

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**  
**CENTRO TECNOLÓGICO**  
**DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**  
**CURSO DE SISTEMAS DE INFORMAÇÃO**

**Ferramenta de conversão de esquemas *XML Schema* para esquemas  
relacionais**

**Crislane Spricigo da Silva**

Trabalho de conclusão de curso submetido à  
Universidade Federal de Santa Catarina  
como parte dos requisitos para obtenção do  
grau de Bacharel em Sistemas de  
Informação.

Orientador: Prof. Ronaldo dos Santos Mello,  
Dr.

**Florianópolis, SC.**

**Junho de 2008.**

**Crislane Spricigo da Silva**

**Ferramenta de conversão de esquemas *XML Schema* para esquemas relacionais**

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

**Orientador:**

---

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

Universidade Federal de Santa Catarina

[ronaldo@inf.ufsc.br](mailto:ronaldo@inf.ufsc.br)

**Banca examinadora**

---

**Prof. Frank Augusto Siqueira, Dr.**

Universidade Federal de Santa Catarina

[frank@inf.ufsc.br](mailto:frank@inf.ufsc.br)

---

**Prof. Fernando Augusto da Silva Cruz**

Universidade Federal de Santa Catarina

[cruz@inf.ufsc.br](mailto:cruz@inf.ufsc.br)

## SUMÁRIO

LISTA DE FIGURAS .....	4
LISTA DE ABREVIATURAS.....	5
RESUMO .....	6
ABSTRACT .....	7
1 INTRODUÇÃO.....	8
1.1 VISÃO GERAL.....	8
1.2 OBJETIVOS.....	9
1.2.1 Geral .....	9
1.2.2 Específicos.....	9
1.3 JUSTIFICATIVA .....	10
1.4 METODOLOGIA.....	11
1.5 ESTRUTURA DO TRABALHO .....	11
2 XML e XML Schema.....	13
2.1 Visão Geral sobre XML .....	13
2.2 Linguagem <i>XML Schema</i> .....	14
2.2.1 Principais componentes da Linguagem <i>XML Schema</i> .....	15
2.2.2 Conceitos <i>XML Schema</i> Considerados na Ferramenta Proposta.....	17
3 TRABALHOS RELACIONADOS .....	19
3.1 Mapeamento de definições <i>XML Schema</i> para SQL:1999 .....	19
3.2 Geração automática de esquemas relacionais a partir de esquemas XML.....	20
3.2.1 Mapping Definition Framework – MDF .....	20
3.3 Conclusão .....	21
4 A FERRAMENTA .....	23
4.1 Funcionamento Geral .....	23
4.2 Construção da Ferramenta XRCONVERSION.....	24
4.2.1 Metodologia.....	24
4.2.2 Diagramas.....	25
4.2.2.1 Casos de uso .....	25
4.2.2.2 Classes do sistema .....	26
4.2.3 Tecnologias utilizadas .....	27
4.3 Módulos.....	28
4.3.1 Módulo de Pré-processamento .....	28
4.3.1.1 Incorporação de conteúdo externo.....	28
4.3.1.2 Tratamento de Grupos de Substituição.....	29
4.3.1.3 Tratamento de Extensões.....	31
4.3.1.4 Eliminação de Referências .....	32
4.3.2 Módulo de Mapeamento.....	33
4.4 Algoritmo .....	49
4.5 Interface gráfica.....	50
4.6 Testes de integração com o banco de dados MySQL 5.0.....	52
5 CONCLUSÃO.....	57
REFERÊNCIAS BIBLIOGRÁFICAS .....	58

## LISTA DE FIGURAS

Figura 1. Exemplo de Documento XML [XML07] .....	13
Figura 2. Exemplo de XML Schema [CRM05] .....	15
Figura 3. Conceitos considerados pela ferramenta XRCONVERSION.....	18
Figura 4. Exemplo de XML Schema .....	18
Figura 5. Funcionamento geral da ferramenta.....	24
Figura 6. Diagrama de casos de uso .....	26
Figura 7. Diagrama de classes .....	27
Figura 8. Declaração de um grupo de substituição.....	30
Figura 9. Criação de choices para tratar grupos de substituição .....	31
Figura 10. Declaração de extensão .....	31
Figura 11. Exemplo de incorporação de conteúdo estendido.....	32
Figura 12. Referência a declarações globais .....	32
Figura 13. Eliminação de referências .....	33
Figura 14. Exemplo de Mapeamento de Atributo Atômico. ....	34
Figura 15. Exemplo de Mapeamento de Atributo do Tipo List.....	35
Figura 16. Exemplo de Mapeamento de Atributo do Tipo Union.....	36
Figura 17. Exemplo de Mapeamento de Atributo do Tipo Derivado.....	37
Figura 18. Exemplo de conversão de atributos.....	38
Figura 19. Resultado da conversão.....	38
Figura 20. Exemplo de Mapeamento de Elemento Simples.....	40
Figura 21. Exemplo de Mapeamento de Atributo Derivado por Restrição .....	41
Figura 22. Exemplo de Mapeamento de Elemento tipo Union .....	43
Figura 23. Exemplo de Mapeamento de Elemento Complexo (I).....	44
Figura 24. Exemplo de Mapeamento de Elemento Complexo (II) .....	44
Figura 25. Exemplo de Mapeamento de Elemento Complexo (III) .....	45
Figura 26. Exemplo de Mapeamento de Elemento Complexo Vazio com atributos .....	46
Figura 27. Exemplo de Mapeamento de Elemento Complexo Vazio sem atributos.....	46
Figura 28. Exemplo de Mapeamento de Elemento Complexo de tipo misto .....	48
Figura 29. Exemplo de Mapeamento de Elemento AnyType .....	49
Figura 30. Algoritmo de conversão [CRM05].....	49
Figura 31. Tela principal do programa XRCONVERSION.....	51
Figura 32. Exemplo de uso da ferramenta XRCONVERSION .....	53
Figura 33. Resultado da conversão.....	54
Figura 34. Criação das tabelas no banco MySQL 5.0 .....	56

## **LISTA DE ABREVIATURAS**

<b>SGBD</b>	Sistema de Gerenciamento de Banco de Dados
<b>XML</b>	Extended Markup Language
<b>W3C</b>	World Wide Web Consortium
<b>SQL</b>	Structured Query Language
<b>JAXP</b>	Java API for XML Processing
<b>DTD</b>	Document Type Definition
<b>XSD</b>	XML Schema Definition

## RESUMO

No atual contexto tecnológico, encontramos a linguagem XML, que promove a interoperabilidade entre sistemas de informação heterogêneos. Esta comunicação de dados provenientes de diversas fontes é um pré-requisito de qualidade de *software*.

Ao escolher a linguagem XML para promover o intercâmbio de dados entre sistemas heterogêneos, é necessário determinar um esquema comum entre as aplicações relacionadas. Isto pode ser feito de diversas maneiras, uma delas é definir o esquema através da linguagem *XML Schema*.

Neste ponto, encontra-se a necessidade de fazer com que os dados que trafegam na rede sejam compatíveis com os bancos de dados dos respectivos sistemas. O modelo de dados relacional é amplamente difundido e utilizado pelos sistemas. Os gerenciadores de banco de dados (SGBDs) do modelo relacional são estáveis e com boa performance, provenientes de muitos anos de aprimoramento das diversas instituições do meio tecnológico.

Tendo em vista todos estes pontos, este trabalho de conclusão de curso propõe-se a implementar e validar uma proposta de uma dissertação de mestrado que contribuiu para a comunidade científica trazendo uma solução para a integração dos dados *XML Schema* com o modelo do banco de dados. Para isso foi construída uma ferramenta que possui como principais funções: leitura de um esquema *XML Schema*, processamento deste e fornecimento de um *script* de criação de tabelas para o modelo relacional.

## **ABSTRACT**

In the current technological context, we find that the XML language promotes the interoperability between heterogeneous information systems. This communication of data proceeding from different sources is a quality prerequisite for software in general.

When choosing language XML to promote the interchange of data between heterogeneous systems, is necessary to determine a common project between the related applications. This can be made through many methodologies; one of them is to define the project through language *XML Schema*.

In this point, the data exchanged on the net needs to be compatible with the databases of the respective systems. The relational database model is widely spread out and used by systems. The relational model database management systems (DBMS) are steady and with good performance, proceeding from many years of improvement on many institutions of Information Technology.

In this context, this course conclusion monography proposes itself to implement and to validate a masters degree monography that intended to bring a solution for the integration between *XML Schema* and relational model database. For this a tool was constructed where its main functions are: Reading and Processing a *XML Schema* and creating a script of it for the relational model database.

# 1 INTRODUÇÃO

## 1.1 VISÃO GERAL

A interoperabilidade entre diversos sistemas de informação é cada vez mais comum no meio computacional. Nas organizações que fazem uso da tecnologia da informação, ter um sistema comunicante com outros sistemas já é um pré-requisito de qualidade de *software*. O intercâmbio de dados funciona mediante a estipulação de um protocolo comum entre as partes envolvidas. Neste contexto, encontramos a linguagem XML [W3C07], que, através de sua ampla flexibilidade e extensibilidade, desempenha com eficiência as funções de importação e exportação de informações.

Quando a linguagem XML surgiu, sua principal função era a troca de informações através de documentos eletrônicos. Com o crescimento das aplicações *web*, XML passou a ser amplamente utilizada, devido à sua simplicidade e portabilidade. Atualmente, o consórcio de empresas de tecnologia conhecido mundialmente como W3C definiu como um dos propósitos da linguagem XML ser um padrão para comunicação na *Web*.

Ao escolher a linguagem XML para promover o intercâmbio de dados entre sistemas heterogêneos, é necessário determinar um esquema comum entre as aplicações relacionadas. Isto pode ser feito de diversas maneiras, uma delas, a qual também é recomendação da W3C, é definir o esquema através da linguagem *XML Schema*. Esta, por sua vez, é baseada nas mesmas regras de sintaxe dos documentos XML, e suas características permitem definir tipos de dados (simples e complexos), chaves primárias, entre outras possibilidades. Dessa forma, torna-se possível garantir a existência de um protocolo de troca de informações organizado e independente do mecanismo de persistência de dados utilizado pelos sistemas [CRM05].

Neste ponto, encontra-se a necessidade de fazer com que os dados que trafegam na rede sejam compatíveis com os bancos de dados dos respectivos sistemas. Ou seja, os dados em XML devem ser transformados para o modelo de dados utilizado pelo banco de dados e vice-versa. O modelo de dados relacional é amplamente difundido e utilizado pelos sistemas. Seus sistemas gerenciadores de banco de dados (SGBDs) são estáveis e com boa performance, provenientes de muitos anos de aprimoramento das diversas instituições do meio tecnológico. Devem ser



adotados mecanismos que exportem dados de suas tabelas no formato XML e vice-versa. A preferência é que esses mecanismos atuem de forma genérica e dinâmica, para garantir uma operação adequada de intercâmbio de dados [CRM05].

Tendo em vista todos esses pontos, este trabalho de conclusão de curso propõe-se a implementar e validar uma proposta de uma dissertação de mestrado que contribuiu para a comunidade científica trazendo uma solução para a integração dos dados XML com o modelo do banco de dados. Foi elaborada uma abordagem de mapeamento de conceitos *XML Schema* para o modelo relacional, automatizando boa parte das tarefas de desenvolvimento de sistemas comunicantes e com persistências em bancos de dados relacionais [CRM05]. Este trabalho trata de uma ferramenta que se baseia nessa abordagem, onde suas principais funções são: ler o *XML Schema*, processá-lo e fornecer um *script* de criação de tabelas para o modelo relacional.

## **1.2 OBJETIVOS**

### **1.2.1 Geral**

No atual cenário tecnológico, encontramos diversos sistemas de informação que comunicam dados de negócio através da utilização de esquemas XML. Tais dados muitas vezes necessitam ser persistidos em bancos de dados relacionais. Devido à flexibilidade e extensibilidade de dados XML, o mapeamento XML-relacional não é uma tarefa simples. Os dois modelos diferem-se na estrutura de formação e tipos de dados, sendo XML um modelo hierárquico com dados complexos e o relacional não-hierárquico com dados simples. Nesse contexto, o presente trabalho propõe uma ferramenta que automatize parte do processo de integração entre esses dois modelos de dados através da geração de um esquema relacional para um dado *XML Schema* de entrada.

### **1.2.2 Específicos**

- Fazer um estudo de *XML Schema* e seus recursos;

- Contribuir para a integração de sistemas de informação baseados em *XML Schema* e com persistência no modelo relacional;
- Estudar as regras e algoritmos de mapeamento propostos em uma dissertação de mestrado no Programa de Pós-Graduação em Ciências da Computação (PPGCC) da Universidade Federal de Santa Catarina (UFSC) em 2005 [CRM05];
- Desenvolver a ferramenta XRConversion baseada nas regras e algoritmos de mapeamento citados no objetivo anterior;
- Apresentar exemplos de uso da ferramenta.

### 1.3 JUSTIFICATIVA

A tarefa de promover a integração dos dados descritos em XML com o modelo de dados relacional é um tanto complexa, pois não é possível fazer diretamente. Os dois modelos baseiam-se em conceitos diferentes: XML possui uma estrutura hierárquica (modelo de árvore) e seus dados são complexos. Já o modelo relacional é definido como não-hierárquico e possui dados simples [CRM05]. Durante a conversão, deve-se evitar erros semânticos, onde um conceito em XML deve possuir o mesmo significado no modelo relacional.

Alguns SGBDs possuem mecanismos embutidos que se propõem a mapear o esquema XML, entretanto são linguagens proprietárias, e possuem algumas desvantagens, tais como: são específicas para cada SGBD, há um tempo adicional para conhecê-las, deixam o mapeamento livre, sem padrão a ser seguido. Escolher a alternativa desses mecanismos acarreta em ter uma elevação de custo durante o desenvolvimento.

Também foram pesquisadas propostas atuais com abordagens semelhantes à deste trabalho. Foram encontradas ferramentas robustas e com embasamento científico, mas que, de maneira geral, não permitem ao desenvolvedor usar sua experiência para gerar o esquema resultante. Outro problema encontrado nessas ferramentas é que há uma consideração parcial dos conceitos XML.

Com base nas deficiências encontradas, este trabalho consiste no desenvolvimento de uma ferramenta que tem como diferencial a análise detalhada de cada conceito do *XML Schema* e constante interação com o usuário para que o mesmo decida a melhor maneira de promover o mapeamento, facilitando o desenvolvimento e o tornando menos custoso.

## 1.4 METODOLOGIA

Para alcançar os objetivos propostos, este trabalho foi dividido nas seguintes etapas:

1. Estudo sobre a estrutura de XML e *XML Schema*;
2. Levantamento de trabalhos com propostas semelhantes;
3. Explicação da abordagem de mapeamento *XML Schema*-Relacional e implementação da ferramenta;
4. Apresentação de exemplos de conversão.

Na primeira etapa é apresentada uma breve descrição de estruturas XML e *XML Schema*, mostrando-se em que se baseiam e como tratam os dados.

A segunda etapa consiste em uma pesquisa sobre trabalhos relacionados com o tema de integração de XML com o modelo relacional, sendo possível traçar uma comparação entre esses trabalhos e verificar quais são os pontos positivos e negativos de cada um.

A terceira etapa consiste no estudo da abordagem de mapeamento que é o fundamento da ferramenta XRConversion. Para que seja possível compreender o seu funcionamento, é necessário ter o entendimento do algoritmo e de cada regra propostos.

Em seguida, há a descrição da ferramenta, que consiste na leitura e processamento de *XML Schema* e geração de uma saída para o modelo relacional. Neste ponto, há a explicação da interface gráfica.

Na quarta etapa, são mostrados exemplos de *schemas* que podem ser convertidos.

## 1.5 ESTRUTURA DO TRABALHO

Este trabalho está dividido da seguinte forma:

- Capítulo 1: Introdução – apresenta uma breve contextualização das áreas envolvidas no tema proposto e o trabalho de forma geral, discorrendo sobre seus objetivos gerais e específicos, justificativa, metodologia e organização;

- Capítulo 2: Fundamentação Teórica – apresenta os conceitos de XML e *XML Schema*, especificando seus elementos e atributos principais;
- Capítulo 3: Trabalhos Relacionados – neste Capítulo são demonstradas propostas atuais relacionadas ao trabalho em questão, tecendo-se comentários sobre elas;
- Capítulo 4: A Ferramenta XRCONVERSION – descreve a arquitetura, composição e funcionamento da ferramenta;
- Capítulo 5: Conclusões – apresenta as conclusões extraídas do período no qual foi desenvolvido este projeto. São apontados alguns aspectos do trabalho, os quais podem ser incorporados em tarefas futuras.

## 2 XML E XML SCHEMA

### 2.1 Visão Geral sobre XML

A World Wide Web Consortium (W3C) é um consórcio de empresas de tecnologia que desenvolve padrões para a criação e a interpretação dos conteúdos para a *Web*. Os padrões estipulados pela W3C permitem uma comunicação eficiente entre pessoas, aplicações e serviços na rede [W3C07]. Entre esses padrões, encontra-se a eXtensible Markup Language (XML), uma linguagem de marcação usada para descrição de informações.

XML é uma linguagem flexível e robusta para descrição de dados, pois não possui elementos pré-definidos. O responsável pelo desenvolvimento é quem define o conjunto de elementos que esteja relacionado com os dados do domínio no qual a aplicação está inserida.

Os documentos XML são baseados em dois principais conceitos de estruturação: elementos e atributos. A Figura 1 mostra um exemplo de documento XML. Os elementos são identificados por *tags* de início e *tags* de fim, que delimitam as suas informações. Os elementos complexos são formados por outros elementos dentro da hierarquia e os elementos simples contêm valores de dados. Os itens chamados *id* e *categoria* são exemplos de atributos.

```
<receita id="117" categoria="sobremesa">
  <titulo>Bolo de Banana</titulo>
  <autor>
    <email>joao@email.com</email>
  </autor>
  <ingredientes>3 bananas, 1 copo de leite,
    1 copo de farinha de trigo,
    1 colher de fermento.
  </ingredientes>
  <preparo>Bata tudo no liquidificador e asse por 20 minutos.
  </preparo>
</receita
```

Figura 1. Exemplo de Documento XML [XML07]

É possível observar que a representação textual do XML é correspondente a uma estrutura de dados em árvore. Na representação de uma árvore, os nodos internos são os

elementos complexos e os nodos-folha representam elementos simples. Dessa forma, os documentos XML são chamados de Modelo de Árvore ou Hierárquicos.

É possível caracterizar os documentos XML dentro de três categorias [ELM05]:

- *Documentos XML centrados em dados*: são documentos cujas mesmas ocorrências de dados são bem estruturadas e homogêneas, seguindo geralmente um esquema (DTD ou *XML Schema*, por exemplo). Esse esquema costuma ser extraído a partir de bancos de dados estruturados, servindo para intercâmbio de dados entre aplicações através da *Web*;
- *Documentos XML centrados em documentos*: são documentos com grande quantidade de texto, ou seja, são fracamente estruturados. Exemplos: artigos de jornais ou livros, anúncios classificados etc.;
- *Documentos XML híbridos*: são os documentos cujos dados podem ter partes bem estruturadas e outras predominantemente textuais ou não-estruturadas.

## 2.2 Linguagem XML Schema

*XML Schema* (XSD) é um padrão para especificação da estrutura de documentos XML. Um documento *XML Schema* é um documento XML e define diversas restrições de conteúdo para documentos XML, como seus elementos, a ordem de apresentação destes, restrições de conteúdo, atributos, grupos re-utilizáveis e tipos de dados.

Um documento XML que esteja de acordo com um esquema XSD é dito um documento *válido* [W3C07]. A linguagem XSD é capaz de definir um esquema bem estruturado e complexo utilizando conceitos de modelos de banco de dados relacionais e orientados a objetos, tais como chaves, estruturas complexas, referências e identificadores [ELM05].

Conforme pode ser visto na Figura 2, um esquema XSD é formado por um elemento *raiz* chamado *schema*. Esse elemento pode ter vários elementos *filhos*.

```

<schema>
  <!--GRUPO DE GERENCIAMENTO-->
  <import>...</import>
  <include>...</include>
  <redefine>...</redefine>
  <!--GRUPO DE COMPONENTES DE CONSTRUÇÃO DO MODELO-->
  <notation>...</notation>
  <simpleType>...</simpleType>
  <complexType>...</complexType>
  <attributeGroup>...</attributeGroup>
  <attribute.../>
  <group>...</group>
  <element>...</element>
</schema>

```

**Figura 2. Exemplo de XML Schema [CRM05]**

O esquema é dividido em dois grupos:

- *Grupo de gerenciamento*: encontra-se sempre no início do documento e auxilia na definição do esquema quando ele está distribuído em vários arquivos;
- *Grupo de componentes de construção do modelo*: são os elementos que compõem o documento propriamente dito.

### 2.2.1 Principais componentes da Linguagem XML Schema

A linguagem *XML Schema* possui um conjunto de elementos pré-definidos para descrever esquemas XML. Cada elemento possui um conjunto de atributos, os quais são utilizados para detalhar suas características, tais como nome, obrigatoriedade do campo, valor *default*, entre outras.

Os principais elementos que compõem um esquema XSD são:

- `<schema>` `</schema>`: é o elemento raiz, dentro dele estão os demais elementos;
- `<element>` `</element>`: esta *tag* define os elementos do sistema. Seus principais atributos são:
  - *name* – nome do elemento;
  - *type* – tipo do elemento;
  - *minOccurs* – número mínimo de ocorrências;
  - *maxOccurs* – número máximo de ocorrências;

- *abstract* – define se o elemento pode ser instanciado.
- `<complexType> </complexType>`: define os tipos de dados complexos, pode ser utilizado na criação de novos tipos ou elementos, e também possui atributos para especificação das características;
- `<simpleType> </simpleType>`: define tipos de dados simples. Pode ser utilizado na construção de atributos, elementos ou novos tipos;
- `<union> </union>`: define novos tipos de dados simples a partir da união de dois ou mais tipos de dados simples;
- `<list> </list>`: define tipos de dados simples, a partir de uma lista de tipos de dados simples que aparecem dentro do atributo *itemType*;
- `<group> </group>`: define um grupo de elementos. Juntamente a esta *tag*, informa-se a ordem dos elementos, com os indicadores: *sequence*, *all*, *choice*;
- `<attribute group> </attribute group>`: é utilizado para declarar um grupo de atributos que aparecem dentro do atributo *itemType*;
- `<attribute> </attribute>`: através das *tags name* e *type*, define atributos de determinado elemento;
- `<sequence> </sequence>`: indica a ordem de elementos complexos pré-definidos;
- `<choice> </choice>`: indica que os elementos que constarem em seguida a esta *tag* devem ter a ocorrência disjunta na estrutura;
- `<all> </all>`: indica que todos os elementos que constarem em seguida a esta *tag* devem existir na estrutura;
- `<restriction> </restriction>`: é utilizado na criação de tipos derivados com restrições de conteúdo. Pode ser aplicado a tipos simples e complexos, gerando assim um novo tipo, o qual será um subconjunto do conteúdo que constar no atributo *base*;
- `<extension> </extension>`: criação de tipo derivado com extensão de conteúdo. É possível sua aplicação a tipos complexos, gerando assim um novo tipo que herda as características do tipo base, podendo estendê-lo;



- `<complexContent> </complexContent>`: elemento usado em conjunto com o elemento *complexType* para indicar conteúdos derivados de tipos complexos;
- `<simpleContent> </simpleContent>`: elemento usado em conjunto com o elemento *complexType* para indicar a definição de conteúdo simples derivado de outro tipo simples ou de um tipo complexo de conteúdo simples;
- `<element type: anytype> </element>`: este elemento, com o tipo *anytype*, é um recurso da linguagem *XML Schema* para indicar que não se enquadra em nenhuma estrutura pré-definida, podendo ser instanciado com qualquer conteúdo;
- `<complexType mixed="true"> </complexType>`: este elemento, com o atributo *mixed* definido como *true*, permite a criação de elementos complexos de vários tipos.

### 2.2.2 Conceitos *XML Schema* Considerados na Ferramenta Proposta

A linguagem *XML Schema* possui um amplo conceito de *tipagem* de dados, sendo esta a principal justificativa de ser atualmente o modelo recomendado pela W3C. A abordagem em que a ferramenta XRCONVERSION se baseia realizou uma análise aprofundada da linguagem *XML Schema* identificando os conceitos que afetam diretamente a estrutura do documento XML modelado. A Figura 3 a seguir apresenta os conceitos que foram considerados [CRM05].

Atributo	Atômico		
	Lista		
	União		
	Derivado		
Elemento	Simples	Atômico	
		Derivado por restrição	
		União	
	Complexo	Conteúdo simples com atributos	Tipo atômico
			Lista
		Conteúdo formado por sub-elementos	União
			Derivado
			<i>Sequence</i>
	Misto	<i>All</i>	
			<i>Choice</i>
Vazio			
<i>anyType</i>			

**Figura 3. Conceitos considerados pela ferramenta XRCONVERSION**

Um exemplo de *XML Schema* com os conceitos considerados pode ser visualizado na Figura 4:

```

<xs:element name="elemento_principal">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="elemento" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="atributo_1" type="xs:ID"/>
    <xs:attribute name="atributo_2" type="xs:integer"/>
  </xs:complexType>
</xs:element>

```

**Figura 4. Exemplo de XML Schema**

### 3 TRABALHOS RELACIONADOS

A fim de traçar uma comparação entre trabalhos similares, este Capítulo apresenta ferramentas relacionadas ao assunto de conversão de *XML Schema* para esquemas relacionais.

#### 3.1 Mapeamento de definições *XML Schema* para SQL:1999

O objetivo da abordagem de mapeamento [MAR05] é prover um algoritmo para o mapeamento das construções XSD que apresentam uso mais comum. O resultado do mapeamento é um esquema objeto-relacional.

Para o início do trabalho, foram feitos levantamentos de conteúdo a partir de 199 esquemas XSD obtidos da *Web*, a fim de identificar as construções efetivamente usadas na prática e identificar mapeamentos eficientes para esquemas SQL:1999. Foram realizados testes estatísticos para verificar quais eram esses esquemas. A partir deste ponto, foram elaborados mapeamentos específicos para esses esquemas.

Para cada XSD, é mapeado um esquema SQL:1999 correspondente, de forma a possibilitar a armazenagem dos dados do XML extraído, sem perdas estruturais significativas.

Como principais características de nossa proposta de Mapeamento de definições *XML Schema* para SQL, têm-se que:

- apenas arquivos XSD válidos são considerados – subentende-se que os esquemas já passaram por um módulo de validação independente da abordagem dessa proposta;
- apenas arquivos XSD que não façam uso de construções contidas em outros esquemas podem ser mapeados;
- são desconsiderados esquemas que utilizem espaços de nomes diferentes do padrão;
- não são contemplados esquemas que possam gerar conteúdo misto, que apresentem recursão e que usem recursos de derivação para a criação de tipos.

### 3.2 Geração automática de esquemas relacionais a partir de esquemas XML

A ferramenta desenvolvida com o nome de **XRMap** é uma proposta de mapeamento automático usando regras pré-definidas [STE06]. Nessa proposta leva-se em consideração a integridade referencial dos dados através da identificação de relacionamentos no esquema XML, e posteriormente é gerado o conjunto de chaves primárias e chaves estrangeiras no modelo relacional.

A ferramenta possui duas formas de mapeamento: uma totalmente automática e outra com intervenção do usuário. A primeira etapa pela qual o XSD passa é uma função de pré-processamento. É feito na saída da ferramenta um XML de metadados, geralmente criado dessa forma:

- Cada elemento do XSD é mapeado como um elemento TABLE. Seu nome é definido em um atributo chamado NAME;
- Cada elemento filho é mapeado como um COLUMN, com atributos NAME e TYPE.

A etapa seguinte é o mapeamento deste XML de metadados para o esquema relacional. Caso tenha sido escolhido o modo automático, o XRMap decide quais colunas serão chaves primárias e estrangeiras, de acordo com a cardinalidade dos elementos. Isto é feito com base no elemento CARDINALITY criado dentro do XML de metadados. Após a geração de cardinalidades, são feitos ajustes pelo usuário nas inconsistências existentes, as quais devem ser corrigidas antes de se gerar o esquema relacional.

Quando se opta pelo mapeamento com intervenção do usuário, o usuário define quais chaves primárias e estrangeiras devem ser criadas durante o mapeamento. Isto é feito através da especificação dos elementos KEY e KEYREF no arquivo XSD de entrada da ferramenta. No final do mapeamento, a ferramenta XRMap cria um *script* SQL com as cláusulas para criação das tabelas.

#### 3.2.1 Mapping Definition Framework – MDF

A proposta deste trabalho consiste na elaboração de um *framework* (*Mapping Definition Framework - MDF*) que mapeia *XML Schema* para o modelo relacional, baseado na definição de anotações dentro do *XML Schema*. Posteriormente, essas anotações são lidas, processadas e dão origem a comandos SQL para a geração de tabelas.

As anotações do *framework* são pré-definidas e em número limitado, controlando dessa maneira os mapeamentos que podem ser definidos pelo usuário.

Um mapeamento é definido por uma quádrupla (XS, EM, AM, SM), onde:

- XS: *XML Schema* de entrada;
- EM: *Element mapping*, é uma função que mapeia um elemento em uma tabela, coluna ou CLOB no esquema relacional;
- AM: *Attribute mapping*, é uma função que mapeia um atributo em uma tabela, coluna ou CLOB no esquema relacional;
- SM: *Structure mapping*, define uma identidade, estrutura e ordem de mapeamento.

As anotações no documento *XML Schema* de entrada, que definem o mapeamento, podem ser associadas a atributos, elementos e grupos. A sintaxe dessas anotações consiste em adicionar atributos de um espaço de nomes chamado MDF em um documento *XML Schema* de entrada. Após a anotação desses atributos, são feitos processos de validação e *parsing*, e como saída há a geração de um repositório de mapeamento e dos comandos CREATE TABLE necessários para a construção do esquema relacional.

### 3.3 Conclusão

Foi verificado que as propostas relacionadas possuem um tratamento bem parecido com as funções da ferramenta XRCONVERSION. Todas as três propostas possuem como saída o SQL para a criação de tabelas. A primeira e a terceira propostas baseiam-se em limitação do escopo de entrada para que a conversão seja mais eficiente. A primeira proposta é baseada em um estudo estatístico das construções mais utilizadas. Já a ferramenta proposta, XRCONVERSION, possui como base os conceitos *XML Schema* que formam as principais construções de XML, conforme recomendação da W3C [W3C07].

A segunda proposta possui uma característica interessante em relação às cardinalidades, pois há a opção de questionar o usuário quanto à decisão que ele deseja tomar

em relação a isso, tornando o processo semi-automático. A ferramenta dessa proposta, porém, não contempla a análise mais aprofundada da *tipagem* dos atributos, como por exemplo o tipo list. A ferramenta XRCONVERSION, por sua vez, analisa essas questões, proporcionando para o tipo list, por exemplo, um tratamento diferenciado, visto que esse atributo pode ter várias ocorrências no esquema, semelhante a um elemento.

A terceira proposta realiza suas tarefas no nível do código e não permite que o usuário interaja durante a conversão, é um modelo caixa-preta. Foi verificado nesses estudos que a intervenção do usuário é muito importante, para que o modelo seja adequado às suas necessidades. Este é um dos principais fundamentos da ferramenta XRCONVERSION.

## **4 A FERRAMENTA**

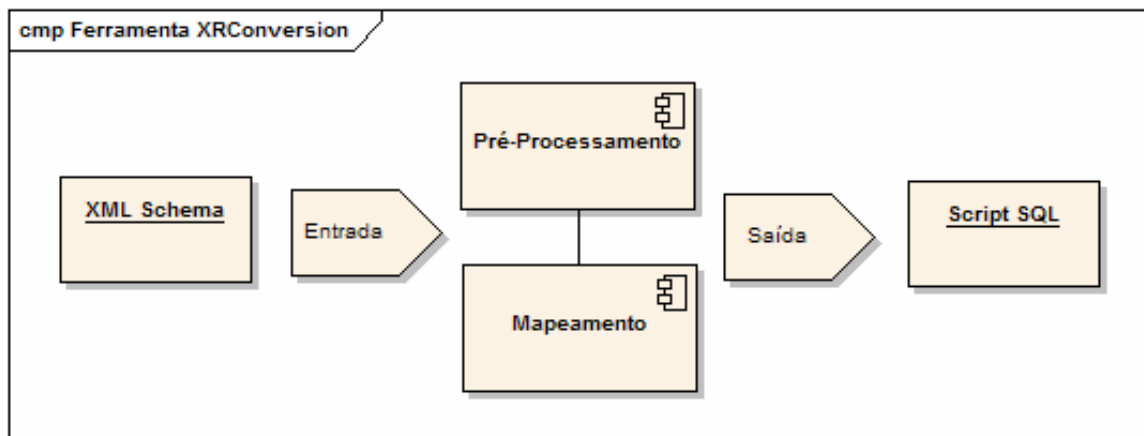
A ferramenta desenvolvida neste trabalho possui como objetivo principal promover a função de integração de dados que trafegam entre aplicações comunicantes, as quais utilizam o padrão XML para essa função, sendo os dados dessas aplicações mantidos em bancos de dados.

A integração proposta utiliza a *XML Schema*, por ser a linguagem mais recomendada para definição de esquemas XML [W3C07], e o modelo de dados relacional, pois é predominante nas estruturas de banco de dados atuais.

Em projetos que envolvem a integração desses dois modelos de representação de dados, os custos e o tempo de desenvolvimento podem ser reduzidos através da utilização da ferramenta XRCONVERSION. Esta aplicação contempla: um pré-processador, que adapta o conteúdo para facilitar o mapeamento; um conjunto de regras, que mapeia a estrutura de entrada para possíveis correspondências na linguagem de definição de dados SQL; e um algoritmo que realiza a varredura no esquema XML.

### **4.1 Funcionamento Geral**

A ferramenta realiza 3 atividades principais: i) leitura do *XML Schema* de entrada; ii) aplicação do pré-processamento e das regras de mapeamento; e iii) criação de um arquivo com o esquema SQL para o modelo de dados relacional. A Figura 5 apresenta uma visão geral do funcionamento da ferramenta.



**Figura 5. Funcionamento geral da ferramenta**

A entrada da ferramenta XRCONVERSION é um arquivo de dados com a extensão *xsd*. Os dados contidos nesse arquivo são lidos e passados para o módulo de pré-processamento. A primeira etapa é um algoritmo avaliador que garante que o conteúdo está de acordo com a estrutura da linguagem *XML Schema*. Em seguida, o pré-processamento invoca as tarefas responsáveis por adequar o esquema XSD de entrada para uma melhor forma de tratamento e leitura pelas regras de mapeamento.

O segundo módulo é chamado de Mapeamento, onde há a aplicação do algoritmo principal, que consiste de uma varredura em cada elemento do esquema, identificando suas possíveis correspondências com o modelo relacional. Ao término do processo de mapeamento, são apresentadas as tabelas resultantes da conversão em um novo arquivo com extensão *sql*.

Mais adiante explicaremos detalhadamente os módulos que compõem a ferramenta XRCONVERSION.

## 4.2 Construção da Ferramenta XRCONVERSION

### 4.2.1 Metodologia

Na fase de concepção da ferramenta XRCONVERSION, decidiu-se dividir o desenvolvimento em 4 etapas, a fim de torná-lo mais organizado e eficiente:



- i. Estudo da abordagem de mapeamento proposta por [CRM05]: consiste no conjunto de regras de mapeamento que são o fundamento da ferramenta;
- ii. Análise da ferramenta e suas principais funções: consiste na criação de diagramas e fluxos de operação da ferramenta para que a mesma efetue sua função principal, que é o mapeamento de esquemas;
- iii. Escolha das tecnologias: pesquisa pelas principais bibliotecas que manipulam XML, bem como os ambientes de desenvolvimento;
- iv. Implementação e testes: é o constante ciclo programação → testes, sendo esta a etapa mais longa de todo o desenvolvimento.

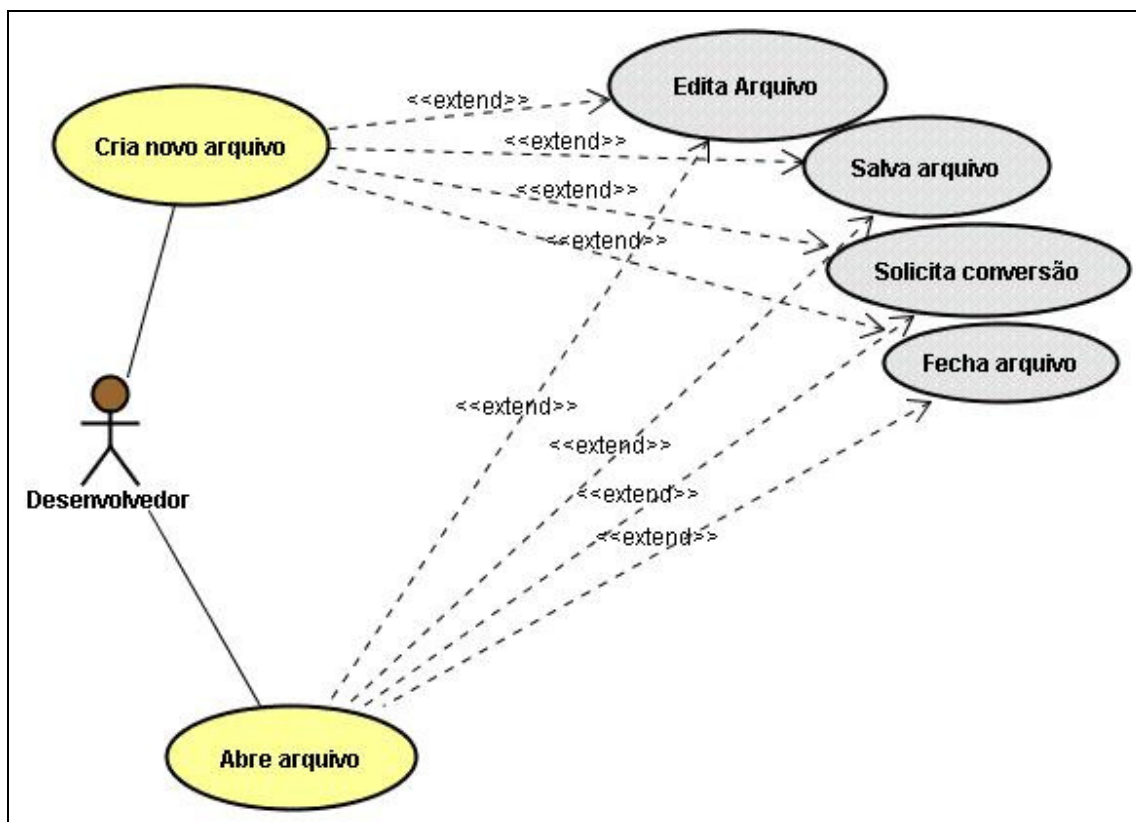
Explicaremos minuciosamente a seguir estes itens, neste mesmo Capítulo.

## **4.2.2 Diagramas**

### **4.2.2.1 Casos de uso**

A partir da identificação das principais interações do usuário com a ferramenta XRCONVERSION, foi possível elaborar um diagrama, apresentado na Figura 6.

Quando o usuário entra em contato com a aplicação, mediante sua interface gráfica, apenas duas ações são possíveis de se tomar: a criação de um novo arquivo ou a abertura de algum arquivo que já esteja salvo em algum diretório.

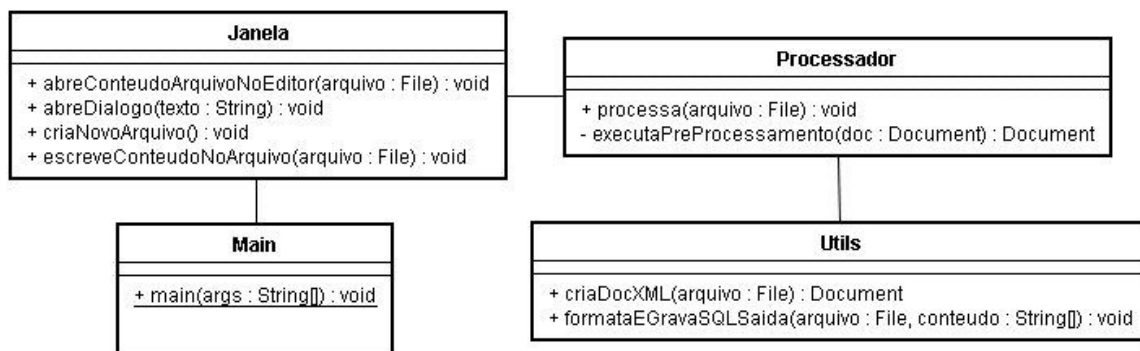


**Figura 6. Diagrama de casos de uso**

No caso de se tomar uma dessas duas ações, novos casos de uso são habilitados, ou seja, novas ações podem ser tomadas pela ferramenta: i) edição do conteúdo; ii) salvamento (gravação) deste em algum diretório; iii) conversão (mapeamento para o modelo relacional) ou iv) fechamento do arquivo convertido. Todos esses casos de uso são as opções de menu que aparecem para o usuário mediante a interface gráfica.

#### **4.2.2.2 Classes do sistema**

Para que as ações descritas anteriormente possam ser contempladas pela ferramenta XRCONVERSION, a implementação foi feita conforme as descrições do diagrama de classes apresentado na Figura 7.



**Figura 7. Diagrama de classes**

O processamento inicia-se mediante o método *main* que está implementado na classe *Main*. A função dessa classe é apenas iniciar o processamento. Em seguida, é chamada a classe *Janela*, que se trata da implementação dos métodos relacionados à interface gráfica. A parte visual da ferramenta consiste de uma tela principal e algumas caixas de diálogo, todas estas processadas com o auxílio da classe *Janela*, que se comunica com a classe *Processador*. Esta, por sua vez, contempla os métodos que executam o mapeamento. A classe *Processador* comunica-se com a classe *Utils*, que implementa funções auxiliares, relacionadas à manipulação de conteúdo texto, criação de novos arquivos, entre outras funções úteis para a aplicação.

#### 4.2.3 Tecnologias utilizadas

Para implementar as regras de mapeamento definidas no trabalho de [CRM05], foi escolhida a linguagem de programação JAVA/JSE na sua versão 6.0 [SUN07]. Diversas bibliotecas foram encontradas, conhecidas como API's externas à linguagem JAVA, que manipulam dados XML. Dentre estas, decidiu-se trabalhar com a API JAXP (JAVA API for XML Processing) na sua versão 1.4.2, que contempla funcionalidades direcionadas ao tema deste trabalho [JAX07].

O ambiente de desenvolvimento escolhido foi o *Oracle JDeveloper* versão 10, o qual auxiliou satisfatoriamente na implementação das rotinas e da interface gráfica [ORA07]. É uma ferramenta gratuita para desenvolvimento de aplicações da linguagem JAVA.

## 4.3 Módulos

### 4.3.1 Módulo de Pré-processamento

O módulo de pré-processamento consiste em adequar o esquema XSD de entrada para uma melhor forma de tratamento e leitura pelas regras de mapeamento. Ele visa a redução de complexidade na aplicação do algoritmo de conversão.

As alterações realizadas por esse módulo são:

- Incorporação de conteúdo externo no mesmo esquema, quando há referência a outro arquivo;
- Conversão de grupos de substituição para a tag *choice* da *XML Schema*;
- Substituição de definições chamadas *extension* pelo próprio conteúdo a ser estendido;
- Eliminação de referências.

Dessa forma, as definições de todas as estruturas do esquema se tornam explícitas nas definições dos elementos.

#### 4.3.1.1 Incorporação de conteúdo externo

Um esquema XML pode ser especificado integralmente em um único arquivo com extensão *.xsd*, mas também é possível relacionar definições de esquema que se encontram em outros arquivos.

A construção de esquemas com referências a arquivos externos pode ser feita com uso dos elementos *include*, *import* e *redefine* da *XML Schema*. O elemento *include* permite incluir definições externas que estão armazenadas no mesmo espaço de nome ou sem definição de espaço de nome, diretamente no esquema em construção. Ele funciona como uma cópia do conteúdo.

Quando o espaço de nome do esquema XML externo a ser usado é diferente, deve-se usar o elemento *import*, que faz a inclusão de conteúdo levando em consideração os espaços de nome do esquema importado e do esquema que está importando.

Já o elemento *redefine* permite importar conteúdos externos e redefinir partes do modelo importado, ajustando o esquema de conteúdo ao esquema em construção.

A ferramenta XRCONVERSION somente realiza o mapeamento para o modelo relacional após a transformação do conteúdo conforme foi explicado anteriormente. O modelo final já deve possuir os conteúdos devidamente incluídos, importados ou redefinidos, ou seja, não deve constar nenhuma referência externa.

#### 4.3.1.2 Tratamento de Grupos de Substituição

Quando houver grupos de substituição no esquema, compostos por um elemento principal chamado “cabeça” e outros elementos que podem substituí-lo, a ferramenta realiza os seguintes passos:

1. Criação de uma *tag choice* contendo o elemento “cabeça”, nos locais onde este é utilizado;
2. Inclusão de uma cópia de cada elemento que pode substituir o “cabeça”, dentro do *choice* criado;
3. Remoção do atributo *substitutionGroup* da declaração dos elementos que podem substituir o “cabeça”.

Após esses passos, o grupo de substituição pode ser mapeado pela regra de mapeamento correspondente.

A Figura 8 a seguir apresenta a declaração de um grupo de substituição onde um elemento *pessoa2* pode ser usado em substituição ao elemento *pessoa1*. Ambos são derivações válidas de *pessoa*.

```

<xs:complexType name="pessoa">
  <xs:sequence>
    <xs:element name="Nome" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="sexo" type="xs:string"/>
</xs:complexType>

<xs:element name="pessoa1" type="pessoa"/>

<xs:element name="pessoa2" substitutionGroup="pessoa1">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="pessoa">
        <xs:sequence>
          <xs:element name="sobrenome"
            type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="aluno">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="pessoa1"/>
      <xs:element name="registro" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

**Figura 8. Declaração de um grupo de substituição**

A Figura 9 apresenta o esquema resultante após a realização das alterações necessárias.

```

<xs:complexType name="pessoa">
  <xs:sequence>
    <xs:element name="Nome" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="sexo" type="xs:string"/>
</xs:complexType>

<xs:element name="pessoa1" type="pessoa"/>

<xs:element name="pessoa2">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="pessoa">
        <xs:sequence>
          <xs:element name="sobrenome" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="aluno">
  <xs:complexType>
    <xs:sequence>

```

```

        <xs:choice>
            <xs:element ref="pessoa1"/>
            <xs:element ref="pessoa2"/>
        </xs:choice>
        <xs:element name="registro" type="xs:int"/>
    </xs:sequence>
</xs:complexType>
</xs:element>

```

**Figura 9. Criação de choices para tratar grupos de substituição**

#### 4.3.1.3 Tratamento de Extensões

Este item do pré-processamento está relacionado ao conceito de herança, que a linguagem *XML Schema* suporta. Nos esquemas que possuem esse conceito, isto é feito declarando-se primeiramente um elemento contendo o atributo *abstract* com valor *true*. Esse elemento não pode ser instanciado, pode apenas ser estendido a partir da definição de outros elementos, os quais passam a obter as características do elemento abstrato, também conhecido como ancestral. Segue um exemplo de utilização do conceito de herança na Figura 10.

```

<xs:complexType name="pessoa" abstract="true">
    <xs:sequence>
        <xs:element name="Nome" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="sexo" type="xs:string"/>
</xs:complexType>

<xs:element name="pessoa2">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="pessoa">
                <xs:sequence>
                    <xs:element name="sobrenome" type="xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>

```

**Figura 10. Declaração de extensão**

A transformação que a ferramenta realiza neste caso consiste em substituir as referências à estrutura estendida (atributo *base*) pelo conteúdo herdado, gerando um esquema XML no qual os elementos incorporam esse conteúdo. A Figura 11 apresenta o resultado da substituição da extensão de tipos abstratos, mostrada na Figura 10.

```

<xs:element name="pessoa2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Nome" type="xs:string"/>
      <xs:element name="sobrenome" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="sexo" type="xs:string"/>
  </xs:complexType>
</xs:element>

```

**Figura 11. Exemplo de incorporação de conteúdo estendido**

#### 4.3.1.4 Eliminação de Referências

Uma definição de esquema de dados em *XML Schema* permite a definição de estruturas globais, que são declaradas como filhas do elemento *schema*. Essas declarações definidas pelo usuário podem ser referenciadas a partir da definição de outras estruturas com o uso dos atributos *ref* ou *type*, como pode ser visto na Figura 12.

```

<schema>
  <xsd:element name="comment" type="xsd:string"/>
  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      ...
    <xsd:element ref="comment" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
</schema>

```

**Figura 12. Referência a declarações globais**

Essas referências existem para que haja o reaproveitamento da definição das estruturas já declaradas globalmente. Elas podem envolver elementos, grupos de elementos, atributos, grupos de atributos e tipos. O objetivo do pré-processamento neste caso é substituir as referências encontradas pelo conteúdo referenciado.

Após a realização das substituições, as declarações de grupos de elementos, grupos de atributos e tipos globais nomeados são removidas, pois seu conteúdo já foi incorporado nas definições de elementos. A Figura 13 a seguir apresenta o resultado da substituição da referência mostrada na Figura 12.



```

<schema>
<xsd:element name="comment" type="xsd:string"/>
<xsd:element name="purchaseOrder">
<xsd:sequence>
...
<xsd:element name="comment" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
</xsd:element>
</schema>

```

**Figura 13. Eliminação de referências**

### 4.3.2 Módulo de Mapeamento

Após a conclusão do pré-processamento, a aplicação chama o módulo de mapeamento, que é responsável pela leitura do esquema e aplicação das regras.

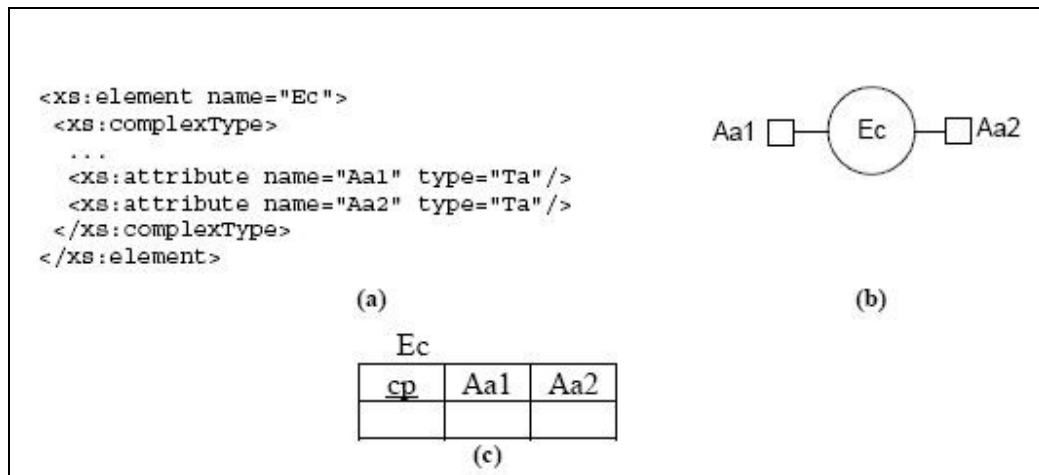
Esse módulo é formado por um conjunto de 11 regras, que fornecem alternativas de mapeamento para atributos e elementos XML descritos no Capítulo 2. O algoritmo desenvolvido varre o esquema XML, identifica seus conceitos e suas possíveis correspondências no modelo relacional e aplica a conversão de acordo com a regra específica. O processo é semi-automático, pois em algumas situações existe mais de uma correspondência no modelo relacional, sendo necessária uma decisão do usuário.

São 4 as regras que tratam atributos e 7 as que tratam elementos, conforme apresentado a seguir, juntamente com exemplos de suas aplicações.

- **Regra ATA (Atributo do Tipo Atômico):** conforme Maurício [CRM 2005], a regra ATA

[...] considera que um atributo do tipo atômico pode ser mapeado para uma coluna na tabela correspondente ao elemento que o contém. Definição: Um atributo atômico AT de nome Aa e tipo de dado Ta contido em um elemento complexo EC gera uma coluna de nome Aa e tipo de dado Ta na tabela que mantém os dados de EC. Se AT for do tipo ID ou IDREF, a coluna gerada deve ser indicada como chave primária ou chave estrangeira, respectivamente [CRM 2005].

A Figura 14 a seguir ilustra melhor essa regra.



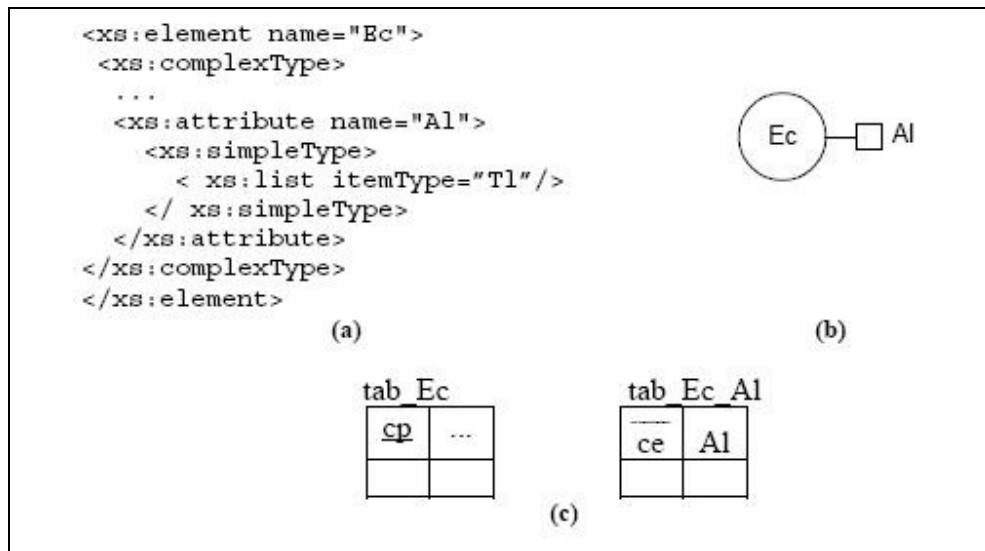
**Figura 14. Exemplo de Mapeamento de Atributo Atômico.**

Nessa figura observa-se um exemplo de Mapeamento de Atributo Atômico onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML *Schema* e (c) é o esquema relacional [CRM05].

- **Regra ATL (Atributo do Tipo List):** Conforme Maurício [CRM 2005], a regra ATL

[...] considera que um atributo de tipo list pode ser mapeado para uma tabela própria, que armazena o conteúdo da lista e faz referência à tabela correspondente ao elemento que contém o atributo. Definição: Um atributo do tipo list ATL de nome Al e tipo de dado da lista Tl contido em um elemento complexo E de nome Ec, gera uma tabela de nome tab\_Ec\_Al, contendo duas colunas: uma sendo a chave estrangeira ce, que referencia a tabela que mantém os dados de E, e a outra sendo uma coluna de nome Al e tipo de dado Tl, que contém um item de Al 5. Se ATL for do tipo IDREFS, a coluna Al gerada deve ser indicada como chave estrangeira [CRM 2005].

A Figura 15 a seguir ilustra melhor essa regra.

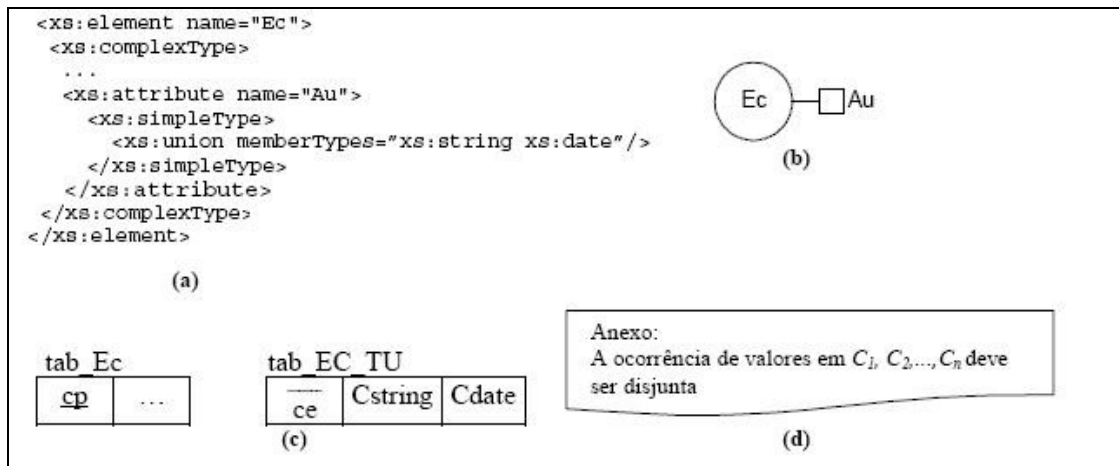


**Figura 15. Exemplo de Mapeamento de Atributo do Tipo List**

Nessa figura observa-se um exemplo de Mapeamento de Atributo do Tipo List, onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML Schema e (c) é o esquema relacional [CRM 2005].

- Regra ATU (Atributo do Tipo União):** conforme Maurício [CRM 2005], a regra ATU [...] considera que um atributo do tipo *union* pode ser mapeado para uma coluna com o tipo mais abrangente da *union* ou uma coluna do tipo texto, na tabela correspondente ao elemento que o contém, ou ainda, para uma tabela própria com uma coluna de referência para a tabela correspondente ao elemento que contém o atributo, com colunas para os tipos de dados pertencentes à *union* [CRM 2005].

A Figura 16 a seguir ilustra melhor essa regra.



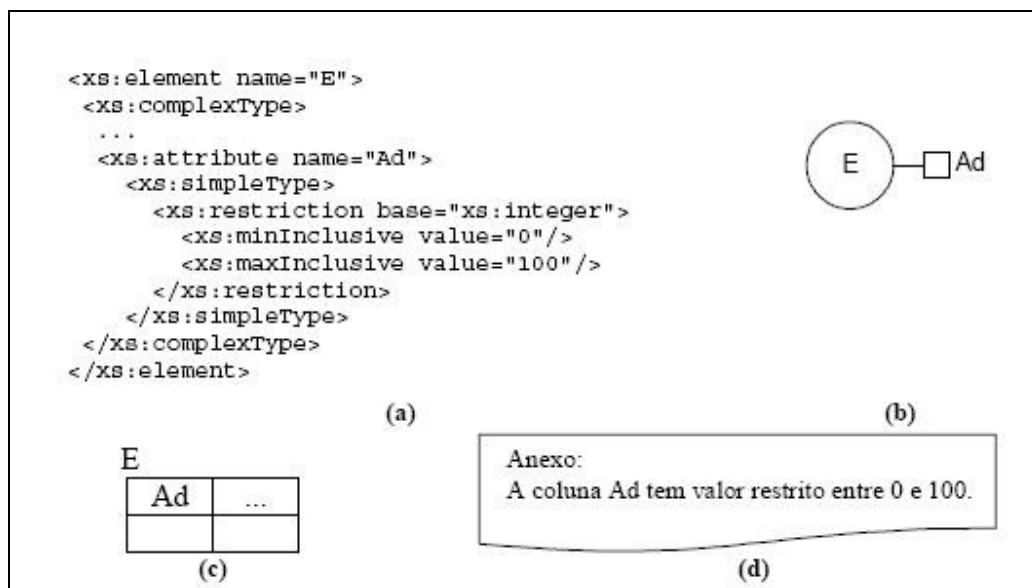
**Figura 16. Exemplo de Mapeamento de Atributo do Tipo Union**

Nessa figura observa-se um exemplo de Mapeamento de Atributo do Tipo Union, onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML Schema, (c) é o esquema relacional e (d) é o anexo ao esquema relacional descrevendo restrições de integridade [CRM05].

- **Regra ATD (Atributo de Tipo Derivado):** conforme Maurício [CRM 2005], a regra ATD

[...] considera que um atributo de tipo derivado é mapeado para uma coluna na tabela correspondente ao elemento que o contém. Definição: Um atributo A de tipo de dado derivado Td de nome Ad, contido em um elemento Ec, gera uma coluna de nome Ad e tipo de dado Td na tabela que mantém os dados de Ec, e um anexo com uma restrição de integridade correspondente à derivação de Td [CRM 2005].

A Figura 16 a seguir ilustra melhor essa regra.

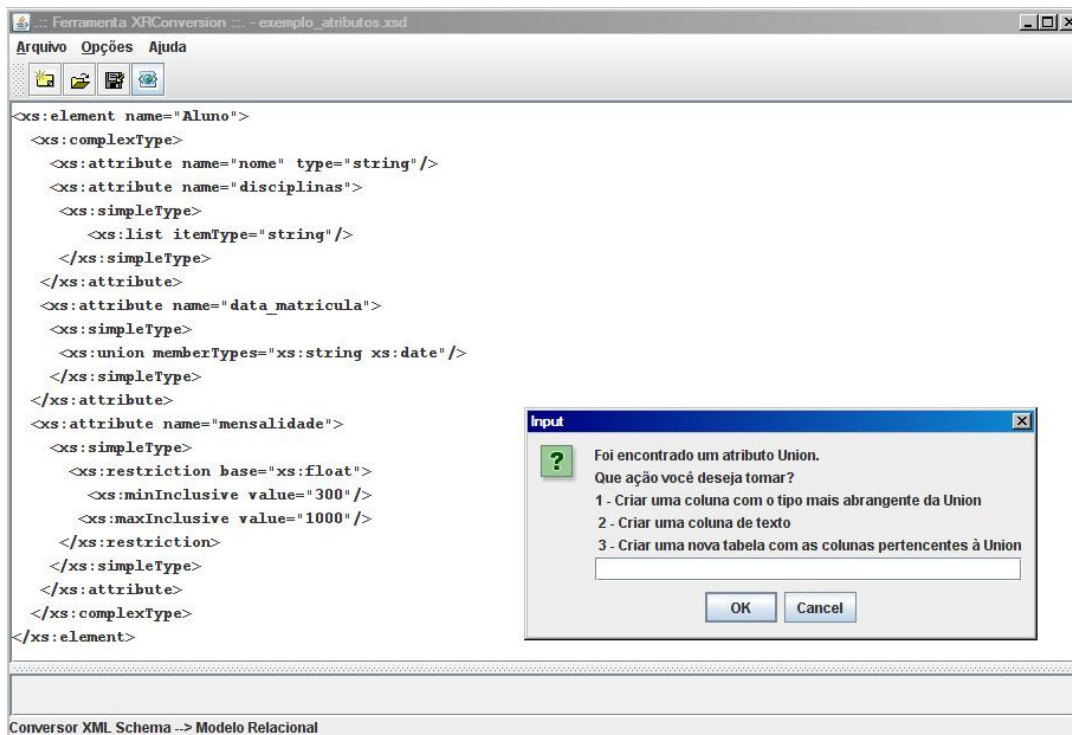


**Figura 17. Exemplo de Mapeamento de Atributo do Tipo Derivado**

Nessa figura observa-se um exemplo de Mapeamento de Atributo do Tipo Derivado, onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML *Schema*, (c) é o esquema relacional e (d) é o anexo ao esquema relacional descrevendo restrições de integridade [CRM05].

Um exemplo de aplicação das regras de mapeamento de atributos é mostrado a seguir para um atributo do tipo união. Neste caso, uma decisão do usuário é necessária, pois ele possui três alternativas de mapeamento, conforme já apresentado na regra ATU.

A Figura 18 mostra a tela principal da ferramenta e uma caixa de diálogo que é apresentada ao usuário quando se necessita tomar uma decisão.



**Figura 18. Exemplo de conversão de atributos**

O resultado da conversão exemplificada nessa figura é salvo em um novo arquivo no mesmo diretório e com o mesmo nome do arquivo de entrada, mas com a extensão *sql*. Ou seja, neste exemplo, o nome do XSD de entrada é *exemplo\_atributos.xsd*. O nome do arquivo de saída é *exemplo\_atributos.sql*. Em seguida, é mostrado o script SQL resultante do exemplo analisado até aqui (ver Figura 19):

```

CREATE TABLE tab_Aluno(
  ID INTEGER NOT NULL,
  nome VARCHAR(80) NULL,
  data_matricula LONG VARCHAR NULL,
  mensalidade INTEGER NULL //Anexo: A Coluna mensalidade
  possui valor restrito entre 300 e 1000

  PRIMARY KEY(ID)
);

CREATE TABLE tab_Aluno_disciplinas(
  ID INTEGER NOT NULL,
  disciplinas VARCHAR(80) NULL

  FOREIGN KEY(tab_Aluno.ID)
);

```

**Figura 19. Resultado da conversão**

Com base nas quatro regras citadas anteriormente, a conversão ocorre da seguinte forma:

- 1) A interface lê o conteúdo e o envia ao módulo de conversão. Como o esquema já está pré-processado, ele é encaminhado para o algoritmo principal;
- 2) Nesse algoritmo, é verificado que a regra correspondente é a ATA, a qual, por sua vez, converte o atributo de tipo atômico *nome* para uma coluna *nome* na tabela *Tab\_Aluno*;
- 3) Essa tabela corresponde à conversão do elemento de mesmo nome no documento XSD. A coluna *mensalidade* é resultado da conversão do atributo de mesmo nome no documento XSD através da aplicação da regra ATD. Em anexo, é colocada a restrição de integridade que limita a sua faixa de valores;
- 4) A aplicação da regra ATL sobre o atributo do tipo *lista disciplinas* gera uma tabela de nome *Tab\_Aluno\_Disciplinas*. Cada linha dessa tabela corresponde a uma das disciplinas que o aluno cursa. O atributo *ID* nessa tabela é uma chave estrangeira para *Tab\_Aluno*;
- 5) A coluna *data\_matricula* foi gerada na tabela *Tab\_Aluno* através da aplicação da regra ATU, que converte o atributo do tipo união *data\_matricula*. Nesse caso, através de interação com a caixa de diálogos, o usuário optou para que fosse gerada uma coluna do tipo texto como resultado da *union*.

A seguir, serão apresentadas as regras de mapeamento que tratam elementos XML.

- **Regra ESTN (Elemento Simples de Tipo Nativo):** conforme Maurício [CRM 2005], a regra ESTN

[...] considera que um elemento simples do tipo nativo pode ser mapeado para uma tabela própria ou para uma coluna na tabela correspondente ao seu elemento ancestral. A regra considera ainda múltiplas ocorrências do elemento no seu elemento ancestral, definindo, neste caso, sempre uma tabela para ele. A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento simples, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido.

Definição: Um elemento simples ES de tipo nativo de nome *Esn* e tipo de dado *Tesn*, com cardinalidade máxima *cm*.

Se ES possui  $cm \leq 1$ :

- i. gera uma coluna de nome Esn e tipo de dado Tesn em tab\_Anc, se ES possui um elemento ancestral que tenha sido mapeado para uma tabela tab\_Anc;
- ii. gera uma tabela de nome tab\_Esn com uma coluna de nome Esn e tipo de dado Tesn que armazena o conteúdo do elemento, se ES não possui um elemento ancestral.

Se ES possui cm > 1:

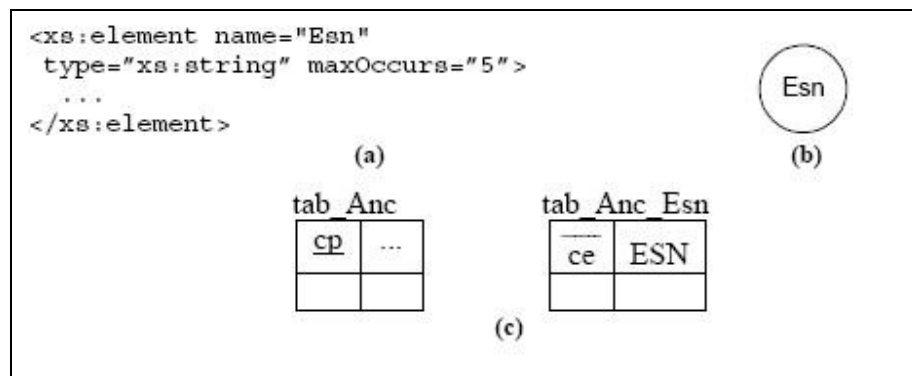
- i. gera uma tabela de nome tab\_Esn com uma coluna de nome Esn e tipo de dado Tesn que armazena o conteúdo do elemento;
- ii. inclui-se uma chave estrangeira Ce em tab\_Esn que referencia tab\_Anc, se ES possui um elemento ancestral mapeado para tab\_Anc.

Se ES está contido em um indicador de ordem *sequence* em um elemento ancestral:

- i. gera uma coluna de nome ordem\_Esn na tabela tab que mantém ES;
- ii. a chave primária de tab é o par (ordem\_Esn, Ce), se tab \_ tab\_Anc.

A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento simples, é também considerada. Nesse caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido [CRM 2005].

A Figura 20 a seguir ilustra melhor essa regra.



**Figura 20. Exemplo de Mapeamento de Elemento Simples**

Nessa figura observa-se um exemplo de Mapeamento de Atributo de Elemento Simples, onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML *Schema* e (c) é o esquema relacional [CRM05].

- **Regra ESDR (Elemento Simples Derivado por Restrição):** conforme Maurício [CRM 2005], a regra ESDR



[...] considera que um elemento simples de tipo derivado por restrição pode ser mapeado para uma tabela própria ou para uma coluna na tabela correspondente ao seu elemento ancestral. A regra considera ainda múltiplas ocorrências do elemento no seu elemento ancestral, definindo, neste caso, sempre uma tabela para ele. A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento simples, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido.

Definição: Um elemento simples ES de tipo de dado derivado por restrição Tdr de nome Edr, com cardinalidade máxima cm.

Se ES possui  $cm \leq 1$ :

- i. gera uma tabela de nome `tab_Edr` com uma coluna de nome `Edr` de tipo `Tdr` que armazena o conteúdo de ES. Incluir uma chave estrangeira `Ce` em `tab_Edr` que referencia `tab_Anc`, se ES possui um elemento ancestral mapeado para uma tabela `tab_Anc`;
- ii. gera uma coluna de nome `Edr` e tipo de dado `Tdr` em `tab_Anc`, se ES possui um elemento ancestral mapeado para `tab_Anc`.

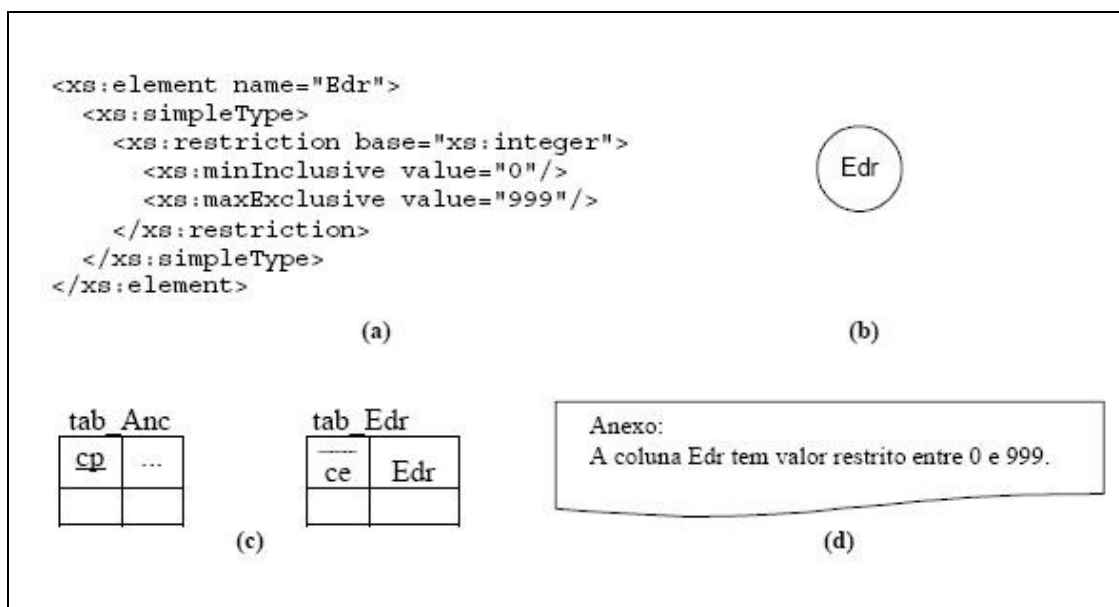
Se ES possui  $cm > 1$ :

- i. gera uma tabela de nome `tab_Edr` com uma coluna de nome `Edr` que armazena o conteúdo do elemento. Incluir uma chave estrangeira `Ce` em `tab_Edr` que referencia `tab_Anc`, se ES possui um elemento ancestral mapeado para `tab_Anc`.
- ii. gera uma restrição de integridade correspondente à `Tdr`.

Se ES está contido em um indicador de ordem *sequence* em um elemento ancestral:

- i. gera uma coluna de nome `ordem_Edr` na tabela `tab` que mantém ES; a chave primária de `tab` é o par (`ordem_Edr`, `ce`), se `tab` \_ `tab_Anc` [CRM 2005].

A Figura 21 a seguir ilustra melhor essa regra.



**Figura 21. Exemplo de Mapeamento de Atributo Derivado por Restrição**

Nessa figura observa-se um exemplo de Mapeamento de Atributo de Elemento Simples, onde (a) é um esquema XML, (b) é uma representação do elemento no modelo XML *Schema*, (c) é o esquema relacional e (d) é o anexo ao esquema relacional descrevendo restrições de integridade [CRM05].

- **Regra ESTU (Elemento Simples de Tipo União):** conforme Maurício [CRM 2005], a regra ESTU

[...] considera que um elemento simples do tipo *union* pode ser mapeado para uma tabela com: (i) uma coluna com o tipo mais abrangente da *union*; (ii) uma coluna do tipo texto ou, (iii) colunas para os tipos de dados pertencentes à *union*. Em todos os casos de mapeamento, deve-se incluir uma coluna de referência para a tabela correspondente ao elemento ancestral, se este existir. A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento simples, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido.

Definição: Um elemento simples ES de nome *Esu* e de tipo *union* TU, com um conjunto de tipos de dados da *union* *tu1, tu2, ..., tun*:

- gera uma tabela de nome *tab\_Anc\_Esu* com uma coluna de nome *Esu* que armazena o conteúdo de *ES* e cujo tipo *T* seja o tipo mais genérico dentre *tu1, tu2, ..., tun*, caso *T* englobe todos os valores admissíveis por *TU*;

ou:

- gera uma tabela de nome *tab\_Anc\_Esu* com uma coluna de nome *ESU* e tipo de dado texto que armazena o conteúdo de *ES*;

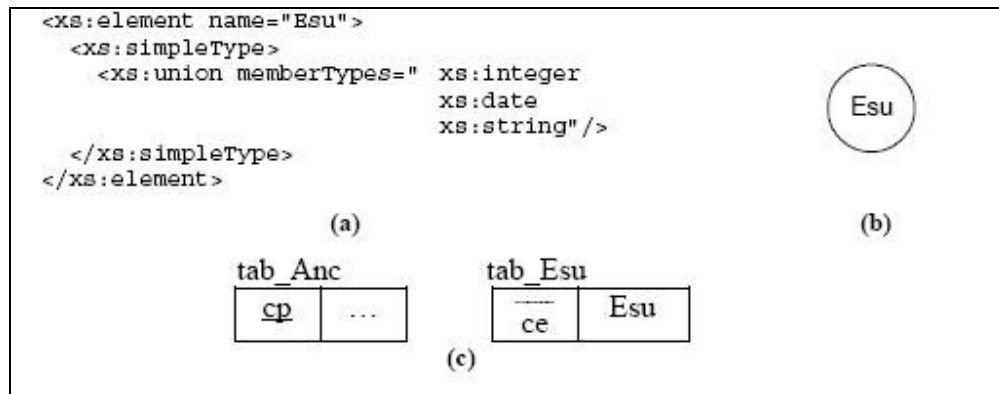
ou:

- gera uma tabela de nome *tab\_Anc\_Esu* com colunas de nome *c1, c2, ..., cn* com tipos correspondentes a cada item do conjunto *tu1, tu2, ..., tun*. Define-se restrições de integridade para garantir que em cada linha da tabela somente uma coluna em *c1, c2, ..., cn* possua valor.

- gera uma coluna chave estrangeira *ce* em *tab\_Anc\_Esu* que referencia *tab\_Anc*, se *ES* possui um elemento ancestral *Anc* que tenha sido mapeado para uma tabela *tab\_Anc*.

- gera: uma coluna de nome *ordem\_Esu* em *tab\_Anc\_Esu*; a chave primária de *tab\_Anc\_Esu* é o par (*ordem\_Esu, ce*), se *ES* está contido em um indicador de ordem *sequence* em um elemento ancestral [CRM 2005].

A Figura 22 a seguir ilustra melhor essa regra.



**Figura 22. Exemplo de Mapeamento de Elemento tipo Union**

Nessa figura observa-se um exemplo de Mapeamento de Elemento tipo Union, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema* e (c) o esquema relacional [CRM05].

- **Regra EC (Elemento Complexo):** conforme Maurício [CRM 2005], a regra EC

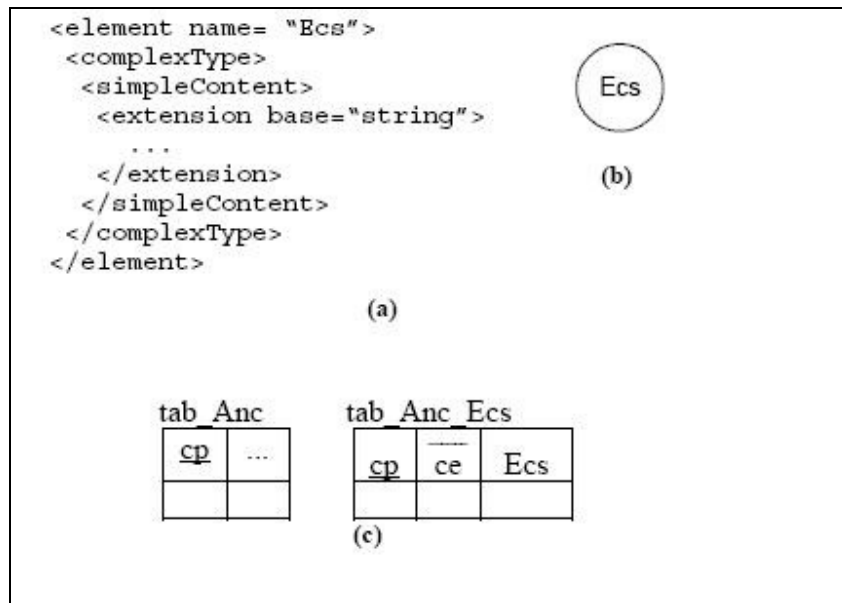
[...] considera que um elemento complexo é mapeado para uma tabela própria. A regra considera ainda o uso do indicador de ordem *choice* no elemento complexo, definindo, neste caso, restrições de integridade para o mapeamento.

A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento complexo, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido.

Definição: Um elemento complexo EC de nome Ecs e tipo Tecs.

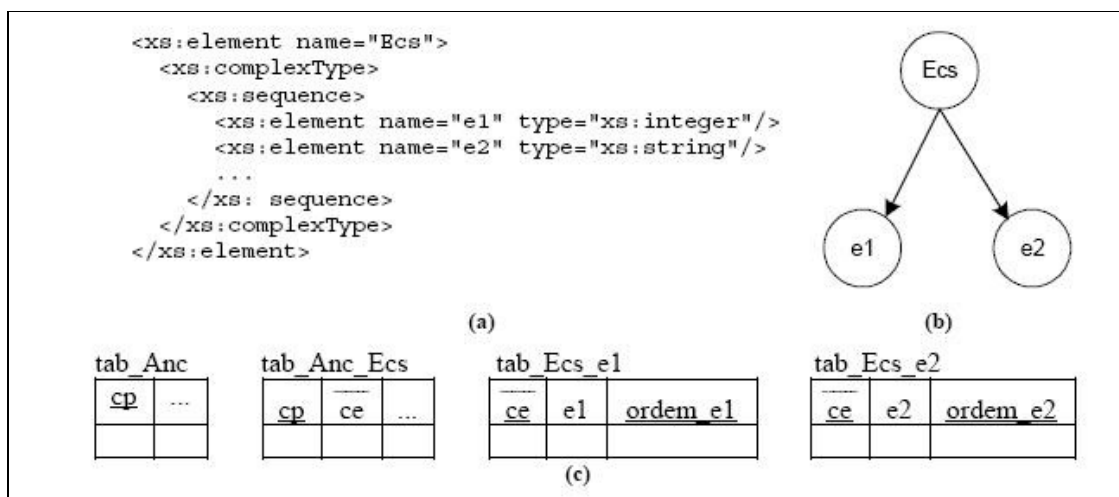
- gera uma tabela de nome *tab\_Ecs* com uma coluna chave primária *Cp*;
- gera uma chave estrangeira *Ce* em *tab\_Ecs* que referencia *tab\_Anc*, se EC possui um elemento ancestral mapeado para uma tabela *tab\_Anc*;
- gera restrições de integridade referentes a uma derivação, se *TEcs* é um tipo derivado por restrição;
- gera uma coluna que armazena o conteúdo de *Ecs*, se EC possui conteúdo simples;
- gera restrições de integridade para garantir que, a cada ocorrência de *Ecs*, somente um elemento entre *e1*, *e2*, ..., *en*, possua valor, se EC usa indicador de ordem *choice* e contém elementos *e1*, *e2*, ..., *en*.
- gera: uma coluna de nome *ordem\_Ecs* em *tab\_Anc\_Ecs*; a chave primária de *tab\_Ecs* é o par (*ordem\_Ecs*, *Ce*), se EC está contido em um indicador de ordem *sequence* em um elemento ancestral [CRM 2005].

As três figuras a seguir ilustram melhor essa regra.



**Figura 23. Exemplo de Mapeamento de Elemento Complexo (I)**

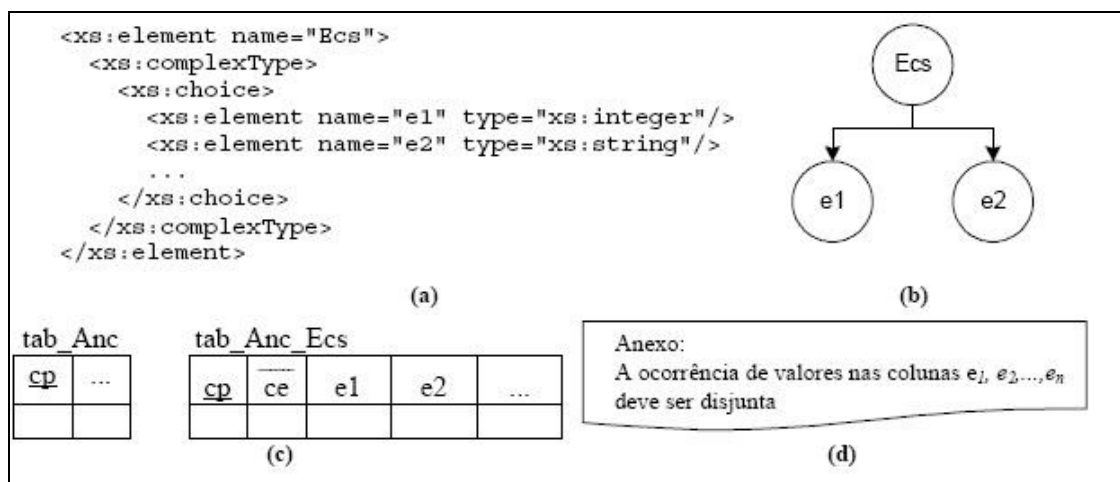
Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema*, e (c) o esquema relacional [CRM05].



**Figura 24. Exemplo de Mapeamento de Elemento Complexo (II)**

Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo com indicador *sequence*, onde (a) é um esquema XML, (b) a representação do elemento no

modelo *XML Schema*, (c) é o esquema relacional, e (d) é o anexo ao esquema relacional descrevendo restrições de integridade [CRM05].



**Figura 25. Exemplo de Mapeamento de Elemento Complexo (III)**

Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo com indicador *choice*, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema*, (c) é o esquema relacional, e (d) é o anexo ao esquema relacional descrevendo restrições de integridade [CRM05].

- **Regra EV (Elemento Vazio):** conforme Maurício [CRM05], a regra EV

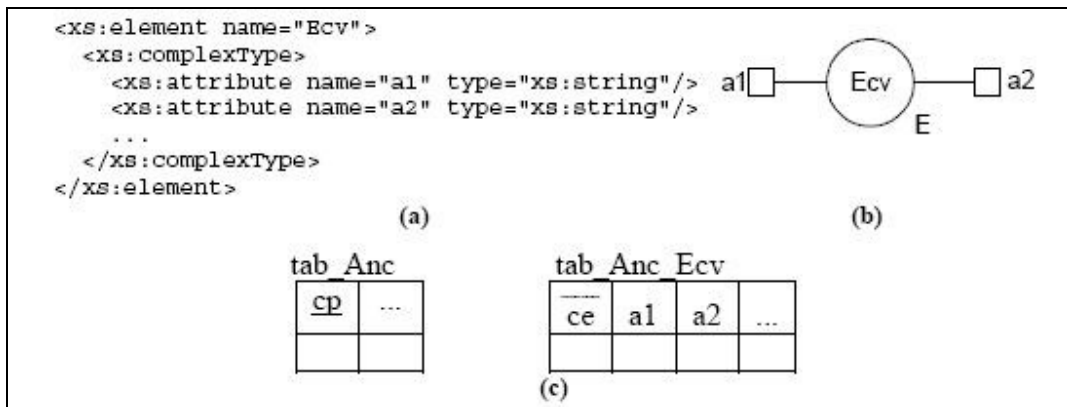
[...] considera que um elemento complexo vazio com atributos pode ser mapeado para uma tabela própria, e caso isto aconteça, a existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento complexo, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido. A regra considera ainda que um elemento complexo vazio sem atributos pode não ter nenhum mapeamento ou então ser mapeado para uma coluna na tabela correspondente ao seu elemento ancestral. Neste segundo caso, o número de ocorrências do elemento é armazenado nesta coluna.

Definição: Um elemento vazio EV de nome Ev: se ES possui um elemento ancestral mapeado para uma tabela tab\_Anc;

- gera uma tabela tab\_Ev;
- gera uma chave estrangeira Ce em tab\_Ev que referencia tab\_Anc e;
- uma coluna de nome ordem\_Ev em tab\_Ev;
- a chave primária de tab\_Ev é o par (ordem\_Ev, Ce) , se EV está contido em um indicador de ordem *sequence* em um elemento ancestral, se EV possui atributos.

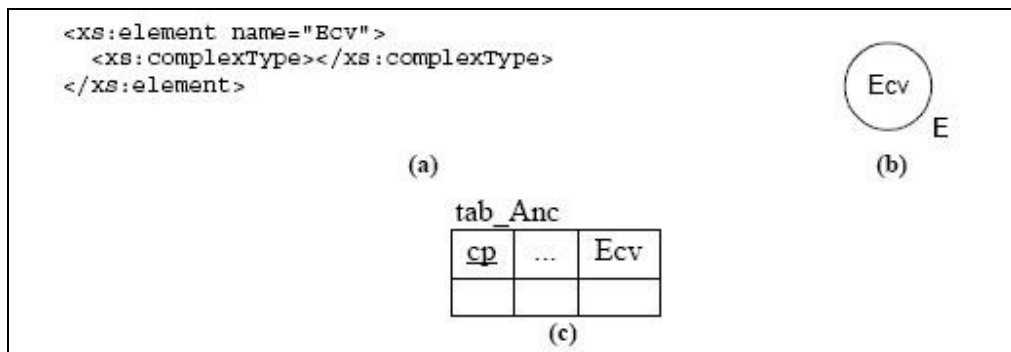
- Ou:
- não gera mapeamento;
  - gera uma coluna numérica em tab\_Anc, que mantém o número de ocorrências de EV, se EV possui um elemento ancestral mapeado para tab\_Anc., se EV não possui atributos (MAURICIO, ano?, páginas ???).

As duas figuras a seguir ilustram melhor essa regra.



**Figura 26. Exemplo de Mapeamento de Elemento Complexo Vazio com atributos**

Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo Vazio com atributos, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema* e (c) é o esquema relacional [CRM05].



**Figura 27. Exemplo de Mapeamento de Elemento Complexo Vazio sem atributos**

Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo Vazio sem atributos, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema* e (c) é o esquema relacional [CRM05].

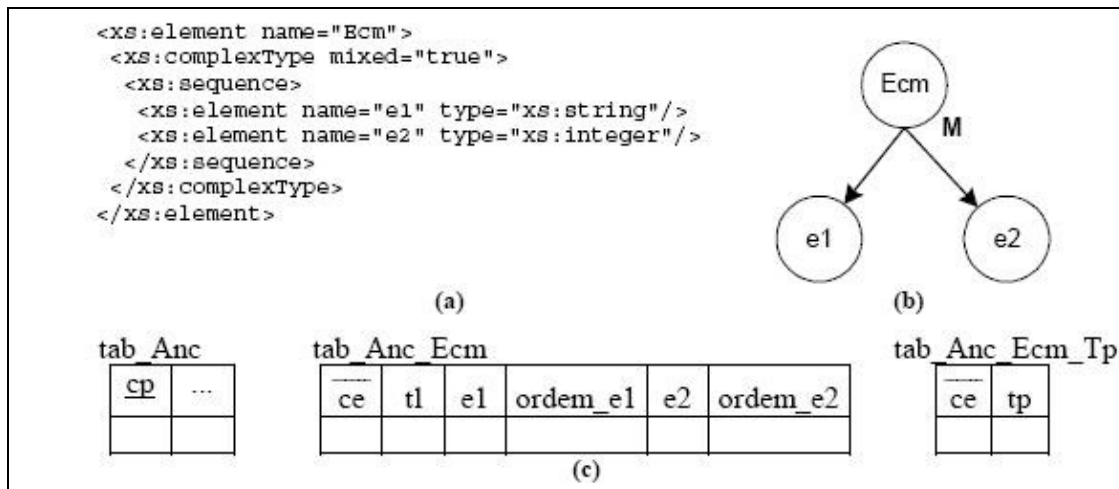
- **Regra ECM (Elemento Complexo do tipo Misto):** conforme Maurício [CRM05], a regra ECM

[...] considera que um elemento complexo de tipo misto pode ser mapeado para uma tabela própria, com ou sem uma coluna de texto longo para armazenar todo o conteúdo do elemento misto (textos e estruturas), e uma tabela que comporta somente a parte textual do elemento. Outra alternativa é o mapeamento do elemento para uma única tabela com uma coluna para armazenar o todo o conteúdo do elemento. A existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento complexo, é também considerada. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido. Considera-se ainda que outras regras são posteriormente aplicadas para o mapeamento dos elementos componentes do elemento misto. Tal tratamento, apesar de gerar redundância no armazenamento dos dados XML, permite que se possa recuperar a estrutura do elemento misto na íntegra (mantida na coluna de texto longo) e, ao mesmo tempo, manter diretamente no esquema relacional as relações hierárquicas do elemento misto com os seus elementos componentes, para facilitar consultas.

Definição: um elemento complexo *EC* de tipo misto *Tm* de nome *Ecm* contendo elementos *e1, e2, ..., en*:

- gera:
  - uma tabela de nome *tab\_Anc\_Ecm* com chave primária *cp* e uma coluna *Ecm* tipo texto longo;
  - uma tabela de nome *tab\_Anc\_Ecm* com chave primária *cp*;
  - uma tabela de nome *tab\_Anc\_Ecm\_Tp* com uma coluna *tp* de tipo texto para armazenar o conteúdo texto puro de *Ecm* e uma coluna chave estrangeira *ce* que referencia *tab\_Anc\_Ecm*;
- Ou:
  - gera uma tabela de nome *tab\_Anc\_Ecm* com chave primária *cp* e uma coluna *Ecm* tipo texto longo;
  - gera uma coluna chave estrangeira *ce* em *tab\_Anc\_Ecm* que referencia *tab\_Anc*, se *Ecm* possui um elemento ancestral *Anc* que tenha sido mapeado para uma tabela *tab\_Anc*.
- gera:
  - uma coluna de nome *ordem\_Ecm* em *tab\_Anc\_Ecm*;
  - a chave primária de *tab\_Anc\_Ecm* é o par (*ordem\_Ecm, ce*), se *EC* está contido em um indicador de ordem *sequence* em um elemento ancestral [CRM05].

A Figura 28 a seguir ilustra melhor essa regra.



**Figura 28. Exemplo de Mapeamento de Elemento Complexo de tipo misto**

Nessa figura observa-se um exemplo de Mapeamento de Elemento Complexo de tipo misto, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema* e (c) é o esquema relacional [CRM05].

- **Regra EAT (Elemento do tipo AnyType):** conforme Maurício [CRM05], a regra EAT

[...] considera que um elemento de tipo *anyType* pode ser mapeado para uma tabela própria com uma coluna para armazenar todo o conteúdo do elemento na forma original, ou uma coluna na tabela correspondente ao seu elemento ancestral, e mais uma tabela para os possíveis atributos do elemento.

A regra considera ainda, a existência de um indicador de ordem *sequence*, na definição do elemento ancestral que contém o elemento *anyType*. Neste caso, um atributo adicional que indica a ordem de ocorrência do elemento é definido.

Definição: um elemento *E* de tipo *anyType* *Ta* de nome *Ea* contendo elementos *e1*, *e2*, ..., *en* e atributos de nomes *a1*, *a2*, ..., *an* com conteúdos *ca1*, *ca2*, ..., *can* respectivamente:

- gera uma tabela de nome *tab\_Anc\_Ea* com chave primária *cp* e uma coluna de tipo texto longo;

Ou:

- gera uma coluna de nome *Ea* de tipo texto longo em *tab\_Anc*, se *E* possui um elemento ancestral *Anc* que tenha sido mapeado para uma tabela *tab\_Anc*;

- gera uma tabela de nome *tab\_Atributos\_Ea* com uma coluna de nome *nomeAtributo* para armazenar os nomes *a1*, *a2*, ..., *an*, outra coluna de nome *conteúdoAtributo* para armazenar *ca1*, *ca2*, ..., *can*, e uma coluna chave estrangeira que referencia tabela que mantém *Ea*.<sup>7</sup>

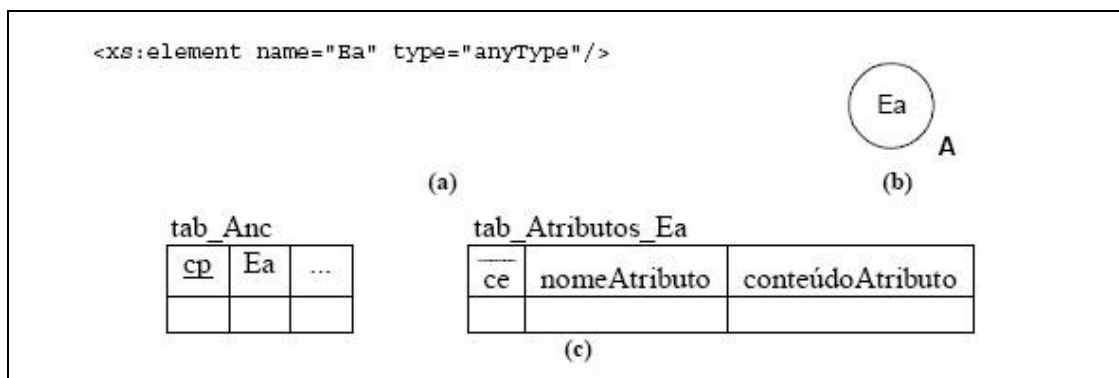
- gera:

- uma coluna de nome *ordem\_Ea* na tabela *tab* que mantém o conteúdo de *E*;



- a chave primária de *tab* é o par (*ordem\_Ea*, *ce*), se *tab* \_ *tab\_Anc.*, se *E* está contido em um indicador de ordem *sequence* em um elemento ancestral [CRM05].

A Figura 29 a seguir ilustra melhor essa regra.

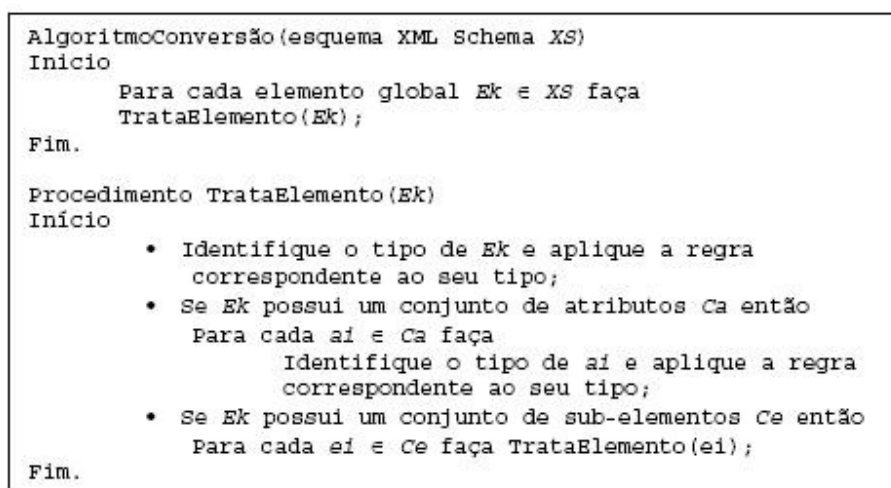


**Figura 29. Exemplo de Mapeamento de Elemento AnyType**

Nessa figura observa-se um exemplo de Mapeamento de Elemento AnyType, onde (a) é um esquema XML, (b) a representação do elemento no modelo *XML Schema* e (c) é o esquema relacional [CRM05].

#### 4.4 Algoritmo

O algoritmo em que se baseia a ferramenta XRCONVERSION para efetuar as conversões é apresentado na Figura 30.



**Figura 30. Algoritmo de conversão [CRM05]**

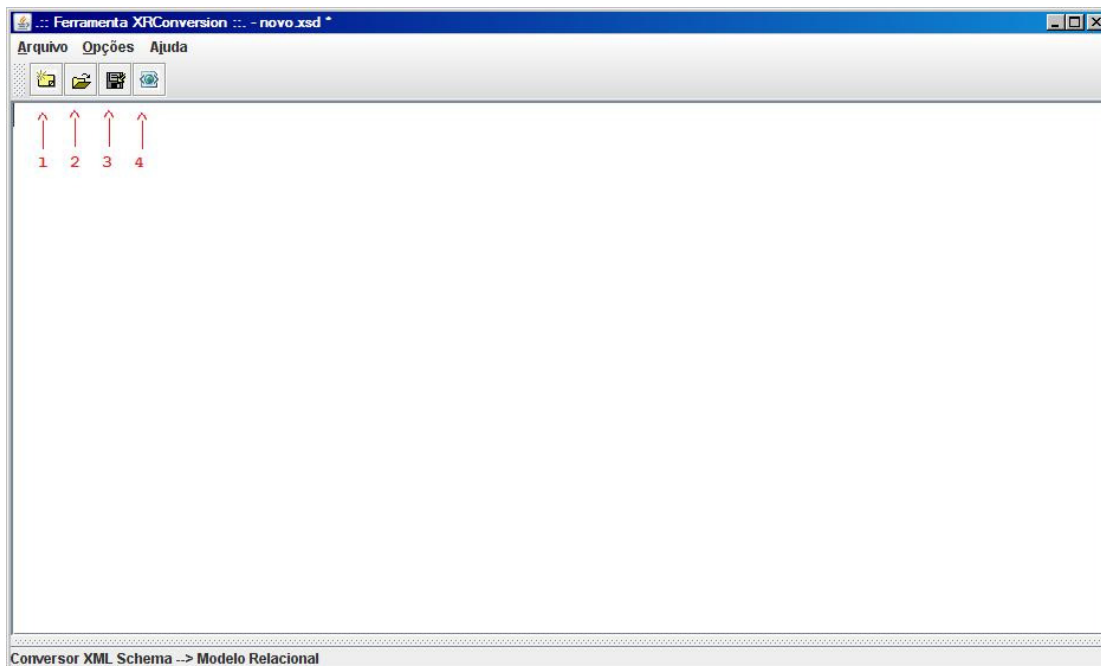
Utilizando recursão, o algoritmo de conversão lê o esquema XSD pré-processado. Para cada elemento global encontrado no XSD, é executado o procedimento *TrataElemento*, também recursivo, passando como parâmetro o elemento global. O procedimento *TrataElemento* identifica o tipo de elemento do parâmetro recebido e encaminha uma regra de mapeamento para o tipo identificado. Neste momento, o *TrataElemento* verifica a existência de atributos e aplica as regras de mapeamento para os atributos. Por último, verifica se há subelementos e, em caso positivo, o *TrataElemento* chama ele mesmo, passando como parâmetro o subelemento.

Nos casos em que se faz necessária a intervenção do usuário para decidir qual é a melhor proposta de mapeamento, o fluxo do programa é direcionado para a interface gráfica questionar qual decisão deve ser tomada. Em seguida, o processamento continua de onde parou. No esquema relacional gerado, as colunas que foram mapeadas a partir de elementos ou atributos indicados como opcionais podem assumir valores nulos e, para aqueles indicados como obrigatórios, as colunas não podem assumir valores nulos.

#### **4.5 Interface gráfica**

A Figura 31 a seguir apresenta a tela principal da ferramenta XRCONVERSION, a qual possui botões com as seguintes finalidades:

- Botão 1: cria novo arquivo;
- Botão 2: abre um arquivo que já está salvo em algum diretório;
- Botão 3: salva um arquivo;
- Botão 4: converte um arquivo.



**Figura 31. Tela principal do programa XRCONVERSION**

Sempre que é passado o mouse em cima dos botões, é aberta uma pequena dica informando a função de cada um. Todas essas funções dos botões podem ser também acessadas pelos menus.

No menu Arquivo, existem os seguintes itens:

- Abrir: mesma função do botão 2. Pode ser acessado pelas teclas Control + A pressionadas ao mesmo tempo;
- Novo: mesma função do botão 1. Pode ser acessado pelas teclas Control + N pressionadas ao mesmo tempo;
- Salvar: mesma função do botão 3. Pode ser acessado pelas teclas Control + S pressionadas ao mesmo tempo;
- Fechar: fecha o programa.

No menu Opções, existe o seguinte item:

- Converter: mesma função do botão 4. Pode ser acessado pelas teclas Control + Y pressionadas ao mesmo tempo.

No menu Ajuda, existe o seguinte item:

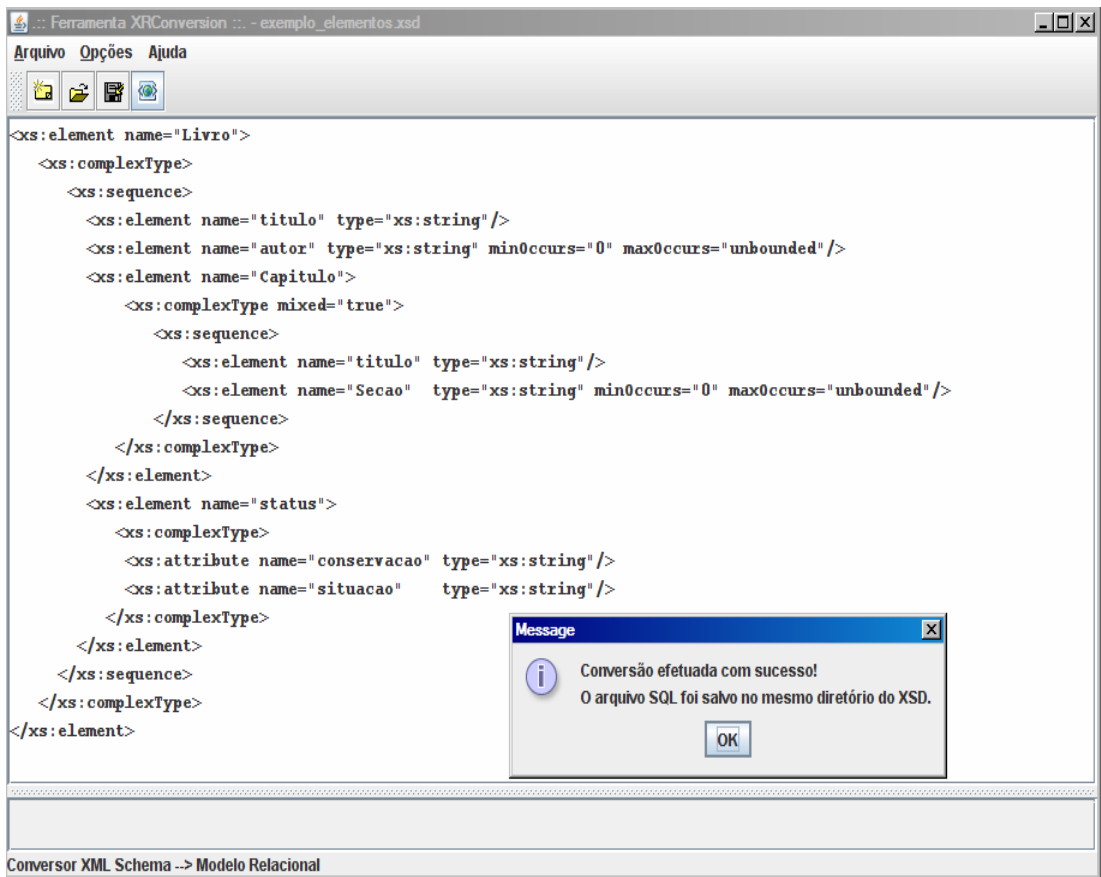
- Sobre: acessa informações sobre a ferramenta.

O fluxo de operação baseia-se no princípio de leitura do esquema de entrada. Para isso, é possível que o usuário utilize o campo texto para digitar o esquema de sua escolha, ou ainda, pode acessar no menu a opção “Abrir”, a qual busca o arquivo já salvo no computador para ler e em seguida executar a conversão. Para que a conversão ocorra, existe a opção “Converter”. Também é possível salvar o arquivo novo que acabou de ser digitado através da opção “Salvar”.

Durante a conversão, a interface abre novas caixas de diálogos para questionar o usuário sobre a decisão a ser tomada nos casos em que houver mais de uma opção de mapeamento. Ao final da conversão, uma nova caixa de diálogo é aberta para informar ao usuário o nome e o local em que foi salvo o *script* SQL que acabara de ser criado.

#### **4.6 Testes de integração com o banco de dados MySQL 5.0**

A Figura 32 a seguir apresenta um exemplo de uso da ferramenta para conversão de elementos e a caixa de diálogo exibida ao término da conversão com a mensagem de confirmação.



**Figura 32. Exemplo de uso da ferramenta XRCONVERSION**

No exemplo da Figura 32, o nome do XSD de entrada é exemplo\_elementos.xsd e o nome do arquivo de saída é exemplo\_elementos.sql. A Figura 33 a seguir apresenta o *script* SQL referente ao exemplo da Figura 32.

```

CREATE TABLE tab_Livro(
ID_LIVRO INTEGER NOT NULL,
titulo VARCHAR(80) NULL,

PRIMARY KEY(ID_Livro)
);

CREATE TABLE tab_Autor(
ID_Livro INTEGER NOT NULL,
autor VARCHAR(80) NULL,
ordem_autor INTEGER NOT NULL,

PRIMARY KEY(ID_Livro),
FOREIGN KEY(ID_Livro) REFERENCES tab_Livro(ID_Livro)
);

CREATE TABLE tab_Capitulo(
ID_Capitulo INTEGER NOT NULL,
titulo VARCHAR(80) NULL,
ID_livro INTEGER NOT NULL,
ordem_capitulo INTEGER NOT NULL,

PRIMARY KEY(ID_Capitulo),
FOREIGN KEY(ID_Livro) REFERENCES tab_Livro(ID_Livro)
);

CREATE TABLE tab_Capitulo_Texto(
ID_Capitulo INTEGER NOT NULL,
texto LONG VARCHAR NULL,
ordem_capitulo_texto INTEGER NOT NULL,

PRIMARY KEY(ID_Capitulo),
FOREIGN KEY(ID_Capitulo) REFERENCES
tab_Livro(ID_Capitulo)
);

CREATE TABLE tab_Livro_status(
ID_livro INTEGER NOT NULL,
ordem_status INTEGER NOT NULL,
conservacao VARCHAR(80) NULL,
situacao VARCHAR(80) NULL,

PRIMARY KEY(ID_Livro),
FOREIGN KEY(ID_Livro) REFERENCES tab_Livro(ID_Livro)
);

```

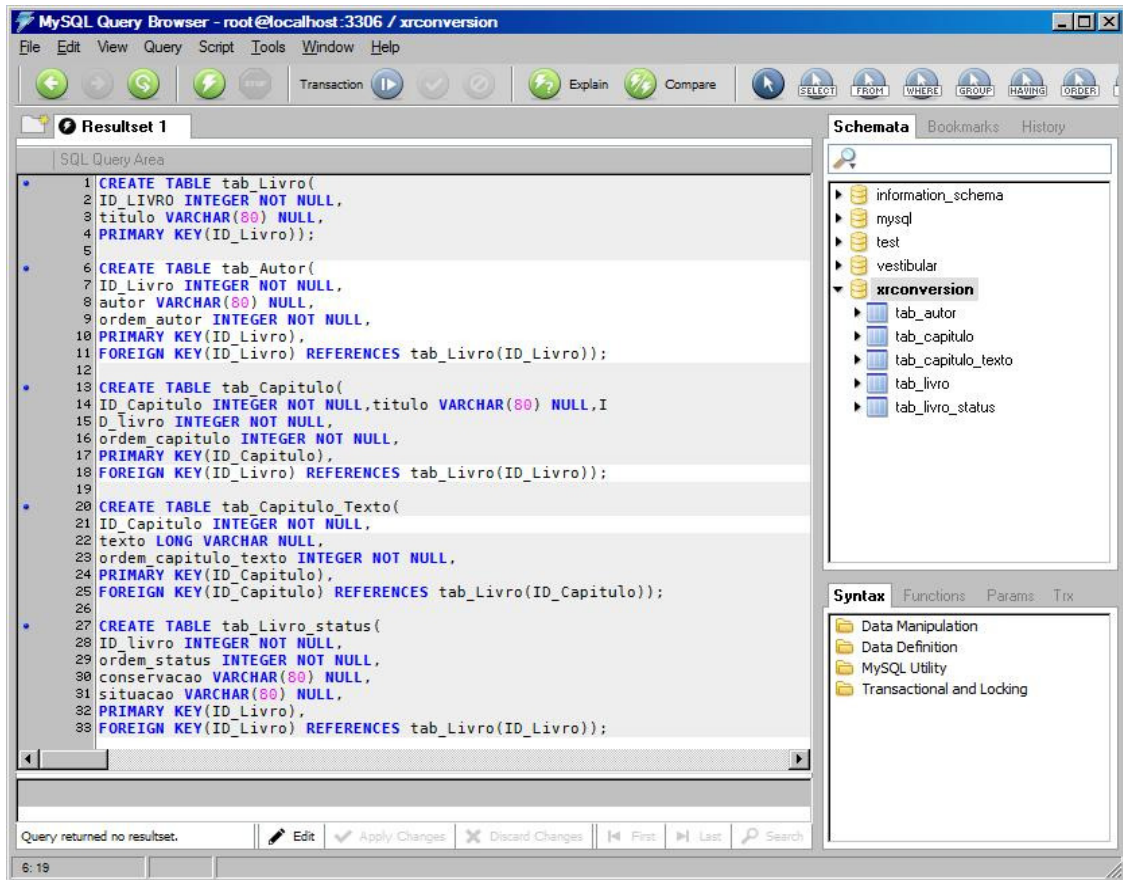
**Figura 33. Resultado da conversão**

A interface leu o conteúdo e o enviou ao módulo de conversão. Como o esquema já estava pré-processado, ele foi encaminhado para o algoritmo principal, onde se verificou:

- Elemento Livro: é um elemento composto, sendo criada uma tabela para ele segundo a regra EC. Ele é composto por dois elementos simples do tipo nativo: título e autor;
- Elemento Título: é do tipo nativo. Seu mapeamento foi realizado conforme a regra ESTN, que gerou a coluna título na tabela Livro;

- Elemento Autor: é do tipo nativo. Seu mapeamento foi realizado conforme a regra ESTN, que gerou a tabela Tab\_Autor. A Tab\_Autor possui uma coluna de referência à Tab\_Livro e uma indicação de ordem do autor no elemento ancestral, uma vez que o conteúdo do elemento Livro é definido com um indicador de ordem *sequence*. A criação dessa tabela foi necessária pois um livro pode ter um ou mais autores associados;
- Elemento Capítulo: a regra ECM é responsável pela conversão do elemento misto Capítulo, que gerou uma tabela Tab\_Capítulo. Essa tabela mantém os dados dos capítulos e referências aos livros aos quais eles pertencem. Foi criada também uma outra tabela, chamada Tab\_Capítulo\_Texto, cuja função é manter de forma ordenada os componentes textuais não-estruturados dos capítulos;
- Elemento Status: a regra EV converte o elemento vazio Status em uma tabela própria, cujos atributos tornam-se colunas da tabela, a qual mantém, além dos atributos, informações de ordem no elemento ancestral.

Em seguida, foi submetido o *script* SQL no banco de dados MySQL versão 5.0. Os resultados dos testes foram positivos, as tabelas foram criadas com sucesso, e os testes podem ser verificados a seguir na Figura 34.



**Figura 34. Criação das tabelas no banco MySQL 5.0**

No campo de texto ao centro da Figura 34 está o *script* resultante da ferramenta XRCONVERSION, para a criação de tabelas em conformidade com o esquema de entrada. Na coluna à direita estão as tabelas que foram criadas após submissão do *script* ao banco de dados (tab\_autor, tab\_capitulo, tab\_capitulo\_texto, tab\_livro e tab\_livro\_status). Neste momento, as tabelas já estão prontas para serem utilizadas pelas aplicações relacionadas. Foi possível, dessa forma, validar a efetividade da ferramenta XRCONVERSION.



## 5 CONCLUSÃO

O uso de documentos XML para interoperabilidade de sistemas, assim como o uso de esquemas *XML Schema* para controlar e validar documentos XML, estão consolidados nos dias de hoje. O mesmo pode-se dizer dos bancos de dados relacionais. Os mecanismos de conversão de dados XML (descritos *XML Schema*) para o modelo relacional são necessários para viabilizar a transferência de informações entre sistemas.

A ferramenta XRCONVERSION pode ser utilizada nesse contexto e, de preferência, por um desenvolvedor que conheça seu modelo de negócio, pois a abordagem permite que ele tenha as decisões mais adequadas mediante a interação com a interface gráfica. As regras de mapeamento que foram implementadas são bem detalhadas, pois contemplam grande parte dos conceitos do modelo *XML Schema*, se comparadas com trabalhos relacionados. A abordagem é também considerada flexível, pois propõe, em diversas regras, várias opções para o mapeamento de um conceito do modelo *XML Schema*.

Por fim, o trabalho contribuiu para concretizar os conhecimentos adquiridos durante o curso de graduação, especialmente nas disciplinas de banco de dados e programação, onde foi utilizada a linguagem de programação JAVA. Como trabalhos futuros, estão previstas melhorias na ergonomia da interface e, complementando, um menu chamado *help*, com dicas de uso, bem como um melhor tratamento das mensagens de erro, ajudando o usuário a entender as falhas que o sistema encontra. Outro trabalho interessante seria a integração de técnicas de inteligência artificial em aprendizagens de conversão XML para o modelo relacional.

## REFERÊNCIAS BIBLIOGRÁFICAS

[ELM05] ELMASRI, R.; NAVATHE S. B. **Sistemas de banco de dados**. 4ª ed. São Paulo: Addison Wesley, 2005.

[W3C07] W3C. Disponível em: <<http://www.w3c.org>>. Acesso em: 2 nov. 2007. Site do World Wide Web Consortium, consórcio responsável pela metodologia XML.

[CRM05] MAURICIO, C.R.M. **Uma proposta de Mapeamento do Modelo XML Schema para o Modelo Relacional**. Dissertação (Mestrado em Ciências da Computação) – PPGCC/Universidade Federal de Santa Catarina, [2007?]. Disponível em: <<http://www.grupobd.inf.ufsc.br>>. Acesso em: 10 nov. 2007.

[MAR05] MARTINS, Patrícia; LAENDER, Alberto H. F. **Mapeamento de definições XML Schema para SQL:1999**. Minas Gerais: DCC/UFMG, 2005. Disponível em: <[http://www.sbbd-sbes2005.ufu.br/arquivos/artigo-07-novo\\_Martins.pdf](http://www.sbbd-sbes2005.ufu.br/arquivos/artigo-07-novo_Martins.pdf)>. Acesso em: 10 nov. 2007.

[STE06] VIAPIANA, Stevan F.; DORNELES, Carina F. **Geração automática de esquemas relacionais a partir de esquemas XML**. RS, Passo Fundo: Instituto de Ciências Exatas e Geociências – UPF, 2006. Disponível em: <<http://ccet.ucs.br/erbd2007/artigos/26119.pdf>>. Acesso em: 10 nov. 2007.

[SUN07] JAVA TECHNOLOGY. Disponível em: <<http://java.sun.com/>>. Acesso em: 12 set. 2007. Site para *download* da ferramenta *JAVA Technology*.

[JAX07] JAXP 1.4.2. Disponível em: <<http://jaxp.dev.java.net/1.4>>. Acesso em: 17 set. 2007. Site para *download* da ferramenta JAXP 1.4.2.

[ORA07] Oracle Jdeveloper 10. Disponível em: <<http://www.oracle.com>>. Acesso em: 10 mar. 2007. Site para *download* da ferramenta Oracle Jdeveloper 10.

[MSQ08] MySQL 5.0. Disponível em: <<http://www.mysql.com>>. Acesso em: 22 jun. 2008. Site para *download* da ferramenta MySQL 5.0.