

**Cláudio de Lima**

***Uma Ferramenta para Conversão de Esquemas  
Conceituais EER para Esquemas Lógicos XML***

**Florianópolis - SC**

**Julho/2008**

**Cláudio de Lima**

***Uma Ferramenta para Conversão de Esquemas  
Conceituais EER para Esquemas Lógicos XML***

Trabalho apresentado à Universidade  
Federal de Santa Catarina para a  
obtenção do grau de Bacharel em  
Ciências da Computação

Orientador:  
**Prof. Ronaldo dos Santos Mello, Dr**

CIÊNCIAS DA COMPUTAÇÃO  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
CENTRO TECNOLÓGICO  
UNIVERSIDADE FEDERAL DE SANTA CATARINA

**Florianópolis - SC**

**Julho/2008**

**Cláudio de Lima**

***Uma Ferramenta para Conversão de Esquemas  
Conceituais EER para Esquemas Lógicos XML***

Trabalho apresentado à Universidade Federal de Santa Catarina para a  
obtenção do grau de Bacharel em Ciências da Computação

---

**Orientador: Prof. Ronaldo dos Santos Mello, Dr**

---

**Co-orientadora: Rebeca Schroeder**

**Banca Examinadora:**

---

**Prof. Renato Fileto, Dr.**

---

**Prof. Mário Dantas, PhD.**

**Florianópolis - SC**

**Julho/2008**

*“...Eu não pedi pra nascer  
Eu não nasci pra perder  
Nem vou sobrar de vítima  
Das Circunstâncias...”*

**Lulu Santos**

## *Agradecimentos*

Dedico meus sinceros agradecimentos para:

- o professor doutor Ronaldo dos Santos Mello e a professora mestranda Rebeca Schroeder pela orientação e incentivo.

- a minha família pelo apoio e compreensão durante o desenvolvimento deste trabalho.

Em especial, ao meu querido avô e as minhas queridas, tia Suely e noiva Camila, pelo carinho e apoio ao longo da vida acadêmica.

## ***Resumo***

O crescente uso de documentos XML nas mais diversas aplicações requer o gerenciamento adequado destes documentos. Um dos tópicos de pesquisa em aberto com relação à gerência de dados XML é a definição de metodologias para projeto de Bancos de Dados (BD) XML, já que não há uma solução consolidada para a modelagem de um BD XML. Este trabalho apresenta uma ferramenta de apoio ao projeto lógico de BD XML, que presta suporte ao processo de conversão de esquemas conceituais EER (*Extended Entity-Relationship*) para esquemas lógicos XML. A ferramenta gera esquemas lógicos XML capazes de serem convertidos para qualquer linguagem de definição de esquema XML, como DTD e XML Schema.

PALAVRAS-CHAVE: XML, EER, Esquema, Conceitual, Lógico, Conversão

## *Abstract*

The extensive use of XML documents by several applications requires enhanced document management techniques. One of the research topics related to XML data management is the definition of methodologies for XML Database (DB) design, given that this technology does not have a consolidated standard to define an XML DB. This paper presents a tool to provide XML DB logical design support, enabling the conversion of an EER (Extended Entity Relationship) conceptual schema to an XML logical schema. The tool generates XML logical schemata that are able to be converted to any XML schema language, like DTD and XML Schema.

**KEYWORDS:** XML, EER, Schema, Conceptual, Logical, Conversion

## *Sumário*

### **LISTA DE FIGURAS**

### **LISTA DE TABELAS**

### **LISTA DE REDUÇÕES**

<b>1.INTRODUÇÃO .....</b>	<b>14</b>
1.1. Objetivos .....	15
1.1.1 Geral .....	15
1.1.2 Específicos .....	15
1.2. Motivação e Justificativa .....	16
1.3. Trabalhos Correlacionados .....	17
1.4. Organização do Trabalho .....	18
<b>2. FUNDAMENTAÇÃO TEÓRICA .....</b>	<b>20</b>
2.1. Modelo EER .....	20
2.1.1 Superclasses, Subclasses e Herança .....	21
2.1.2 Generalização e Especialização .....	22
2.1.3 Categorias ou Tipo União .....	23
2.2. XML .....	24
2.2.1 DTD ( <i>Document Type Definition</i> ) .....	25
2.2.2 XML Schema .....	26
2.3. Modelo Lógico XML .....	28
<b>3. ABORDAGEM DE CONVERSÃO .....</b>	<b>30</b>
3.1 Regras de Conversão .....	30
3.1.1 Entidades e Atributos .....	30
3.1.2 Generalização/Especialização e Categorias .....	31
3.1.3 Relacionamentos .....	34
3.2 O Processo de Conversão.....	37
<b>4. IMPLEMENTAÇÃO E RESULTADOS.....</b>	<b>41</b>
4.1 A Ferramenta EER2XML .....	41
4.1.1 Projeto .....	42



4.1.2 Funcionamento e Interface Gráfica.....	47
4.2 Testes e Resultados .....	49
4.3 Estudo de Caso .....	51
<b>5. CONCLUSÕES E TRABALHOS FUTUROS .....</b>	<b>54</b>
5.1 Conclusões .....	54
5.2 Trabalhos Futuros .....	54
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>56</b>
<b>ANEXO I - Artigo</b>	
<b>ANEXO II - Documento XML de entrada -Estudo de caso</b>	
<b>ANEXO III - Documento XML gerado da conversão do documento XML de entrada - Estudo de caso</b>	

## *Lista de Figuras*

FIGURA 2.1 - Exemplo de um esquema conceitual EER.....	21
FIGURA 2.2 - Exemplo Tipos União.....	23
FIGURA 2.3 - Exemplo de especificação DTD.....	26
FIGURA 2.4 - Exemplo de XML Schema.....	28
FIGURA 2.5 - Exemplo de um esquema lógico XML.....	29
FIGURA 3.1 - Regra 1 para generalização e categorias.....	31
FIGURA 3.2 - Regra 2 para generalização e categorias.....	32
FIGURA 3.3 - Regra 3 para generalização e categorias.....	33
FIGURA 3.4 - Regra 4 para generalização e categorias.....	34
FIGURA 3.5 - Regra 1 para relacionamentos.....	35
FIGURA 3.6 - Regra 2 para relacionamentos.....	36
FIGURA 3.7 - Regra 3 para relacionamentos.....	36
FIGURA 3.8 - Visão geral da ferramenta.....	37
FIGURA 3.9 - Hierarquia de múltiplos níveis.....	39
FIGURA 3.10 - Hierarquia gerada após ordenação top-down.....	39
FIGURA 3.11 - Hierarquia gerada após ordenação botton-up.....	39
FIGURA 3.12 - Diagrama algoritmo do processo de conversão.....	40
FIGURA 4.1 - Pacote de classes EERmodel.....	42
FIGURA 4.2 - Pacote de classes XMLmodel.....	43
FIGURA 4.3 - Pacote de classes ConversionRules.....	44
FIGURA 4.4 - Método convertGeneralizationsCategories.....	44

FIGURA 4.5 - Método convertRelationships.....	45
FIGURA 4.6 - Método defineRootElement.....	46
FIGURA 4.7 - Método generateXSD.....	47
FIGURA 4.8 - Ferramenta EER2XML.....	48
FIGURA 4.9 - Tela Conceptual Sechema Location.....	48
FIGURA 4.10 - Principais componentes e a relação entre eles.....	49
FIGURA 4.11 - Adição de Tipos União ao documento XML de entrada.....	51
FIGURA 4.12 - Adição de Disjunção ao documento XML de entrada.....	51
FIGURA 4.13 - Entrada de dados para a conversão.....	52

## *Lista de Tabelas*

TABELA 3.1 - Tabela de Prioridades para Aplicação das regras de Generalização e	
Categorias.....	34
TABELA 3.2 - Tabela de Prioridades para Aplicação das regras de Relacionamentos	
Associativos.....	37

## *Lista de Reduções*

SGBD	Sistema de Gerenciamento de Banco de Dados
EER	<i>Extended Entity-Relationship</i>
XML	<i>Extended Markup Language</i>
BD	Banco de Dados
DB	<i>Database</i>
API	<i>Application Programming Interface</i>
DTD	<i>Document Type Definition</i>
W3C	<i>World Wide Web Consortium</i>
XSL	<i>Extensible Stylesheet Language</i>
GUI	<i>Graphical User Interface</i>
ER	Entidade Relacionamento
EReX	ER estendido para XML
GBD	Grupo de Banco de Dados
GIS	<i>Geographic Information System</i>

# 1 INTRODUÇÃO

Hoje em dia a linguagem XML (*Extensible Markup Language*) tem emergido e se consolidado como um formato de representação de dados [XML, 2007]. Documentos XML são um importante mecanismo de troca de dados em muitos contextos de aplicação como comércio eletrônico e cadastro bibliográfico, e cada vez mais aplicações e pessoas disponibilizam e manipulam informações XML na *Web* [MELLO, 2003]. O uso cada vez mais intensivo de XML vem despertando o interesse de diversas áreas da ciência da computação, como linguagens de programação, engenharia de *software* e bancos de dados, todas elas preocupadas com a definição de mecanismos para o tratamento de dados neste formato.

Do ponto de vista da área de Banco de Dados (BD), um documento XML é uma coleção de dados, da mesma forma que um BD. Porém, um documento XML tem a vantagem de manter dados flexíveis, auto-descritivos e facilmente portáteis. A tecnologia XML assemelha-se bastante com a tecnologia de Sistemas Gerenciadores de BD (SGBDs), uma vez que possui um formato de armazenamento específico (documento), permite a definição de esquemas, possui linguagens de consulta e define interfaces (*APIs*) para o acesso a dados.

Entretanto, a tecnologia XML não é equivalente à tecnologia de SGBD, pois não existem soluções para todos os aspectos de gerenciamento de dados, como controle de integridade, gerenciamento de transações e visões. Além disso, o formato de um dado XML é altamente irregular e muitas vezes combina texto em linguagem natural com informações estruturadas. Assim sendo, BDs tradicionais, como os BDs relacionais, são incapazes de armazenar diretamente ocorrências de dados XML [MELLO, 2003].

Neste contexto, surgiram esforços para resolver o problema da modelagem de dados XML em diferentes níveis de abstração, visando, em muitos casos, definir o esquema de um banco de dados XML. Alguns destes esforços propõem novos formalismos para a representação de dados XML. No entanto, não há consenso sobre metodologias para o projeto de bancos de dados XML.

Alguns trabalhos propõem uma tradução direta do modelo conceitual tradicional para o modelo físico XML e outros propõem novos modelos e formalismos para modelagem de dados XML, sendo que, nenhum deles são validados através de uma ferramenta específica para conversão. Na literatura existem algumas ferramentas que propõem a conversão de BDs relacionais (através de SGBDs) em documentos XML específicos e outras (pagas) que transformam esquemas conceituais UML, também, em documentos XML específicos.

Como no projeto tradicional de bancos de dados, a modelagem em três níveis pode ser aplicada para bancos de dados XML: conceitual, lógica e física. Um modelo conceitual tradicional pode ser usado no primeiro nível, como o modelo Entidade-Relacionamento Estendido (*Extended Entity-Relationship* – EER). O modelo de dados XML é considerado no nível lógico, sendo que um esquema conceitual é convertido para um esquema em um modelo lógico XML. Um modelo lógico XML deve considerar construtores e restrições específicos do modelo de dados XML e este modelo deve ser capaz de abstrair diferentes modelos de implementação para XML, como *Document Type Definition* (DTD) e *XML Schema* [W3C, 2007].

Este trabalho propõe uma ferramenta que aplica um processo de conversão [SCHROEDER, 2008] que gera um esquema lógico XML a partir de um esquema conceitual EER. A estratégia de conversão utilizada é gerar estruturas hierárquicas para representar cada um dos tipos de relacionamento do EER. Esta estratégia permite a geração de esquemas lógicos XML que podem ser convertidos para qualquer modelo físico XML.

## **1.1 OBJETIVOS**

### **1.1.1 GERAL**

Desenvolver uma ferramenta de apoio ao Projeto Lógico de Banco de Dados XML, que presta suporte ao processo de conversão de um esquema conceitual EER para um esquema lógico XML.

### **1.1.2 ESPECÍFICOS**

A ferramenta recebe como entrada um documento XML que representa um esquema conceitual EER. Após o processo de conversão, a ferramenta disponibiliza um documento XML contendo o esquema lógico XML gerado como saída, o qual pode ser visualizado no formato de um XML *Schema*. O processo de conversão é automático, permitindo que o usuário especialista forneça apenas o esquema conceitual EER de entrada, em forma de um documento XML, e escolha a entidade principal, ou seja, a entidade na qual o processo inicia a conversão. O processo de conversão está baseado em regras que possuem pré-condições e prioridades de aplicação, e que busca obter o esquema lógico XML mais compacto e conexo.

## 1.2 MOTIVAÇÃO E JUSTIFICATIVA

XML é uma linguagem de marcação de padrão aberto, criado por um grupo de empresas e acadêmicos chamado *World Wide Web Consortium (W3C)*, e adequada ao armazenamento de dados estruturados e semi-estruturados [XML, 2007]. Ela permite que informações de diferentes sistemas sejam intercaladas independentemente da plataforma utilizada pelos sistemas, através de arquivos que possuem o formato XML (documentos XML).

Devido ao crescente uso de dados XML por pessoas e aplicações, existe a necessidade de um gerenciamento destes documentos XML por Sistemas de Gerência de Banco de Dados (SGBDs). Atualmente, existem SGBDs para o gerenciamento de dados XML, porém, estes não são considerados robustos devido a algumas limitações, como armazenamento eficiente, segurança, integridade, acessos simultâneos e metodologias de projeto de bancos de dados XML [RPBOURRET, 2007].

Como mencionado, um dos tópicos de pesquisa em aberto com relação à gerência de dados XML é a definição de metodologias para projeto de bancos de dados XML. No que diz respeito a BDs relacionais, o projeto de um BD envolve três etapas: (i) a criação de um esquema conceitual, que é uma descrição do BD independente da implementação em um SGBD; (ii) a modelagem lógica, que representa a estrutura de um BD de acordo com um modelo de dados (por exemplo, o modelo de dados



relacional) e; (iii) a modelagem física, que descreve a implementação do esquema lógico em um SGBD específico.

A tecnologia XML não possui um padrão de projeto consolidado para a criação de um BD XML, que é uma categoria recente de BD disponível comercialmente. Em função disto, uma metodologia de projeto de BD XML está sendo desenvolvida como dissertação de mestrado no âmbito do Grupo de Banco de Dados (GBD) da UFSC [SCHROEDER, 2008]. A idéia deste Trabalho de Conclusão de Curso (TCC) é contribuir com esta dissertação através da implementação de uma ferramenta que preste suporte ao projeto lógico de um BD XML.

### 1.3 TRABALHOS RELACIONADOS

Vários trabalhos têm sido propostos para modelagem de dados XML em diferentes níveis de abstração. Alguns propõem uma tradução direta do modelo conceitual tradicional para o modelo físico XML. Outros propõem novos modelos e formalismos para modelagem de dados XML. Podemos classificar os trabalhos relacionados de acordo com o modelo usado em suas conversões em três categorias: (i) ER para linguagem de esquema para XML. (ii) ER para um modelo lógico XML; (iii) novo modelo conceitual para dados XML [SCHROEDER, 2008].

Os trabalhos inseridos na primeira categoria propõem um algoritmo para gerar um XML *Schema* a partir de um esquema ER. Outros apresentam um processo de conversão para mapear esquemas EER para DTD.

Os trabalhos da segunda categoria apresentam a conversão do modelo ER para o modelo lógico XML, ou seja, um esquema lógico hierárquico. Este modelo lógico depende apenas da estrutura baseada em árvore e não representa construtores e restrições específicas do modelo de dados XML. O trabalho relacionado mais parecido com o proposto neste trabalho descreve um algoritmo para traduzir um esquema EReX (ER estendido para XML) para um esquema lógico XML definido por uma gramática.

A terceira e última categoria compreende trabalhos que propõem novos modelos de dados para XML, como uma modelagem XML baseada em redes semânticas e um novo modelo hierárquico para representação de dados semi-estruturados.

As abordagens da primeira categoria são fortemente vinculadas a um esquema específico para XML, já o processo de conversão utilizado neste trabalho propõe a conversão para um nível lógico no qual permite uma maior abstração do nível físico. A segunda categoria baseia-se em uma nova extensão para o modelo ER. No entanto, conclui-se que o banco de dados XML deva ser baseado em um modelo conceitual consolidado e mais abrangente, como o modelo considerado neste trabalho, o modelo EER. As propostas da terceira categoria são uma mistura de modelos lógico XML com conceitos semânticos de um modelo conceitual. A abordagem proposta deste trabalho abrange dois modelos independentes: EER e um adequado modelo lógico XML.

O modelo EER foi escolhido como modelo conceitual porque é um modelo simples e poderoso para prover uma abstração de alto nível para a informação do domínio de aplicação. Além disso, o EER é o modelo mais utilizado para modelagem conceitual de banco de dados. Consideramos no processo de conversão todos os construtores do EER, inclusive o conceito de categoria ou tipo união.

Na literatura existem algumas ferramentas que propõem a conversão de BDs relacionais, geralmente a partir de SGBDs, em documentos XML [RPBOURRET, 2007]. Existem ainda algumas ferramentas pagas que possuem módulos para transformar esquemas conceituais UML em documentos XML *Schema* [SPARX, 2008], [RATIONAL, 2008] e [MTF, 2008]. No entanto, não foi encontrada uma ferramenta similar a apresentada neste trabalho, pois nenhuma outra trata da conversão de um esquema conceitual (EER) para um esquema lógico XML que considera construtores e restrições específicos comuns aos mais relevantes modelos de implementação para XML.

## **1.4 ORGANIZAÇÃO DO TRABALHO**

Este trabalho está dividido em mais quatro capítulos. O capítulo 2 apresenta a fundamentação teórica, que dá suporte ao entendimento do processo de conversão utilizado neste trabalho. São apresentados os modelos de dados mais relevantes ao projeto e a tecnologia XML. O capítulo 3 apresenta uma abordagem de conversão do construtor EER para o modelo lógico XML, apresentando as regras de conversão e

definindo o processo de conversão em que estas regras são utilizadas. O capítulo 4 apresenta basicamente o projeto da ferramenta na sua versão atual, os resultados já obtidos e ainda um estudo de caso de utilização da ferramenta. No capítulo 5 são apresentadas as conclusões finais e as atividades futuras.

## **2 FUNDAMENTAÇÃO TEÓRICA**

Neste capítulo são descritos alguns estudos que formam o embasamento teórico para o trabalho.

### **2.1 MODELO EER**

Os conceitos de modelagem ER são suficientes para a representação de muitos esquemas para as aplicações de bancos de dados “tradicionais”, que incluem, principalmente, as aplicações de processamento de dados na indústria e no comércio. Desde o início dos anos 80, entretanto, os projetistas de banco de dados têm tentado projetar esquemas de banco de dados mais exatos, que reflitam as propriedades e as restrições dos dados mais precisamente. Isso foi particularmente importante nas, relativamente, recentes aplicações da tecnologia de banco de dados, como os bancos de dados para o projeto de engenharia e manufatura (CAD/CAM), telecomunicações, sistemas de software complexos e Sistemas de Informações Geográficas (GIS), entre outras. Muitos desses conceitos também foram desenvolvidos independentemente, em áreas relacionadas da ciência da computação, como nas áreas de modelagem de objeto da engenharia de software e de representação de conhecimento da inteligência artificial. [ELMASRI, 2005].

O modelo EER engloba todos os conceitos de modelagem do modelo ER, além disso, inclui os conceitos relacionados de especialização e generalização, atributos complexos e ainda o conceito de categoria ou tipo união. Associado a esses conceitos está o importante mecanismo de herança de atributo e relacionamento [ELMASRI, 2005]. A figura 2.1 apresenta um exemplo de um esquema conceitual no modelo EER.

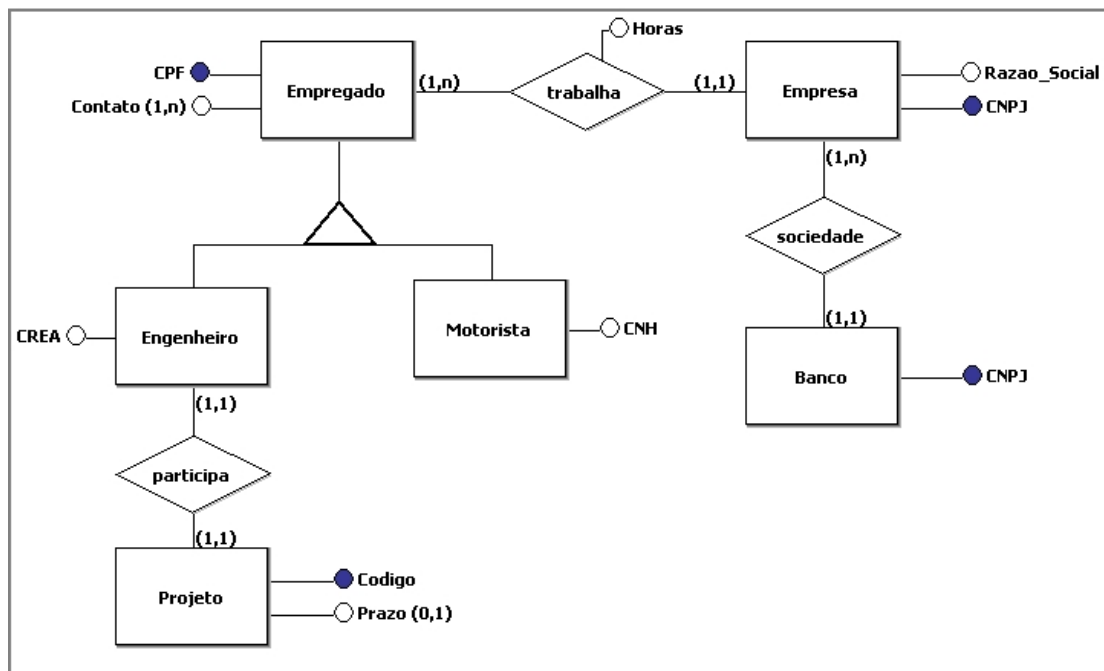


Figura 2.1: Exemplo de um esquema conceitual EER.

### 2.1.1 SUPERCLASSES, SUBCLASSES E HERANÇA

Em muitos casos, um tipo entidade tem numerosos subgrupos dessas entidades, que são significativos e que necessitam ser representados explicitamente, em virtude de sua importância para a aplicação de banco de dados. Por exemplo, na figura 2.1 as entidades que são membros do tipo entidade Empregado podem ser posteriormente agrupadas em Engenheiro e Motorista. O conjunto de entidades de cada desses grupos é um subconjunto das entidades que pertencem ao conjunto de entidades Empregado, significando que toda entidade que é um membro de um desses subgrupos, é também um empregado. Chamamos cada um desses subgrupos uma *subclasse* do tipo Empregado e o tipo entidade Empregado é conhecido como *superclasse* para cada uma dessas subclasses.

Chamamos o relacionamento entre uma superclasse e qualquer uma de suas subclasses relacionamento superclasse/subclasse (ou classe/subclasse). No exemplo anterior, Empregado /Motorista é um relacionamento classe/subclasse.

Uma entidade pode ser incluída opcionalmente com membro de várias subclasses. Por exemplo, um engenheiro que também é um motorista, pertence às duas subclasses, Engenheiro e Motorista do tipo entidade Empregado. Entretanto, não é

necessário que toda entidade de uma superclasse seja também membro de alguma subclasse.

Um importante conceito associado a subclasses é o de tipo de herança. Dizemos que uma entidade, que é um membro de uma subclasse, herda todos os atributos da entidade como membro da superclasse. A entidade também herda todos os relacionamentos do qual a superclasse participa.

### **2.1.2 GENERALIZAÇÃO E ESPECIALIZAÇÃO**

A especialização é o processo de definir um conjunto de subclasses de um tipo entidade; este tipo entidade é chamado superclasse da especialização. O conjunto de subclasses que forma uma especialização é definido como base em algumas características de distinção das entidades da superclasse. Podemos ter diversas especializações para o mesmo tipo entidade, baseadas nas diferentes características que as distinguem. Por exemplo, na figura 2.1 uma especialização do tipo entidade Empregado produz o conjunto de subclasses { Engenheiro, Motorista } mas poderíamos criar outro conjunto de subclasses {EmpregadoDiurno, EmpregadoNoturno}. A primeira especialização distinguiria os empregados com base na função e a segunda distinguiria com base no período de trabalho.

Há duas razões para incluir relacionamentos classe/subclasse e especializações em um modelo de dados. A primeira é que certos atributos podem ser usados em algumas, mas não em todas as entidades da superclasse. Uma subclasse é definida de forma a agrupar as entidades para as quais esses atributos se aplicam. Os membros da subclasse podem, ainda, compartilhar a maioria de seus atributos com os membros da superclasse. A segunda razão para usar as subclasses é que apenas as entidades que são membros de alguma subclasse podem participar de algum tipo relacionamento [ELMASRI, 2005].

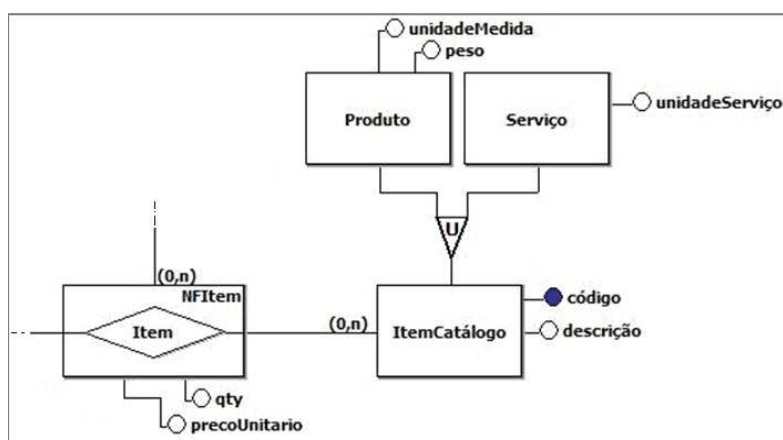
Em resumo, o processo de especialização nos permite: (i) definir um conjunto de subclasses de um tipo entidade; (ii) estabelecer atributos específicos adicionais para cada subclasse; (iii) estabelecer tipos relacionamento adicionais específicos entre cada subclasse e outros tipos entidade, ou outras subclasses.

Podemos pensar em um processo contrário de abstração, no qual suprimos as diferenças entre os diversos tipos entidade, identificamos suas características comuns e os generalizamos em uma única superclasse, da qual os tipos entidade originais são subclasses especiais.

Duas restrições importantes podem ser aplicadas a uma especialização (e generalização). A primeira é chamada restrição de integralidade, que pode ser *total* ou *parcial*. Uma restrição de especialização total especifica que toda entidade na superclasse deve ser um membro de pelo menos uma das subclasses na especialização. Já uma restrição de especialização parcial admite que uma entidade não pertença a nenhuma das subclasses. A segunda é a restrição de *disjunção*, especificando que as subclasses da especialização devem ou não ser mutuamente exclusivas. No primeiro caso, significa que uma entidade pode ser membro de no máximo uma das subclasses da especialização [ELMASRI, 2005].

### 2.1.3 CATEGORIAS OU TIPOS UNIÃO

Há a possibilidade de surgir necessidade de modelar um único relacionamento superclasse/subclasse com mais de uma superclasse, na qual a superclasse represente os diferentes tipos entidade. Nesse caso a superclasse representará uma coleção de objetos que é um subconjunto da União de diferentes tipos entidade; chamamos essa superclasse tipo união ou categoria [ELMASRI, 2005].



**Figura 2.2:** Exemplo de Tipos União.

Por exemplo, na figura 2.2 temos dois tipos de entidade: Serviço e Produto. Em um banco de dados para o registro de itens de um catálogo, um item de catálogo pode ser um serviço ou um produto. Precisamos criar uma classe (coleção de entidades) que inclua entidades de todos os dois tipos, para desempenharem o papel de item de catálogo. Uma categoria ItemCatálogo, que é a união dos dois conjuntos de entidade Serviço e Produto, é criada com este propósito.

O atributo de herança trabalha mais seletivamente quando aplicado às categorias. Cada entidade ItemCatálogo (exemplo anterior) herda os atributos do Serviço ou do Produto, dependendo da superclasse à qual a entidade pertence.

Uma categoria pode ser total ou parcial. Uma categoria total controla a união de todas as entidades em suas superclasses, ao passo que uma categoria parcial pode controlar um subconjunto da união.

## 2.2 XML

A XML é uma linguagem de marcação que utiliza *tags* para delimitação das informações. Ela foi desenvolvida pelo *World Wide Web Consortium* (W3C), um consórcio formado por acadêmicos e empresários com o objetivo de troca de conhecimento e definição padrões para as linguagens utilizadas na *Web* [W3C, 2007].

A XML é bastante utilizada na representação e transferência de dados semi-estruturados na *Web*. Dados descritos em XML estão contidos em um documento texto chamado documento XML. De natureza estruturada, porém não formatada, um documento XML possui inúmeras possibilidades de utilização, como por exemplo, o intercâmbio de dados presentes em BDs diferentes. A simplicidade da XML faz com que ela seja acessível a todos os níveis de programadores, tornando-a cada vez mais difundida e utilizada [BARCAROLI, 2005].

Cada dado contido num documento XML recebe o nome de *elemento* e é constituído de uma *tag* inicial, de um conteúdo e de uma *tag* final, além de alguns atributos. XML é um dado semi-estruturado cujas principais características são: (i) representação estrutural heterogênea, ou seja, não é completamente não-estruturado, porém não pode ser considerado como um dado totalmente tipado; (ii) dados auto-



descritivos, ou seja, seu esquema de representação está presente explicitamente. Pela análise destes dados, pode-se encontrar e extrair sua estrutura.

Um documento XML, para ser considerado “Bem Formado”, deve obedecer à especificação da sintaxe XML. Tal documento contém um ou mais elementos delimitados pelas *tags*, aninhados adequadamente um dentro do outro, formando uma estrutura hierárquica na forma de árvore.

Um documento XML pode ter a organização de seus elementos definida basicamente de duas formas: através de uma DTD ou de um XML *Schema*. Estes definem quais são os elementos válidos em um documento, especificam restrições de número e ordem de ocorrência desses elementos, além de outras restrições quanto à estrutura e conteúdo dos mesmos [W3C, 2007].

### **2.2.1 DTD (*Document Type Definition*)**

A DTD é um artefato opcional para se especificar formalmente um conjunto de regras para definição da estrutura de documentos XML. Estas regras definem quais elementos podem ser usados e definem onde eles podem ser aplicados em relação a cada outro elemento. Além disso, especifica a hierarquia e a granularidade do documento [BRADLEY, 2003].

Uma DTD pode ser composta por um número arbitrário de declarações, podendo cada declaração ser do tipo *ELEMENT* (definição de elementos), *ATTLIST* (definição de atributos de elementos), *ENTITY* (declaração de entidades) e *NOTATION* (definição de tipos de dados). Essas declarações são agrupadas juntas dentro de uma declaração de tipo de documento [BRADLEY, 2003]. Um exemplo de especificação DTD pode ser visto na Figura 2.3.

```

<?xml version="1.0"?>
<!DOCTYPE loja [
  <!ELEMENT loja (entrega+, livro+) >
  <!ELEMENT entrega (duracao) >
  <!ATTLIST entrega entregaID ID #REQUIRED>
  <!ELEMENT livro (#PCDATA) >
  <!ATTLIST livro tipoEntrega IDREF #IMPLIED>
  <!ELEMENT duracao (#PCDATA) >
]
<loja>
  <entrega entregaID = "s1">
    <duracao>2 a 4 dias</duracao>
  </entrega>

  <entrega entregaID = "s2">
    <duracao>1 dia</duracao>
  </entrega>

  <livro tipoEntrega = "s2">
    Java: Como programar
  </livro>

  <livro tipoEntrega = "s2">
    C: Como programar
  </livro>

  <livro tipoEntrega = "s1">
    C++: Como programar
  </livro>
</loja>

```

**Figura 2.3:** Exemplo de especificação DTD.

No exemplo apresentado acima, pela figura 2.3, temos a definição do elemento ‘loja’ como o elemento raiz (*root*) do documento, definido pela declaração *DOCTYPE*. A linha “*ELEMENT loja (entrega+, livro+)*” define que o elemento ‘loja’ contém dois sub-elementos (‘entrega’ e ‘livro’) e o símbolo ‘+’ define a cardinalidade [1..N]. Já a linha “*ATTLIST entrega entregaID ID #REQUIRED*”, define que o elemento ‘entrega’ possui o atributo identificador ‘entregaID’ e que este atributo é requerido (obrigatório).

## 2.2.2 XML SCHEMA

O XML *Schema* (recomendado desde 2001 pela W3C) consiste num conjunto de regras com o objetivo de padronizar a estrutura de um documento XML bem formado, porém com algumas facilidades a mais como, por exemplo, o suporte a tipos de dados e extensão de tipos [BRADLEY, 2003]. Além das facilidades citadas anteriormente, o XML *Schema* conta com recursos adicionais voltados a restrições de valores de dados como, por exemplo, a definição de qual deve ser a estrutura e conteúdo dos dados, além do número e ordem de sua ocorrência.

O XML *Schema* possui em sua composição um elemento chamado “*schema*”, que é o elemento raiz. Ele pode conter um atributo opcional chamado “*version*”, usado para a identificação da versão atual do esquema [BRADLEY, 2003].

Outro elemento que forma um XML *Schema* recebe o nome de “*annotation*”. Ele é utilizado tanto para a inserção de anotações e/ou comentários julgados necessários, como para a inclusão de informações de processamento automático.

Outras duas importantes declarações da XML *Schema* são chamadas de *element* e *attribute*, usadas para a definição de elementos e seus atributos, respectivamente. Tanto os atributos como os elementos devem possuir a definição de seus nomes e tipos.

Conforme mencionado anteriormente, o XML *Schema* permite definir tipos de dados, podendo estes representar elementos simples ou complexos. Um elemento é dito simples quando contém apenas texto (ou seja, *string*, *boolean*, etc...) ou qualquer outro tipo personalizado definido pelo próprio usuário. O tipo simples não pode conter outros elementos (elementos filhos) nem atributos no seu conteúdo. Na Figura 2.3, os elementos identificados como “idade” e “nascimento” são exemplos de elementos simples.

Na figura 2.4 temos também um exemplo de declaração de um elemento complexo, sendo o elemento “pai” de nome “nome” e os elementos filhos correspondendo a “primeiroNome” e “sobrenome”. Estes filhos, pela definição do construtor de ordem “*sequence*”, devem aparecer na ordem definida, ou seja, primeiro deve vir o “primeiroNome” e depois o “sobrenome”. Além do construtor “*sequence*”, pode ser utilizado o construtor “*choice*” que especifica que somente um dos elementos filhos (definidos no grupo) pode aparecer em cada instância. Além destes, ainda podemos citar o construtor “*all*” que define que os elementos do grupo podem aparecer uma vez desde que juntos (em qualquer ordem) ou então nenhuma vez.

```

<?xml version="1.0"?>
<xs: schema>
  <xs: annotation>
    <xs: documentation> Exemplo
  </xs: documentation>
</xs: annotation>
<xs: element name = "idade" type="xs:integer"/>
<xs: element name = "nascimento" type="xs:date"/>
<xs: element name=" nome ">
  <xs:complexType>
    <xs:sequence>
      <xs:element name "primeiroNome" type="xs:string"/>
      <xs:element name "sobrenome" type="xs:string">
    </xs:sequence>
  </xs:complexType>
</xs: element name=" nome ">
</xs: schema>

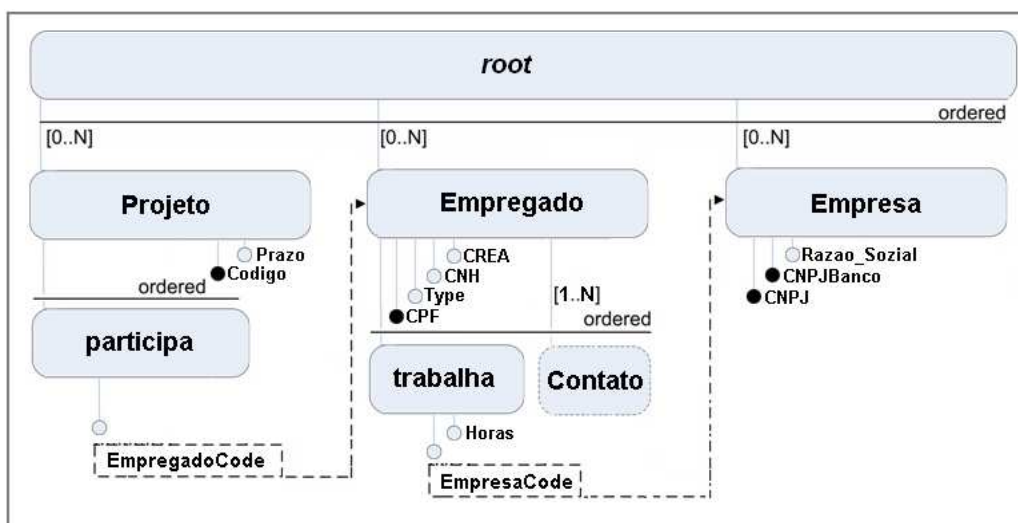
```

Figura 2.4: Exemplo de XML Schema.

## 2.3 MODELO LÓGICO XML

O modelo lógico XML considerado representa composições hierárquicas de dados e se baseia nos conceitos de elemento, atributo e relacionamento. Este modelo lógico está sendo desenvolvido por uma dissertação de mestrado do Grupo de Banco de Dados da UFSC [SCHROEDER, 2008]. Este modelo não é um novo formalismo para a modelagem lógica XML, mas uma adaptação de modelos hierárquicos para modelagem de dados semi-estruturados, como [DOBBIE, 2001] e [CHANG, 2002]. Para garantir a abstração do modelo lógico, foram considerados como conceitos deste modelo somente construtores do modelo XML que são comuns às mais relevantes linguagens de definição de esquemas XML.

A Figura 2.5 mostra um exemplo de um esquema no modelo lógico XML. Atributos são representados por um círculo e podem ser do tipo *normal*, *identificador* e *de referência*. Um atributo normal representa uma propriedade de um elemento XML. Atributos identificadores são aqueles que fazem parte do conjunto de atributos que identificam unicamente uma instância de elemento. Um atributo de referência é um atributo que faz referência a outro elemento. Os atributos *prazo*, *código* e *empregadocode* são, respectivamente, exemplos de atributo normal, identificador e de referência.



**Figura 2.5:** Exemplo de um esquema lógico XML.

Um elemento pode ser *simples* ou *complexo*. Um elemento simples representa informações que são definidas por um tipo de dado simples que não possui atributos, como o elemento *Contato* da Figura 2.5. Um elemento complexo pode ser formado por atributos e/ou elementos componentes. Os elementos componentes são organizados por um construtor de ordem definido para o elemento complexo que os contém. O construtor de ordem de um elemento complexo pode ser *ordered* ou *exclusive*. Um construtor *ordered* estabelece os elementos componentes de forma ordenada. Já um construtor *exclusive* define uma disjunção entre os elementos componentes. O elemento *Projeto* é um exemplo de elemento complexo com construtor de ordem *ordered*.

Os relacionamentos podem ser hierárquicos ou de referência. Relacionamentos hierárquicos são representados por linhas contínuas entre o conceito origem e o conceito destino. Um relacionamento define as ocorrências mínima e máxima do conceito destino no conceito origem. Por exemplo, o elemento *Empregado* pode ter um ou muitos *Contato*. Quando não especificado no esquema, as ocorrências mínimas e máximas correspondem a 1 ([1..1]). Relacionamentos de referência são representados por linhas pontilhadas entre um atributo de referência (ou um conjunto de atributos de referência) e um elemento complexo. O relacionamento entre o atributo *empresacode* e o elemento *Empresa* é um exemplo deste tipo de relacionamento.

### **3 ABORDAGEM DE CONVERSÃO**

Este capítulo apresenta uma descrição sucinta das regras de conversão, bem como do processo de conversão de esquemas EER para esquemas lógicos XML.

#### **3.1 REGRAS DE CONVERSÃO**

As regras de conversão são propostas para conversão do construtor do esquema EER para a representativa composição lógica XML. Estas regras são propostas em três grupos: (1) Regras para Entidade e Atributos; (2) Regras para Generalização/Especialização e Categorias; (3) Regras para Relacionamento. Durante o processo de conversão uma destas regras é escolhida automaticamente mediante critério de escolha. A seguir é descrito sucintamente cada um destes três grupos. Estas regras são baseadas nas regras de conversão usadas no projeto lógico de BDs convencionais, definidas em [BATINI, 92].

##### **3.1.1. ENTIDADES E ATRIBUTOS**

As regras de conversão de entidades e atributos são relativamente simples: uma entidade torna-se um elemento; um atributo monovalorado obrigatório torna-se atributo do seu respectivo elemento e se for opcional, atribui-se a cardinalidade [0..1]; um atributo multivalorado gera um sub-elemento (simples) para esse atributo com construtor de ordem “*ordered*” e cardinalidade [0..n] para o relacionamento; para atributo composto monovalorado, incluir “cada atributo” como atributo do seu respectivo elemento. Caso ele seja multivalorado, define-se um sub-elemento para este atributo e inclui-se neste, se também for composto, “cada atributo”; para atributo identificador, adicioná-lo como atributo identificador do seu respectivo elemento e criar uma restrição de identificação (para uso posterior).

### 3.1.2. GENERALIZAÇÃO/ESPECIALIZAÇÃO E CATEGORIAS

Quatro regras para conversão de generalização/especialização e categorias foram definidas, além de uma tabela de prioridades para a aplicação delas. Elas são apresentadas a seguir.

#### Regra 1:

Criar elementos para superclasse e subclasse (se já não foram criados). Se for disjunta, tornar o construtor de ordem do elemento da superclasse “*exclusive*”, senão definir como “*ordered*”. Se for parcial, a cardinalidade dos elementos das subclasses deve ser [0..1]; senão deve ser [1..1]. A Figura 3.1 ilustra a aplicação desta regra de forma simplificada.

Se a superclasse já tiver sido convertida e o elemento que a representa é exclusivo, criar um sub-elemento agrupador (com construtor de ordem “*exclusive*”) para a superclasse e colocar como sub-elementos do agrupador todos os sub-elementos que já existiam. Então proceda a conversão da generalização em questão: se a generalização em questão é exclusiva, então crie um outro sub-elemento agrupador (com construtor de ordem “*exclusive*”) para a superclasse e coloque todos os elementos para representar as subclasses como filhos desse agrupador.

**Pré-condições:** Pelo menos uma subclasse não foi processada por outra generalização/categoria como filha de outro elemento. Para as demais que já foram processadas como filha de outro elemento, deve-se criar atributos de referência para estes elementos.

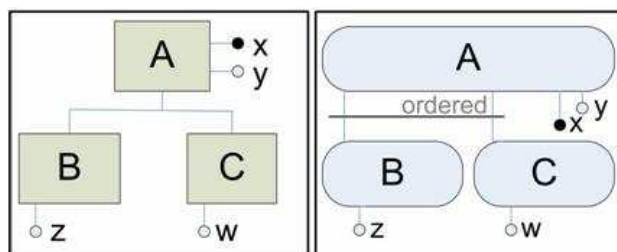


Figura 3.1: Regra 1 para generalização e categorias.

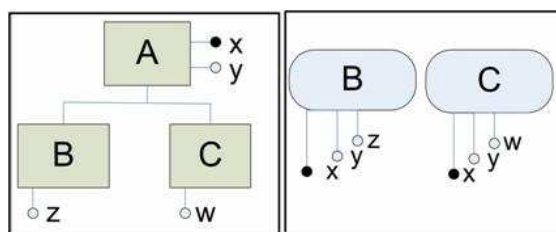
#### Regra 2:

Criar elementos somente para as subclasses. Em cada elemento da subclasse adicionar os atributos da superclasse. A Figura 3.2 ilustra a aplicação desta regra de forma simplificada.

**Pré-condições:** (i) As subclasses não foram processadas por outra generalização/categoria como filhas de outros elementos; (ii) O número de atributos da superclasse ser menor ou igual ao número máximo de atributos que uma superclasse/conjunto de subclasses pode ter.; (iii) O número de relacionamentos associativos com a superclasse deve ser menor ou igual ao número máximo de relacionamentos associativos que uma superclasse pode ter.

O número máximo de atributos mencionado nas pré-condições, atualmente, é pré-definido internamente na ferramenta e busca delimitar um valor máximo para que não haja um número excessivo de atributos no elemento da superclasse. Posteriormente, este número máximo de atributos deverá ser parametrizado pelo usuário especialista.

**Pós-condições:** (i) Um mesmo valor para o atributo identificador da superclasse não deve ser assumido por ambas as subclasses (somente se ocorre disjunção, para generalização); (ii) Outros relacionamentos com a superclasse devem ser mapeados para cada uma das subclasses; (iii) Se houver disjunção na generalização, os atributos que eram identificadores nas subclasses (se existirem) tornam-se atributos simples e os atributos que eram identificadores na superclasse tornam-se os atributos identificadores dos elementos das subclasses; senão, se existiam atributos identificadores para as subclasses eles se tornam os identificadores dos elementos criados e os identificadores da superclasse tornam-se atributos simples (ocorre em categorias). Caso contrário ocorre o mesmo processo com disjunção.



**Figura 3.2:** Regra 2 para generalização e categorias.

### Regra 3:

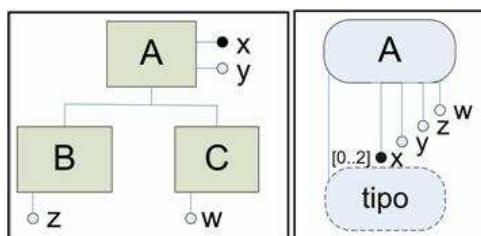


Criar um elemento apenas para a superclasse. Adicionar neste elemento os atributos das subclasses como opcional. Adicionar um elemento simples *TYPE* com uma cardinalidade que represente as restrições ([0..1] se for parcial e disjunta; [0..n] se for parcial e compartilhada; [1..1] se for total e disjunta; [1..n] se for total e compartilhada). A Figura 3.3 ilustra a aplicação desta regra de forma simplificada.

**Pré-condições:** (i) As subclasses não foram processadas por outras generalizações/categorias como filhas de outros elementos; (ii) Subclasses não podem ter outros relacionamentos (associativos, generalização ou categorias); (iii) A somatória de atributos de subclasse deve ser menor ou igual ao número máximo de atributos que uma superclasse/conjunto de subclasses pode ter.

O número máximo de atributos mencionado nas pré-condições, atualmente, é pré-definido internamente na ferramenta e busca delimitar um valor máximo para que não haja um número excessivo de atributos no elemento da superclasse. Posteriormente, este número máximo de atributos deverá ser parametrizado pelo usuário especialista.

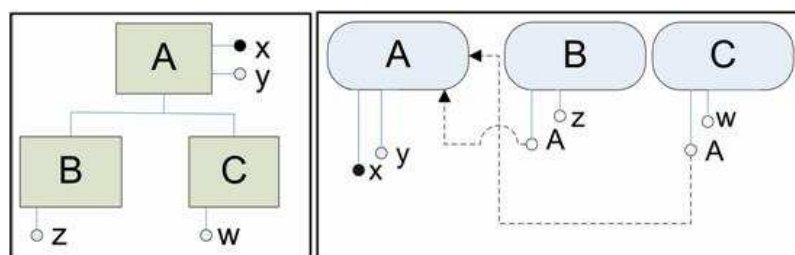
**Pós-condições:** A aplicação deverá tratar e garantir a disjunção (se existir) entre atributos das subclasses (com base no atributo *TYPE*); Os atributos que eram identificadores nas subclasses (se existirem), tornam-se atributos simples.



**Figura 3.3:** Regra 3 para generalização e categorias.

**Regra 4:**

Criar elementos para superclasse e subclasses. No elemento da superclasse criar atributos de referência para as subclasses. Criar atributo de referência para a superclasse nos elementos das subclasses. A Figura 3.4 ilustra a aplicação desta regra de forma simplificada.



**Figura 3.4:** Regra 4 para generalização e categorias.

A tabela 3.1 apresenta as prioridades para aplicação das regras recém-descritas para generalização/especialização e categorias. As prioridades são dadas às regras que gerarem o menor esquema possível, considerando as pré-condições de cada regra [SCHROEDER, 2008].

**Tabela 3.1 -** Prioridades para Aplicação das Regras de Generalização e Categorias.

<b>Tipo Generalização</b>	<b>Regra 1</b>	<b>Regra2</b>	<b>Regra 3</b>	<b>Regra 4</b>
Total e Compartilhada	-	-	1º	2º
Total e Disjunta	3º	2º	1º	4º
Parcial e Compartilhada	2º	-	1º	3º
Parcial e Disjunta	2º	-	1º	3º
<b>Tipo Categoria</b>				
Total	3º	2º	1º	4º
Parcial	-	1º	-	2º

Ao se tratar da conversão de uma categoria, deve-se lembrar que os papéis das superclasses e subclasse devem ser invertidos para esta regra, isto é, o elemento que representa a subclasse deve ser o elemento pai dos elementos que representam as superclasses.

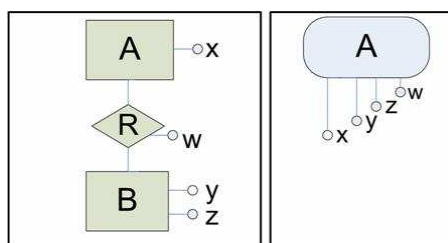
### 3.1.3 RELACIONAMENTOS

A seguir são apresentadas as três regras de conversão para relacionamentos associativos, seguidas da tabela de prioridades para aplicação destas regras.

#### **Regra 1:**

É criado um elemento único para representar E1 (entidade 1) e E2 (entidade 2). A Figura 3.5 ilustra a aplicação desta regra de forma simplificada.

**Pré-condições:** (i) Ser relacionamento binário; (ii) E2 não foi processada por outros relacionamentos associativos como um sub-elemento de outro elemento, ou por fusão com outro elemento.



**Figura 3.5:** Regra 1 para relacionamentos.

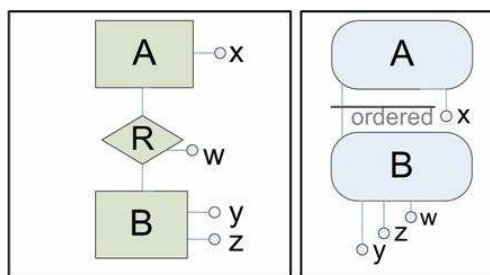
### Regra 2:

São criados elementos para representar cada entidade envolvida no relacionamento. E1 torna-se o elemento pai de E2. A cardinalidade de E2 em E1 deve ser definida conforme a sua cardinalidade definida para esse relacionamento no esquema conceitual. Os tributos do relacionamento ficam em E2. A Figura 3.6 ilustra a aplicação desta regra de forma simplificada.

O primeiro relacionamento processado no processo de conversão é aquele que contém a entidade de partida do processo, e neste caso, E1 será a entidade de partida. Posteriormente, são percorridos caminhos através do primeiro relacionamento e serão definidos como E1 as entidades envolvidas nos relacionamentos seguintes da seqüência desses caminhos, ou seja, se E2 do primeiro relacionamento processado possuir outros relacionamentos, E2 será definida como E1 dos seus outros relacionamentos, e assim por diante. Caso E2 do primeiro relacionamento processado não tenha outros relacionamentos, a escolha torna-se aleatória.

Se o elemento que representa E1 já havia sido criado com o construtor de ordem definido como “*exclusive*”, criar um sub-elemento agrupador (com construtor de ordem “*exclusive*”) para E1 e colocar todos os sub-elementos que existiam como filhos do agrupador.

**Pré-condição:** E2 não foi processada por outros relacionamentos associativos como um sub-elemento de outro elemento, ou por fusão com outro elemento.



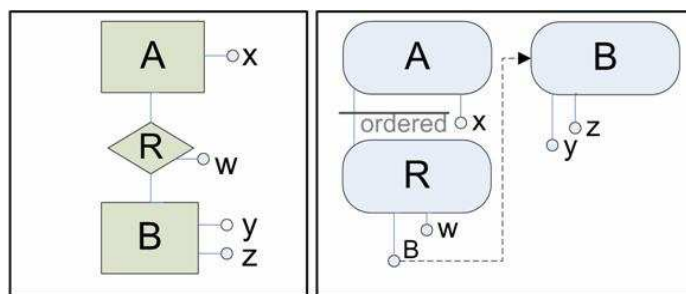
**Figura 3.6:** Regra 2 para relacionamentos.

**Regra 3:**

É criado um elemento para representar E1 e outro elemento para representar o relacionamento juntamente com seus atributos. O elemento do relacionamento se torna filho de E1. São criados elementos à parte para as demais entidades participantes do relacionamento. A associação com esses elementos (demais entidades participante) é estabelecida por meio de atributos de referência no elemento que representa o relacionamento. A Figura 3.7 ilustra a aplicação desta regra de forma simplificada.

O primeiro relacionamento processado no processo de conversão é aquele que contém a entidade de partida do processo, e neste caso, E1 será a entidade de partida. Posteriormente, são percorridos caminhos através do primeiro relacionamento e serão definidos como E1 as entidades envolvidas nos relacionamentos seguintes da seqüência desses caminhos, ou seja, se E2 do primeiro relacionamento processado possuir outros relacionamentos, E2 será definida como E1 dos seus outros relacionamentos, e assim por diante. Caso E2 do primeiro relacionamento processado não tenha outros relacionamentos, a escolha torna-se aleatória.

Se o elemento que representa E1 já havia sido criado como exclusivo, criar um sub-elemento agrupador (com construtor de ordem “exclusive”) para E1 e colocar todos os sub-elementos que existiam como filhos do agrupador.



**Figura 3.7:** Regra 3 para relacionamentos.

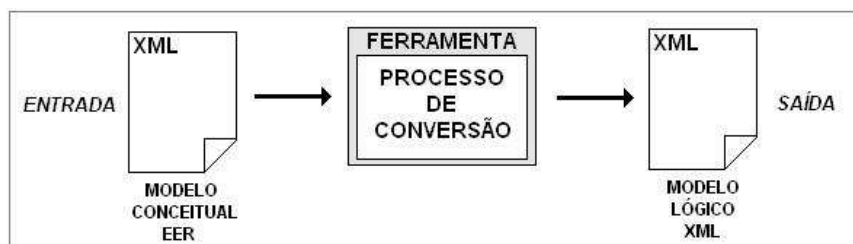
A tabela 3.2 apresenta as prioridades para aplicação das regras, descritas acima, para relacionamentos. As prioridades são dadas às regras que gerar o menor esquema possível, considerando as pré-condições de cada regra [SCHROEDER, 2008].

**Tabela 3.2 - Prioridades** para Aplicação das Regras de Relacionamentos Associativos.

Tipo Relacionamento	Alternativas de Mapeamento		
	Regra 1	Regra 2	Regra 3
(0,1) – (0,1)	-	-	1º
(0,1) – (1,1)	1º	2º	3º
(1,1) – (1-1)	1º	2º	3º
(0,1) – (0,N)	-	-	1º
(0,1) – (1,N)	-	-	1º
(1,1) – (0,N)	-	1º	2º
(1,1) – (1,N)	-	1º	2º
N – N	-	-	1º
N-ários (n>2)	-	-	1º

### 3.2 O PROCESSO DE CONVERSÃO

O processo de conversão é baseado em um algoritmo que promove uma seqüência de procedimentos de conversão. Este algoritmo pode ser estruturado esquematicamente da seguinte forma: (i) Entrada e Saída; (ii) Conversões.



**Figura 3.8:** Visão geral do processo de conversão.

A entrada do algoritmo requer um esquema conceitual do modelo EER especificado em um documento XML (entrada de ferramenta), como mostra a Figura 3.8. Também é definida uma Entidade de Partida, ou seja, a partir de qual entidade o processo de conversão inicia. Além disso, são definidos o número máximo de atributos

que uma superclasse (ou conjunto de subclasses) pode ter e o número máximo de relacionamentos associativos que uma superclasse pode ter.

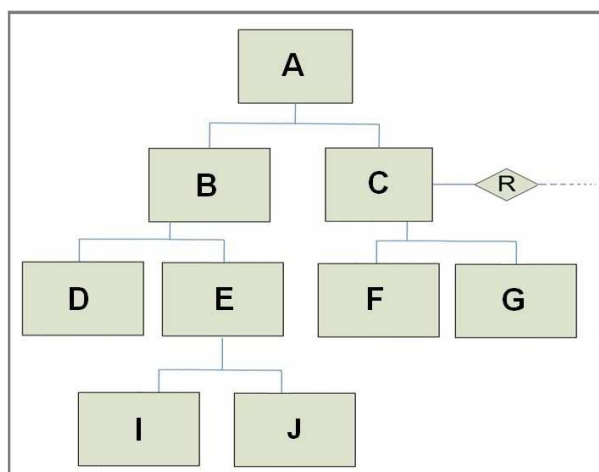
Estes números máximos de atributos e relacionamentos mencionados acima, atualmente, são pré-definidos internamente na ferramenta e buscam delimitar valores máximos para que não haja um número excessivo de atributos e relacionamentos no elemento da superclasse. Posteriormente, estes números máximos de atributos e relacionamentos deverão ser parametrizados pelo usuário especialista.

Atualmente, o formato dos documentos XML de entrada é gerado por uma ferramenta de modelagem conceitual de BD desenvolvida no Grupo de Banco de Dados da UFSC e chamada *BrModelo* [CANDIDO, 2005].

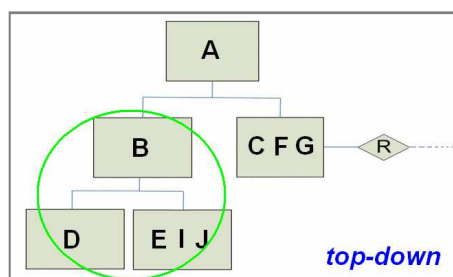
A saída apresenta um documento XML contendo o esquema lógico XML devidamente convertido e formatado para visualização. Atualmente o documento de saída é apresentado no formato de um XML *Schema*.

Para iniciar as conversões é necessário agrupar as categorias e as generalizações/especializações em um único conjunto (chamado de GENERAL\_CAT, por exemplo). Caso existam hierarquias de múltiplos níveis no esquema conceitual de entrada, os integrantes deste conjunto são ordenados de modo que a conversão ocorra no sentido *bottom-up*, ou seja, dos elementos dos níveis inferiores para os elementos dos níveis superiores. A escolha do sentido *bottom-up* para ordenação deve-se ao fato de que do contrário (sentido *top-down*) estaríamos gerando elemento(s) para representar generalizações/Especializações e Categorias cujas subclasses ainda não foram definidas. isto é, poderiam existir sub-hierarquias não convertidas ou ainda não definidas.

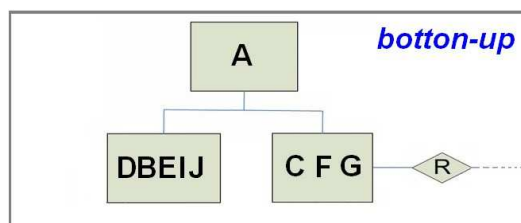
A figura 3.9 apresenta um exemplo de uma hierarquia de múltiplos níveis. Se ordenássemos esta hierarquia de modo que o sentido da conversão se desse no sentido *top-down* e *bottom-up*, após a aplicação das regras de conversão na hierarquia ordenada (seguindo critérios como as pré-condições e as tabelas de prioridades) teríamos, esquematicamente, as hierarquias de elementos gerados representados pelas figuras 3.10 e 3.11, respectivamente.



**Figura 3.9:** Hierarquia de múltiplos níveis.



**Figura 3.10:** Hierarquia gerada após ordenação top-down.



**Figura 3.11:** Hierarquia gerada após ordenação bottom-up.

As conversões iniciam-se por generalizações e categorias porque se convertêssemos relacionamentos antes ou juntamente com generalizações e categorias poderia acontecer de, por exemplo, separar duas subclasses que na verdade dizem respeito a um mesmo conceito (superclasse). E ainda, na conversão de generalizações nós temos maior chance de reduzir o esquema através de regras de fusão.

Desta forma, um procedimento de conversão de generalizações e categorias é chamado e recebe como parâmetro o conjunto GENERAL\_CAT. Este procedimento processa (converte) os elementos do conjunto GENERAL\_CAT a partir do tipo de cada

elemento, da tabela de prioridades de aplicação das regras e das precondições das regras.

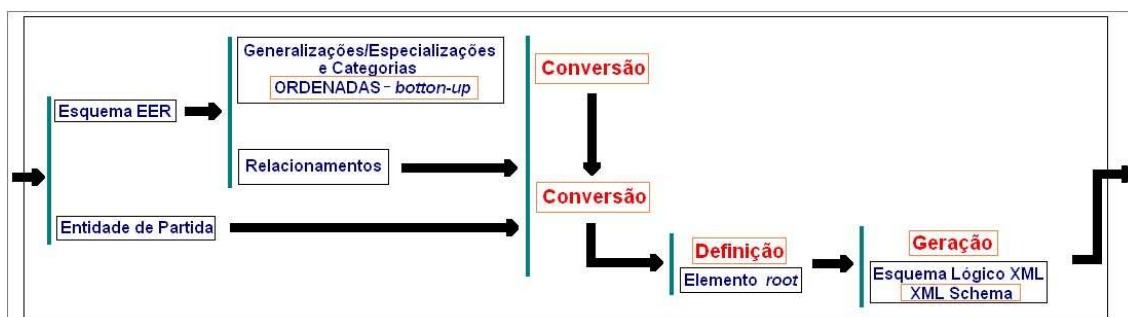
Na seqüência do algoritmo é chamado um procedimento de conversão de relacionamentos, que recebe como parâmetro o conjunto de relacionamentos (chamado de REL, por exemplo) e a Entidade de Partida. Da mesma forma que o procedimento descrito anteriormente, este procedimento processa (converte) todos os elementos do conjunto REL a partir do tipo de relacionamento, da tabela de prioridades de aplicação das regras e das precondições das regras.

A Entidade de Partida, que é um dos parâmetros para a conversão de relacionamentos, é escolhida pelo usuário especialista na entrada do processo juntamente com o esquema EER.

Por fim, é definido um elemento raiz (*root*) para o esquema, chamando-se o procedimento responsável por esta ação. Posteriormente é gerado o documento XML de saída.

A definição do elemento *root* funciona basicamente da seguinte maneira: se houver apenas um elemento no esquema lógico que não constitui destino para nenhum relacionamento hierárquico, este elemento é o elemento *root* do modelo lógico. Caso contrário, é criado um novo elemento para assumir o papel de elemento *root* do modelo lógico em questão.

A figura 3.12 apresenta um diagrama bastante simples (esquemático) que demonstra o algoritmo do processo de conversão descrito. Note que esta figura “abre a caixa preta” apresentada na figura 3.8.



**Figura 3.12:** Diagrama algoritmo do processo de conversão.



## 4 IMPLEMENTAÇÃO E RESULTADOS

Este capítulo apresenta o projeto e a implementação da ferramenta EER2XML, bem como os testes, os resultados obtidos e um estudo de caso pontual.

### 4.1 A FERRAMENTA EER2XML

A ferramenta desenvolvida tem como objetivo contribuir com uma dissertação de mestrado, em desenvolvimento no Grupo de Banco de Dados (GBD) da UFSC, visando dar suporte ao processo de conversão de um esquema conceitual Entidade-Relacionamento Estendido (EER) para um esquema lógico XML [SCHROEDER, 2008].

A ferramenta EER2XML foi desenvolvida na linguagem Java através do ambiente de desenvolvimento Netbeans 6.0.1. A API JDOM [JDOM, 2007] foi utilizada para a leitura do documento XML de entrada que representa um esquema conceitual EER. Para o desenvolvimento da interface gráfica da ferramenta foi utilizado o pacote Swing juntamente com o *framework* JGoodies [JGOODIES, 2007], que disponibiliza componentes e soluções de projeto que complementam o pacote Swing.

A figura 3.8 mostra a visão geral de funcionamento da ferramenta. A entrada é um documento XML que contém o esquema conceitual EER. O formato deste documento é gerado, atualmente, por uma ferramenta de modelagem conceitual de BD chamada *brModelo* [CANDIDO, 2005]. O documento de entrada é então traduzido para objetos do modelo EER da ferramenta para serem convertidos para objetos do modelo lógico XML. A saída da ferramenta é um documento XML, que, atualmente, pode ser visualizado no formato de um documento XML *Schema*.

As atividades de implementação da ferramenta foram divididas em três etapas. A primeira etapa trata da criação do modelo de classes para representar os conceitos dos modelos EER e do modelo lógico XML através de classes Java. A segunda etapa trata da implementação do algoritmo do processo de conversão e da construção da interface

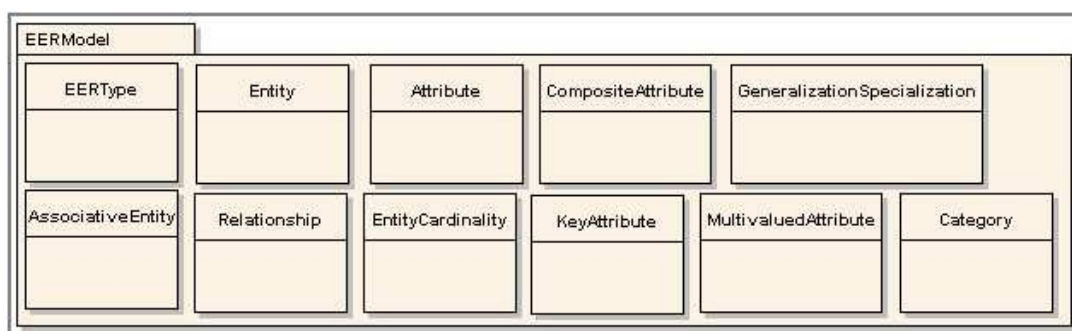
gráfica da ferramenta. Por fim, a última etapa trata do refinamento da ferramenta, da realização de testes e do aprimoramento da interface gráfica.

A seguir é apresentado o projeto da ferramenta EER2XML seguindo as etapas de implementação.

#### 4.1.1 PROJETO

As primeiras etapas da implantação da ferramenta tratam da criação dos modelos de classes para representar os conceitos dos modelos EER e do modelo lógico XML, da implementação das regras de conversão, do algoritmo do processo de conversão e da construção da interface gráfica da ferramenta através de classes Java.

As classes Java desenvolvidas para representar os conceitos dos modelos EER e do modelo lógico XML foram agrupadas em dois pacotes de classes, *EERmodel* e *XMLmodel*, como mostram, respectivamente, as figuras 4.1 e 4.2.

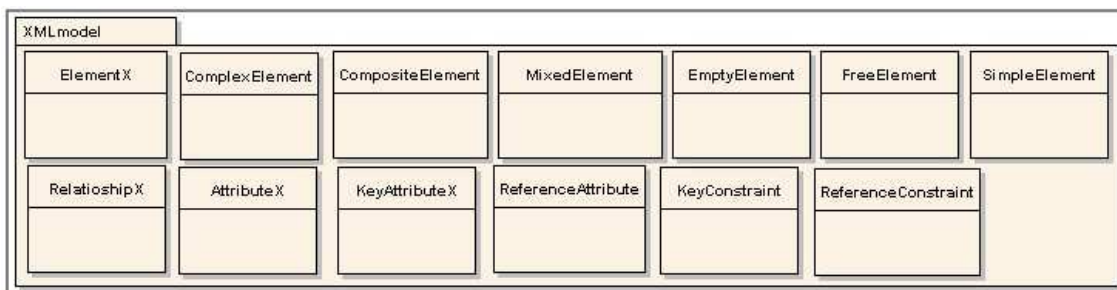


**Figura 4.1:** Pacote de classes *EERmodel*.

O pacote *EERmodel* contém as classes que tratam dos construtores do modelo EER. A definição (ou função) de cada classe contida neste pacote é bastante intuitiva. Por exemplo, as classes *Attribute*, *KeyAttribute*, *CompositeAttribute* e *MultivaluedAttribute* definem atributos normal, identificador, composto e multivalorado respectivamente. A classe *Relationship* define um relacionamento e é composta por componentes definidos na classe *EntityCardinality*, que são entidades com suas respectivas cardinalidades no relacionamento em questão.

É importante ressaltar que as classes *Category* e *GeneralizationSpecialization* implementam a interface *MultiLevelHierarchy*, que é responsável por ordenar

categorias e generalizações/especializações, quando houver hierarquias de múltiplos níveis. O processo ordena e generalizações/especializações e categorias de forma que a ordem de conversão se dê no sentido *bottom-up*, como explicado no capítulo anterior.



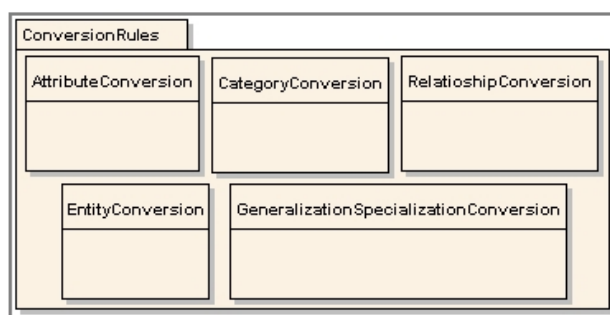
**Figura 4.2:** Pacote de classes *XMLmodel*.

O pacote *XMLmodel* possui os elementos que são gerados a partir do pacote *EERmodel*. A definição (função) de cada classe contida neste pacote, como no pacote *EERmodel*, também é bastante intuitiva.

A classe abstrata *ElementX* possui o método abstrato *toXSD* que tem a função de processar os elementos do documento XML de saída (no formato de um XML Schema). As classes *ComplexElement* e *SimpleElement* estendem a classe *ElementX*, assim como as classes *CompositeElement*, *FreeElement* e *EmptyElement* estendem a classe *ComplexElement*. Já a classe *MixedElement* estende a classe *CompositeElement*. Deste conjunto de classes, somente as classes *CompositeElement* e *SimpleElement* implementam o método abstrato da classe *ElementX*.

As classes *ReferenceConstraint*, *KeyConstraint*, *ReferenceAttribute* e *KeyAttributeX* utilizadas juntas, são responsáveis por manter a consistência dos relacionamentos de referência.

Após a implementação das classes que representam os conceitos dos modelos EER e do modelo lógico XML, as regras de conversão (apresentadas na seção 3.1) foram implementadas e foram agrupadas no pacote *ConversionRules*, como mostra a figura 4.3.



**Figura 4.3:** Pacote de classes *ConversionRules*.

O algoritmo do processo de conversão foi implementado na classe *ConversionProcess*, que é a classe mais importante do projeto. Esta classe está agrupada no pacote *Main* do projeto juntamente com a classe de testes *Test*.

A classe *ConversionProcess* possui quatro métodos principais que são aplicados sequencialmente durante o processo de conversão. O primeiro método, apresentado na figura 4.4, diz respeito à conversão de generalizações e categorias. Neste método, o conjunto de objetos de generalizações e categorias (*generalCat*), que implementam a interface *MultiLevelHierarchy*, é processado. Neste ponto do processo, o conjunto *generalCat* já se encontra ordenado nos casos em que existe hierarquias de múltiplos níveis. Ele é então iterado (a coleção é percorrida) e cada objeto deste conjunto é convertido, através das regras de conversão, segundo a sua instância (generalização ou categoria).

```
//convert Generalizations and Categories
public void convertGeneralizationsCategories(){
    Iterator <MultiLevelHierarchy> it = this.generalCat.iterator();
    //for each object (generalization or category)
    while(it.hasNext()){
        Object o = it.next();
        if(o instanceof Category){
            Category cat = (Category)o;
            CategoryConversion convertCat = new CategoryConversion();
            //convert
            convertCat.convertCategory(cat, maxAtt, maxRel);
            //set to processed
            cat.setToProcessed();
        }
        else{
            GeneralizationSpecialization general = (GeneralizationSpecialization)o;
            GeneralizationSpecializationConversion convertGeneral = new GeneralizationSpecializationConversion();
            //convert
            convertGeneral.convertGeneralization(general, maxAtt, maxRel);
            //set to processed
            general.setToProcessed();
        }
    }
}
```

**Figura 4.4:** Método *convertGeneralizationsCategories*.

O segundo método na seqüência dos quatro principais é apresentado na figura 4.5. Este método promove a conversão de relacionamentos. Inicialmente, dois conjuntos de relacionamentos (*remainderRelationships* e *relationships*) recebem todos os relacionamentos do esquema EER a ser convertido. Na medida em que as conversões ocorrem, os relacionamentos processados são removidos do primeiro conjunto. Os objetos do segundo conjunto são iterados e cada instância é processada se ainda não a foi. A conversão ocorre através da instância da classe *RelationshipConversion*, que implementa as regras de conversão para relacionamentos e que recebe como parâmetros o objeto relacionamento do segundo conjunto iterado, a entidade de partida (definida pelo usuário na entrada de dados da ferramenta) e o primeiro conjunto, que tem os seus objetos removidos a cada relacionamento processado. A cada conversão é verificada a possibilidade de processar o relacionamento seguinte automaticamente, ou seja, verifica-se se outros relacionamentos em que as entidades do relacionamento a ser convertido possuem e se possível os converte na seqüência.

Por fim, as entidades de cada relacionamento processado são verificadas e tratadas para manter a consistência do esquema lógico XML de saída.

```
//convert Relationships
public void convertRelationships(){
    RelationshipConversion convertRel = new RelationshipConversion();
    EntityCardinality ec;
    Entity ent;
    while(!this.remainderRelationships.isEmpty()){
        Iterator <Relationship> it = this.relationships.iterator();
        while(it.hasNext()){
            Relationship rel = it.next();
            if(rel.isProcess()){
                continue;
            }else{
                //convert
                convertRel.convertRelationship(rel, this.initialEntity, this.remainderRelationships);
                //verify relationship entities
                Iterator <EntityCardinality> it2 = rel.getComponents().iterator();
                while(it2.hasNext()) {
                    ec = (EntityCardinality)it2.next();
                    ent = ec.getEntity();
                    if(ent.getConvertCase()==18){
                        this.remainderFirstLevel.add(ent);
                    }
                    if(ent.isFirstLevel() && !this.firstLevel.contains(ent)){
                        this.firstLevel.add(ent);
                    }
                }
            }
        }
    }
}
```

**Figura 4.5:** Método *convertRelationships*.

O terceiro método trata da definição do elemento *root* (raiz do esquema). Este método, apresentado na figura 4.6, verifica quantas entidades não constituem destino para nenhum relacionamento hierárquico (conjunto *firstLevel*). Havendo mais do que uma entidade neste conjunto, cria-se um novo elemento para ser o elemento *root* do esquema e neste elemento são adicionados relacionamentos filhos onde os elementos destinos são os elementos em que as entidades do conjunto *firstLevel* foram convertidas.

Se houver apenas uma entidade no conjunto *firstLevel*, o elemento em que esta entidade foi convertida será o elemento *root* do esquema. Caso não exista nenhuma entidade no conjunto, é criado um novo elemento para ser o elemento *root* do sistema, e neste elemento é adicionado um relacionamento filho onde o elemento destino é o elemento em que a entidade de partida foi convertida.

```
//define root element
public void defineRootElement(){
    //create a new element
    if(this.firstLevel.size()>1){
        this.root = new CompositeElement("root");
        //add first-level entities as root children
        Iterator <Entity> it = this.firstLevel.iterator();
        while(it.hasNext()) {
            Entity e = (Entity)it.next();
            RelationshipX relX = new RelationshipX(this.root, (ComplexElement)e.getXMLconverted());
            this.root.addRelChild(relX);
        }
    }else {
        //there is one first-level entity
        Iterator <Entity> it3 = this.firstLevel.iterator();
        if(!this.firstLevel.isEmpty()){
            Entity e = (Entity)it3.next();
            this.root = (CompositeElement)e.getXMLconverted();
        }
    }
    //there isn't first-level entity
    if (this.root == null){
        this.root = new CompositeElement("root");
        RelationshipX relX = new RelationshipX(this.root, (ComplexElement)this.initialEntity.getXMLconverted());
        this.root.addRelChild(relX);
    }
}
```

**Figura 4.6:** Método *defineRootElement*.

O último método na seqüência trata da geração do documento XML de saída no formato de um XML *Schema*. A figura 4.7 apresenta este método.

As classes *SimpleElement* e *CompositeElement* do pacote *XMLmodel* implementam o método da superclasse abstrata *ElementX* chamado *toXSD* que processa os elementos do esquema, que por sua vez é utilizado no método *generateXSD*.

```

public void generateXSD(){
    sb.append("<?xml version='1.0' encoding='ISO-8859-1' ?>\n");
    sb.append("<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>\n");
    //verify remainder first-level elements
    if(!this.remainderFirstLevel.isEmpty()){
        Iterator <Entity> itt = this.remainderFirstLevel.iterator();
        while(itt.hasNext()){
            Entity e = itt.next();
            RelationshipX relX = new RelationshipX(this.root, (ComplexElement)e.getXMLconverted());
            this.root.addRelChild(relX);
        }
    }
    //process root's children
    if(!this.root.hasNotChildren()){
        sb.append(this.root.toXSD(1, 10));
    }else
        throw new RuntimeException ("Root Element hasn't RelChild!!!");
    sb.append("</xs:schema>");
    this.output.writeString(sb.toString());
    this.output.close();
}

```

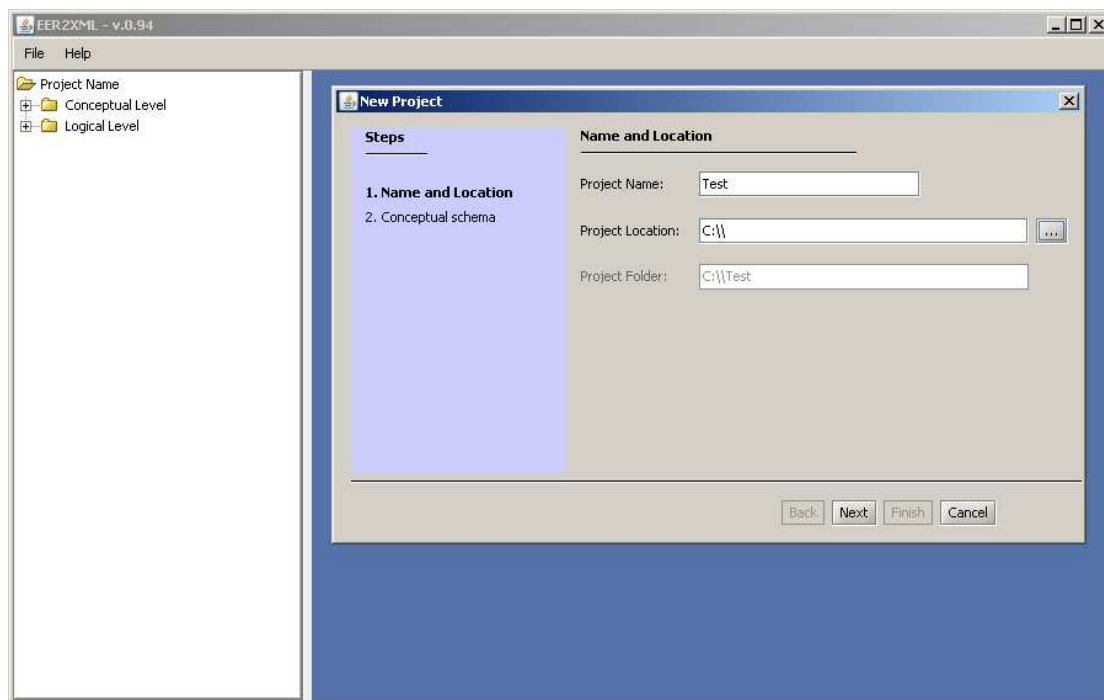
**Figura 4.7:** Método *generateXSD*.

#### 4.1.2 FUNCIONAMENTO E INTERFACE GRÁFICA

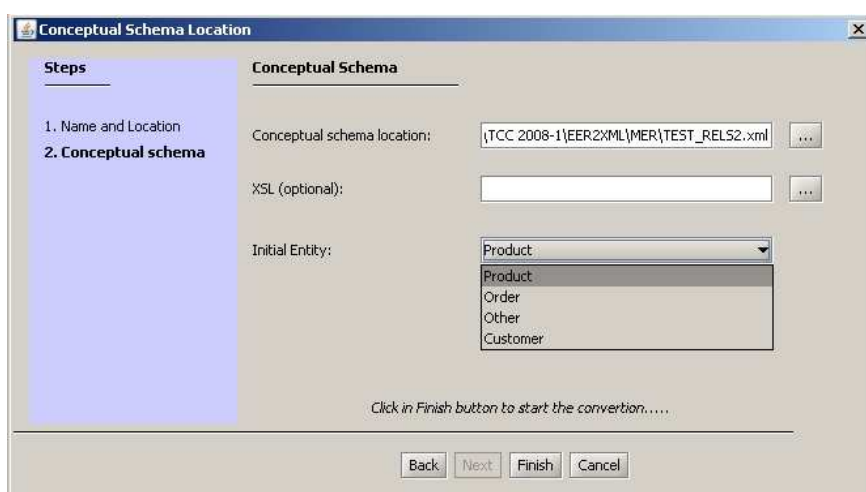
A interface gráfica da ferramenta é bastante simplificada, pois trata-se de um processo de conversão automático. O usuário especialista precisa interagir em apenas dois passos sequenciais para realizar a conversão.

O primeiro passo, apresentado na figura 4.8, deve-se informar o nome do projeto e o local onde será salvo o documento XML de saída gerado pela ferramenta.

Na seqüência, o usuário deve informar a localização do documento XML de entrada que contém o esquema conceitual EER. Opcionalmente, pode-se informar também um documento XSL (*Extensible Stylesheet Language*) para a transformação do documento XML de entrada no padrão aceito pela ferramenta. Para concluir, o usuário especialista deve escolher uma das entidades como a entidade inicial (ou entidade de partida), como mostra a figura 4.9. A entidade de inicial é um dos parâmetros para a conversão de relacionamentos e é importante para determinar o encadeamento entre os elementos do esquema lógico.



**Figura 4.8:** Ferramenta EER2XML.



**Figura 4.9:** Tela *Conceptual Sechema Location*.

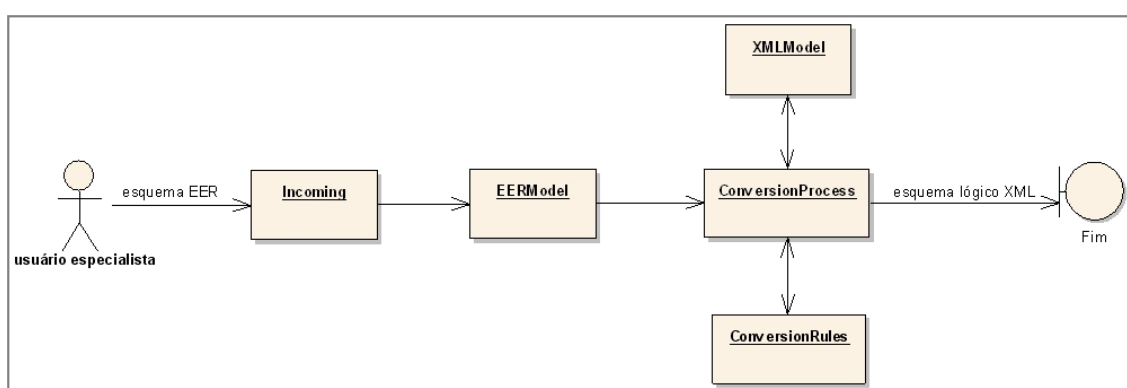
A Figura 4.10 apresenta um diagrama esquemático que possibilita uma visualização geral do funcionamento interno da ferramenta. Neste diagrama estão representados os principais pacotes de classes da ferramenta e a relação entre eles.

O pacote *Incoming* contém as classes que são responsáveis em processar o documento XML que contém o esquema EER de entrada do processo, transformando-os em objetos do pacote *XMLmodel*.



O pacote *EERmodel* contém classes que representam cada um dos conceitos do modelo EER.

A classe *ConversionProcess* (agrupada no pacote *Main* do projeto e representada como pacote *ConversionProcess* neste diagrama) contém as classes necessárias para transformar um esquema EER em um esquema lógico XML. A classe *ConversionProcess* é assistida pelas classes do pacote *ConversionRules*, que contém a implementação das regras de conversão e também pelo pacote *XMLmodel*, que contém classes que representam os conceitos do modelo lógico XML.



**Figura 4.10:** Principais componentes e a relação entre eles.

A classe *ConversionProcess* é responsável também pela geração do documento XML de saída no formato de XML Schema.

## 4.2 TESTES E RESULTADOS

Os testes de software verificam através de uma execução controlada se o seu comportamento corre de acordo com o especificado. O objetivo principal desta tarefa é encontrar o número máximo de erros, ou seja, verificar se os resultados estão ou não de acordo com os padrões estabelecidos.

Os testes aplicados à ferramenta EER2XML foram realizados ao longo das etapas da implantação da mesma. Inicialmente, os testes se assemelharam aos testes de caixa branca, ou seja, trabalhou-se diretamente sobre o código-fonte de cada componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos. As classes contidas nos

pacotes *EERmodel*, e *XMLmodel* passaram com maior frequência por estes tipos de testes.

Os resultados destes primeiros testes foram bastante positivos, ou seja, eles corresponderam aos resultados esperados, que por sua vez validam a modelagem das classes que representam os conceitos dos modelos EER e do modelo lógico XML.

No decorrer da implementação, os testes mudaram do perfil. Mudaram de testes de caixa branca para os testes de caixa preta, ou seja, não se considerou tanto o comportamento interno da ferramenta como nos testes iniciais. Os dados de entrada eram fornecidos, o teste era executado e o resultado obtido foi comparado a um resultado esperado previamente conhecido ou simulado. A segunda metade da implementação (das regras de conversão e do processo de conversão) passaram com maior frequência por esta abordagem de testes.

Os resultados para a conversão das regras são bastante positivos. As conversões isoladas não demonstraram problema algum. A conversão de esquemas (conceituais EER) menores e simples trouxe bons resultados, poucos erros foram encontrados e a maior parte destes já estão corrigidos.

No entanto, quando submetida a esquemas maiores e mais complexos, a ferramenta se mostra um pouco instável, ou seja, dependendo da entidade de partida escolhida, o documento XML de saída traz trechos confusos, com algumas repetições e conversões não mostradas. Por este motivo, a ferramenta precisa e está recebendo novos testes de caixa branca e também de caixa preta.

Como já mencionado, o formato do documento de entrada é gerado, atualmente, por uma ferramenta de modelagem conceitual de BD chamada *BrModelo*. Esta ferramenta trata da modelagem de BD e engloba apenas os conceitos do modelo ER (Entidade-Relacionamento), o que é uma “restrição” para os testes da ferramenta EER2XML que trata dos conceitos do modelo EER (Entidade-Relacionamento Extendido).

Conceitos como Disjunção (para especializações/generalizações) e Categoria (Tipo União) não são tratados pela ferramenta *BrModelo*. Por isto, para a realização dos testes destes conceitos foram adicionados manualmente sub-elementos aos elementos do documento XML gerado pela ferramenta *BrModelo*, que é o documento de entrada da ferramenta EER2XML.

As figuras 4.11 e 4.12 demonstram a adição destes sub-elementos ao documento XML gerado pela ferramenta *BrModelo*.

```

<Entidade nome="U2" id="33">
  <Left Valor="721"/>
  <Top Valor="532"/>
  <Width Valor="102"/>
  <Height Valor="66"/>
  <Fonte default="-1"/>
  <Atributos/>
  <AtributosOcultos/>
  <Dicionario></Dicionario>
  <Nula Valor="0"/>
  <Observacao></Observacao>
  <Anexos/>
  <AutoRelacoes AutoRelacionado="0"/>
  <TiposUniao>
    <Uniao id="88">
      <Parcial Valor="0"/>
      <Ligacoes>
        <Ligacao Destino_ID="11"/>
        <Ligacao Destino_ID="35"/>
      </Ligacoes>
    </Uniao>
  </TiposUniao>
  <Especializacoes ehEsp="0"/>
</Entidade>

```

**Figura 4.11:** Adição de Tipos União ao documento XML de entrada.

```

<Especializacoes ehEsp="-1">
  <Especializacao nome="Esp_1" id="2">
    <Left Valor="249"/>
    <Top Valor="124"/>
    <Width Valor="50"/>
    <Height Valor="31"/>
    <Fonte default="0">
      <FonteNome Valor="Tahoma"/>
      <FonteTamanho Valor="8"/>
      <FonteEstilo Valor="[fsBold,fsItalic]"/>
      <FonteCor Valor="32896"/>
    </Fonte>
    <Atributos/>
    <AtributosOcultos/>
    <Dicionario></Dicionario>
    <Nula Valor="0"/>
    <Observacao></Observacao>
    <Anexos/>
    <Parcial Valor="0"/>
    <Disjoint Valor="-1"/>
    <Ligacoes>
      <Ligacao Destino_ID="1">
        <MostraCardinalidade Valor="0" Card_id="3"/>
      </Ligacao>
    </Ligacoes>
  </Especializacao>
</Especializacoes>

```

**Figura 4.12:** Adição de Disjunção ao documento XML de entrada.

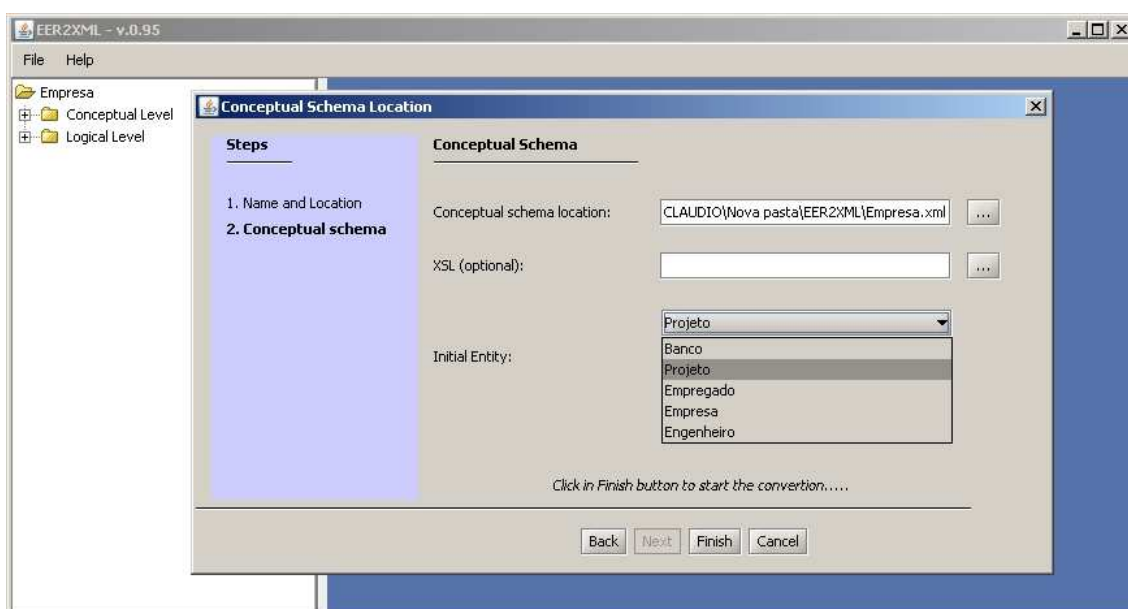
### 4.3 ESTUDO DE CASO

Esta seção apresenta um estudo de caso que realiza a conversão de um esquema conceitual através da ferramenta EER2XML. A Figura 2.1 apresenta um diagrama EER simples do domínio de uma Empresa, gerado pela ferramenta *BrModelo* [CANDIDO, 2005]. O documento XML de entrada encontra-se nos anexos deste trabalho.

O esquema conceitual é passado na forma de um documento XML como entrada da ferramenta através da tela apresentada pela Figura 4.13. Opcionalmente, um documento XSL pode ser informado para a transformação do documento XML de entrada no padrão aceito pela ferramenta EER2XML.

O usuário especialista deve selecionar uma das entidades do esquema conceitual como entidade principal. Uma vez informados estes dados, dá-se início o processo automático de conversão.

A Figura 2.5 apresenta o diagrama do esquema lógico gerado pela ferramenta correspondendo ao esquema EER apresentado pela Figura 2.1.



**Figura 4.13:** Entrada de dados para a conversão.

No processo de conversão, a generalização e a categoria são convertidas primeiramente. A generalização total e compartilhada estabelecida pela superclasse *Empregado* pode ser convertida pela *Regra 3* devido a inexistência de relacionamentos específicos com a subclasse *Motorista*. O atributo opcional *Contato* torna-se elemento simples, sendo adicionado o atributo *Type* ao elemento criado *Empregado* para identificar qual das subclasses está sendo instanciada.

Os relacionamentos EER são convertidos começando pela entidade principal, neste caso indicada como sendo a entidade *Projeto*. A partir dela, o relacionamento *participa* é convertido aplicando-se a *Regra 3*, pois a entidade *Engenheiro* já foi

processada, como ressaltado nas pré-condições das *Regras 1 e 2*. Cria-se o elemento *Projeto* e o sub-elemento *participa*, onde um relacionamento de referência é estabelecido entre os elementos *participa* e *Engenheiro (Empregado)* através do atributo de referência *EmpregadoCode*.

O próximo relacionamento, *sociedade*, é convertido pela *Regra 1*, pois trata-se de um relacionamento binário [1..1] em que as entidades participantes ainda não foram processadas, conforme ressaltado nas pré-condições desta regra. Cria-se o elemento *Empresa* para representar todo o relacionamento e adiciona-se o atributo da entidade *Banco, CNPJBanco*, no elemento criado.

O terceiro e último relacionamento convertido é *trabalha* e nele se aplica a *Regra 3*, pois trata-se de um relacionamento binário [1..N] em que as entidades envolvidas já foram processadas. Cria-se o sub-elemento *trabalha* ao elemento *Empregado* e um relacionamento de referência é estabelecido entre os elementos *trabalha* e *Empresa* através do atributo de referência *EmpresaCode*.

Finalmente, um elemento *root* é criado e relacionamentos hierárquicos são estabelecidos entre o elemento *root* e os elementos que não constituem destino para qualquer outro relacionamento hierárquico, neste caso: *Projeto, Empregado, e Empresa*.

Após este passo, é criado um documento XML que contém o esquema lógico XML gerado pela conversão no formato de XML *Schema*. Este documento gerado pode ser conferido nos anexos deste trabalho.

## **5 CONCLUSÕES E TRABALHOS FUTUROS**

### **5.1 CONCLUSÕES**

Este trabalho apresenta uma ferramenta de apoio ao projeto lógico de BD XML, específica para a tradução de esquemas conceituais EER em esquemas lógicos XML. A ferramenta implementa um processo de conversão que considera todas as possíveis restrições e construções do modelo EER, gerando esquemas lógicos XML capazes de serem traduzidos para qualquer linguagem de definição de esquemas XML.

A principal contribuição da ferramenta, certamente, é a validação (e posterior aplicação prática) deste processo de conversão [SCHROEDER, 2008]. O fato de nenhuma outra ferramenta ser encontrada na literatura, que implemente um processo de conversão totalmente automático, que considera todos os conceitos do modelo EER e que gera esquemas lógicos de alta abstração, também é uma importante contribuição

O projeto da ferramenta encontra-se em fase de testes e aprimoramentos na interface gráfica. Para a realização dos testes estão sendo aplicados diversos estudos de caso reais de projeto de BD XML para, ocasionalmente, levantar pontos do projeto que necessitam de refinamento.

### **5.2 TRABALHOS FUTUROS**

Como atividades futuras, considera-se:

- A expansão da ferramenta para considerar atividades de projeto físico de um BD XML.
- Uma análise de usabilidade da interface gráfica, de modo a torná-la mais agradável e eficaz para o usuário, como, por exemplo, agregar componentes de visualização à ferramenta.

- A implementação do algoritmo que busque a entidade de partida que apresente o maior *fechamento funcional*, ou seja, a entidade na qual inicia-se a conversão e se consegue uma estrutura de aninhamento mais conexa e compacta.
- A implementação de um processo de conversão semi-automático, permitindo a intervenção do usuário especialista durante o processo na escolha das estratégias de conversão para cada porção do esquema conceitual.

## *Referências Bibliográficas*

[SCHROEDER, 2008] SCHROEDER, R. e MELLO, R. S. (2008) “Conversion of Generalization Hierarchies and Union Types from Extended Entity-Relationship Model to an XML Logical Model”, In: 23<sup>rd</sup> Annual ACM Symposium on Applied Computing.

[MELLO, 2003] MELLO, R. S. (2003) “Gerenciamento de Dados XML”, In: V Escola de Informática Norte da SBC, v.1, p. 15-34.

[W3C, 2007] W3C, World Wide Web Consortium. Disponível em <<http://www.w3.org>>. Acesso em 21 de agosto de 2007.

[JDOM, 2007] JDOM. (2007) “JDOM: Documentation” Disponível em: <http://www.jdom.org/downloads/docs.html>. Último Acesso: 29 set. 2007.

[JGOODIES, 2007] JGOODIES. (2007) “JGOODIES: Documentation”, Disponível em: <http://www.jgoodies.org/>. Último Acesso: 30 nov. 2007.

[W3SCHOOLS, 2007] W3SCHOOLS. (2007) XML Tutorials. Disponível em: <<http://www.w3schools.com>>. Último Acesso em: 15 jun. 2007.

[RPBOURRET, 2007] RPBOURRET. (2007) XML and Databases. Disponível em: <<http://www.rpbouret.com/xml/XMLAndDatabases.htm>>. Último Acesso em: 22 nov. 2007.

[ELMASRI, 2005] ELMASRI, Ramez. e NAVATHE, Shamkant. Sistemas de Banco de Dados. 4<sup>a</sup> edição. São Paulo, 2005. Pearson / Prentice Hall.

[BARCAROLI, 2005] BARCAROLI, Velcir. e MELLO, Ronaldo dos Santos. Análise Comparativa de Linguagens de Consulta para Banco de Dados XML Nativos. Florianópolis, mar. 2005. GBD-UFSC.

[CANDIDO, 2005] CANDIDO, C. H. Aprendizagem em Banco de Dados: Implementação de Ferramenta de Modelagem ER. Monografia de Especialização. Disponível em: <http://www.grupobd.inf.ufsc.br/bibliografiaGBD/especializacoes/Especializacao-CarlosCandido-FerramentaModelagemER-2005>. Último acesso: dezembro de 2007.

[BRADLEY, 2003] BRADLEY, Neil. The XML Schema Companion. Addison-Wesley Longman Publishing Co., 2003 apud FRANTZ, Arthur Pereira. Um Processo de



Conversão de XML Schemas para Esquemas Conceituais. Trabalho de Conclusão de Curso - UFSC, 2004.

[CHANG, 2002] CHANG, E., DILLON, T. and FENG, L. (2002) “A Semantic Network-Based Design Methodology for XML Documents”, In: ACM TOIS, p.390-421.

[CHOI, 2003] CHOI, M., LIM, J. and JOO, K. (2003) “Developing a Unified Design Methodology Based on Extended Entity-Relationship Model for XML”, In: International Conference on Computational Science, p. 920-929.

[DOBBIE, 2001] DOBBIE, G., XIAOYING, W., Ling, T. and Lee, M. (2001) “ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data”, In: Technical Report TR21/00, National University of Singapore.

[EMBLEY, 2004] EMBLEY, D., LIDDLE, S. and KAMHA, S. (2004) “Enterprise Modeling with Conceptual XML”, In: International Conference on Conceptual Modeling, p. 150-165.

[PIGOZZO, 2005] PIGOZZO, P. e QUINTARELLI, E. (2005) “An algorithm for generating XML Schemas from ER Schemas”, In: Italian Symposium on Advanced Database Systems, pp. 192-199.

[BATINI, 1992] BATINI, C. e CERI, S. e NAVATHE, S.B. (1992) “Conceptual Database Design: An Entity-Relationship Approach” , Benjamin/Cummings Publishing Company

[SPARX, 2008] SPARX Systems. (2008) XSD Transformation. Disponível em: < [http://www.sparxsystems.com/resources/mda/xsd\\_transformation.html](http://www.sparxsystems.com/resources/mda/xsd_transformation.html)>. Último Acesso em: 10 mai. 2008.

[RATIONAL, 2008] Rational Software. (2008) Generating XSD Schemas from UML models. Disponível em: < <http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.xtools.transformations.doc/topics/txsdoover.html>>. Último Acesso em: 10 mai. 2008.

[MTF, 2008] Model Transformation Framework. (2008) . Disponível em: <http://www.alphaworks.ibm.com/tech/mtf/faq>>. Último Acesso em: 10 mai. 2008.

***Anexo I - Artigo***

# Uma Ferramenta para Conversão de Esquemas Conceituais EER para Esquemas Lógicos XML

Cláudio de Lima<sup>1</sup>, Rebeca Schroeder<sup>1</sup>, Ronaldo dos Santos Mello<sup>1</sup>

<sup>1</sup>Departamento de Informática e Estatística – Universidade Federal de Santa Catarina  
Caixa Postal 476 – 88.040-900 – Florianópolis –SC – Brasil

{clima, rebecks, ronaldo}@inf.ufsc.br

**Abstract.** *The extensive use of XML documents by several applications requires enhanced document management techniques. One of the research topics related to XML data management is the definition of methodologies for XML Database (DB) design, given that this technology does not have a consolidated standard to define an XML DB. This paper presents a tool to provide XML DB logical design support, enabling the conversion of an EER (Extended Entity Relationship) conceptual schema to an XML logical schema. The tool generates XML logical schemata that are able to be converted to any XML schema language, like DTD and XML Schema.*

**Resumo.** *O crescente uso de documentos XML nas mais diversas aplicações requer o gerenciamento adequado destes documentos. Um dos tópicos de pesquisa em aberto com relação à gerência de dados XML é a definição de metodologias para projeto de Bancos de Dados (BD) XML, já que não há uma metodologia consolidada para a modelagem de um BD XML. Este artigo apresenta uma ferramenta de apoio ao projeto lógico de BD XML, que presta suporte ao processo de conversão de esquemas conceituais EER (Extended Entity-Relationship) para esquemas lógicos XML. A ferramenta gera esquemas lógicos XML capazes de serem convertidos para qualquer linguagem de definição de esquema XML, como DTD e XML Schema.*

## 1. Introdução

O formato de documento XML (*eXtensible Markup Language*) é um importante recurso em muitos contextos de aplicações como comércio eletrônico e cadastro bibliográfico, uma vez que cada vez mais aplicações e pessoas disponibilizam e manipulam informações XML na *Web* [XML 2007].

A tecnologia XML assemelha-se à tecnologia de Sistemas Gerenciadores de BD (SGBDs), uma vez que possui um formato de armazenamento específico (documento), possui linguagens de consulta, permite a definição de esquemas e define interfaces (*APIs*) para o acesso a dados, dentre outras características [Mello 2003]. Entretanto, a tecnologia XML não é equivalente à tecnologia de SGBD, pois não existem soluções consolidadas para todos os aspectos de gerenciamento de dados, como controle de integridade, gerenciamento eficiente de transações e metodologias para projeto ou modelagem de BD XML.

Neste contexto, surgiram esforços para resolver o problema da modelagem de dados XML em diferentes níveis de abstração, visando definir o esquema de um documento XML [Liu et al. 2006]. Alguns trabalhos como [Pigozzo et al. 2005] e [Choi et al. 2003] estão focados em converter modelos conceituais de alta abstração como o ER e a UML para uma linguagem específica de definição de esquema XML. Outros trabalhos propõem a utilização de novos modelos para serem aplicados na modelagem de documentos XML. Por exemplo, o trabalho de [Embley et al. 2004] introduz um novo modelo conceitual para ser utilizado no projeto de documentos XML, assim como [Dobbie et al. 2001] e [Chang et al. 2002] apresentam novos modelos lógicos para XML. No entanto, não há consenso sobre metodologias para o projeto de BD XML.

A modelagem tradicional de BD em três níveis pode ser aplicada para BD XML: níveis conceitual, lógico e de implementação [Batini et al. 1992]. Um modelo conceitual tradicional pode ser utilizado no primeiro nível, como o modelo EER (*Extended Entity-Relationship*). O modelo de dados XML passa a ser considerado a partir do nível lógico. O modelo lógico XML a ser utilizado neste nível deve considerar construtores e restrições específicas do modelo de dados XML, de modo a abstrair diferentes modelos de implementação para XML. No nível de implementação são considerados como modelos as linguagens de definição de esquema XML, como DTD (*Document Type Definition*) e XML Schema [XML W3C 2007].

O objetivo deste artigo é apresentar uma ferramenta que presta suporte ao projeto lógico de BD XML através da conversão de esquemas conceituais EER para esquemas lógicos XML. O processo de conversão é baseado em um conjunto de regras que consideram cada uma das construções do EER para a geração de esquemas lógicos correspondentes. Por estar baseada na modelagem de três níveis, a abordagem de conversão proposta não é dependente de qualquer linguagem de definição de esquema XML, permitindo a geração de esquemas lógicos que podem ser convertidos para qualquer modelo de implementação XML. Além disto, a consideração de todos os construtores do modelo EER pelas regras de conversão constitui um diferencial frente aos trabalhos correlatos mencionados anteriormente.

Este artigo está organizado em mais três seções. A seção 2 descreve a metodologia do processo de conversão, apresentando o modelo lógico utilizado pela abordagem bem como as regras e o processo de conversão. Detalhes do projeto da ferramenta e uma pequena exemplificação do processo são apresentados na seção 3. A última seção é dedicada às considerações finais.

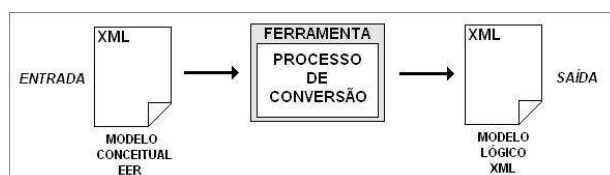
## **2. Metodologia de Conversão EER-XML**

O processo de conversão suportado pela ferramenta inicia com a entrada de um documento XML que contém a descrição de um esquema conceitual EER<sup>1</sup>. Esta entrada é lida pela ferramenta e então processada para gerar um esquema lógico XML. Uma vez concluído o processo de conversão, a ferramenta gera como saída um documento XML que descreve o esquema lógico gerado. A Figura 1 apresenta a visão geral do

---

<sup>1</sup> Atualmente, o formato deste documento XML é gerado por uma ferramenta de modelagem conceitual de BD desenvolvida no Grupo de Banco de Dados da UFSC, chamada *BrModelo* [Candido 2005].

funcionamento da ferramenta. O documento XML de saída pode ser exibido na forma de diagrama pela ferramenta desenvolvida.



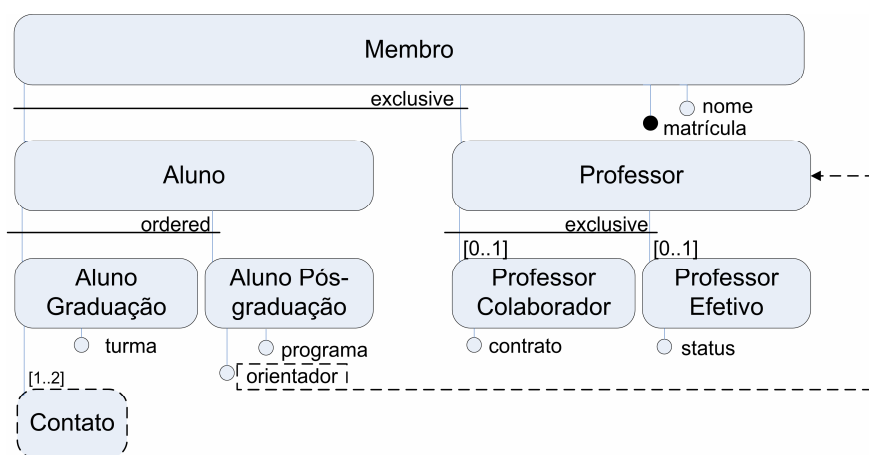
**Figura 1. Visão geral da ferramenta.**

O processo de conversão, bem como o modelo lógico, está sendo desenvolvido por uma dissertação de mestrado do Grupo de Banco de Dados da UFSC. As sub-seções seguintes apresentam uma visão geral do modelo lógico, das regras e do processo de conversão.

### 2.1. Modelo Lógico

O modelo lógico XML considerado representa composições hierárquicas de dados e se baseia nos conceitos de elemento, atributo e relacionamento [Schroeder e Mello 2008]. Este modelo não é um novo formalismo para a modelagem lógica XML, mas uma adaptação de modelos hierárquicos para modelagem de dados semi-estruturados, como [Dobbie et al. 2001] e [Chang et al. 2002]. Para garantir a abstração do modelo lógico, foram considerados como conceitos deste modelo somente construtores do modelo XML que são comuns às mais relevantes linguagens de definição de esquemas XML.

A Figura 2 mostra um exemplo de um esquema no modelo lógico XML. Atributos são representados por um círculo e podem ser do tipo normal, identificador e de referência. Um atributo normal representa uma propriedade de um elemento XML. Atributos identificadores são aqueles que fazem parte do conjunto de atributos que identificam unicamente uma instância de elemento. Um atributo de referência é um atributo que faz referência a outro elemento. Os atributos *nome*, *matrícula* e *orientador* são, respectivamente, exemplos de atributo normal, identificador e de referência.



**Figura 2. Exemplo de um esquema lógico XML.**

Um elemento pode ser simples ou complexo. Um elemento simples representa informações que são definidas por um tipo de dado simples que não possui atributos, como o elemento *Contato* da Figura 2. Um elemento complexo pode ser formado por

atributos e/ou elementos componentes. Os elementos componentes são organizados por um construtor de ordem definido para o elemento complexo que os contém. O construtor de ordem de um elemento complexo pode ser *ordered* ou *exclusive*. Um construtor *ordered* estabelece os elementos componentes de forma ordenada. Já um construtor *exclusive* define uma disjunção entre os elementos componentes. Os elementos *Aluno* e *Professor* são exemplos de elementos complexos com construtor de ordem *ordered* e *exclusive*, respectivamente.

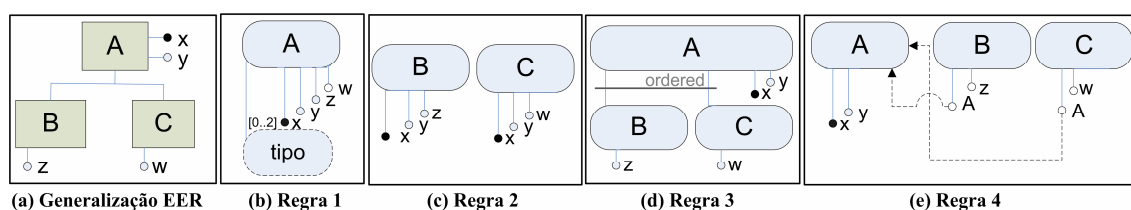
Os relacionamentos podem ser hierárquicos ou de referência. Relacionamentos hierárquicos são representados por linhas contínuas entre o conceito origem e o conceito destino. Um relacionamento define as ocorrências mínima e máxima do conceito destino no conceito origem. Por exemplo, o elemento *Professor*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

## 2.2. Regras de Conversão

O processo de conversão é baseado em regras que consideram os conceitos EER para a tradução em um esquema lógico XML. As regras de conversão envolvidas no processo podem ser divididas em dois grupos: (a) regras para Generalização/Especialização e Categorias; e (b) regras para Relacionamento. Para cada um destes grupos são fornecidas regras alternativas para conversão. Durante o processo de conversão uma destas regras é escolhida automaticamente mediante critério de escolha que é descrito na seção 2.3. Os dois grupos de regras existentes são descritos a seguir.

### 2.2.1. Generalização/Especialização e Categorias

Basicamente, existem quatro regras alternativas para a conversão de generalizações/especializações e categorias, conforme apresentado na Figura 3.



**Figura 3. Estratégias de conversão para generalizações e categorias EER.**

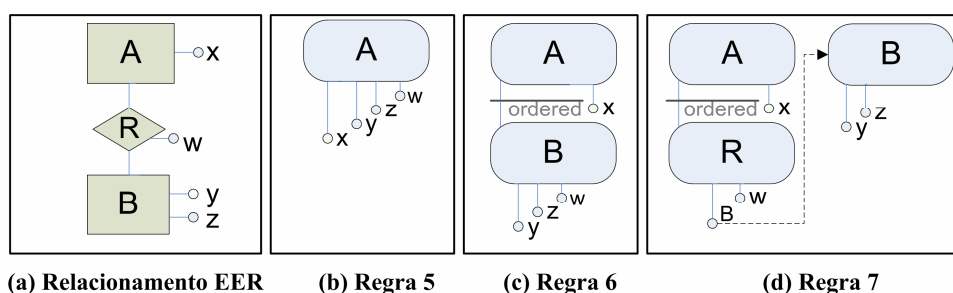
A *Regra 1* (Figura 3(b)) propõe a criação de um único elemento complexo para representar uma generalização ou categoria. A este elemento são adicionados os atributos das subclasses e um atributo ou elemento simples para identificar qual(is) subclasse(s) está(ão) sendo representada(s) pela instância do elemento que representa a superclasse. A *Regra 2* (Figura 3(c)) gera elementos somente para representar as subclasses, adicionando os atributos da superclasse em cada subclasse.

A *Regra 3* (Figura 3 (d)) propõe a criação de elementos tanto para a superclasse quanto para as subclasses. O construtor de ordem do elemento que representa a superclasse deve ser *exclusive* para representar generalizações disjuntas, e *ordered* para

representar as compartilhadas. Para generalizações parciais, a cardinalidade dos elementos das subclasses deve ser opcional ([0..1]). Por fim, a *Regra 4* propõe a criação de elementos para a superclasse e para as subclasses. Para os elementos das subclasses são criados atributos de referência para a superclasse. Ao se tratar da conversão de uma categoria, os papéis das superclasses e subclasse devem ser invertidos para as regras em questão. Isto é, o elemento que representa a subclasse deve ser o elemento pai dos elementos que representam as superclasses<sup>2</sup>.

### 2.2.2. Relacionamentos

Há três regras alternativas para a conversão de relacionamentos, conforme apresentado na Figura 4. A *Regra 5* (Figura 4(b)) propõe a criação de um elemento único para representar as entidades participantes do relacionamento. A *Regra 6* (Figura 4(c)) sugere a criação de elementos para representar cada entidade envolvida no relacionamento, sendo que um dos elementos deve se tornar sub-elemento do outro. Os atributos do relacionamento, para esta regra, devem então ser mantidos no elemento filho e a cardinalidade do relacionamento deve ser representada pela mínima e máxima ocorrência do elemento filho no elemento pai.



**Figura 4. Estratégias de conversão para relacionamentos EER.**

A proposta da *Regra 7* (Figura 4(d)) é a criação de um elemento para representar uma das entidades do relacionamento e a criação de um sub-elemento para representar o relacionamento, juntamente com os seus atributos. Deve-se criar elementos à parte para as demais entidades participantes do relacionamento, sendo que a associação com estes elementos é estabelecida por meio de atributos de referência no elemento que representa o relacionamento.

### 2.3. Processo de Conversão

O processo de conversão é dividido em três passos sequenciais: (i) conversão de generalizações e categorias; (ii) conversão de relacionamentos e; (iii) definição de elemento *root* do esquema lógico XML.

Nos dois primeiros passos do processo de conversão são aplicadas as regras apresentadas na seção 2.2. Por se tratar de um processo automático, para cada instância de generalização ou relacionamento é escolhida apenas uma destas regras para a aplicação. O critério de escolha é pela regra capaz de gerar a menor porção de esquema lógico e que também seja adequada para representar as restrições da porção do esquema EER que está sendo convertida. Para cada regra de conversão existem pré-condições

<sup>2</sup> Uma discussão sobre a aplicabilidade destas regras pode ser encontrada em [Schroeder e Mello 2008].

para a sua aplicação, o que torna possível identificar se tal regra é adequada ou não para ser utilizada para uma determinada porção de esquema conceitual. Para gerar a menor porção de esquema possível, as pré-condições das regras são verificadas na seguinte ordem: (i) para generalizações a seqüência é Regra 1, 2, 3 e 4, e; (ii) para relacionamentos Regra 5, 6 e 7. Por exemplo, generalizações com subclasses que possuem relacionamentos com outras entidades não podem ser convertidas pela Regra 1, pois esta regra impossibilita a conversão destes relacionamentos já que superclasse e subclasses estariam representadas em um único elemento. Neste caso, a Regra 1 é descartada, e então verifica-se a viabilidade da aplicação da Regra 2, e assim por diante. No caso de relacionamentos, por exemplo, a Regra 7 é a única opção para converter relacionamentos N:N, já que as regras 5 e 6 gerariam, neste caso, redundância de informação.

Na conversão de generalizações e categorias, inicialmente é verificada a existência de hierarquias de múltiplos níveis de especialização. Caso existam, o processo ordena as generalizações e categorias envolvidas de forma que a ordem de conversão se dê no sentido bottom-up, isto é, dos níveis inferiores da hierarquia para os superiores.

Na conversão de relacionamentos, o processo inicia a conversão por uma entidade identificada como Entidade Principal, que é sugerida pela ferramenta ou, opcionalmente, fornecida pelo usuário especialista na entrada do processo juntamente com o esquema EER. Esta entidade principal é importante para determinar o encadeamento entre os elementos do esquema lógico, visto que nas regras de relacionamento apresentadas é necessário identificar qual das entidades é representada como elemento pai das demais entidades participantes de um relacionamento. A ferramenta sugere como Entidade Principal a entidade que apresentar a maior participação no fechamento funcional das demais entidades do esquema EER. Este método para identificar a Entidade Principal de esquemas EER através de fechamentos funcionais encontra-se definido em [Mok et al. 2006].

Como último passo, é determinado ou criado o elemento root do esquema lógico. Caso exista apenas um elemento complexo que não constitui destino para qualquer relacionamento hierárquico, tal elemento é determinado como root do esquema lógico. Caso não exista, um novo elemento é criado. Seus elementos filhos diretos são todos os elementos que não constituem destino de qualquer relacionamento hierárquico do esquema lógico gerado.

### **3. A Ferramenta EER-XML**

A ferramenta foi desenvolvida na linguagem Java através do ambiente de desenvolvimento Netbeans 5.5.1. A API JDOM [JDOM 2007] foi utilizada para a leitura do documento XML de entrada que representa um esquema conceitual EER. Para a interface gráfica da ferramenta foi utilizado o pacote Swing juntamente com o framework JGoodies [JGOODIES 2007], que disponibiliza componentes e soluções de projeto que complementam o pacote Swing.

As atividades de implementação da ferramenta foram divididas em três etapas. A primeira etapa trata da criação do modelo de classes para representar os conceitos dos modelos EER e do modelo lógico XML através de classes Java. A segunda etapa trata da implementação do algoritmo do processo de conversão e da construção da interface



gráfica da ferramenta. Por fim, a última etapa trata do refinamento da ferramenta, da realização de testes (inclusive utilizando unidades de testes automatizadas JUnit) e do aprimoramento da interface gráfica da ferramenta.

### 3.1. Principais Componentes do Projeto

A Figura 5 apresenta um diagrama dos principais pacotes de classe da ferramenta e a relação entre eles. O pacote Entrada contém as classes que são responsáveis em processar o documento XML que contém o esquema EER de entrada do processo, transformando-os em objetos do pacote ModeloEER. O pacote ModeloEER contém classes que representam cada um dos conceitos do modelo EER.

O pacote ProcessoConversão contém todas as classes necessárias para transformar um esquema EER em um esquema lógico XML. As classes deste pacote são assistidas pelas classes do pacote RegrasConversão, que contém a implementação das regras de conversão, e também pelo pacote ModeloXML, que contém classes que representam os conceitos do modelo lógico XML.

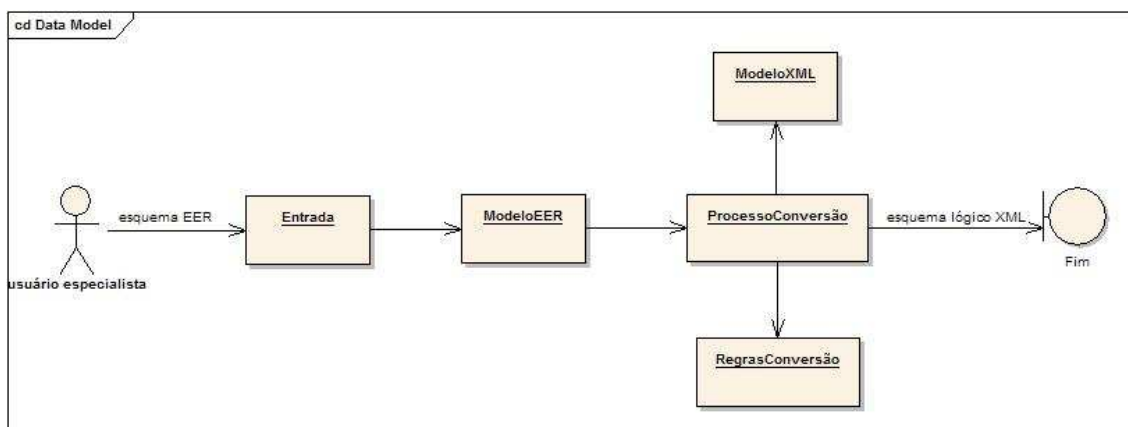


Figura 5. Principais componentes do projeto da ferramenta.

### 3.2. Exemplo da Aplicação do Processo de Conversão

Esta seção apresenta a conversão de um esquema conceitual através da ferramenta desenvolvida. A Figura 6 apresenta o diagrama EER do domínio de Notas Fiscais gerado pela ferramenta BrModelo [Candido 2005].

O esquema conceitual é passado na forma de um documento XML como entrada a ferramenta EER-XML através da tela apresentada pela Figura 7. Opcionalmente, um documento XSL pode ser informado para a transformação do documento XML de entrada no padrão aceito pela ferramenta EER-XML. Na seqüência, uma das entidades do esquema conceitual deve ser informada como Entidade Principal. Uma vez informados os dados do esquema conceitual, dá-se início ao processo automático de conversão. A Figura 8 apresenta o esquema lógico gerado pela ferramenta, correspondendo ao esquema EER apresentado pela Figura 6.

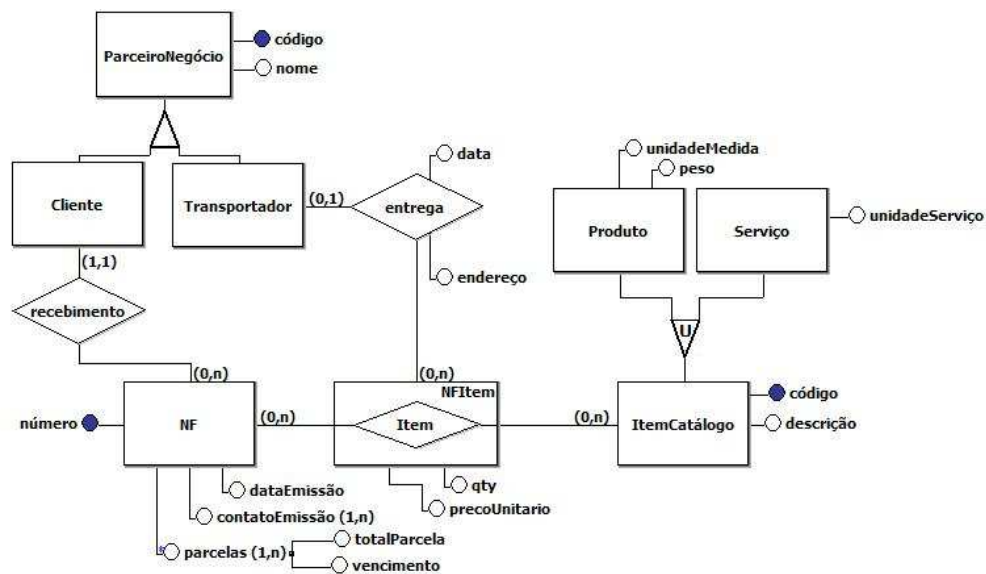


Figura 6. Esquema EER de entrada.

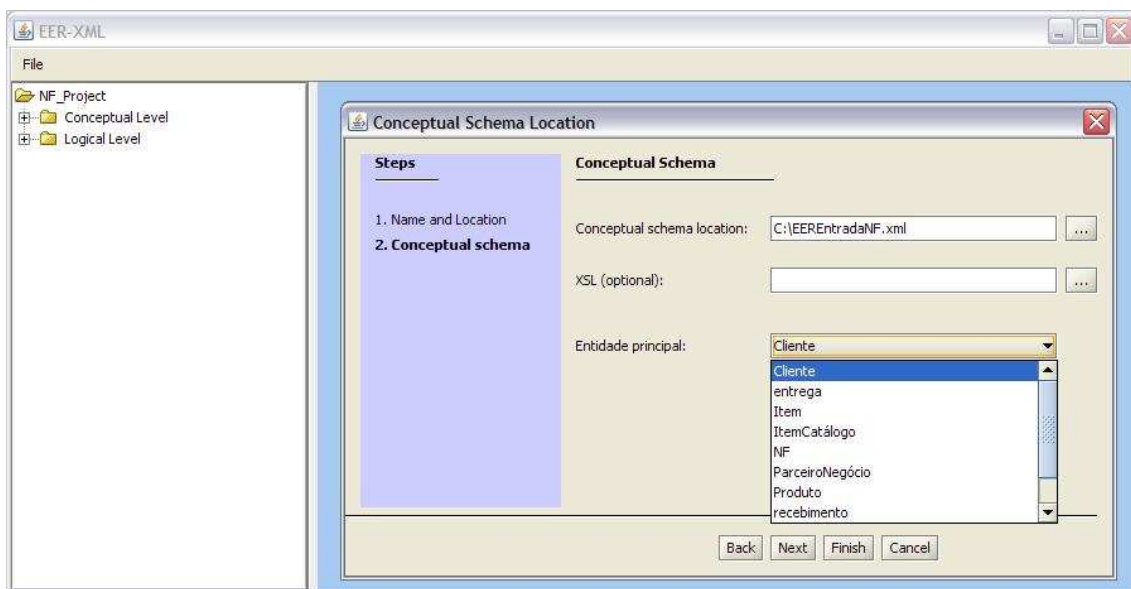


Figura 7. Entrada de dados para início da conversão.

No processo de conversão a generalização e a categoria são convertidas primeiramente. A generalização total e disjunta estabelecida pela superclasse *ParceiroNegocio* não pode ser convertida pela *Regra 1* devido a existência de relacionamentos específicos com as subclasses *Cliente* e *Transportador*. Entretanto a *Regra 2* se mostra adequada para a representação desta generalização considerando a restrição de totalidade, pois a superclasse sempre é especializada em uma de suas subclasses. Quanto à categoria estabelecida pela entidade *ItemCatálogo*, a *Regra 1* foi aplicada pois nenhuma restrição foi encontrada para a aplicação desta regra.

Os relacionamentos EER são convertidos começando pela entidade principal, neste caso indicada como sendo a entidade *Cliente*. A partir dela, o relacionamento *recebimento* é convertido aplicando-se a *Regra 6*, fazendo com que o elemento *NF* se torne sub-elemento de *Cliente*. Em seguida, a entidade associativa *NFIItem* e o

relacionamento *entrega* são convertidos através da *Regra 7*, onde relacionamentos de referência são estabelecidos entre as entidades participantes *ItemCatálogo* e *Transportador*.

Finalmente, um elemento *root* é criado, e relacionamentos hierárquicos são estabelecidos entre o elemento *root* e os elementos que não constituem destino para qualquer outro relacionamento hierárquico, neste caso: *Cliente*, *Transportador* e *ItemCatálogo*. Após este passo, é criado um documento XML que contém o esquema lógico XML gerado pela conversão. O usuário pode visualizar o esquema lógico gerado através de um diagrama exibido pela ferramenta, semelhante ao apresentado na Figura 8.

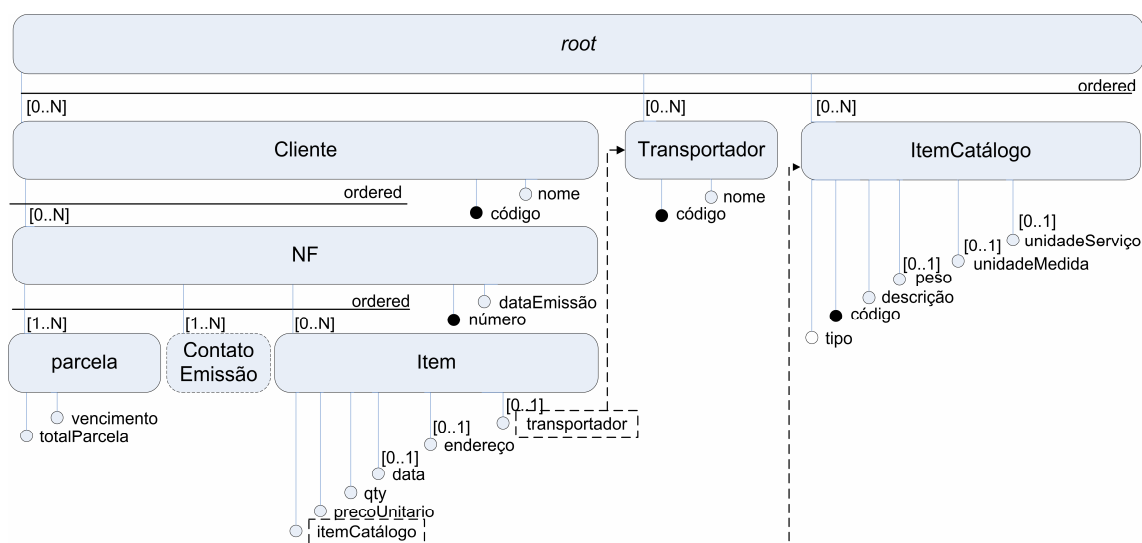


Figura 8. Esquema lógico XML final.

#### 4. Considerações Finais

Este artigo apresenta uma ferramenta de apoio ao projeto lógico de BD XML, específica para a tradução de esquemas conceituais EER em esquemas lógicos XML. A ferramenta implementa um processo de conversão que considera todas as possíveis restrições e construções do modelo EER, gerando esquemas lógicos XML capazes de serem traduzidos para qualquer linguagem de definição de esquemas XML. A principal contribuição da ferramenta, certamente, é a validação (e posterior aplicação prática) deste processo de conversão.

O projeto da ferramenta encontra-se em fase de testes e aprimoramentos na interface gráfica. Para a realização dos testes serão aplicados diversos estudos de caso reais de projeto de BD XML para, ocasionalmente, levantar pontos do projeto que necessitam de refinamento. Trabalhos futuros incluem a expansão da ferramenta para considerar atividades de projeto físico de um BD XML, bem como uma análise de usabilidade da interface gráfica, de modo a torná-la mais agradável e eficaz para o usuário. Além disto, considera-se a possibilidade de implementação de um processo de conversão semi-automático, permitindo a intervenção do usuário especialista durante o processo na escolha das estratégias de conversão para cada porção do esquema conceitual.

Um estudo, que se encontra em andamento, diz respeito à coleta de estimativas referentes às principais operações que são realizadas sobre o banco de dados XML. Esta coleta deve informar quais são as operações mais relevantes, a frequência de execução destas operações, e quais são as entidades e/ou relacionamentos envolvidos. Pretende-se, com estas informações, gerar um melhor encadeamento dos elementos do esquema lógico, minimizando a complexidade para recuperar informações durante a execução destas operações.

## Referências

- Batini, C., Ceri, S. and Navathe, S. B. (1992). *Conceptual Database Design: An Entity-Relationship Approach*, Benjamin/Cummings Publishing Company.
- Candido, C. H. *Aprendizagem em Banco de Dados: Implementação de Ferramenta de Modelagem ER*. Monografia de Especialização. Disponível em: <http://www.grupobd.inf.ufsc.br/bibliografiaGBD/especializacoes/Especializacao-CarlosCandido-FerramentaModelagemER-2005>. Último acesso: dezembro de 2007.
- Chang, E., Dillon, T. and Feng, L. (2002) “A Semantic Network-Based Design Methodology for XML Documents”, In: *ACM TOIS*, p.390-421.
- Choi, M., Lim, J. and Joo, K. (2003) “Developing a Unified Design Methodology Based on Extended Entity-Relationship Model for XML”, In: *International Conference on Computational Science*, p. 920-929.
- Dobbie, G., Xiaoying, W., Ling, T. and Lee, M. (2001) “ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data”, In: *Technical Report TR21/00*, National University of Singapore.
- Embley, D., Liddle, S. and Kamha, S. (2004) “Enterprise Modeling with Conceptual XML”, In: *International Conference on Conceptual Modeling*, p. 150-165.
- Jdom. (2007) “JDOM: Documentation” Disponível em: <http://www.jdom.org/downloads/docs.html>. Último Acesso: 29 set. 2007.
- Jgoodies. (2007) “JGOODIES: Documentation”, Disponível em: <http://www.jgoodies.org/>. Último Acesso: 30 nov. 2007.
- Liu, H., Lu, Y. and Yang, Q. (2006) “XML Conceptual Modeling with XUML”, In: *International Conference on Software Engineering*, pp. 973-976.
- Mello, R. (2003) “Gerenciamento de Dados XML”, In: *V Escola de Informática Norte da SBC*, v.1, p. 15-34.
- Mok, W. Y. and Embley, D. W. (2006) “Generating Compact Redundancy-Free XML Documents from Conceptual-Model Hypergraphs”, In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1082-1096.
- Pigozzo, P. e Quintarelli, E. (2005) “An algorithm for generating XML Schemas from ER Schemas”, In: *Italian Symposium on Advanced Database Systems*, pp. 192-199.
- Schroeder, R. e Mello, R. S. (2008) “Conversion of Generalization Hierarchies and Union Types from Extended Entity-Relationship Model to an XML Logical Model”, In: *23<sup>rd</sup> Annual ACM Symposium on Applied Computing* (to appear).

*Anexo II - Documento XML de entrada - Estudio de caso*

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<MER>
  <Informacoes>
    <Posicao Left="0" Top="0"/>
    <TotalItens Valor="124"/>
    <Tipo Valor="CONCEITUAL"/>
    <Versao Valor="1.0.0"/>
  </Informacoes>
  <Entidades>
    <Entidade nome="Empregado" id="1">
      <Left Valor="360"/>
      <Top Valor="100"/>
      <Width Valor="102"/>
      <Height Valor="66"/>
      <Fonte default="-1"/>
      <Atributos>
        <Atributo nome="CPF" id="37">
          <Left Valor="285"/>
          <Top Valor="114"/>
          <Width Valor="47"/>
          <Height Valor="16"/>
          <Fonte default="-1"/>
          <Atributos/>
          <AtributosOcultos/>
          <Dicionario></Dicionario>
          <Nula Valor="0"/>
          <Observacao></Observacao>
          <Anexos/>
          <MaxCard Valor="1"/>
          <MinCard Valor="1"/>
          <Composto Valor="0"/>
          <Identificador Valor="-1"/>
          <Tipo Valor="Texto(1)"/>
          <Multivalorado Valor="0"/>
          <Orientacao Valor="2"/>
          <TamAuto Valor="-1"/>
          <Desvio Valor="10"/>
          <BarraID Valor="39"/>
          <Ligacoes>
            <Ligacao Destino_ID="1">
              <MostraCardinalidade Valor="0" Card_id="38"/>
              <Cardinalidades Cardinalidade="4"/>
              <Orientacao Valor="0"/>
              <Fracas Valor="0"/>
            </Ligacao>
          </Ligacoes>
        </Atributo>
        <Atributo nome="Contato" id="43">
          <Left Valor="231"/>
          <Top Valor="136"/>
          <Width Valor="102"/>
          <Height Valor="16"/>
          <Fonte default="-1"/>
          <Atributos/>
          <AtributosOcultos/>
          <Dicionario></Dicionario>
          <Nula Valor="0"/>
          <Observacao></Observacao>
          <Anexos/>
        </Atributo>
      </Atributos>
    </Entidade>
  </Entidades>
</MER>

```

```

<MaxCard Valor="21"/>
<MinCard Valor="1"/>
<Composto Valor="0"/>
<Identificador Valor="0"/>
<Tipo Valor="Texto(1)"/>
<Multivalorado Valor="-1"/>
<Orientacao Valor="2"/>
<TamAuto Valor="-1"/>
<Desvio Valor="10"/>
<BarraID Valor="45"/>
<Ligacoes>
  <Ligacao Destino_ID="1">
    <MostraCardinalidade Valor="0" Card_id="44"/>
    <Cardinalidades Cardinalidade="4"/>
    <Orientacao Valor="0"/>
    <Fraca Valor="0"/>
  </Ligacao>
</Ligacoes>
</Atributo>
</Atributos>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<AutoRelacoes AutoRelacionado="0"/>
<Especializacoes ehEsp="-1">
  <Especializacao nome="Esp_1" id="2">
    <Left Valor="388"/>
    <Top Valor="205"/>
    <Width Valor="50"/>
    <Height Valor="31"/>
    <Fonte default="0">
      <FonteNome Valor="Tahoma"/>
      <FonteTamanho Valor="8"/>
      <FonteEstilo Valor="[fsBold,fsItalic]"/>
      <FonteCor Valor="32896"/>
    </Fonte>
  </Especializacao>
</Especializacoes>
<Atributos/>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<Parcial Valor="0"/>
<Disjoint Valor="-1"/>
<Ligacoes>
  <Ligacao Destino_ID="1">
    <MostraCardinalidade Valor="0" Card_id="3"/>
    <Cardinalidades Cardinalidade="4"/>
    <Orientacao Valor="1"/>
    <Fraca Valor="0"/>
  </Ligacao>
  <Ligacao Destino_ID="4">
    <MostraCardinalidade Valor="0" Card_id="6"/>
    <Cardinalidades Cardinalidade="4"/>
    <Orientacao Valor="1"/>
    <Fraca Valor="0"/>
  </Ligacao>
</Ligacoes>

```

```

        <Ligacao Destino_ID="5">
            <MostraCardinalidade Valor="0" Card_id="7"/>
            <Cardinalidades Cardinalidade="4"/>
            <Orientacao Valor="1"/>
            <Fraca Valor="0"/>
        </Ligacao>
    </Ligacoes>
</Especializacao>
</Especializacoes>
</Entidade>
<Entidade nome="Engenheiro" id="4">
    <Left Valor="265"/>
    <Top Valor="299"/>
    <Width Valor="102"/>
    <Height Valor="66"/>
    <Fonte default="-1"/>
    <Atributos>
        <Atributo nome="CREA" id="64">
            <Left Valor="195"/>
            <Top Valor="324"/>
            <Width Valor="56"/>
            <Height Valor="16"/>
            <Fonte default="-1"/>
            <Atributos/>
            <AtributosOcultos/>
            <Dicionario></Dicionario>
            <Nula Valor="0"/>
            <Observacao></Observacao>
            <Anexos/>
            <MaxCard Valor="1"/>
            <MinCard Valor="1"/>
            <Composto Valor="0"/>
            <Identificador Valor="0"/>
            <Tipo Valor="Texto(1)"/>
            <Multivalorado Valor="0"/>
            <Orientacao Valor="2"/>
            <TamAuto Valor="-1"/>
            <Desvio Valor="10"/>
            <BarraID Valor="66"/>
            <Ligacoes>
                <Ligacao Destino_ID="4">
                    <MostraCardinalidade Valor="0" Card_id="65"/>
                    <Cardinalidades Cardinalidade="4"/>
                    <Orientacao Valor="0"/>
                    <Fraca Valor="0"/>
                </Ligacao>
            </Ligacoes>
        </Atributo>
    </Atributos>
    <AtributosOcultos/>
    <Dicionario></Dicionario>
    <Nula Valor="0"/>
    <Observacao></Observacao>
    <Anexos/>
    <AutoRelacoes AutoRelacionado="0"/>
    <Especializacoes ehEsp="0"/>
</Entidade>
<Entidade nome="Motorista" id="5">
    <Left Valor="460"/>

```



```

<Top Valor="294"/>
<Width Valor="102"/>
<Height Valor="66"/>
<Fonte default="-1"/>
<Atributos>
  <Atributo nome="CNH" id="61">
    <Left Valor="577"/>
    <Top Valor="319"/>
    <Width Valor="49"/>
    <Height Valor="16"/>
    <Fonte default="-1"/>
    <Atributos/>
    <AtributosOcultos/>
    <Dicionario></Dicionario>
    <Nula Valor="0"/>
    <Observacao></Observacao>
    <Anexos/>
    <MaxCard Valor="1"/>
    <MinCard Valor="1"/>
    <Composto Valor="0"/>
    <Identificador Valor="0"/>
    <Tipo Valor="Texto(1)"/>
    <Multivalorado Valor="0"/>
    <Orientacao Valor="3"/>
    <TamAuto Valor="-1"/>
    <Desvio Valor="10"/>
    <BarraID Valor="63"/>
    <Ligacoes>
      <Ligacao Destino_ID="5">
        <MostraCardinalidade Valor="0" Card_id="62"/>
        <Cardinalidades Cardinalidade="4"/>
        <Orientacao Valor="0"/>
        <Fraca Valor="0"/>
      </Ligacao>
    </Ligacoes>
  </Atributo>
</Atributos>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<AutoRelacoes AutoRelacionado="0"/>
<Especializacoes ehEsp="0"/>
</Entidade>
<Entidade nome="Projeto" id="8">
  <Left Valor="265"/>
  <Top Valor="535"/>
  <Width Valor="102"/>
  <Height Valor="66"/>
  <Fonte default="-1"/>
  <Atributos>
    <Atributo nome="Codigo" id="70">
      <Left Valor="403"/>
      <Top Valor="549"/>
      <Width Valor="65"/>
      <Height Valor="16"/>
      <Fonte default="-1"/>
    </Atributo>
  </Atributos>

```

```

<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<MaxCard Valor="1"/>
<MinCard Valor="1"/>
<Composto Valor="0"/>
<Identificador Valor="-1"/>
<Tipo Valor="Texto(1)"/>
<Multivalorado Valor="0"/>
<Orientacao Valor="3"/>
<TamAuto Valor="-1"/>
<Desvio Valor="10"/>
<BarraID Valor="72"/>
<Ligacoes>
  <Ligacao Destino_ID="8">
    <MostraCardinalidade Valor="0" Card_id="71"/>
    <Cardinalidades Cardinalidade="4"/>
    <Orientacao Valor="0"/>
    <Fraca Valor="0"/>
  </Ligacao>
</Ligacoes>
</Atributo>
<Atributo nome="Prazo" id="76">
  <Left Valor="403"/>
  <Top Valor="571"/>
  <Width Valor="89"/>
  <Height Valor="16"/>
  <Fonte default="-1"/>
  <Atributos/>
  <AtributosOcultos/>
  <Dicionario></Dicionario>
  <Nula Valor="0"/>
  <Observacao></Observacao>
  <Anexos/>
  <MaxCard Valor="1"/>
  <MinCard Valor="0"/>
  <Composto Valor="0"/>
  <Identificador Valor="0"/>
  <Tipo Valor="Texto(1)"/>
  <Multivalorado Valor="-1"/>
  <Orientacao Valor="3"/>
  <TamAuto Valor="-1"/>
  <Desvio Valor="10"/>
  <BarraID Valor="78"/>
  <Ligacoes>
    <Ligacao Destino_ID="8">
      <MostraCardinalidade Valor="0" Card_id="77"/>
      <Cardinalidades Cardinalidade="4"/>
      <Orientacao Valor="0"/>
      <Fraca Valor="0"/>
    </Ligacao>
  </Ligacoes>
</Atributo>
</Atributos>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>

```

```

<Observacao></Observacao>
<Anexos/>
<AutoRelacoes AutoRelacionado="0"/>
<Especializacoes ehEsp="0"/>
</Entidade>
<Entidade nome="Banco" id="12">
  <Left Valor="677"/>
  <Top Valor="287"/>
  <Width Valor="102"/>
  <Height Valor="66"/>
  <Fonte default="-1"/>
  <Atributos>
    <Atributo nome="CNPJ" id="67">
      <Left Valor="814"/>
      <Top Valor="312"/>
      <Width Valor="54"/>
      <Height Valor="16"/>
      <Fonte default="-1"/>
      <Atributos/>
      <AtributosOcultos/>
      <Dicionario></Dicionario>
      <Nula Valor="0"/>
      <Observacao></Observacao>
      <Anexos/>
      <MaxCard Valor="1"/>
      <MinCard Valor="1"/>
      <Composto Valor="0"/>
      <Identificador Valor="-1"/>
      <Tipo Valor="Texto(1)"/>
      <Multivalorado Valor="0"/>
      <Orientacao Valor="3"/>
      <TamAuto Valor="-1"/>
      <Desvio Valor="10"/>
      <BarraID Valor="69"/>
      <Ligacoes>
        <Ligacao Destino_ID="12">
          <MostraCardinalidade Valor="0" Card_id="68"/>
          <Cardinalidades Cardinalidade="4"/>
          <Orientacao Valor="0"/>
          <Fraca Valor="0"/>
        </Ligacao>
      </Ligacoes>
    </Atributo>
  </Atributos>
  <AtributosOcultos/>
  <Dicionario></Dicionario>
  <Nula Valor="0"/>
  <Observacao></Observacao>
  <Anexos/>
  <AutoRelacoes AutoRelacionado="0"/>
  <Especializacoes ehEsp="0"/>
</Entidade>
<Entidade nome="Empresa" id="17">
  <Left Valor="677"/>
  <Top Valor="100"/>
  <Width Valor="102"/>
  <Height Valor="66"/>
  <Fonte default="-1"/>
  <Atributos>

```

```

<Atributo nome="CNPJ" id="49">
  <Left Valor="814"/>
  <Top Valor="136"/>
  <Width Valor="54"/>
  <Height Valor="16"/>
  <Fonte default="-1"/>
  <Atributos/>
  <AtributosOcultos/>
  <Dicionario></Dicionario>
  <Nula Valor="0"/>
  <Observacao></Observacao>
  <Anexos/>
  <MaxCard Valor="1"/>
  <MinCard Valor="1"/>
  <Composto Valor="0"/>
  <Identificador Valor="-1"/>
  <Tipo Valor="Texto(1)"/>
  <Multivalorado Valor="0"/>
  <Orientacao Valor="3"/>
  <TamAuto Valor="-1"/>
  <Desvio Valor="10"/>
  <BarraID Valor="51"/>
  <Ligacoes>
    <Ligacao Destino_ID="17">
      <MostraCardinalidade Valor="0" Card_id="50"/>
      <Cardinalidades Cardinalidade="4"/>
      <Orientacao Valor="0"/>
      <Fracas Valor="0"/>
    </Ligacao>
  </Ligacoes>
</Atributo>
<Atributo nome="Razao_Social" id="52">
  <Left Valor="813"/>
  <Top Valor="114"/>
  <Width Valor="102"/>
  <Height Valor="16"/>
  <Fonte default="-1"/>
  <Atributos/>
  <AtributosOcultos/>
  <Dicionario></Dicionario>
  <Nula Valor="0"/>
  <Observacao></Observacao>
  <Anexos/>
  <MaxCard Valor="1"/>
  <MinCard Valor="1"/>
  <Composto Valor="0"/>
  <Identificador Valor="0"/>
  <Tipo Valor="Texto(1)"/>
  <Multivalorado Valor="0"/>
  <Orientacao Valor="3"/>
  <TamAuto Valor="-1"/>
  <Desvio Valor="10"/>
  <BarraID Valor="54"/>
  <Ligacoes>
    <Ligacao Destino_ID="17">
      <MostraCardinalidade Valor="0" Card_id="53"/>
      <Cardinalidades Cardinalidade="4"/>
      <Orientacao Valor="0"/>
      <Fracas Valor="0"/>
    </Ligacao>
  </Ligacoes>
</Atributo>

```

```

        </Ligacao>
    </Ligacoes>
    </Atributo>
</Atributos>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<AutoRelacoes AutoRelacionado="0"/>
<Especializacoes ehEsp="0"/>
</Entidade>
</Entidades>
    <Relacoes>
    <Relacao nome="trabalha" id="28">
        <Left Valor="522"/>
        <Top Valor="108"/>
        <Width Valor="102"/>
        <Height Valor="51"/>
        <Fonte default="-1"/>
        <Atributos>
            <Atributo nome="Horas" id="46">
                <Left Valor="583"/>
                <Top Valor="85"/>
                <Width Valor="60"/>
                <Height Valor="16"/>
                <Fonte default="-1"/>
                <Atributos/>
                <AtributosOcultos/>
                <Dicionario></Dicionario>
                <Nula Valor="0"/>
                <Observacao></Observacao>
                <Anexos/>
                <MaxCard Valor="1"/>
                <MinCard Valor="1"/>
                <Composto Valor="0"/>
                <Identificador Valor="0"/>
                <Tipo Valor="Texto(1)"/>
                <Multivalorado Valor="0"/>
                <Orientacao Valor="3"/>
                <TamAuto Valor="-1"/>
                <Desvio Valor="10"/>
                <BarraID Valor="48"/>
                <Ligacoes>
                    <Ligacao Destino_ID="28">
                        <MostraCardinalidade Valor="0" Card_id="47"/>
                        <Cardinalidades Cardinalidade="4"/>
                        <Orientacao Valor="0"/>
                        <Fraca Valor="0"/>
                    </Ligacao>
                </Ligacoes>
            </Atributo>
        </Atributos>
        <AtributosOcultos/>
        <Dicionario></Dicionario>
        <Nula Valor="0"/>
        <Observacao></Observacao>
        <Anexos/>
    </Ligacoes>

```

```

<Ligacao Destino_ID="1">
  <MostraCardinalidade Valor="-1" Card_id="29"/>
  <Cardinalidades Cardinalidade="3">
    <Cardinalidade nome="" id="29">
      <Left Valor="462"/>
      <Top Valor="118"/>
      <Width Valor="40"/>
      <Height Valor="20"/>
      <Fonte default="-1"/>
      <Atributos/>
      <AtributosOcultos/>
      <Dicionario></Dicionario>
      <Nula Valor="0"/>
      <Observacao></Observacao>
      <Anexos/>
      <Cor Valor="15780518"/>
      <TamAuto Valor="-1"/>
      <Tipo Valor="0"/>
      <TextAlin Valor="0"/>
      <Card Valor="3"/>
      <Fixa Valor="0"/>
      <Ligacoes/>
    </Cardinalidade>
  </Cardinalidades>
  <Orientacao Valor="0"/>
  <Fraca Valor="0"/>
</Ligacao>
<Ligacao Destino_ID="17">
  <MostraCardinalidade Valor="-1" Card_id="30"/>
  <Cardinalidades Cardinalidade="1">
    <Cardinalidade nome="" id="30">
      <Left Valor="639"/>
      <Top Valor="118"/>
      <Width Valor="40"/>
      <Height Valor="20"/>
      <Fonte default="-1"/>
      <Atributos/>
      <AtributosOcultos/>
      <Dicionario></Dicionario>
      <Nula Valor="0"/>
      <Observacao></Observacao>
      <Anexos/>
      <Cor Valor="15780518"/>
      <TamAuto Valor="-1"/>
      <Tipo Valor="0"/>
      <TextAlin Valor="0"/>
      <Card Valor="1"/>
      <Fixa Valor="0"/>
      <Ligacoes/>
    </Cardinalidade>
  </Cardinalidades>
  <Orientacao Valor="0"/>
  <Fraca Valor="0"/>
</Ligacao>
</Ligacoes>
</Relacao>
<Relacao nome="participa" id="31">
  <Left Valor="265"/>
  <Top Valor="427"/>

```

```

<Width Valor="102"/>
<Height Valor="51"/>
<Fonte default="-1"/>
<Atributos/>
<AtributosOcultos/>
<Dicionario></Dicionario>
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<Ligacoes>
  <Ligacao Destino_ID="8">
    <MostraCardinalidade Valor="-1" Card_id="32"/>
    <Cardinalidades Cardinalidade="1">
      <Cardinalidade nome="" id="32">
        <Left Valor="316"/>
        <Top Valor="520"/>
        <Width Valor="40"/>
        <Height Valor="20"/>
        <Fonte default="-1"/>
        <Atributos/>
        <AtributosOcultos/>
        <Dicionario></Dicionario>
        <Nula Valor="0"/>
        <Observacao></Observacao>
        <Anexos/>
        <Cor Valor="15780518"/>
        <TamAuto Valor="-1"/>
        <Tipo Valor="0"/>
        <TextAlin Valor="0"/>
        <Card Valor="1"/>
        <Fixa Valor="0"/>
        <Ligacoes/>
      </Cardinalidade>
    </Cardinalidades>
    <Orientacao Valor="0"/>
    <Fraca Valor="0"/>
  </Ligacao>
  <Ligacao Destino_ID="4">
    <MostraCardinalidade Valor="-1" Card_id="33"/>
    <Cardinalidades Cardinalidade="1">
      <Cardinalidade nome="" id="33">
        <Left Valor="316"/>
        <Top Valor="366"/>
        <Width Valor="40"/>
        <Height Valor="20"/>
        <Fonte default="-1"/>
        <Atributos/>
        <AtributosOcultos/>
        <Dicionario></Dicionario>
        <Nula Valor="0"/>
        <Observacao></Observacao>
        <Anexos/>
        <Cor Valor="15780518"/>
        <TamAuto Valor="-1"/>
        <Tipo Valor="0"/>
        <TextAlin Valor="0"/>
        <Card Valor="1"/>
        <Fixa Valor="0"/>
        <Ligacoes/>
      </Cardinalidade>
    </Cardinalidades>
  </Ligacao>

```

```

        </Cardinalidade>
    </Cardinalidades>
    <Orientacao Valor="0"/>
    <Fraca Valor="0"/>
    </Ligacao>
</Ligacoes>
</Relacao>
<Relacao nome="sociedade" id="34">
    <Left Valor="677"/>
    <Top Valor="202"/>
    <Width Valor="102"/>
    <Height Valor="51"/>
    <Fonte default="-1"/>
    <Atributos/>
    <AtributosOcultos/>
    <Dicionario></Dicionario>
    <Nula Valor="0"/>
    <Observacao></Observacao>
    <Anexos/>
    <Ligacoes>
        <Ligacao Destino_ID="17">
            <MostraCardinalidade Valor="-1" Card_id="35"/>
            <Cardinalidades Cardinalidade="3">
                <Cardinalidade nome="" id="35">
                    <Left Valor="728"/>
                    <Top Valor="167"/>
                    <Width Valor="40"/>
                    <Height Valor="20"/>
                    <Fonte default="-1"/>
                    <Atributos/>
                    <AtributosOcultos/>
                    <Dicionario></Dicionario>
                    <Nula Valor="0"/>
                    <Observacao></Observacao>
                    <Anexos/>
                    <Cor Valor="15780518"/>
                    <TamAuto Valor="-1"/>
                    <Tipo Valor="0"/>
                    <TextAlin Valor="0"/>
                    <Card Valor="3"/>
                    <Fixa Valor="0"/>
                    <Ligacoes/>
                </Cardinalidade>
            </Cardinalidades>
            <Orientacao Valor="0"/>
            <Fraca Valor="0"/>
        </Ligacao>
    <Ligacao Destino_ID="12">
        <MostraCardinalidade Valor="-1" Card_id="36"/>
        <Cardinalidades Cardinalidade="1">
            <Cardinalidade nome="" id="36">
                <Left Valor="728"/>
                <Top Valor="272"/>
                <Width Valor="40"/>
                <Height Valor="20"/>
                <Fonte default="-1"/>
                <Atributos/>
                <AtributosOcultos/>
                <Dicionario></Dicionario>
            </Cardinalidade>
        </Cardinalidades>
        <Orientacao Valor="0"/>
        <Fraca Valor="0"/>
    </Ligacao>
</Ligacoes>

```



```
<Nula Valor="0"/>
<Observacao></Observacao>
<Anexos/>
<Cor Valor="15780518"/>
<TamAuto Valor="-1"/>
<Tipo Valor="0"/>
<TextAlin Valor="0"/>
<Card Valor="1"/>
<Fixa Valor="0"/>
<Ligacoes/>
  </Cardinalidade>
</Cardinalidades>
<Orientacao Valor="0"/>
<Fraca Valor="0"/>
</Ligacao>
</Ligacoes>
</Relacao>
</Relacoes>
  <EntAssoss>
  </EntAssoss>
  <Texto>
  </Texto>
</MER>
```

***Anexo III - Documento XML gerado da conversão do documento XML de entrada - Estudo de caso***

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="root" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Projeto" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:attribute name="Prazo" use="required" type="xs:string"/>
            <xs:attribute name="Codigo" use="required" type="xs:ID"/>
            <xs:element name="participa" minOccurs="1" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:attribute name="EmpregadoCode" use="required" type="xs:IDREF"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Empregado" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:attribute name="CNH" use="optional" type="xs:string"/>
            <xs:attribute name="CPF" use="required" type="xs:ID"/>
            <xs:attribute name="Type" use="required" type="xs:string"/>
            <xs:attribute name="CREA" use="optional" type="xs:string"/>
            <xs:element name="Contato" minOccurs="1" maxOccurs="21">
              <xs:simpleType>
                <xs:restriction base="xsd:string"/>
              </xs:simpleType>
            </xs:element>
            <xs:element name="trabalha" minOccurs="1" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:attribute name="EmpresaCode" use="required" type="xs:IDREF"/>
                  <xs:attribute name="Horas" use="required" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Empresa" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:attribute name="CNPJ" use="required" type="xs:ID"/>
            <xs:attribute name="Razao_Social" use="required" type="xs:string"/>
            <xs:attribute name="CNPJBanco" use="required" type="xs:ID"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```