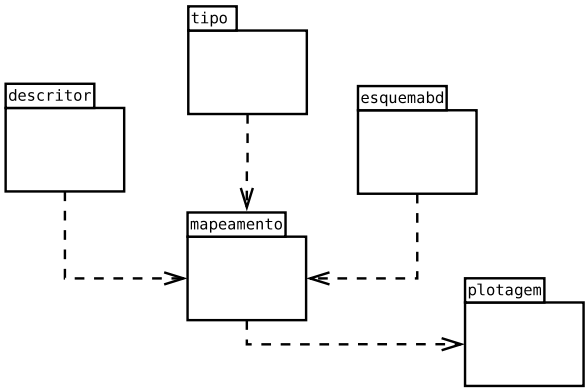

Figura 4.1: Diagrama dos pacotes do módulo XMap.

Inicialmente, a Figura 4.1 ilustra os relacionamentos de dependência entre os pacotes do módulo XMap do framework X2Rel. As próximas Seções detalham as particularidades de cada pacote contendo o diagrama de classes em particular.

4.2.1 Os pacotes `esquemabd` e `descritor`

Para melhor adequação da ilustração, as classes dos pacotes `descritor`, conforme (Figura 4.2), e `esquemabd`, conforme mostrado na Figura 4.3, tiveram somente seus atributos privados ilustrados. O rótulo “atributos encapsulados set e get” indica que as classes em questão possuem somente métodos de leitura e escrita que encapsulam as variáveis de acesso privado.

Uma vez que os descritores XML são documentos auxiliares no processo de mapeamento para armazenamento de conteúdo XML em banco de dados relacionais, a estrutura definida para cada descritor segue uma convenção pré-estabelecida. A estrutura do documento de mapeameamento segue uma convenção definida na concepção do módulo XMap do framework X2Rel, ao passo que o esquema XML do banco de dados relacional teve a estrutura elaborada na proposta da ferramenta OntoRel. Dessa maneira, o uso dos descritores XML pela biblioteca Castor é uma opção conveniente, visto que é de conhecimento prévio sobre a estrutura dos documentos auxiliares de XMap.

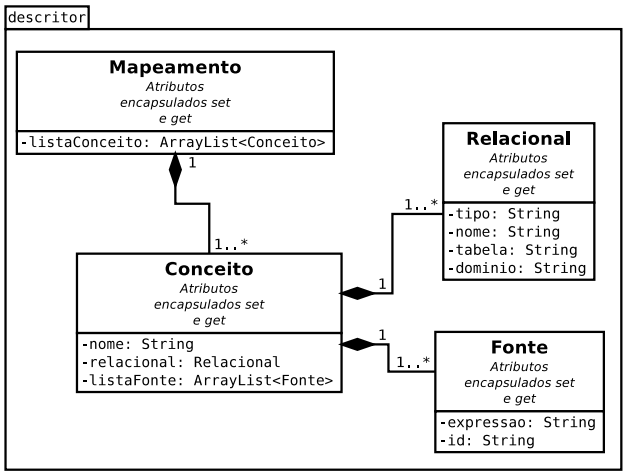

Figura 4.2: Pacote `descritor` e suas classes.

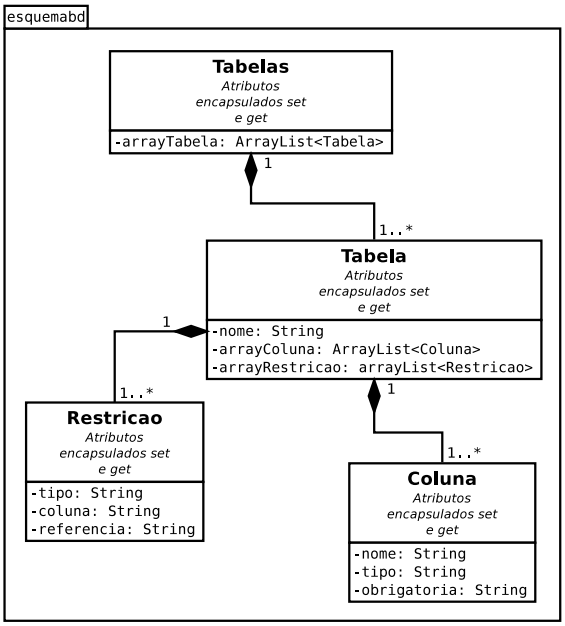

Figura 4.3: Pacote `esquemabd` e suas classes.

4.2.2 O pacote `mapeamento`

O pacote `mapeamento` (Figura 4.4) contém as classes Java para a resolução de conflitos de inserção no banco de dados. Com os objetos em memória do documento de mapeamento e os arquivos XML de entrada, o objetivo é criar objetos referentes ao cabeçalho de inserção no banco de dados juntamente com as respectivas tuplas (Figura 3.16). Associado a resolução de conflitos, o resultado é o *script* SQL de inserção do conteúdo de arquivos XML em banco de dados relacionais.

Para carregar em memória o documento de mapeamento, a classe `Mapeador` possui a finalidade de, através do descritor XML correspondente, utilizar os métodos da biblioteca Castor para criação do objeto `mapeamento` em memória. Primeiramente todas as entidades são extraídas do

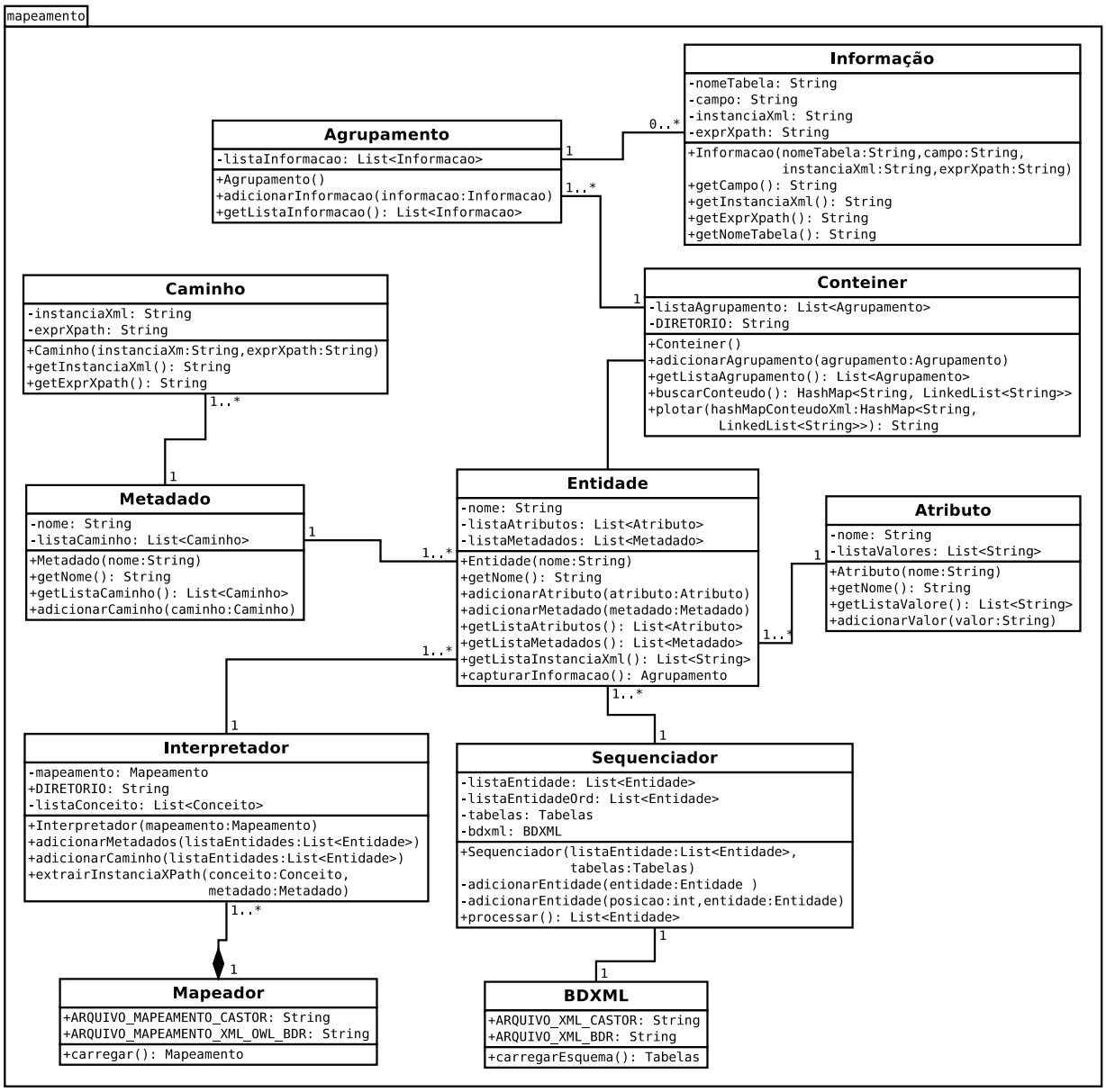

Figura 4.4: Pacote `mapeamento` e suas classes.

documento de mapeamento, sendo objetos da classe `Entidade`.

Novamente através do documento de mapeamento, correspondente em memória ao objeto da classe `Mapeamento`, os atributos são retirados e representados como objetos de `Atributo` a partir de uma lista de entidades. Como classes auxiliares no processo de busca do conteúdo XML para armazenamento em banco de dados, `Metadado` e `Caminho` definem os objetos contendo as expressões XPath do conteúdo XML. Mais especificamente, objetos da classe `Caminho` encapsulam a expressão XPath com o nome de arquivo XML, ao passo que objetos da classe `Metadado` reúnem objetos da classe `Caminho`.

Considerando o documento XML do esquema lógico relacional, a classe `BDXML`, através do descritor XML correspondente, implementa o carregamento do documento XML de descrição

do esquema relacional em memória na forma de objeto da classe `Tabelas` (vide Figura 4.3). De acordo com as restrições definidas no elemento `<restrição>` do documento XML do esquema de banco de dados, a classe `Sequenciador` organiza a lista de entidades na ordem de dependência de chave primária e chave estrangeira. Tabelas que possuem somente chave primária devem ser criadas antes das quais possuem restrição de chave estrangeira. Desse modo, o *script* de inserção de dados apresenta uma sequência correta de criação das tabelas.

Uma vez finalizando o sequenciamento da lista de entidades, o próximo passo é a criação do cabeçalho da instrução de inserção, conforme consta na Figura 3.16. Objetos da classe `Informação` associam as expressões XPath de uma instância XML para o campo de uma tabela. A partir disso, toda informação é reunida em um objeto da classe `Agrupamento`. Objetos da classe `Contêiner` reúnem os agrupamentos de cada entidade. Nesta classe, ocorre efetivamente a busca do conteúdo XML interpretando as expressões XPath de cada instância de dados.

4.2.3 O pacote `plotagem`

O pacote `plotagem` (Figura 4.5) contém as classes destinadas para montagem da instrução de inserção do conteúdo XML em banco de dados relacionais. É considerando como a lista ordenada de entidades e o conteúdo XML presente em memória na forma de objetos do pacote `mapeamento`.

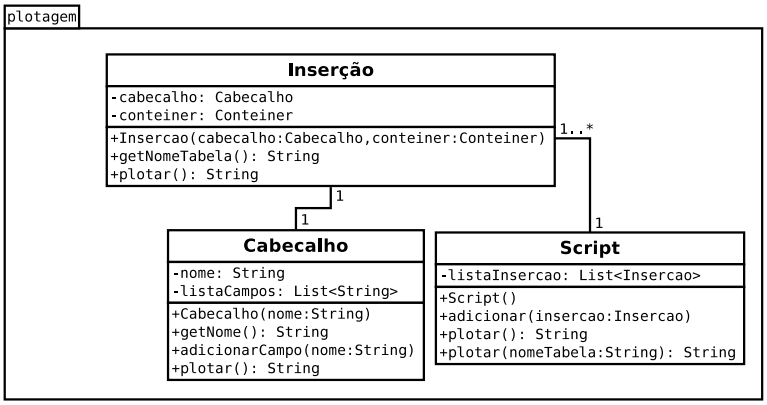

Figura 4.5: Pacote `plotagem` e suas classes.

Associando o cabeçalho com o contêiner de dados, a classe `Inserção` implementa a criação da instrução de inserção de dados correspondente a uma determinada tabela do banco de dados. A classe `Script` reúne as inserções de todas as tabelas resultando na inserção do conteúdo XML para todas as tabelas do esquema relacional.

4.2.4 O pacote `tipo`

Os conceitos não-léxicos e conceitos léxicos presentes no documento de mapeamento são convertidos para tabela e coluna, respectivamente, no banco de dados relacional. Para a criação

do *script* de inserção, é necessário diferenciar as estruturas para a montagem do cabeçalho e do contêiner de tuplas.

Considerando a definição de tipos, o processo de interpretação do documento de mapeamento considera um tipo genérico, sendo modelado pela classe abstrata `Tipo`. A partir disso, uma vez verificando o elemento `<relational>` do documento de mapeamento, os tipos mais específicos, definidos pelas classes `Tabela` e `Coluna`, são determinados por vinculação dinâmica. O pacote `tipo` está ilustrado na Figura 4.6.

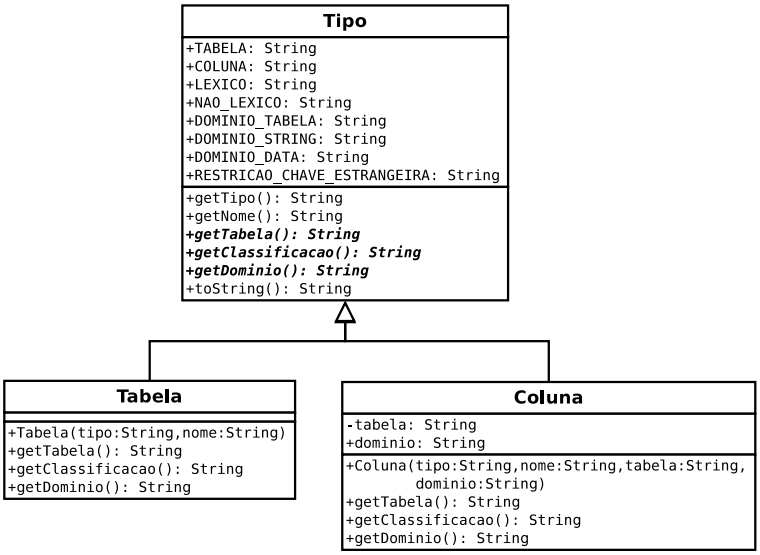

Figura 4.6: Pacote `tipo` e suas classes.

A diferenciação entre um conceito da ontologia a ser traduzida seja para uma tabela ou coluna no esquema relacional está presente no elemento `<type>` do documento de mapeamento. Conforme ilustrado na Figura 4.7, o conceito `conference` (linha 3) é traduzido para a tabela de nome `conference` (linha 7), sendo determinado pela palavra reservada `Table` do documento de mapeamento como conteúdo do elemento `<type>` (linha 6).

Analisando ainda a Figura 4.7, o conceito `title` (linha 10) é interpretado como sendo atributo de nome `title` (linha 14) da tabela `conference` (linha 15). A identificação ocorre pela palavra reservada `column` do documento de mapeamento como conteúdo do elemento `<type>` (linha 13).

4.3 Protótipo de XMap

Como parte integrante do framework X2Rel, o módulo XMap possui a responsabilidade de mapear o conteúdo dos documentos XML para armazenamento em banco de dados relacional. Para isso, considera-se a ontologia, os documentos XML e o esquema lógico relacional integrados no documento de mapeamento.

```
01 <?xml version="1.0" encoding="utf-8" ?>
02 <mapping>
03    <concept name="conference">
04       ...
05       <relational>
06          <type>Table</type>    <-- tipo tabela
07          <name>conference</name>
08       </relational>
09    </concept>
10    <concept name="title">
11       ...
12       <relational>
13          <type>column</type>   <-- tipo coluna
14          <name>title</name>
15          <table>conference</table>
16          <domain>string</domain>
17       </relational>
18    </concept>
19 <mapping>
```

Figura 4.7: Fragmento do documento de mapeamento demonstrando a diferenciação entre tabela e coluna no esquema relacional.

4.3.1 Carregamento do documento de mapeamento em memória

O processamento da estrutura XML em memória exige o uso de descritor em razão do uso da biblioteca Castor. O documento de mapeamento deve ser inicialmente interpretado e carregado pela ferramenta XMap para interpretação de conflitos e da busca de conteúdo XML pelas expressões XPath. A Figura 4.8 ilustra o método que realiza o carregamento do documento de mapeamento em memória.

```
01 public static Mapeamento carregar() throws IOException, MappingException, MarshalException, ValidationException
02 {
03    Mapping mapping = new Mapping();
04    Mapeamento mapeamento = null;
05    mapping.loadMapping(ARQUIVO_MAPEAMENTO_CASTOR);
06    Unmarshaller unmarshaller = new Unmarshaller(mapping);
07    mapeamento = (Mapeamento)unmarshaller.unmarshal(new InputSource(
08       new FileReader(ARQUIVO_MAPEAMENTO_XML_OWL_BDR))
09    );
10    return mapeamento;
11 }
```

Figura 4.8: Método de carregamento do documento de mapeamento.

A partir do carregamento do documento de mapeamento (linha 7), a próxima tarefa é o processar o descritor para interpretar o documento de mapeamento (linha 8). Como resultado, o documento de mapeamento passa a ser representado em memória como objetos em linguagem Java de programação (linha 9).

Observando a Figura 4.8, é possível concluir a simplicidade para carregar um documento XML como objetos através do uso de descritores. Na linha 7, o `ARQUIVO_MAPEAMENTO_CASTOR` representa o caminho para o descritor XML do documento de mapeamento. Na linha 10, o `ARQUIVO_MAPEAMENTO_XML_OWL_BDR` é o caminho para o documento de mapeamento propriamente dito.

4.3.2 Extração de entidades e de atributos

Uma vez tendo sido interpretado o documento de mapeamento, o resultado é a manipulação em memória das equivalências entre conceitos, documentos XML e o esquema lógico relacional.

A extração de entidades e atributos ocorre pela identificação do conceito da ontologia. Conceitos não-léxicos correspondem a tabelas no banco de dados, ao passo que conceitos léxicos são colunas de tabelas. A Figura 4.9 apresenta o método responsável pela extração de entidades a partir do documento de mapeamento.

```
01 public List<Entidade> extrairEntidades(Mapeamento mapeamento)
02 {
03   List<Entidade> listaEntidades = new ArrayList<Entidade>();
04   List<Conceito> listaConceito = mapeamento.getListaConceito();
05   for(Conceito conceito : listaConceito)
06   {
07     Relacional relacional = conceito.getRelacional();
08     if((relacional.getTipo().compareTo(Tipo.TABELA)) == 0)
09       listaEntidades.add(new Entidade(relacional.getNome());
10   }
11   return listaEntidades;
12 }
```

Figura 4.9: Método de extração de entidades a partir do documento de mapeamento.

O objeto `mapeamento`, passado como parâmetro para o método `extrairEntidades`, corresponde ao documento de mapeamento em memória na forma de objetos em Java. O objeto `conceito` (linha 5) são os conceitos da ontologia, assim como o objeto `relacional` são as informações do esquema lógico relacional. A comparação na linha 8 verifica se o conceito possui um elemento relacional do tipo `TABELA`.

Para a extração de atributos, a implementação é semelhante. A comparação da linha 8 na Figura 4.9 é modificada para o tipo `COLUNA` na Figura 4.10 (linha 9). Além disso, uma verificação adicional é necessária para associar a coluna com a tabela correspondente (linha 11).

```
01 public void adicionarAtributos(List<Entidade> listaEntidades)
02 {
03    Relacional relacional;
04    for(Entidade entidade : listaEntidades)
05      for(Conceito objeto : this.listaConceito)
06      {
07         relacional = objeto.getRelacional();
08         if((relacional.equals(Tipo.COLUNA)))
09            if(relacional.getTabela().equals(entidade.getNome()))
10               entidade.adicionarAtributo( new Atributo(relacional.getNome()) );
11      }
12 }
```

Figura 4.10: Método de extração de atributos a partir do documento de mapeamento.

Através dos dois métodos ilustrados nessa seção, as tabelas e colunas estarão disponíveis para montar o cabeçalho de inserção no banco de dados. Cada entidade possuirá atributos, assim compor a cláusula `INSERT INTO` pode ser feita com manipulação de `strings`.

4.3.3 Extração do conteúdo XML

Cada conceito da ontologia presente no documento de mapeamento possui os documentos XML para recuperação de conteúdo e armazenamento no banco de dados. O caminho XPath apontado para a informação de interesse deve ser interpretado. Para isso, o método ilustrado na Figura 4.11 associa cada documento XML com a expressão em XPath de endereçamento.

```
01 public List<String> buscarConteudoXml(String instanciaXml, String exprXpath) throws FileNotFoundException, XPathExpressionException
02 {
03   XPathFactory  factory = XPathFactory.newInstance();
04   XPath xPath = factory.newXPath();
05   ArrayList<String> listaConteudo = new ArrayList<String>();
06   File documentoXml = new File(instanciaXml);
07     XPathExpression xPathExpression = xPath.compile(exprXpath);
08     InputSource is = new InputSource(new FileInputStream(documentoXml));
09     Object resultado = xPathExpression.evaluate(is, XPathConstants.NODESET);
10     NodeList nodos = (NodeList)resultado;
11     if(nodos != null)
12       for(int el = 0; el < nodos.getLength(); el++)
13         listaConteudo.add(nodos.item(el).getTextContent());
14   return listaConteudo;
15 }
```

Figura 4.11: Método de extração de conteúdo dos documentos XML.

A partir do resultado da compilação da expressão XPath (linha 9) e da criação de fluxo de arquivo (linha 10), ambos são vinculados (linha 11) para recuperação do conteúdo XML. O laço de repetição (linha 15) processa os nós resultantes da expressão XPath avaliada sobre o documento XML. O conteúdo é associado sequencialmente em um vetor dinâmico de `strings`.

O método ilustrado na Figura 4.11 possui influência na resolução de conflitos de nomenclatura do tipo homonímia e sinonímia. O documento de mapeamento é o mecanismo utilizado para resolver ambos os tipos de conflitos. Desse modo, o processo de extração do conteúdo XML proposto pelo método em questão implementa a solução proposta.

A chamada do método considera a existência de expressão XPath associado com o documento de mapeamento. Para o conflito de elemento inexistente, não há uma expressão XPath. Desse modo, uma verificação prévia deve ocorrer quanto a existência de expressão. Em caso negativo, o valor `null` deve ser inserido no banco de dados.

4.4 Considerações finais

Partindo da proposta teórica, recursos de programação foram utilizados para implementação do módulo XMap. O diagrama de classes foi apresentado para entendimento da organização de implementação. O detalhamento de pacotes e as respectivas classes oferece um melhor entendimento de programação em código-fonte.

Métodos principais de importância para o módulo XMap do framework X2Rel foram apresentados. Desse modo, evidencia-se a existência de uma implementação para os algoritmos de resolução de conflitos. Através da definição dos métodos, experimentos e análise dos resultados

foram realizados de maneira a avaliar a implementação, conforme apresentados no capítulo a seguir.

5 EXPERIMENTOS E AVALIAÇÃO DE RESULTADOS

A implementação do módulo XMap da ferramenta X2Rel e os algoritmos propostos para a resolução de conflitos, devem ser verificados de maneira a validar a proposta de solução.

Os experimentos correspondem a ensaios realizados com arquivos XML, representando as instâncias de dados pertencentes a um domínio de conhecimento prévio. A partir disso, o processo seguinte passa a ser a análise da ontologia descritiva e o esquema lógico relacionado resultantes das ferramentas OntoGen e OntoRel, respectivamente. As instâncias de dados com a ontologia e o esquema relacional, o documento de mapeamento experimental, juntamente com a solução dos conflitos existentes, fazem parte da verificação do módulo XMap.

5.1 Arquivos XML

A base inicial de testes para este trabalho deve possuir uma variedade de informação e um volume de dados para possibilitar a execução mais abrangente de experimentos. A Internet foi o ambiente mais apropriado para busca das fontes de dados XML devido ao compartilhamento de informações heterogêneas.

Partindo de um conteúdo mais específico, foi utilizada as fontes de dados XML de previsão do tempo de cidades arbitrariamente escolhidas. Em razão de ser um tipo de informação cuja atualização ocorre diversas vezes no período de um dia, a gestão desse tipo de informação deve ser eficiente. Trata-se de informações de interesse na tomada de decisões, como na agricultura, na aviação e no cotidiano. Além disso, atualizações frequentes, no período de um dia, são necessárias na divulgação de dados mais precisos da previsão do tempo.

Em função da natureza dos dados utilizados inicialmente na etapa de experimentação, as fontes são constituídas por documentos RSS, os quais representam um formato XML que permite que a informação publicada seja interpretada por diversos programas. Leitores RSS ou agregadores são aplicações de software utilizados na interpretação desse formato de sindicância de dados.

Essencialmente, os documentos XML são constituídos por boletins de previsão diária do tempo contendo a temperatura mínima e máxima, dados de precipitação e radiação solar, descrição da previsão, além de outras informações meteorológicas. Instituições de divulgação da previsão do tempo publicam o prognóstico na Internet. Os documentos XML em formato RSS são diferenciados em canais, como monitoramento e avisos meteorológicos, condições atuais para os aeroportos, etc. O interesse são os boletins de previsão do tempo para as cidades escolhidas de acordo com a Tabela 5.1.

Tabela 5.1: Relação das cidades com as respectivas fontes XML experimentais.

Cidade	**Fonte**	**Documento XML**
Boston	(SUPERPAGES, 2012)	Boston_Superpages.xml
Canberra	(WEATHERZONE, 2012)	Canberra_weatherzone.xml
Caxias do Sul	(TEMPOAGORA, 2012)	Caxias_do_Sul_TempoAgora.xml
Curitiba	(TEMPOAGORA, 2012)	Curitiba_TempoAgora.xml
Orlando	(SUPERPAGES, 2012)	Orlando_Superpages.xml
Porto Alegre	(CPTEC, 2012)	Porto_Alegre_INPE.xml
Santa Maria	(CPTEC, 2012)	Santa_Maria_INPE.xml
Sydney	(WEATHERZONE, 2012)	Sydney_weatherzone.xml
Capitais	(CLIMATEMPO, 2012)	Capitais_ClimaTempo.xml

Considerando as fontes citadas de divulgação do boletim de previsão do tempo, o conteúdo dos documentos XML foi mantido inalterado. Além disso, as capitais das unidades federativas brasileiras estão reunidas em um único documento XML. Outras cidades, em documentos distintos, também foram arbitrariamente escolhidas para diversificar as fontes de dados.

A Tabela 5.1 apresenta as fontes de dados XML e os respectivos nomes dos arquivos. Para facilitar o entendimento da estrutura dos arquivos XML utilizados experimentalmente, foi criado o diagrama gráfico XML de cada documento (ZANELLA, 2011).

O conteúdo XML experimental e as representações gráficas podem ser encontrados no repositório próprio (AVELAR, 2012).

5.2 Ontologia resultante gerada pela OntoGen

A função da ontologia é descrever o domínio de conhecimento constituído pelas instâncias de dados XML, assim oferecendo semântica à informação e abstraindo as diferenciações de cada documento XML. A responsabilidade de reunir toda a informação em formato XML para constituir a ontologia, assim como identificar o domínio de conhecimento correspondente, é atribuída ao usuário do framework X2Rel.

Partindo do domínio de informações da previsão o tempo, os arquivos XML considerados na base inicial de testes foram utilizados como entrada para a ferramenta OntoGen. O resultado gráfico da ontologia está ilustrado na Figura 5.1. A ontologia descritiva dos documentos XML experimentais demonstra a presença de cinco conceitos não-léxicos, quatorze conceitos léxicos tipo `string`, dois conceitos léxicos do tipo numérico inteiro, um conceito léxico do tipo numérico de ponto flutuante e um conceito léxico do tipo lógico. De maneira geral, os conceitos não-léxicos são descritos para melhor entendimento do domínio de conhecimento experimental:

- rss: os boletins de previsão do tempo utilizados experimentalmente são documentos

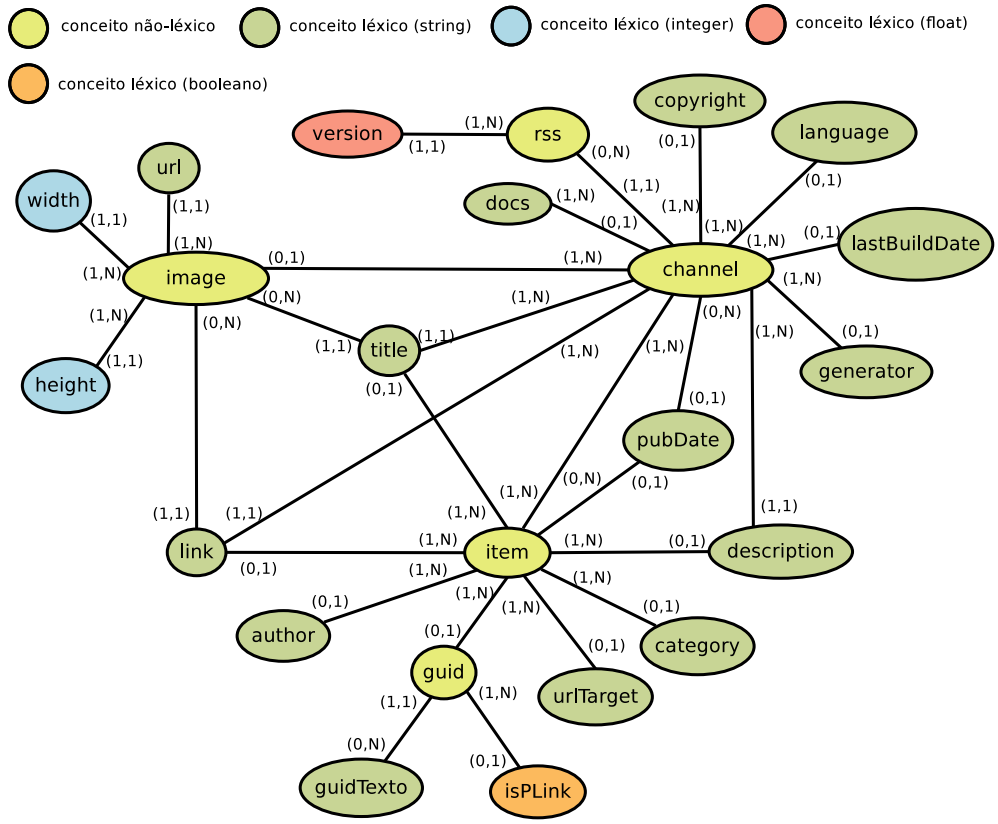

Figura 5.1: Resultado da ontologia descritiva dos documentos XML experimentais.

RSS fundamentados no padrão XML. O conceito `rss` representa os documentos RSS, incluindo a versão de sindicância de dados;

- image: representa as informações de imagem utilizadas como o logotipo do website de divulgação do boletim de previsão do tempo;
- channel: representa o canal de divulgação das informações de previsão do tempo, incluindo restrições de direitos autorais, idioma, data de atualização, descrição sobre o canal, fonte de geração dos dados de previsão do tempo e informações sobre o documento RSS;
- item: corresponde as informações de previsão do tempo contendo a data de publicação, a descrição meteorológica, categoria, que pode ser local, regional ou federativa; e o autor dos dados. Em especial, `urlTarget` indica a ação quando usuário clicar no link de hipertexto, podendo a página ser aberta na mesma janela ou em outra janela do navegador Internet;
- guid: representa o endereço de Internet para acesso ao boletim do tempo, podendo ser permanente, conforme indicado pelo conceito léxico lógico.

Os conceitos presentes nas ontologias representam as equivalências estabelecidas a partir dos documentos XML. Conforme apresentado na Seção 3.4.1, elementos XML podem ser agru-

pados como sinônimos pela ferramenta OntoGen para representação na ontologia. Além disso, os relacionamentos definem as dependências entre conceitos resultantes a partir da presença dos elementos em documentos XML. Estabelecendo uma analogia, as ontologias são como mapas conceituais na modelagem de um domínio de conhecimento.

5.3 Esquema lógico relacional gerado pela OntoRel

Em se tratando de documentos XML, a informação está disposta de maneira estrutural tanto em elementos como atributos de elementos. No caso de ontologias, as definições estruturais de XML são abstraídas em conceitos não-léxicos e conceitos léxicos. Todavia, o esquema lógico relacional utiliza definições como entidades, atributos e relacionamentos para estruturar a informação.

Considerando a ontologia ilustrada na Figura 5.1 para os documentos XML do domínio de previsão do tempo, a Figura 5.2 apresenta graficamente o esquema lógico relacional criado pela ferramenta OntoRel. Verifica-se que os conceitos não-léxicos da ontologia correspondem às tabelas no esquema relacional. Conceitos léxicos são colunas das respectivas tabelas. Os tipos de dados são equivalentes entre a ontologia e o esquema do banco de dados, como ocorre entre `string` na ontologia e `varchar` no esquema relacional.

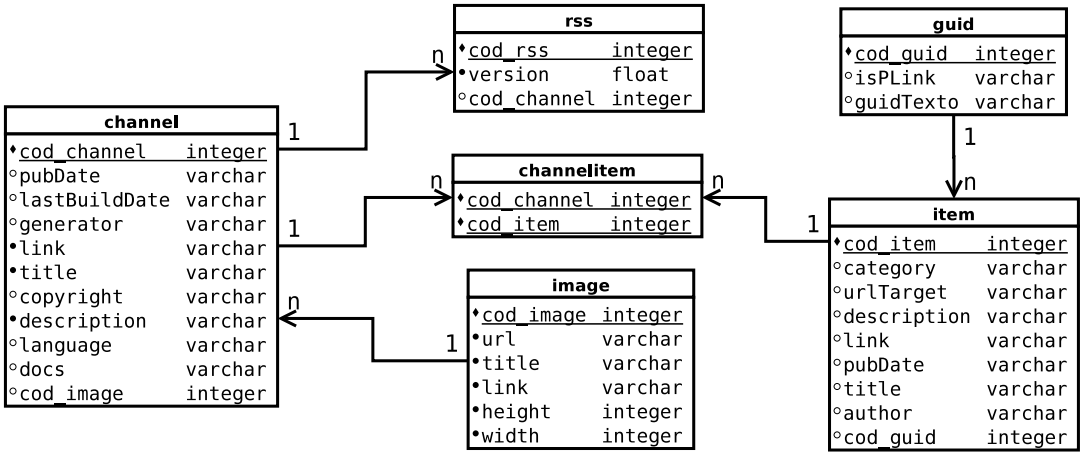

Figura 5.2: Esquema lógico relacional criado a partir da ontologia experimental.

Quando há um relacionamento entre conceitos não-léxicos, uma nova entidade é criada para a cardinalidade mínima direta e inversa igual a um, como exemplificado no relacionamento entre os conceitos `channel` e `item` na ontologia e a tabela `channelitem` no esquema relacional. Tal fato não ocorre no relacionamento entre os conceitos `image` e `channel`, pois há cardinalidade mínima igual a zero. Ressalta-se que a cardinalidade mínima igual a zero para os conceitos da ontologia resultaram em atributos com permissão de armazenamento de valor `null` no banco de dados. Maiores detalhes do funcionamento da ferramenta OntoRel podem ser lidas nas Seções 2.5 e 3.1.2.

As etapas apresentadas nas Seções 5.1, 5.2 e nesta seção, correspondem ao módulo XMap do framework X2Rel. As Seções 5.4, 5.5 e 5.6 abordarão os resultados referentes às etapas de XMap.

5.4 Documento de mapeamento

As fontes de dados inicialmente estão dispostas no formato de documentos XML, sendo o conteúdo de interesse para armazenamento em banco de dados relacional. A estrutura dos documentos XML conta com a ontologia para modelar semanticamente o domínio de conhecimento das instâncias de dados. A criação da base de dados relacional ocorre a partir da definição da ontologia, possibilitando uma abstração quanto as diferenciações estruturais de cada arquivo XML.

O documento de mapeamento resultante associa os conceitos presentes na ontologia com cada arquivo do boletim de previsão do tempo em formato XML. Conforme o fragmento ilustrado na Figura 5.3 retirado do documento de mapeamento, o conteúdo de cada conceito `<concept>` designado pelo atributo `name` é apontado por um caminho em `<xpath>` para busca no arquivo XML identificado pelo atributo `id` do elemento `<source>`.

```
01 <concept name="author">
02    <source id="Boston_Superpages.xml" />
03    <source id="Canberra_weatherzone.xml" />
04    <source
05       <xpath>/rss/channel/item/author</xpath>
06    </source>
07    <source id="Curitiba_TempoAgora.xml">
08       <xpath>/rss/channel/item/author</xpath>
09    </source>
10    <source id="Orlando_Superpages.xml" />
11    <source id="Porto_Alegre_INPE.xml">
12       <xpath>/rss/channel/item/author</xpath>
13    </source>
14    <source id="Santa_Maria_INPE.xml">
15       <xpath>/rss/channel/item/author</xpath>
16    </source>
17    <source id="Sydney_weatherzone.xml" />
18    <source id="capitais_ClimaTempo.xml">
19       <xpath>/channel/item/author</xpath>
20    </source>
21    <relational>
22       <type>column</type>
23       <name>author</name>
24       <table>item</table>
25       <domain>string</domain>
26    </relational>
27 </concept>
```

Figura 5.3: Fragmento do documento de mapeamento experimental.

Para facilitar a identificação, os arquivos XML foram indicados pelo nome da cidade acrescentado pela instituição de divulgação do boletim da previsão do tempo. Por exemplo, o arquivo `Porto_Alegre_INPE.xml` (linha 11) indica que o conteúdo XML da previsão do tempo refere-se à cidade de Porto Alegre retirado do website do Instituto Nacional de Pesquisas Espaciais (CPTEC, 2012).

Além de cada conceito da ontologia estar associado com o conteúdo das fontes de dados

XML, o documento de mapeamento contém os parâmetros do esquema lógico relacional, delimitados pelo elemento `<relational>`, para inserção dos dados. Considerando o fragmento exposto, o conceito léxico `author` terá o conteúdo, presente nas fontes de dados apontados pelo caminho XPath, armazenados na coluna `author` da tabela `item` como uma `string`. Tais afirmações podem ser verificadas a partir da linha 21 até a linha 26.

O módulo XMap do framework X2Rel não interpreta diretamente o documento OWL da ontologia. Os conceitos são lidos do documento de mapeamento associados com os demais componentes necessários para mapeamento do conteúdo XML a ser armazenado em banco de dados relacionais. O documento de mapeamento, proposto nesta dissertação e implementado como trabalho de conclusão de curso por (NEGRINI, 2011), pode ser acessado integralmente em repositório próprio (AVELAR, 2012).

5.5 Solução dos conflitos existentes

O processo de armazenamento do conteúdo das fontes de dados XML exige a criação da ontologia representativa do domínio de conhecimento e do esquema lógico relacional consequente. O documento de mapeamento integra os documentos XML com a modelagem ontológica e relacional da informação.

Além da função da integração entre modelos de dados, o documento de mapeamento é uma solução proposta para a resolução dos conflitos existentes na transformação do conteúdo XML a ser mapeado para armazenamento em banco de dados relacionais. Conforme apresentado na Seção 3.5, os conflitos de nomenclatura e de estrutura estão relacionados principalmente à estrutura dos documentos XML. Os conflitos de representação e de elemento inexistente estão mais voltados às regras de armazenamento em banco de dados. Em se tratando de ambos, considera-se as diferenças de representação de dado presente em documentos XML para armazenamento em banco de dados e a regra de integridade referencial de elemento nulo, respectivamente.

5.5.1 Conflitos de nomenclatura

O conflito de nomenclatura tipo homonímia pode ser verificado observando a ontologia da Figura 5.1. Os conceitos léxicos de relacionamento em comum com conceitos não-léxicos exemplificam os casos existentes em que um mesmo nome é utilizado em dois ou mais conceitos distintos. Tal fato ocorre com os conceitos `title`, `link`, `pubDate` e `description`. A Tabela 5.2 apresenta os casos de conflito de homonímia considerando o caminho XPath do conteúdo XML e o correspondente campo de armazenamento na tabela do esquema relacional.

Os caminhos XPath considerados no exemplo da Tabela 5.2 estão presentes nas fontes de dados XML utilizados na experimentação. Devido à quantidade de documentos, convencionou-se listar apenas o caminho XPath de maneira geral para ilustrar os casos de conflito de homonímia.

Tabela 5.2: Exemplos de ocorrências do conflito de nomenclatura do tipo homonínia.

Caminho XPath	**Tabela(campo)**
/rss/channel/title	channel(title)
/rss/channel/image/title	image(title)
/rss/channel/item/title	item(title)
/rss/channel/image/link	image(link)
/rss/channel/link	channel(link)
/rss/channel/item/link	item(link)
/rss/channel/pubDate	channel(pubDate)
/rss/channel/item/pubDate	item(pubDate)
/rss/channel/item/description	item(description)
/rss/channel/description	channel(description)

Em razão do esquema lógico relacional estar associado à ontologia, a solução para o conflito de homonímia pode ser verificada nos nomes de atributos iguais entre entidades diferentes na Figura 5.2. Considerando os atributos `title` e `link`, ambos estão presentes nas tabelas `channel`, `image` e `item`. Já os atributos `pubDate` e `description` estão presentes nas tabelas `channel` e `item`. Embora tais atributos possuam nomes iguais, cada situação de significado está explicitada como atributo de entidades relacionais distintas. O documento de mapeamento possui a finalidade de diferenciar os conceitos de mesma nomenclatura para atributos de entidades distintas juntamente com o caminho XPath das fontes de dados XML.

O conflito de nomenclatura do tipo sinonímia, da mesma forma como no caso de homonímia, também pode ser analisado pela ontologia da Figura 5.1. Os conceitos `guid` e `guidTexto`, embora sejam conceito não-léxico e conceito léxico, respectivamente, ilustram o caso de nomes diferentes que possuem significados equivalentes. A decorrência desse tipo de conflito ocorre em razão de alguns documentos XML possuírem o elemento `guid` sem a presença do atributo `@isPLink`, ao passo que outros possuem o elemento com o atributo.

Tabela 5.3: Exemplos de ocorrências do conflito de nomenclatura do tipo sinonínia.

Caminho XPath	**Descrição relacional**
/rss/channel/item/guid	coluna `guidTexto` da Tabela `guid`
/rss/channel/item/guid	coluna `guid`

Caso todos os documentos XML contassem somente com o elemento `guid` sem atributos, o elemento seria traduzido para um conceito léxico. Para considerar o atributo `@isPLink` associado ao elemento `guid`, o conceito `guidTexto` foi criado na ontologia para representar o conteúdo do elemento `guid`, ao passo que o conceito `isPLink` foi criado para representar o conteúdo do atributo. Conforme exemplificado na Tabela 5.3, ambos os caminhos XPath apontam

para o conteúdo de guid, embora estejam diferenciados entre atributo e entidade.

```
01  <concept name="guidTexto">
02    <source id="Boston_Superpages.xml" />
03    <source id="Canberra_weatherzone.xml">
04      <xpath>/rss/channel/item/guid</xpath>
05    </source>
06    <source id="Caxias_do_Sul_TempoAgora.xml">
07      <xpath>/rss/channel/item/guid</xpath>
08    </source>
09    <source id="Curitiba_TempoAgora.xml">
10      <xpath>/rss/channel/item/guid</xpath>
11    </source>
12    <source id="Orlando_Superpages.xml" />
13    <source id="Porto_Alegre_INPE.xml" />
14    <source id="Santa_Maria_INPE.xml" />
15    <source id="Sydney_weatherzone.xml">
16      <xpath>/rss/channel/item/guid</xpath>
17    </source>
18    <source id="capitais_ClimaTempo.xml">
19      <xpath>/channel/item/guid</xpath>
20    </source>
21    <relational>
22      <type>column</type>
23      <name>guidTexto</name>
24      <table>guid</table>
25      <domain>string</domain>
26    </relational>
27  </concept>
```

```
01  <concept name="guid">
02    <source id="Boston_Superpages.xml" />
03    <source id="Canberra_weatherzone.xml">
04      <xpath>/rss/channel/item/guid</xpath>
05    </source>
06    <source id="Caxias_do_Sul_TempoAgora.xml" />
07    <source id="Curitiba_TempoAgora.xml" />
08    <source id="Orlando_Superpages.xml" />
09    <source id="Porto_Alegre_INPE.xml" />
10    <source id="Santa_Maria_INPE.xml" />
11    <source id="Sydney_weatherzone.xml">
12      <xpath>/rss/channel/item/guid</xpath>
13    </source>
14    <source id="capitais_ClimaTempo.xml" />
15    <relational>
16      <type>table</type>
17      <name>guid</name>
18    </relational>
19  </concept>
```

Figura 5.4: Replicação dos caminhos XPath para solucionar o conflito de sinonímia entre conceitos léxicos e não-léxicos.

No documento de mapeamento, conforme o fragmento da Figura 5.4, os documentos XML, os quais somente contendo o elemento guid, o conteúdo é mapeado para o campo guidTexto. No caso de elemento e atributo, o conteúdo é mapeado para a tabela guid. Em se tratando da implementação do módulo XMap do framework X2Rel, o fato de guid ser uma tabela no esquema relacional fará com que o mapeamento do conteúdo para inserção no banco de dados somente ocorra com o atributo guidTexto. Sendo assim, os caminhos XPath são replicados para a inserção correta. A Figura 5.4 exemplifica o caso na linha 4 para ambos os conceitos e nas linhas 16 e 12, para os conceitos guidTexto e guid, respectivamente. Como a inserção do conteúdo XML não ocorre em uma tabela propriamente dita, mas sim em seus campos, os caminhos XPath do conceito guid estão replicados no conceito guidTexto de acordo com as as linhas mencionadas da Figura 5.4 do fragmento do documento de mapeamento.

5.5.2 Conflito de elemento inexistente

Considerando o relacionamento entre dois conceitos na ontologia, a cardinalidade mínima indica a presença de um determinado conceito na forma de elemento ou atributo nas fontes de dados XML. Caso a cardinalidade mínima seja zero, significa que o relacionamento não existe para todas as fontes de dados XML. De outro modo, a cardinalidade mínima igual a um demonstra que os conceitos pertinentes do relacionamento modelam as fontes XML de maneira global.

Ocorre o conflito de elemento inexistente quando há um relacionamento entre conceitos não-léxicos. Conforme observado na Figura 5.1, o relacionamento direto entre `channel` e `image` possui a cardinalidade mínima igual a zero. A interpretação consequente implica `channel` possuir no máximo um `image`. Nem todos os documentos XML possuem um `<image>` em `<channel>`.

Observando o esquema lógico relacional da Figura 5.2, o campo `cod_image` da tabela `channel` é chave estrangeira da tabela `image`. O fragmento do documento de mapeamento ilustrado na Figura 5.5 a seguir demonstra a presença de fontes de dados XML sem o caminho XPath para o conteúdo do conceito `image` (linhas 2, 6, 7, 8 e 18).

```
01 <concept name="image">
02   <source id="Boston_Superpages.xml" />
03   <source id="Canberra_weatherzone.xml">
04     <xpath>/rss/channel/image</xpath>
05   </source>
06   <source id="Caxias_do_Sul_TempoAgora.xml" />
07   <source id="Curitiba_TempoAgora.xml" />
08   <source id="Orlando_Superpages.xml" />
09   <source id="Porto_Alegre_INPE.xml">
10     <xpath>/rss/channel/image</xpath>
11   </source>
12   <source id="Santa_Maria_INPE.xml">
13     <xpath>/rss/channel/image</xpath>
14   </source>
15   <source id="Sydney_weatherzone.xml">
16     <xpath>/rss/channel/image</xpath>
17   </source>
18   <source id="capitais_ClimaTempo.xml" />
19   <relational>
20     <type>table</type>
21     <name>image</name>
22   </relational>
23 </concept>
```

Figura 5.5: Fragmento do documento de mapeamento para o conflito de elemento inexistente.

Conforme indicado no esquema relacional, `cod_image` permite armazenar conteúdo `null`. Como solução para o conflito de elemento inexistente, para as fontes de dados XML que não possuírem caminho XPath para o conceito `image`, a chave estrangeira assumirá o valor nulo. Observe que o mesmo tipo de conflito ocorre no relacionamento entre os conceitos `rss` e `channel`.

5.6 Inserção dos dados

O esquema lógico relacional, criado pela ferramenta OntoRel a partir da ontologia OWL, constitui a estrutura da informação a ser armazenada no banco de dados. Seguindo o modelo estabelecido, a próxima etapa é a inserção dos dados a partir do formato XML para o esquema relacional através do módulo XMap do framework X2Rel.

Para qualquer documento XML, assume-se a inexistência de elemento identificador de registro. Durante o processo de montagem das tuplas para inserção em banco de dados relacional, os valores de chaves primárias são criadas aleatoriamente na implementação de XMap. Como exemplo, a Figura 5.6 exibe a instrução SQL de inserção do conteúdo XML na tabela `image` do

banco de dados. Os valores da chave primária `cod_image` foram criados de maneira aleatória, pois o conteúdo não estava presente em documentos XML. Chave primária é um campo obrigatório para identificação de tuplas no esquema lógico relacional.

```
INSERT INTO image(cod_image, link, width, url, title, height) VALUES
(82, null, null, null, null, null),
(21, "http://www.weatherzone.com.au", 88, "http://www.weatherzone.com.au/images/logos/weatherzone_logo_88x20.gif", "Weatherzone", 20),
(47, null, null, null, null, null),
(35, null, null, null, null, null),
(60, null, null, null, null, null),
(39, "http://www7.cptec.inpe.br", 144, "http://img0.cptec.inpe.br/~rgrafico/portal_home/img/new_rss.jpg", "CPTEC/INPE", 58),
(27, "http://www7.cptec.inpe.br", 144, "http://img0.cptec.inpe.br/~rgrafico/portal_home/img/new_rss.jpg", "CPTEC/INPE", 58),
(55, "http://www.weatherzone.com.au", 88, "http://www.weatherzone.com.au/images/logos/weatherzone_logo_88x20.gif", "Weatherzone", 20),
(78, null, null, null, null, null);
```

Figura 5.6: Inserção de conteúdo na tabela `image` do banco de dados.

Conforme apresentado na Tabela 5.1, são nove documentos XML utilizados na experimentação. Desse modo, a Figura 5.6 exibe nove linhas de inserção de dados. Há linhas as quais há somente o valor da chave primária, sendo as demais iguais a `null`. Preferencialmente as linhas em questão foram mantidas na instrução de maneira a corresponder com a quantidade de documentos XML utilizados na etapa de experimentos.

Após resolvidas as situações de conflito, o resultado do processo de mapeamento do conteúdo XML em banco de dados relacionais é o *script* de inserção de dados em SQL (AVELAR, 2012). A partir de então, é possível a execução em banco de dados para armazenar os dados no esquema relacional previamente estabelecido.

5.7 Experimentos adicionais

O domínio de conhecimento de previsão do tempo abordado anteriormente considerou documentos XML contendo informações meteorológicas de cidades aleatoriamente escolhidas, conforme listado na Tabela 5.1. Para diversificar os casos de teste e envolver mais conflitos, experimentos adicionais foram realizados para verificação dos resultados.

Ao invés de utilizar documentos XML encontrados na Internet, foram criados arquivos de maneira convencional para experimentação. Nesse caso, o domínio de conhecimento adotado são informações de publicações em conferências, incluindo título do trabalho, dados dos autores e instituição. Embora existam conflitos abordados anteriormente, os experimentos realizados nesta seção somente abordarão aqueles não tratados no primeiro caso de testes.

Ilustrado na Figura 5.7, há três fragmentos de documentos XML, sendo que o destaque está para os elementos `<day>`, linha 5 de `Doc_A.xml` e linha 7 de `Doc_C.xml`. Além disso, considera-se o elemento `<code>` na linha 9 de `Doc_B.xml` e nas linhas 11 e 16 de `Doc_C.xml`, assim como o atributo `code` na linha 11 de `Doc_B.xml`. Para o nome de autor, o nome completo está presente em um único elemento `<name>`, como nas linhas 11, 14 e 17 de `Doc_A.xml`, e nas linhas 12 e 17 de `Doc_C.xml`. Além disso, nome e sobrenome de autor estão separados em

elementos específicos e agrupados pelo elemento <names>, conforme ilustrado na linha 12 até a 15 de Doc_B.xml.

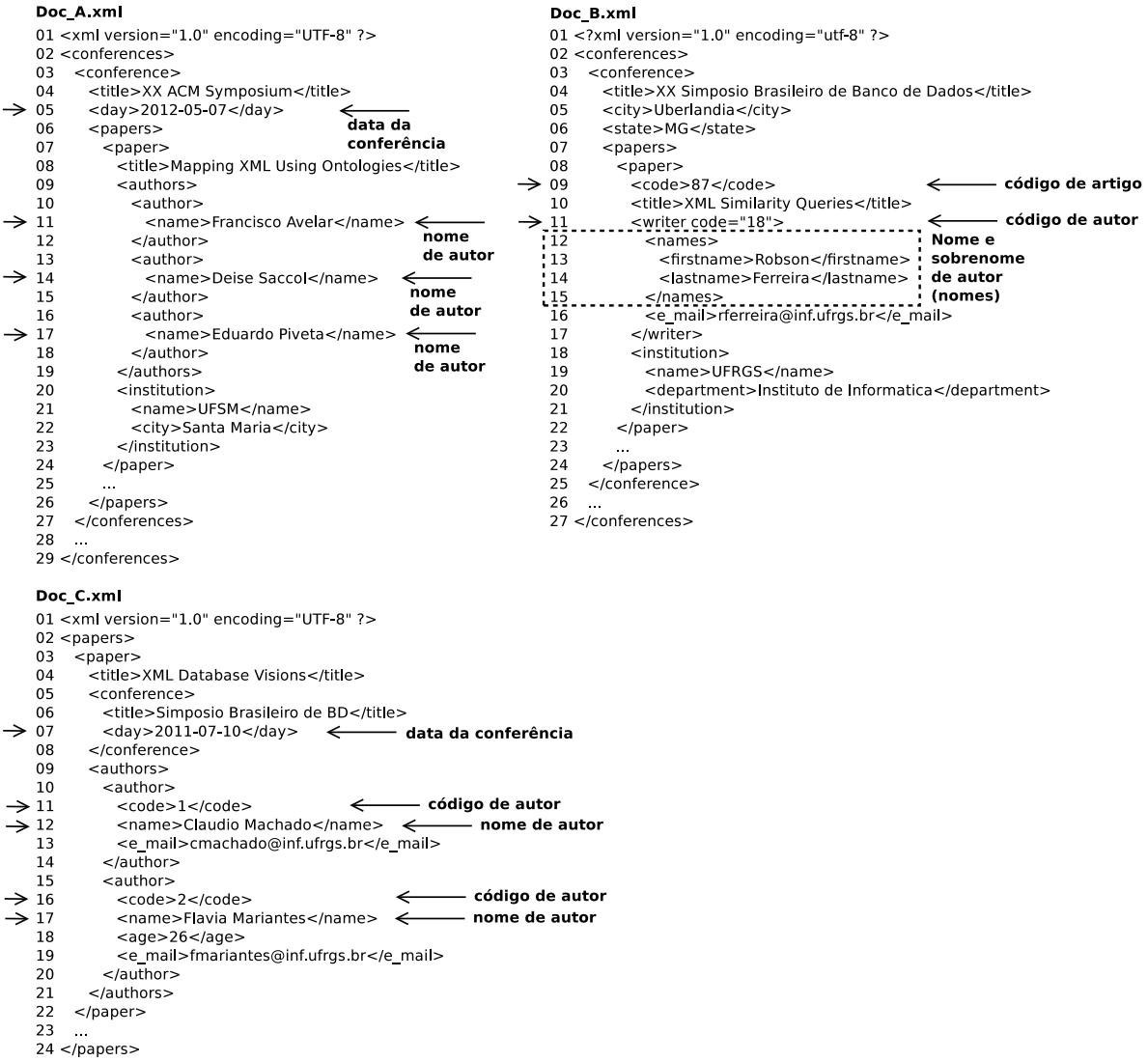

Figura 5.7: Fragmento dos documentos XML para o domínio de publicação de artigos em eventos.

No caso de elemento e atributo relativos ao código, evidencia-se um exemplo de conflito de estrutura modalidade tipo. A data no formato dddd-mm-aa foi propositalmente utilizada para interpretação do conteúdo XML para conceito léxico do tipo date pela ferramenta OntoGen. O formato de armazenamento de datas em banco de dados adotado é de acordo com a máscara dd/mm/aaaa, assim causando um conflito de representação. Os nomes de autores em um único elemento comparado com nome e sobrenome em elementos distintos indicam um conflito de generalização.

Os elementos <author> e <writer> dos documentos XML da Figura 5.7 foram considerados sinônimos e o conceito author foi criado pela ferramenta OntoGen como agrupamento. Em razão do elemento complexo <names>, linha 12 de Doc_B.xml, o conceito não-léxico names

está presente na ontologia resultante relacionando-se com o conceito `author`. A ontologia resultante para o domínio de publicações em conferências está exemplificada de forma gráfica na Figura 5.8.

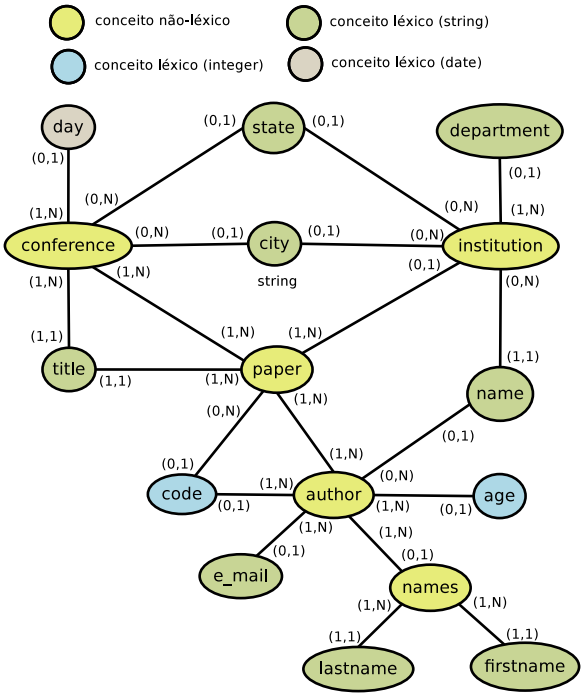

Figura 5.8: Ontologia resultante para o domínio de conhecimento relativo à publicação de eventos.

O esquema lógico relacional criado pela ferramenta OntoRel está ilustrado na Figura 5.9. Cada conceito não-léxico da ontologia passou a ser uma entidade. Os relacionamentos entre conceitos não-léxicos resultaram em uma associação com cardinalidade N:N no esquema relacional, fazendo com que as tabelas `paperwriter` e `conferencepaper` estivessem no esquema.

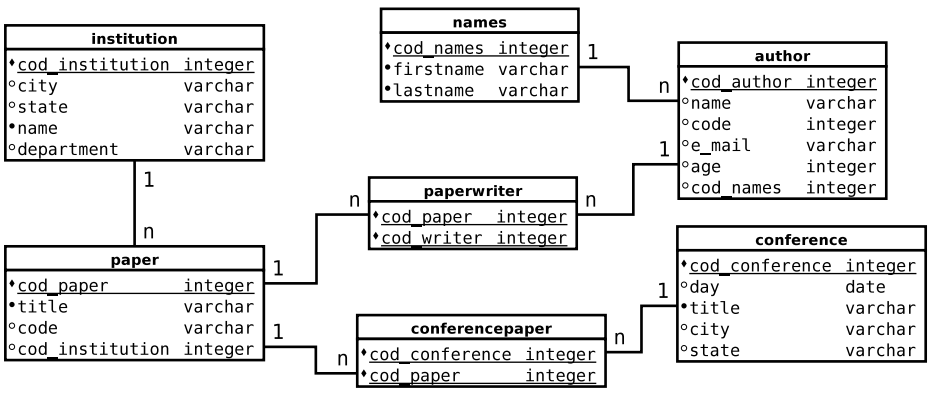

Figura 5.9: Esquema lógico relacional criado a partir da ontologia.

A resolução do conflito de estrutura modalidade tipo está no caminho XPath no fragmento do documento de mapeamento ilustrado na Figura 5.10. Para o atributo `code` do elemento `<writer>` (linha 11 de `Doc_B.xml`), o caminho XPath correspondente está na linha 6 do docu-

mento de mapeamento ilustrado. Nos demais casos nos quais o código está como elemento `<code>`, o caminho XPath está registrado na linha 9 na Figura 5.10.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <mapping>
03   <concept name="code">
04     <source id="Doc_A.xml" />
05     <source id="Doc_B.xml">
06       <xpath>/conferences/conference/papers/paper/writer/@code</xpath>
07     </source>
08     <source id="Doc_C.xml">
09       <xpath>/papers/paper/authors/author/code</xpath>
10     </source>
11     <relational>
12       <type>column</type>
13       <name>code</name>
14       <table>author</table>
15       <domain>integer</domain>
16     </relational>
17   </concept>
18 </mapping>
```

Figura 5.10: Fragmento do documento de mapeamento para o conflito de tipo.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <mapping>
03   <concept name="name">
04     <source id="Doc_A.xml">
05       <xpath>/conferences/conference/papers/paper/authors/author/name</xpath>
06     </source>
07     <source id="Doc_B.xml" />
08     <source id="Doc_C.xml">
09       <xpath>/papers/paper/authors/author/name</xpath>
20     </source>
21     <relational>
22       <type>column</type>
23       <name>name</name>
24       <table>author</table>
25       <domain>string</domain>
26     </relational>
27   </concept>
28   <concept name="firstname">
29     <source id="Doc_A.xml" />
30     <source id="Doc_B.xml">
31       <xpath>/conferences/conference/papers/paper/writer/names/firstname</xpath>
32     </source>
33     <source id="Doc_C.xml" />
34     <relational>
35       <type>column</type>
36       <name>firstname</name>
37       <table>names</table>
38       <domain>string</domain>
39     </relational>
40   </concept>
41   <concept name="lastname">
42     <source id="Doc_A.xml" />
43     <source id="Doc_B.xml">
44       <xpath>/conferences/conference/papers/paper/writer/names/lastname</xpath>
45     </source>
46     <source id="Doc_C.xml" />
47     <relational>
48       <type>column</type>
49       <name>lastname</name>
50       <table>names</table>
51       <domain>string</domain>
52     </relational>
53   </concept>
54 </mapping>
```

Figura 5.11: Versão oficial do documento de mapeamento criada pela ferramenta CMap para o conflito de generalização (fragmento).

A abordagem adotada para resolução do conflito de generalização pode ser vista no fragmento do documento de mapeamento da Figura 5.11. O caminho XPath da linha 5 corresponde ao nome de autor na forma do elemento único `<name>` para a tabela `author`. Além disso, os

caminhos XPath das linhas 31 e 44 correspondem ao documento XML com nome e sobrenome em elementos distintos, ambos para a tabela `names`.

Partindo da intervenção do usuário, uma segunda abordagem seria suprimir a tabela `names` do esquema lógico relacional da Figura 5.9. Desse modo, o documento de mapeamento deveria ser modificado manualmente de maneira a utilizar somente o campo `name` da tabela `author`. Como resultado, a Figura 5.12 ilustra o documento de mapeamento adaptado pelo usuário com o uso de funções XPath (linhas 8 e 9). Comparando o documento de mapeamento hipotético com o gerado pelo framework X2Rel (Figura 5.12 versus 5.11, respectivamente), verifica-se a ausência dos conceitos léxicos `firstname` e `lastname` (linhas 28 e 41 da Figura 5.11). Os conteúdos dos elementos `<firstname>` e `<lastname>` (linhas 13 e 14 de `Doc_B.xml` da Figura 5.7) são concatenados com a função `concat` de XPath.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <mapping>
03   <concept name="name">
04     <source id="Doc_A.xml">
05       <xpath>/conferences/conference/papers/paper/authors/author/name</xpath>
06     </source>
07     <source id="Doc_B.xml">
08       concat(/conferences/conference/papers/paper/writer/names/firstname, '',
09       /conferences/conference/papers/paper/writer/names/lastname)
10     </source>
11     <source id="Doc_C.xml">
12       <xpath>/papers/paper/authors/author/name</xpath>
13     </source>
14     <relational>
15       <type>column</type>
16       <name>name</name>
17       <table>author</table>
18       <domain>string</domain>
19     </relational>
20   </concept>
21 </mapping>
```

Figura 5.12: Versão hipotética do documento de mapeamento para o conflito de generalização (fragmento).

Considerando o fragmento do documento de mapeamento da Figura 5.10, verifica-se que a inserção do código do autor de artigo, seja na forma de elemento ou atributo em XML, assim como demais informações, procede-se na instrução `INSERT` da linha 4 da Figura 5.13.

Para o conflito de representação, as datas foram inseridas de acordo com a máscara definida como `dd/mm/aaaa`, conforme consta nas linhas 13 e 15 da Figura 5.13. Nesse caso, não há informação no documento de mapeamento, pois ocorre uma adaptação do conteúdo XML para a inserção no banco de dados.

O documento de mapeamento criado pela ferramenta CMap, ilustrado na Figura 5.11, é considerado como uma solução para o conflito de generalização. O resultado é a inserção no banco de dados parcialmente reproduzido na Figura 5.13. Nesse caso, há a utilização da tabela `names` e a instrução de inserção de dados consta nas linhas 1 e 2.

A chave estrangeira `cod_names` na tabela `author` ocasiona a presença do campo adicional

na inserção, a partir da linha 4 até a linha 10. O relacionamento entre as tabelas `names` e `author` faz com que o valor de `cod_names`, na tabela `names` (linha 2), esteja como o valor da correspondente chave estrangeira na tabela `author` (linha 8). Os demais casos, `cod_names`, como chave estrangeira na tabela `author`, assume valor `null`.

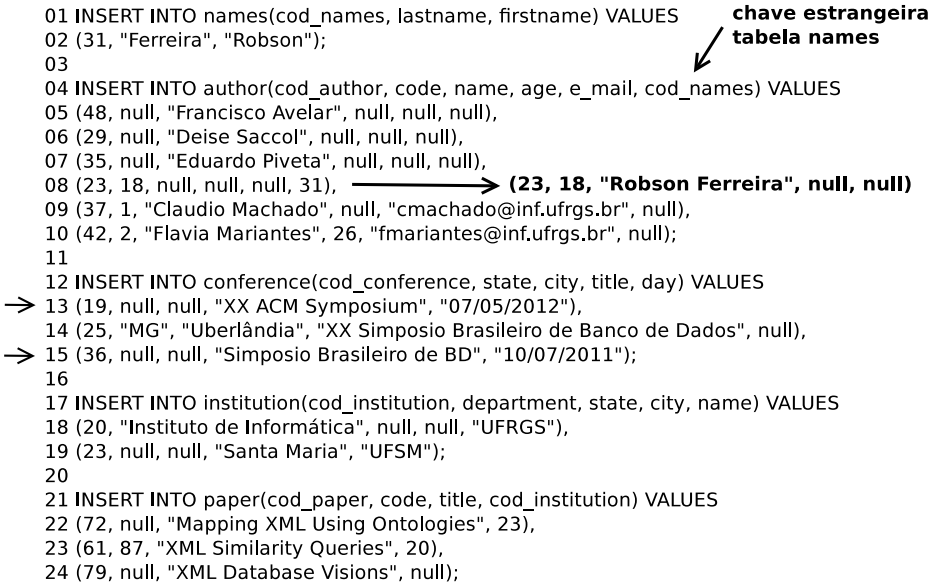

Figura 5.13: Parte da inserção do banco de dados.

Para o documento de mapeamento hipotético, ilustrado na Figura 5.12, a inserção no banco de dados passaria por algumas modificações. A tabela `names` não existiria, assim a inserção nas linhas 1 e 2 (Figura 5.13) não existia. O campo `cod_names` como chave estrangeira da tabela `author`, assim como os respectivos valores de inserção, não mais fariam parte do *script*. Além disso, a linha 8 passaria a ser conforme indicado pela seta na Figura 5.13, pois nome e sobrenome estariam concatenados pela função `concat` de XPath.

Os arquivos XML utilizados nos experimentos adicionais, assim como os resultados ilustrados ao longo desta seção encontram-se em repositório em repositório próprio (AVELAR, 2012).

5.8 Considerações finais

Para a etapa de experimentação, documentos XML diversos foram utilizados para armazenamento do conteúdo em banco de dados relacionais. Quanto a diversidade, trata-se das diferenças estruturais entre documentos XML definidos em um mesmo domínio de conhecimento. O usuário possui a responsabilidade de identificar o domínio de aplicação dos documentos XML cujo conteúdo seja de interesse para armazenar em banco de dados.

Considerando a Tabela 5.4, os conflitos classificados foram analisados em duas etapas de experimento. Desse modo, buscou-se abranger os casos de conflito desde da interpretação dos documentos XML até a criação do *script* de inserção dos dados.

Tabela 5.4: Manifestação dos conflitos na etapa de experimentação.

Conflito	**Caso de teste**
Nomenclatura (homonímia)	Primeiro
Nomenclatura (sinonímia)	Primeiro
Estrutura (tipo)	Segundo
Estrutura (generalização)	Segundo
Elemento inexistente	Primeiro
Representação	Segundo

Partindo dos documentos XML definidos em um domínio de conhecimento, as etapas de processamento e geração de esquemas de dados foram realizadas. Através da ferramenta OntoGen, a ontologia do domínio de aplicação foi apresentada. A partir da ontologia, a ferramenta OntoRel criou a estrutura lógica relacional do banco de dados.

Com a definição da ontologia, da estrutura XML do banco de dados e o documento de mapeamento, o módulo XMap passou a atuar na resolução de conflitos e criação de inserções com o conteúdo XML. Cada módulo do framework X2Rel gera uma determinada saída, sendo necessária para a próxima etapa. Os documentos XML e os artefatos de saída utilizados na experimentação, assim como o *script* de inserção dos dados, podem ser acessados em repositório próprio (AVELAR, 2012).

Para a análise experimental do módulo XMap do framework X2Rel, foram utilizados documentos XML de boletins da previsão meteorológica para cidades aleatoriamente escolhidas. A busca foi realizada em feeds RSS presentes em websites de divulgação da previsão do tempo, conforme especificado na Tabela 5.1. Em uma segunda etapa de experimentos, documentos XML foram criados convencionalmente abordando os conflitos não manifestados na primeira parte experimental.

Em razão da estrutura dos documentos XML considerados, somente alguns dos conflitos apresentados no processo de mapeamento do conteúdo XML a ser armazenado em bancos de dados relacionais foram constatados em uma única etapa de experimentação. Testes com diferentes documentos XML podem apresentar comportamento distinto.

6 CONCLUSÕES

O processo de mapeamento do conteúdo XML para armazenamento em bancos de dados relacionais é constituído por etapas intermediárias de transformação de modelos de documentos. Partindo dessa divisão em tarefas menores, cada módulo do framework X2Rel contribui para transformação de dados XML para o formato relacional.

Primeiramente, a identificação de domínio de conhecimento dos documentos XML é tarefa do usuário. A partir disso, a ontologia descritiva poderá ser criada usando a ferramenta OntoGen. É importante a definição de um domínio de conhecimento, por parte do usuário, para agrupamento dos documentos XML para a representação em uma ontologia. A seguir, a ontologia passa a ser processada pela ferramenta OntoRel para a criação do esquema lógico relacional.

No âmbito do módulo XMap, a próxima tarefa é a criação do documento de mapeamento de maneira a representar as equivalências entre as fontes de dados XML, a ontologia descritiva e o esquema lógico relacional. A estrutura do documento de mapeamento, proposto neste trabalho, foi implementado à parte (NEGRINI, 2011) para que a criação de documentos de mapeamento seja feita de maneira automatizada.

Além de gerar as equivalências entre os documentos XML, a ontologia e o esquema lógico relacional, a proposta do documento de mapeamento foi de possibilitar uma abordagem para a resolução de conflitos pertinentes no processo de transformação do conteúdo XML para ser armazenado em banco de dados relacionais. Os algoritmos desenvolvidos para a resolução de conflitos consideram o documento de mapeamento como artefato fundamental para propor abordagens simplificadas na resolução de problemas.

Além do documento de mapeamento, a estrutura em XML do esquema lógico relacional, criada pela ferramenta OntoRel, constitui um mecanismo importante para verificação do esquema relacional. No âmbito de implementação, é mais conveniente processar a estrutura XML relacional em comparação com a descrição em SQL do esquema lógico relacional.

Por fim, a responsabilidade do módulo XMap é interpretar o documento de mapeamento e buscar o conteúdo nas fontes de dados XML pelo caminho XPath relativo a cada conceito da ontologia associando com o armazenamento no esquema lógico relacional. Nesse intermédio, os conflitos existentes devem ser resolvidos em tempo de execução. Uma vez criados o cabeçalho de inserção e as tuplas, o *script* de inserção em banco de dados é criado conforme o esquema XML relacional de maneira a obedecer às restrições de integridade referencial.

Através de experimentação, a presença de conflitos pode variar conforme os documentos XML utilizados. Uma maior diferenciação estrutural entre as fontes de dados pode resultar em ontologias mais complexas e também abranger uma resolução maior de conflitos.

6.1 Contribuições da dissertação

Conforme mencionado na seção 1.2, as contribuições deste trabalho são:

- a análise e a classificação de tipos de conflitos estruturais e semânticos em documentos XML;
- a definição de regras de mapeamento do conteúdo XML para o esquema relacional, com base nos conflitos definidos;
- a especificação de funções de transformação para eliminar conflitos estruturais e semânticos;
- a definição da estrutura do documento de mapeamento que descreve as equivalências de conceitos entre documentos XML, a ontologia e o esquema lógico relacional de banco de dados;
- a implementação de um protótipo do mecanismo proposto;

Quanto a publicações, estava sob processo de avaliação os artigos listados a seguir até a data de defesa final desta dissertação (30 de março de 2012). Na ocasião da entrega do texto pós-defesa, ambos foram aceitos conforme especificados abaixo:

- "An ontology-based approach for storing XML data into relational databases", por Francisco Tiago Machado de Avelar, Deise de Brum Saccol e Eduardo Kessler Piveta. Conferência "The 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012)". Classificada como estrato B2 no qualis da CAPES.
- "CMAP: Tracking Equivalences from XML Documents to Relational Schemas", por Douglas Negrini, Deise de Brum Saccol, Francisco Tiago Machado de Avelar e Eduardo Kessler Piveta. Conferência "The 13th International Conference on Information Reuse and Integration (IRI 2012)". Classificada como estrato B2 no qualis da CAPES.

Embora o documento de mapeamento tenha sido proposto neste trabalho, a abordagem teórica e implementação foi tema de trabalho de conclusão de curso à parte, sendo proposto por (NEGRINI, 2011).

6.2 Trabalhos futuros

Como continuidade no tema de pesquisa, a seguir alguns trabalhos futuros são apontados e podem ser de interesse:

- Consultas definidas a partir da estrutura XML, normalmente desenvolvidas em XQuery, poderão ser mapeadas para consultas em SQL. A estrutura dos documentos XML é abstraída em uma ontologia para criação do esquema lógico relacional, assim como o conteúdo dos documentos XML é armazenado em banco de dados. A migração completa ocorre também com o mapeamento das consultas XML para SQL.

- O esquema lógico relacional resultante poderá ser normalizado para melhor otimização no armazenamento da informação. Como consequência, o esquema proposto pela ferramenta OntoRel será modificado.

- Aliada à normalização, técnicas de eliminação de redundância e replicação de dados podem ser elaboradas para criação de uma base de dados mais refinada.

- Novos tipos de conflitos devido a complexidade na presença de fontes de dados XML estruturalmente distintas pode resultar em tipos diferentes de conflitos não classificados neste trabalho.

- O *script* de inserção em banco de dados pode ser feito em uma linguagem SQL genérica. Desse modo, a instrução de inserção não fica dependente de um determinado banco de dados comercial. Uma camada de tradução do SQL geral para o formato específico dependente de implementação deve ser implementado. Há soluções dependentes de tecnologia que realizam a tradução de maneira facilitada através de linguagem SQL genérica (HSQLDB, 2012).

APÊNDICE A – Documento XML de representação do modelo lógico

Documento XML de representação do modelo lógico relacional de banco de dados conforme exemplo da figura 3.3. Criado pela ferramenta OntoRel juntamente com o script SQL de criação das tabelas.

```
<tabelas>
  <tabela>
    <nome>paper</nome>
    <coluna>
      <nome>cod_paper</nome>
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>title</nome>
      <tipo>string</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>cod_institution</nome>
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <restricao>
      <tipo>chave_primaria</tipo>
      <coluna>cod_paper</coluna>
    </restricao>
    <restricao>
      <tipo>chave_estrangeira</tipo>
      <coluna>cod_institution</coluna>
      <referencia>institution</referencia>
    </restricao>
  </tabela>
  <tabela>
    <nome>conference</nome>
    <coluna>
      <nome>cod_conference</nome>
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>day</nome>
```

```
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <coluna>
      <nome>title</nome>
      <tipo>string</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>city</nome>
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <coluna>
      <nome>state</nome>
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <restricao>
      <tipo>chave_primaria</tipo>
      <coluna>cod_conference</coluna>
    </restricao>
  </tabela>
  <tabela>
    <nome>institution</nome>
    <coluna>
      <nome>cod_institution</nome>
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>city</nome>
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <coluna>
      <nome>state</nome>
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <coluna>
      <nome>name</nome>
      <tipo>string</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>department</nome>
      <tipo>string</tipo>
      <obrigatoria>false</obrigatoria>
    </coluna>
    <restricao>
      <tipo>chave_primaria</tipo>
      <coluna>cod_institution</coluna>
    </restricao>
  </tabela>
  <tabela>
    <nome>writer</nome>
    <coluna>
      <nome>cod_writer</nome>
```

```
    <tipo>integer</tipo>
    <obrigatoria>true</obrigatoria>
  </coluna>
  <coluna>
    <nome>name</nome>
    <tipo>string</tipo>
    <obrigatoria>false</obrigatoria>
  </coluna>
  <coluna>
    <nome>code</nome>
    <tipo>integer</tipo>
    <obrigatoria>false</obrigatoria>
  </coluna>
  <coluna>
    <nome>e_mail</nome>
    <tipo>string</tipo>
    <obrigatoria>false</obrigatoria>
  </coluna>
  <coluna>
    <nome>age</nome>
    <tipo>integer</tipo>
    <obrigatoria>false</obrigatoria>
  </coluna>
  <restricao>
    <tipo>chave_primaria</tipo>
    <coluna>cod_writer</coluna>
  </restricao>
</tabela>
<tabela>
  <nome>conferencepaper</nome>
  <coluna>
    <nome>cod_conference</nome>
    <tipo>integer</tipo>
    <obrigatoria>true</obrigatoria>
  </coluna>
  <coluna>
    <nome>cod_paper</nome>
    <tipo>integer</tipo>
    <obrigatoria>true</obrigatoria>
  </coluna>
  <restricao>
    <tipo>chave_primaria_composta</tipo>
    <coluna>cod_conference,cod_paper</coluna>
  </restricao>
  <restricao>
    <tipo>chave_estrangeira</tipo>
    <coluna>cod_conference</coluna>
    <referencia>conference</referencia>
  </restricao>
  <restricao>
    <tipo>chave_estrangeira</tipo>
    <coluna>cod_paper</coluna>
    <referencia>paper</referencia>
  </restricao>
</tabela>
<tabela>
  <nome>paperwriter</nome>
  <coluna>
    <nome>cod_paper</nome>
```

```
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <coluna>
      <nome>cod_writer</nome>
      <tipo>integer</tipo>
      <obrigatoria>true</obrigatoria>
    </coluna>
    <restricao>
      <tipo>chave_primaria_composta</tipo>
      <coluna>cod_paper,cod_writer</coluna>
    </restricao>
    <restricao>
      <tipo>chave_estrangeira</tipo>
      <coluna>cod_paper</coluna>
      <referencia>paper</referencia>
    </restricao>
    <restricao>
      <tipo>chave_estrangeira</tipo>
      <coluna>cod_writer</coluna>
      <referencia>writer</referencia>
    </restricao>
  </tabela>
</tabelas>
```

APÊNDICE B – Estruturas utilizadas nos algoritmos de resolução de conflitos

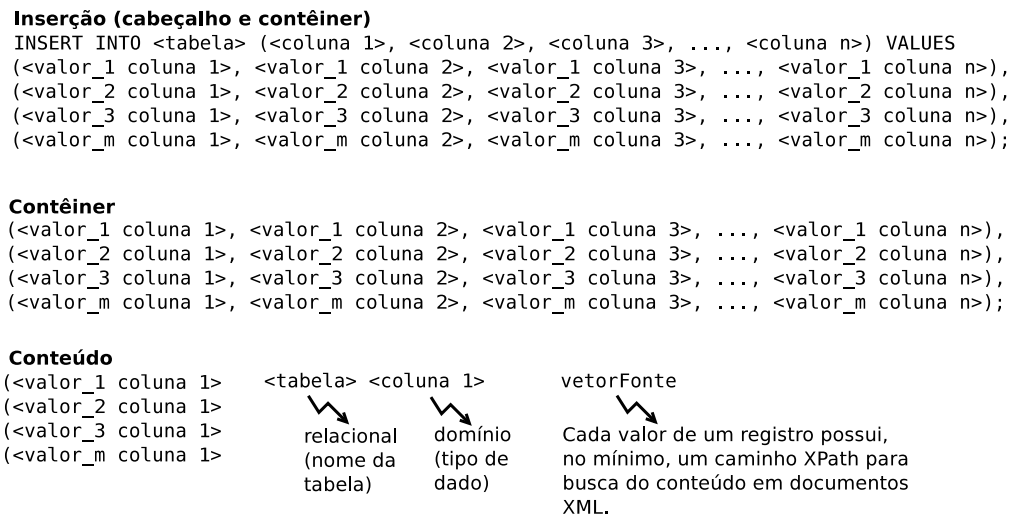

Figura B.1: Estruturas utilizadas no algoritmo de resolução dos conflitos.

APÊNDICE C – Descritores XML utilizados na biblioteca Castor

Descritor para o documento de mapeamento apresentado na seção 3.4.

```
<?xml version="1.0"?>
<!DOCTYPE mapping PUBLIC "-//EXOLAB/Castor Mapping DTD Version 1.0//EN"
                         "http://castor.org/mapping.dtd">
<mapping>
   <class name="descritor.Mapeamento">
      <map-to xml="mapeamento"/>
      <field name="ListaConceito" type="descritor.Conceito" collection="arraylist">
         <bind-xml name="conceito"/>
      </field>
   </class>
   <class name="descritor.Conceito">
      <field name="Nome" type="java.lang.String">
         <bind-xml name="nome" node="attribute"/>
      </field>
      <field name="Relacional" type="descritor.Relacional">
         <bind-xml name="relacional"/>
      </field>
      <field name="ListaFonte" type="descritor.Fonte" collection="arraylist">
         <bind-xml name="fonte"/>
      </field>
   </class>
   <class name="descritor.Fonte">
      <field name="Id" type="java.lang.String">
         <bind-xml name="id" node="attribute"/>
      </field>
      <field name="Expressao" type="java.lang.String">
         <bind-xml name="expressao" node="element"/>
      </field>
   </class>
   <class name="descritor.Relacional">
      <field name="Tipo" type="java.lang.String">
         <bind-xml name="tipo" node="element"/>
      </field>
      <field name="Nome" type="java.lang.String">
         <bind-xml name="nome" node="element"/>
      </field>
      <field name="Tabela" type="java.lang.String">
         <bind-xml name="tabela" node="element"/>
      </field>
      <field name="Dominio" type="java.lang.String">
```

```
         <bind-xml name="dominio" node="element"/>
      </field>
   </class>
</mapping>
```

Descritor para a representação XML do esquema lógico criado pela ferramenta OntoRel apresentado na seção 3.1.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapping PUBLIC "-//EXOLAB/Castor Mapping DTD Version 1.0//EN"
                         "http://castor.org/mapping.dtd">
<mapping>
   <class name="esquemabd.Tabelas">
      <map-to xml="tabelas"/>
      <field name="arrayTabela" type="esquemabd.Tabela" collection="arraylist">
         <bind-xml name="tabela"/>
      </field>
   </class>
   <class name="esquemabd.Tabela">
      <field name="nome" type="java.lang.String">
         <bind-xml name="nome" node="element"/>
      </field>
      <field name="arrayColuna" type="esquemabd.Coluna" collection="arraylist">
         <bind-xml name="coluna"/>
      </field>
      <field name="arrayRestricao" type="esquemabd.Restricao" collection="arraylist">
         <bind-xml name="restricao"/>
      </field>
   </class>
   <class name="esquemabd.Coluna">
      <field name="nome" type="java.lang.String">
         <bind-xml name="nome" node="element"/>
      </field>
      <field name="tipo" type="java.lang.String">
         <bind-xml name="tipo" node="element"/>
      </field>
      <field name="obrigatoria" type="java.lang.String">
         <bind-xml name="obrigatoria" node="element"/>
      </field>
   </class>
   <class name="esquemabd.Restricao">
      <field name="tipo" type="java.lang.String">
         <bind-xml name="tipo" node="element"/>
      </field>
      <field name="coluna" type="java.lang.String">
         <bind-xml name="coluna" node="element"/>
      </field>
      <field name="referencia" type="java.lang.String">
         <bind-xml name="referencia" node="element"/>
      </field>
   </class>
</mapping>
```

APÊNDICE D – Exemplo de documento de mapeamento

Documento de mapeamento representando a integração entre os esquemas XML, a ontologia em linguagem OWL e o modelo lógico relacional. A partir do elemento raiz `<mapeamento>`, o atributo `nome` do elemento `<conceito>` refere-se ao conceito da ontologia. A partir disso, o elemento `<fonte>` possui as fontes de dados em XML identificados pelo atributo `id` pertencentes ao conceito da ontologia. O elemento `<relacional>` possui `<tipo>` e `<nome>` para conceitos não-léxicos, ao passo que os elementos adicionais `<tabela>` e `<domínio>` correspondem a conceitos léxicos.

```
<?xml version="1.0" encoding="utf-8" ?>
<mapeamento>
   <conceito nome="conference">
      <fonte id="Doc_A__EstudoDeCaso.xml">
         <expressao>/conferences/conference</expressao>
      </fonte>
      <fonte id="Doc_B__EstudoDeCaso.xml">
         <expressao>/conferences/conference</expressao>
      </fonte>
      <fonte id="Doc_C__EstudoDeCaso.xml">
         <expressao>/papers/paper/conference</expressao>
      </fonte>
      <relacional>
         <tipo>Tabela</tipo>
         <nome>conference</nome>
      </relacional>
   </conceito>
   <conceito nome="title">
      <fonte id="Doc_A__EstudoDeCaso.xml">
         <expressao>/conferences/conference/title</expressao>
      </fonte>
      <fonte id="Doc_B__EstudoDeCaso.xml">
         <expressao>/conferences/conference/title</expressao>
      </fonte>
      <fonte id="Doc_C__EstudoDeCaso.xml">
         <expressao>/papers/paper/conference/title</expressao>
      </fonte>
      <relacional>
         <tipo>coluna</tipo>
         <nome>title</nome>
         <tabela>conference</tabela>
```

```
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="day">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/day</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/conference/day</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>day</nome>
      <tabela>conference</tabela>
      <dominio>date</dominio>
   </relacional>
</conceito>
<conceito nome="city">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/city</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>city</nome>
      <tabela>conference</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="state">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/state</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>state</nome>
      <tabela>conference</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="paper">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
```

```
      <expressao>/conferences/conference/papers/paper</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper</expressao>
   </fonte>
   <relacional>
      <tipo>Tabela</tipo>
      <nome>paper</nome>
   </relacional>
</conceito>
<conceito nome="title">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/title</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/title</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/title</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>title</nome>
      <tabela>paper</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="writer">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/authors/author</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/writer</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/authors/author</expressao>
   </fonte>
   <relacional>
      <tipo>Tabela</tipo>
      <nome>writer</nome>
   </relacional>
</conceito>
<conceito nome="code">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/writer/@code</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/authors/author/code</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>code</nome>
      <tabela>writer</tabela>
      <dominio>integer</dominio>
   </relacional>
</conceito>
```

```
<conceito nome="name">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/writer/name</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/authors/author/name</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>name</nome>
      <tabela>writer</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="e_mail">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/writer/e_mail</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/authors/author/e_mail</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>e_mail</nome>
      <tabela>writer</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="age">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao>/papers/paper/authors/author/age</expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>age</nome>
      <tabela>writer</tabela>
      <dominio>integer</dominio>
   </relacional>
</conceito>
<conceito nome="institution">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
```

```
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>Tabela</tipo>
      <nome>institution</nome>
   </relacional>
</conceito>
<conceito nome="name">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution/name</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution/name</expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>name</nome>
      <tabela>institution</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="city">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution/city</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>city</nome>
      <tabela>institution</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="state">
   <fonte id="Doc_A__EstudoDeCaso.xml">
      <expressao>/conferences/conference/papers/paper/institution/state</expressao>
   </fonte>
   <fonte id="Doc_B__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <fonte id="Doc_C__EstudoDeCaso.xml">
      <expressao></expressao>
   </fonte>
   <relacional>
      <tipo>coluna</tipo>
      <nome>state</nome>
      <tabela>institution</tabela>
      <dominio>string</dominio>
   </relacional>
</conceito>
<conceito nome="department">
```

```
        <fonte id="Doc_A__EstudoDeCaso.xml">
           <expressao></expressao>
        </fonte>
        <fonte id="Doc_B__EstudoDeCaso.xml">
           <expressao>/conferences/conference/papers/paper/institution/department</expressao>
        </fonte>
        <fonte id="Doc_C__EstudoDeCaso.xml">
           <expressao></expressao>
        </fonte>
        <relacional>
           <tipo>coluna</tipo>
           <nome>department</nome>
           <tabela>institution</tabela>
           <dominio>string</dominio>
        </relacional>
     </conceito>
  </mapeamento>
```

REFERÊNCIAS BIBLIOGRÁFICAS

ABITEBOUL, S.; SEGOUFIN, L.; VIANU, V. Representing and querying xml with incomplete information. *ACM Trans. Database Syst.*, ACM, New York, NY, USA, v. 31, n. 1, p. 208–254, mar 2006. ISSN 0362-5915. Disponível em: <http://doi.acm.org/10.1145/1132863.1132869>.

ANDRADE, T. de C. *Mapeamento de Esquemas XML Integrados para Bancos de Dados Relacionais*. Dezembro 2010. Trabalho de Graduação.

ASTROVA, I.; KORDA, N.; KALJA, A. Storing OWL Ontologies in SQL Relational Databases. In: . [S.l.]: International Journal of Electrical, Computer, and Systems Engineering, 2007. v. 1, n. 4.

ATAY, M. et al. Efficient schema-based XML-to-Relational data mapping. *Information Systems*, v. 32, n. 3, p. 458 – 476, 2007. ISSN 0306-4379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437905001286>.

AVELAR. *Dados XML Experimentais*. Avelar, 2012. Disponível em: <http://www.infovisao.com/xml_data/>. Acesso em: 27 de fevereiro de 2012.

BOHRING, H.; AUER, S. Mapping XML to OWL Ontologies. In: *Leipziger Informatik-Tage, volume 72 of LNI*. [S.l.]: GI, 2005. p. 147–156.

BORST, W. N. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. 215 p. Tese (Doutorado) — University of Twente, Enschede, 1997.

BOZZI, E. V.; MELLO, R. dos S. Gerência de Informações de Mapeamento no Sistema de Integração de Esquemas XML BInXS. *VII Escola Regional de Banco de Dados (ERBD 2011)*, 2011.

BRAGANHOLO, V. de P. *From XML to Relational View Updates: applying old solutions to solve a new problem*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2004. PPGC da UFRGS.

CHAMBERLIN, D.; ROBIE, J.; FLORESCU, D. Quilt: An xml query language for heterogeneous data sources. In: GOOS, G. et al. (Ed.). *The World Wide Web and Databases*. Springer Berlin / Heidelberg, 2001, (Lecture Notes in Computer Science, v. 1997). p. 1–25. ISBN 978-3-540-41826-9. 10.1007/3-540-45271-0_1. Disponível em: <http://dx.doi.org/10.1007/3-540-45271-0_1>.

CLIMATEMPO. *Empresa privada de meteorologia da América Latina*. ClimaTempo, 2012. Disponível em: <http://www.climatempo.com.br/>. Acesso em: 10 de janeiro de 2012.

CPTEC. *Centro de Previsão do Tempo e Estudos Climáticos*. INPE, 2012. Disponível em: <http://www.cptec.inpe.br/>. Acesso em: 10 de janeiro de 2012.

DAYEN, I. *Storing XML in Relational Databases*. O'Reilly, 2001. Disponível em: <http://www.xml.com/pub/a/2001/06/20/databases.html>. Acesso em: 11 de março de 2012.

DO, N.; RAHAYU, W.; TORABI, T. Conflict Detection Method in Adopting Global XML Standard for Database Systems. *International Conference on Ubiquitous Information Management and Comunication (ACM ICUIMC)*, 2011.

EXOLAB. *Using Castor XML*. ExoLab Group, 2012. Disponível em: <http://castor.org/xml-framework.html>. Acesso em: 11 de março de 2012.

FAN, W.; LIBKIN, L. On xml integrity constraints in the presence of dtds. *J. ACM*, ACM, New York, NY, USA, v. 49, n. 3, p. 368–406, maio 2002. ISSN 0004-5411. Disponível em: <http://doi.acm.org/10.1145/567112.567117>.

FLORESCU, D.; KOSSMANN, D. Storing and querying xml data using an rdmbs. *IEEE Data Engineering Bulletin*, v. 22, p. 27–34, 1999.

GRUBER, T. R. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, v. 43, n. 5, p. 907–928, 1993.

HSQLDB. *HyperSQL*. The HSQL Development Group, 2012. Disponível em: <http://hsqldb.org/>. Acesso em: 20 de fevereiro de 2012.

JONGE, A. de. *Comparing XML database approaches*. IBM Developer Works, 2012. Disponível em: <http://www.ibm.com/developerworks/library/x-comparexmldb/>. Acesso em: 16 de abril de 2012.

KEDAD, Z.; XUE, X. Mapping Discovery for XML Data Integration. In: *OTM Conferences (1)'05*. [S.l.: s.n.], 2005. p. 166–182.

KHAN, L.; RAO, Y. A Performance Evaluation of Storing XML Data in Relational Database Management Systems. In: *Proceedings of the 3rd International Workshop on Web Information and Data Management*. New York, NY, USA: ACM, 2001. (WIDM '01), p. 31–38. ISBN 1-58113-444-4.

LEEO, K.-H. et al. Conflict Classification and Resolution in Heterogeneous Information Integration based on XML Schema. *IEEE TENCON*, 2002.

LEGLER, F.; NAUMANN, F. A classification of schema mappings and analysis of mapping tools. *GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW 2007).*, 2007.

LEHTI, P.; FANKHAUSER, P. XML data integration with OWL: experiences and challenges. In: *Proceedings of the 2004 International Symposium on Applications and the Internet*. [S.l.: s.n.], 2004. p. 160 – 167.

MADRIA, S.; PASSI, K.; BHOWMICK, S. An XML Schema Integration and Query Mechanism System. *Data Knowl. Eng.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 65, n. 2, p. 266–303, maio 2008. ISSN 0169-023X. Disponível em: <http://dx.doi.org/10.1016/j.datak.2007.09.008>.

MAEDCHE, A.; STAAB, S. Measuring similarity between ontologies. In: GÓMEZ-PÉREZ, A.; BENJAMINS, V. (Ed.). *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*. [S.l.]: Springer Berlin / Heidelberg, 2002, (Lecture Notes in Computer Science, v. 2473). p. 15–21. ISBN 978-3-540-44268-4. 10.1007/3-540-45810-7_24.

MARTINS, P.; LAENDER, A. H. F. Mapeamento de Definições XML Schema para SQL: 1999. In: *SBBD'05*. [S.l.: s.n.], 2005. p. 100–114.

MELLO, M. R. de. *OntoGen: Uma Ferramenta para Integração de Esquemas XML*. Novembro 2007. Trabalho de Graduação.

MELLO, R. dos S. *Uma Abordagem Bottom-Up para a Integração Semântica de Esquemas XML*. 145 p. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2002.

NEGRINI, D. *CMAP: Geração e Representação de Equivalências entre Documentos XML, Ontologia e Modelo Relacional*. Dezembro 2011. Trabalho de Graduação.

NOLL, R. P. Uma proposta para análise de similaridade entre documentos XML e ontologias definidas em OWL. *Universidade Federal do Rio Grande do Sul*, 2007. Curso de Especialização em WEB e Sistemas de Informação.

ORACLE. *Using XMLType*. Oracle, 2012. Disponível em: <http://docs.oracle.com/cd/B10501_01/appdev.920/a96620/xdb04cre.htm>. Acesso em: 16 de março de 2012.

POGGI, A.; ABITEBOUL, S. Xml data integration with identification. In: *Proceedings of the 10th International Conference on Database Programming Languages*. Berlin, Heidelberg: Springer-Verlag, 2005. (DBPL'05), p. 106–121. ISBN 3-540-30951-9, 978-3-540-30951-2. Disponível em: <http://dx.doi.org/10.1007/11601524_7>.

SACCOL, D. de B. et al. Managing Application Domains in P2P Systems. *IEEE International Conference on Information Reuse and Integration*, 2008.

SACCOL, D. de B.; PIVETA, E. K.; ANDRADE, T. de C. Mapping OWL ontologies to relational schemas . *The 12th IEEE International Conference on Information Reuse and Integration.*, p. 71 – 76, 2011.

SCHAFFNER, B. *Store XML data in a relational database*. TechRepublic, 2012. Disponível em: <http://www.techrepublic.com/article/store-xml-data-in-a-relational-database/5075453>. Acesso em: 16 de abril de 2012.

STUDER, R. et al. Knowledge Engineering: Survey and Future Directions. In: *XPS'99*. [S.l.: s.n.], 1999. p. 1–23.

SUPERPAGES. *RSS Weather Forecasts*. Superpages, 2012. Disponível em: <http://www.weatherzone.com.au/services/rss.jsp>. Acesso em: 21 de janeiro de 2012.

TAMINO. *Manage structured and unstructured data efficiently with webMethods Tamino*. Software AG, 2012. Disponível em: <http://www.softwareag.com/Corporate/products/wm/tamino/default.asp>. Acesso em: 16 de abril de 2012.

TEMPOAGORA. *Previsão do tempo em formato RSS*. TempoAgora, 2012. Disponível em: <http://www.tempoagora.com.br/>. Acesso em: 10 de janeiro de 2012.

VYSNIAUSKAS, E.; NEMURAITE, L. Transforming Ontology Representation from OWL to Relational Database. In: . [S.l.]: 124X Information Technology and Control, 2006. v. 35, n. 3.

W3C. *Extensible Markup Language (XML)*. The World Wide Web Consortium, 2012a. Disponível em: <http://www.w3.org/XML/>. Acesso em: 24 de abril de 2012.

W3C. *OWL Web Ontology Language Overview*. The World Wide Web Consortium, 2012b. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: 11 de fevereiro de 2012.

W3C. *W3C XML Query (XQuery)*. The World Wide Web Consortium, 2012c. Disponível em: <http://www.w3.org/XML/Query/>. Acesso em: 24 de abril de 2012.

WEATHERZONE. *RSS Weather Feeds*. Weatherzone, 2012. Disponível em: <http://www.superpages.com/about/rss_weather.html>. Acesso em: 21 de janeiro de 2012.

WORDNET. *WordNet - A lexical database for the English language*. Princeton, 2011. Disponível em: <http://wordnet.princeton.edu>. Acesso em: 27 de julho de 2011.

WU, J.; HUANG, S.-Y. An efficient mapping schema for storing and accessing xml data in relational databases. *International Journal of Web Information Systems*, v. 5, p. 327 – 347, 2009.

ZANELLA, M. K. *Representação Gráfica de Documentos XML*. 2011. Trabalho de Graduação.