UNIVERSIDADE FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA
MESTRADO EM INFORMÁTICA

MAPEAMENTO DE ATRIBUTOS COMPLEXOS E MULTIVALORADOS NA EXTRAÇÃO DE ESQUEMAS UTILIZANDO *XML*

CURITIBA – PR
AGOSTO – 2001

JOSÉ SIMÃO DE PAULA PINTO

MAPEAMENTO DE ATRIBUTOS COMPLEXOS E MULTIVALORADOS NA EXTRAÇÃO DE ESQUEMAS UTILIZANDO *XML*

Proposta de dissertação de mestrado apresentada à Universidade Federal do Paraná, Programa de Pós-Graduação em Informática, Mestrado em Informática, como pré-requisito à obtenção do título de mestre em informática.

Orientador: Prof. Dr. Marcos Sfair Sunye

CURITIBA – PR

AGOSTO – 2001

TERMO DE APROVAÇÂO

JOSÉ SIMÃO DE PAULA PINTO

MAPEAMENTO DE ATRIBUTOS COMPLEXOS E MULTIVALORADOS NA EXTRAÇÃO DE ESQUEMAS UTILIZANDO *XML*

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: Prof. Dr. Marcos Safir Sunye
Setor de Ciências Exatas, UFPR

Profa. Dra. Laura Sánches Garcia
Setor de Ciências Humanas, Letras e Artes, UFPR

Profa. Dra. Maria Salete Marcon Gomes Vaz
Setor de Ciências Agrárias e de Tecnologia, UEPG

"*XML* can do for data what Java has done for programs, which is to make the data both platform-independent and vendor-independent."[1]

[1] JON BOSAK, **MEDIA-INDEPENDENT PUBLISHING: FOUR MYTHS ABOUT "XML"**, IEEE COMPUTER, VOLUME 31, NUMBER 10, OCTOBER 1998

AGRADECIMENTOS

Agradeço profundamente a todos que coloboraram com o desenvolvimento deste trabalho, seja de forma técnica, seja na forma do apoio recebido nas horas difíceis, nas sugestões e estímulos, em especial:

Ao meu colega Prof. Ezequiel Gueiber, pelos fontes do Sistema *VIQUEN*,

Ao meu orientador, Prof. Marcos Sfair Sunye, pela sua paciência e pelas inúmeras idéias, *e-mails* e sugestões recebidas.

Este trabalho é dedicado
À minha esposa, Katy, e ao meu filho, Henrique.

SUMÁRIO

LISTA DE TABELAS vi
LISTA DE QUADROS vii
LISTA DE FIGURAS viii
LISTA DE ABREVIATURAS E SIGLAS x
RESUMO xi
ABSTRACT xii
1 INTRODUÇÃO 13
1.1 O SISTEMA VIQUEN 13
1.2 DESCRIÇÃO DO PROBLEMA 13
1.2.1 LIMITAÇÃO DE ACESSO 16
1.3 OBJETIVO 17
1.4 CONTRIBUIÇÃO DO TRABALHO 18
2 TRABALHOS RELACIONADOS 19
2.1 CONCLUSÕES 23
3 INTEGRAÇÃO DE ESQUEMAS 25
3.1 O MODELO ENTIDADE RELACIONAMENTO COMPLEXO (*ERC+*) 26
4 A LINGUAGEM DE MARCAÇÃO *XML* 29
4.1 CONCEITOS DE *XML* 29
4.2 DIFERENÇAS ENTRE *HTML* e *XML* 32
4.3 COMPOSIÇÃO DE UM DOCUMENTO *XML* 34
4.3.1 O PRÓLOGO (*PROLOG*) DE UM DOCUMENTO *XML* 35
4.3.2 CONTEÚDO DE UMA INSTÂNCIA DE DOCUMENTO *XML* 35
4.3.3 REGRAS DE FORMAÇÃO 37
4.4 TIPOS DE PROCESSADORES (*PARSERs*) 39
4.4.1 TIPO 1 - *NON-VALIDATING PARSER* 40
4.4.2 TIPO 2 - *VALIDATING PARSER* 40
4.4.3 TIPO 3 - *TREE-BASED PARSING* 40
4.4.4 TIPO 4 - *EVENT-BASED PARSING* 41
4.5 DEFINIÇÕES DO DOCUMENTO: *DTD* E *SCHEMA* 42
4.5.1 EXEMPLOS EXTRAÍDOS A PARTIR DA BASE DO SISTEMA *VIQUEN* 42
4.6 LINGUAGENS DE TRANSFORMAÇÃO *XSL* E *XSLT* 46

4.7 CONSULTAS POR MEIO DA *XML QUERY LANGUAGE* (*XQUERY*) 47
4.7.1 A NOTAÇÃO *XQUERY* 48
4.8 EXEMPLO DE USO DA LINGUAGEM *XQUERY* (*W3C*) 52
4.8.1 PROJEÇÃO 54
4.8.2 PROJEÇÃO DE ATRIBUTOS 54
4.8.3 PROJEÇÃO DE VALORES DE ATRIBUTOS 54
4.8.4 PROJEÇÃO DE ATRIBUTOS MULTIVALORADOS 54
4.8.5 ITERAÇÃO 54
4.8.6 SELEÇÃO 55
4.8.7 JUNÇÕES (“JOINS”) 56
4.8.8 REFERÊNCIAS PARA NÓS DE DOCUMENTOS 57
4.8.9 REESTRUTURAÇÃO E REAGRUPAMENTO DOS NÓS DE UM DOCUMENTO 57
4.8.10 ORDENAÇÃO E RECUPERAÇÃO ORDENADA 58
4.8.11 CLASSIFICAÇAO (“SORT”) 58
4.8.12 AGREGAÇÃO 58
4.8.13 FUNÇÕES 59
5 O PROTÓTIPO X-VIQUEN 61
5.1 EXTRAÇÃO DO ESQUEMA DA BASE DE DADOS 62
5.2 ATRIBUTOS COMPLEXOS / MULTIVALORADOS 63
5.3 FUNCIONAMENTO 66
5.3.1 EXEMPLO DE DOCUMENTO 67
5.4 FORMAÇÃO DA REPRESENTAÇÃO GRÁFICA 69
5.5 FORMULAÇÃO DE CONSULTAS 71
5.6 EXIBIÇÃO DOS RESULTADOS DA CONSULTA 75
6 CONCLUSÕES 78
6.1 A ABORDAGEM *X-VIQUEN VERSUS* OS MODELOS PESQUISADOS 79
6.2 TRABALHOS FUTUROS 80
7 REFERÊNCIAS BIBLIOGRÁFICAS 82
ANEXOS 85

LISTA DE TABELAS

Tabela 1 - Marcações para controle de nulos em documentos *XML* 36

Tabela 2 - Notações para controle de cardinalidade em documentos *XML* 36

Tabela 3 - Declarações da sintaxe *XML* 39

Tabela 4 - Mapeamento de tipos *XML* para tipos padrão de bases de dados. 45

LISTA DE QUADROS

Quadro 1 – Principais diferenças entre as linguagens de marcação *HTML* e *XML* ..33

Quadro 2 - Comparação entre *CSS* e *XSL* para controlar a apresentação de documentos *XML* 46

LISTA DE FIGURAS

Figura 1 – Modelo lógico do protótipo do Sistema *VIQUEN*, obtido a partir de ferramenta comercial 14
Figura 2 – Representação de um atributo complexo no Sistema *VIQUEN*. 15
Figura 3 – Conjunto de entidades exibidas pelo protótipo do Sistema *VIQUEN*. 15
Figura 4 – Atributos complexos/multivalorados. 16
Figura 5 – Limitação da possibilidade de conexões de uma ferramenta comercial (*ER-Win*) 17
Figura 6 – Representação de uma consulta em *XML-GL*. 21
Figura 7 – Poder expressivo de seis linguagens de consulta *XML* de acordo com BONIFATI e LEE. 22
Figura 8 – A *Interface* de consulta do projeto *EquiX*. 23
Figura 9 – Atributos que não podem ser diretamente representados em uma base relacional. 25
Figura 10 – Atributo complexo/multivalorado representado no modelo relacional. 26
Figura 11 – Exemplo de modelo *ERC+* 27
Figura 12 – Tela da ferramenta *SUPER*, que implementa o modelo *ERC+*. 28
Figura 13 - Diferenças entre as linguagens de marcação *HTML* e *XML* 34
Figura 14 – Exibição da estrutura de um documento *XML* e sua representação em uma estrutura de árvore. 34
Figura 15 – O conceito de documento bem formado (*well formed*) 37
Figura 16 – O conceito de um documento válido (*valid*). 38
Figura 17 – A estrutura hierárquica e operações possíveis no modelo *DOM*. 41
Figura 18 – Fragmento de arquivo *DTD* para a base de dados proposta, sem utilizar a notação de pontos. 43
Figura 19 – Fragmento de arquivo *XML* para a base de dados proposta utilizando *DTD* sem a notação de pontos. 43

Figura 20 – Fragmento de arquivo *DTD* para a base de dados proposta, utilizando notação com pontos. 44
Figura 21 – Fragmento de arquivo *XML* para a base de dados proposta utilizando notação com pontos. 45
Figura 22 – Transformação de um documento *XML* em um documento *HTML* por meio da utilização de transformações *XSL* 47
Figura 23 – Documento de amostra do *W3C* para demonstração da linguagem *XQuery* 53
Figura 24 – Notação em *XQuery* e criação de variáveis. 53
Figura 25 – Exemplo para demonstrações de junções (“joins”) na linguagem “XQuery” 56
Figura 26 - Diagrama funcional do protótipo do Sistema *VIQUEN* 61
Figura 27 - Proposições *is-a* e *maybe-a* do modelo *ERC+* 63
Figura 28 – Utilização da interface *dbxml* para geração de arquivos *XML* a partir de bases relacionais. 66
Figura 29 - Um documento de integração possui conhecimento sobre os documentos integrados. 67
Figura 30 - Visão geral do protótipo *X-VIQUEN* 70
Figura 31 – Esquema *ERC+* gerado 71
Figura 32 – Seleção de objetos por meio do Sistema *VIQUEN* 71
Figura 33 – Subesquema gerado pela seleção de objetos pelo usuário. 71
Figura 34 – Execução de consultas às bases originais pelo protótipo *X-VIQUEN*. 72
Figura 35 – Retorno dos conjuntos de dados e seu mapeamento. 73
Figura 36 - O *parser X-VIQUEN* chamando *APIs* do Sistema *VIQUEN*. 75
Figura 37 – Exibição dos resultados da consulta no Sistema *VIQUEN*. 76
Figura 38 - Representação aninhada para atributos complexos sugerida por ERWIG(2000). 77
Figura 39 - Tela sugerida para o protótipo do sistema *X-VIQUEN++*. 81

LISTA DE ABREVIATURAS E SIGLAS

CASE – Computer aided software engineering

CDF - Channel Definition Format

CSS - Cascading Style Sheets (CSS1, CSS2, CSS3 - Níveis 1, 2, 3)

DOM - Document Object Model

DSSSL - Dynamic Style Semantics and Specification Language

DTD – Document type definitions

ERC+ - (Modelo) Entidade Relacionamento Complexo

HTML - HyperText Markup Language

MathML - Mathematical Markup Language

ORACLE – Nome de um SGBD comercial e também razão social de seu fabricante

RDF - Resource Description Framework; linguagem para escrita de metadados

SCHEMA – Esquema. Descrição do conteúdo de uma base de dados

SGBD – Sistema Gerenciador de Bases de Dados

SGML - Standard Generalized Markup Language (pré-web)

TAG – Marca; marcação; declaração de comando ou atributo

W3C – World Wide Web Consortium

XHTML - Extensible HyperText Markup Language

XLink, XPointer - Extensible Link e Extensible Pointer Languages

XLL - Extensible Link Language

XML – Extensible Markup Language

XML Schema – Extensão dos DTDs; permitem modelagem, herança, tipos de dados

XSL - Extensible Style Language

XSLT- Extensible Stylesheet Language Transformations

RESUMO

Este trabalho aborda uma maneira de mapear atributos complexos e/ou multivalorados de um ambiente visual para banco de dados relacionais, usando *XML*. Estes atributos usualmente surgem após a aplicação de uma metodologia de integração, como o modelo *ERC+*, independentemente de existirem nos modelos originais.

O mapeamento foi produzido em bases já integradas, usando arquivos *DTD* para reter o conhecimento sobre as bases pré-integradas, e uma ferramenta para consultas visuais desenvolvida no protótipo do Sistema *VIQUEN*, que gera a visualização do esquema utilizando o modelo *ERC+*.

As consultas e seus resultados são visualizadas em ambiente gráfico.

Expressar consultas em um ambiente gráfico possibilita a usuários comuns, não especializados em informática, criar suas próprias consultas, usando uma ferramenta que esconde detalhes complexos de bancos de dados e não exige conhecimento sobre a linguagem de consulta *SQL*.

Palavras-chave: *XML*; integração de esquemas; consultas em ambiente visual; *QBE*; *ERC+*.

ABSTRACT

This work describes a way to map complex and / or multivalued attributes from a visual query environment to relational databases, using XML. These attributes are usually produced after the application of integration methodologies, like ERC+, whenever exists in original databases.

The mapping was produced in databases already integrated, using XML DTD files to retain knowledge about pre-integration databases, and a tool, called VIQUEN, that generates a schema visualization by use the ERC+ model.

Queries and their results are both expressed in a visual environment.

Express queries in a visual environment enables common users to create your own queries, since the tool hide complex database details and don't demand user knowledge about SQL statements.

Keywords: *XML; schema integration; visual queries; QBE; ERC+.*

1 INTRODUÇÃO

A grande profusão de SGBDs baseados no modelo relacional existente no mercado, e a eventual necessidade da integração de esquemas[1] neles construídos, criam a demanda por ferramentas para a visualização destes esquemas. Por outro lado, o poder semântico do modelo relacional deixa a desejar, seja qual for o método de integração adotado (BATINI e LENZERINI, 1986 e KLAS e outros, 1994). Baseado nestes fatos foi elaborado o Sistema *VIQUEN*[2] (GUEIBER, 2001), que tem como objetivo extrair um esquema no modelo *ERC+*[3] a partir de um banco de dados relacional.

1.1 O SISTEMA VIQUEN

O Sistema *VIQUEN* é um sistema que permite a visualização de informações de um banco de dados por meio do Modelo *ERC+* (SPACCAPIETRA e outros, 1989). O projeto foi implementado para ser utilizado em conjunto com o banco de dados comercial *ORACLE*©, e faz uso de cursores para possibilitar o mapeamento de atributos multivalorados no SGBD. Sua utilização é restrita a este SGBD (GUEIBER, 2001).

1.2 DESCRIÇÃO DO PROBLEMA

Na integração de esquemas, torna-se necessária a existência de uma ferramenta visual que facilite o processo. Esta ferramenta será responsável pela exibição dos "achados" dos SGBDs a integrar.

Embora existam trabalhos relacionados à representação de modelos *ER*[4] em bancos de dados relacionais (MAROKOWITZ, 1992), no processo inverso as ferramentas comerciais existentes no mercado são específicas para alguns tipos de

---

[1] Em uma base de dados o ESQUEMA é a descrição de seu conteúdo.

[2] *Visual Query Environment* - Ambiente de consulta visual

[3] Entidade-Relacionamento Complexo

[4] Entidade-Relacionamento

SGBDs, são limitadas aos drivers disponíveis fornecidos, são incapazes de representar / tratar atributos multivalorados, uma representação no modelo *ERC+*, ou uma combinação destas limitações.

A Figura 1 representa o modelo lógico do banco de dados utilizado no protótipo original do Sistema *VIQUEN*.

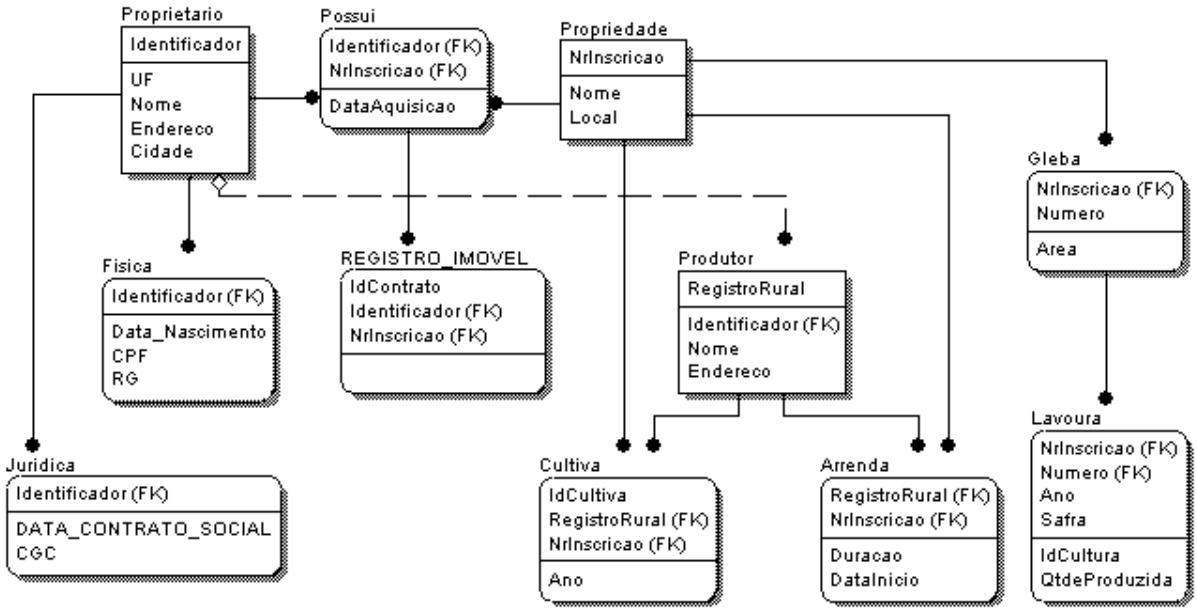

Figura 1 – Modelo lógico do protótipo do Sistema *VIQUEN*, obtido a partir de ferramenta comercial

A figura foi obtida a partir de uma ferramenta comercial[5]. Apesar dela corresponder ao modelo lógico da base de dados em questão, notamos que o poder de expressão é pequeno. O modelo representa somente as ligações, que são formadas a partir das definições de chaves primárias e estrangeiras existentes na base.

Para constraste, observemos a Figura 2, obtida pelo Sistema *VIQUEN*, que demonstra a relação existente entre os objetos PROPRIEDADE, GLEBA e LAVOURA: GLEBA é um atributo complexo de PROPRIEDADE, e LAVOURA um atributo complexo de GLEBA (GUEIBER, 2001).

[5] Foi utilizado o produto Erwin, da Logic Works.

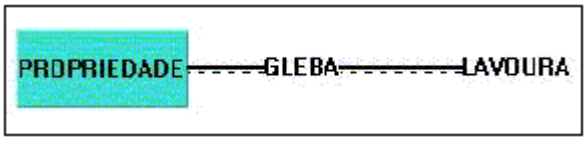

Figura 2 – Representação de um atributo complexo no Sistema *VIQUEN*.

Desta maneira podemos obter uma representação que realmente espelhe as entidades do mundo real, conforme mostrado na Figura 3, também obtida por meio do Sistema *VIQUEN*.

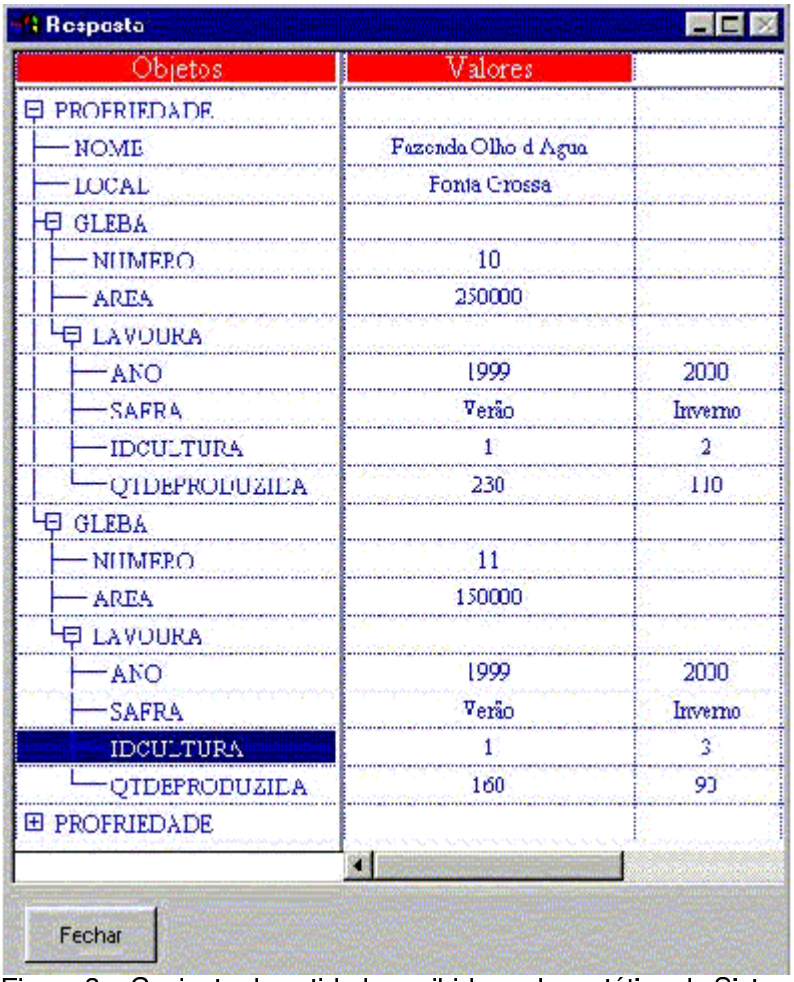

Figura 3 – Conjunto de entidades exibidas pelo protótipo do Sistema *VIQUEN*.

Notamos nesta figura que a entidade PROPRIEDADE possui os atributos NOME, LOCAL e GLEBA. GLEBA é um atributo multivalorado, composto de NÚMERO, ÁREA e LAVOURA, que por sua vez também é multivalorado.

Além do caso mostrado poderão ocorrer casos em que a representação de um atributo complexo (não atômico) possa ser composta de atributos que poderão estar presentes em quantidades variáveis. Embora possamos

intuitivamente pensar nestes atributos em termos de tabelas separadas, é desejável a representação semântica como entidade única. Por exemplo, a representação de uma base de dados de filmes, no qual um nome do diretor corresponde a várias obras e cada obra a vários atores (ABITEBOUL e outros, 1995), ou a representação dos atributos de uma pessoa, conforme sugerido por PARENT e SCCAPAPIETRA (1989) e exibido na Figura 4.

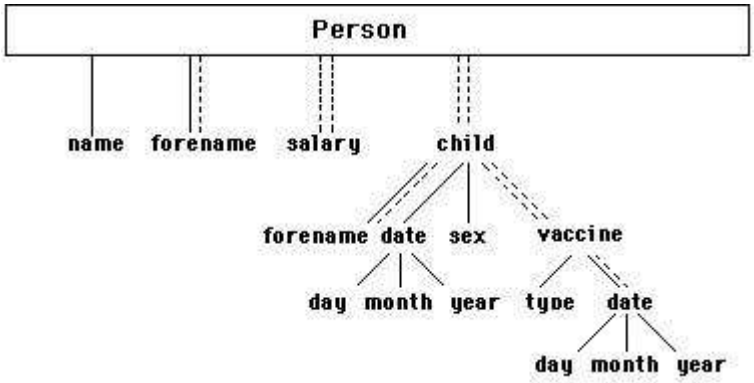

Figura 4 – Atributos complexos/multivalorados.

1.2.1 LIMITAÇÃO DE ACESSO

Nas ferramentas comerciais, a obtenção do diagrama que representa a base, seguindo algum tipo de modelo físico ou lógico, é efetuada a partir de conexão direta com a base de dados. A ferramenta efetua a leitura do esquema existente e, a partir de regras internas, constrói um modelo gráfico representando os objetos encontrados e seus relacionamentos. A construção, porém, só poderá ser efetuada para uma das bases de dados para a qual a ferramenta possui um *driver* disponível, conforme exibido na Figura 5.

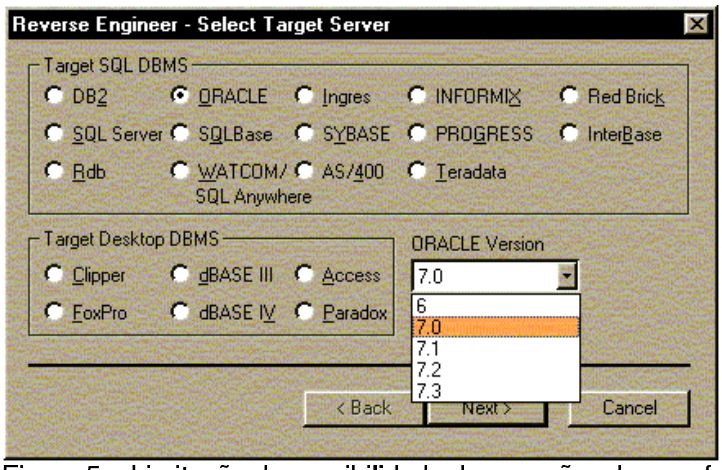

Figura 5 – Limitação da possibilidade de conexões de uma ferramenta comercial (*ER-Win*).

A limitação da possibilidade de acesso somente àquelas bases que possuem *drivers* disponíveis também limita a possibilidade de expansão do uso, já que para cada nova versão de bases de dados lançada no mercado, obrigatoriamente, teremos que atualizar também a ferramenta *CASE*.

Torna-se necessária, então, a existência de uma ferramenta capaz de efetuar a visualização de esquemas de quaisquer SGBDs, independentemente da existência de *drivers* de comunicação ou de sua organização (existência / inexistência de atributos multivalorados, por exemplo), e que possua um poder semântico maior em sua representação do modelo de dados.

1.3 OBJETIVO

Neste trabalho pretende-se utilizar a interface gráfica desenvolvida no Sistema *VIQUEN* (GUEIBER, 2001) e ampliá-la para a utilização com quaisquer SGBDs relacionais. Para isso será implementada uma interface *XML* (padrão do *World Wide Web Consortium, W3C*). Pela utilização de *XML*, e dada a possibilidade de mapeamento de objetos para esta linguagem de marcação, pode-se também utlizar a ferramenta para a integração quaisquer bases de dados que suportem o modelo definido para a linguagem *SQL*.

A representação *XML* foi escolhida pela sua grande flexibilidade e poder semântico. Com muitas implementações derivadas, a *XML* dá suporte a várias características importantes do ambiente pretendido, existindo uma representação

específica para esquemas (BROWN e outros, 2001, KLARLUND e outros, 2000). Sua principal característica é a representação dados e metadados, ou seja, sua habilidade na representação do conteúdo e sua descrição.

1.4 CONTRIBUIÇÃO DO TRABALHO

Com este trabalho pretende-se apresentar uma maneira de realizar a leitura e representação de informações de esquema de uma base de dados (CONNOLLY, 1996) no modelo *ERC+,* utilizando a linguagem de marcação *XML* de forma a possibilitar a visualização de esquemas vindos de bases heterogêneas em um modelo comum. Isto possibilitará a geração de subesquemas, para os quais será exprimida uma consulta à(s) base(s) de dados correspondente(s), sendo que o resultado das consultas será exibido em forma gráfica.

Também será demonstrada a aplicação da linguagem de marcação *XML* e suas extensões, em especial o *XML Schema* (HAROLD, 1999) e os arquivos de definição de documentos (*DTD*) ao domínio de integração de bancos de dados.

Após esta introdução do primeiro capítulo, o capiítulo dois deste trabalho mostrará alguns trabalhos existentes relacionados com a abordagem escolhida. No capítulo três aborda-se a metodologia *ERC+* para a integração de esquemas de bancos de dados e no capítulo quatro a linguagem de marcação *XML*.

O capítulo 5 descreve o protótipo *X-VIQUEN*, correspondente ao núcleo deste trabalho. Foi adicionada a letra "X" ao nome do Sistema *VIQUEN*, desenvolvido por Ezequiel Gueiber (GUEIBER, 2001), de maneira a representar a utilização da linguagem de marcação *XML* junto ao Sistema *VIQUEN*.

O capítulo seis apresenta uma comparação do trabalho desenvolvido com alguns dos trabalhos relacionados expostos no capítulo dois e também apresenta a conclusão do presente trabalho.

2 TRABALHOS RELACIONADOS

Devido a seu grande poder expressivo, a utilização de *XML* e suas variantes tem se generalizado e aproximado cada vez mais dos estudos na área de bases de dados e suas problemáticas para a representação e recuperação de dados. Desta maneira, existe um grande número de referências a pesquisas realizadas com linguagens de consulta para *XML* e, em menor número, formulação de tais consultas em um ambiente visual. A seguir estão resumidos alguns deles, de significância para este trabalho.

BISKUP e EMBLEY (2001) discutem formas de extrair informações de bases de dados heterogêneas, inclusive arquivos de texto, e sugerem uma metodologia capaz de relacionar os objetos extraídos e/ou gerados pela extração com os originais. No trabalho é sugerido um algoritmo capaz de realizar a tarefa. O mapeamento é efetuado criando-se uma tabela para conter os "achados" e na qual realizam-se inferências. O *framework* sugerido é válido para pesquisas em fontes de informação estruturadas e não estruturadas.

MURATA (1997) propõe transformações baseadas em estruturas de árvore, as quais podem ser aplicadas em documentos e esquemas de documentos. Especialmente aplicável em documentos do tipo *SGML*. As transformações são baseadas em padrões e em condições, as quais combinadas com a descrição do documento (esquema), permitem transformar suas características e/ou gerar um novo esquema. O trabalho discute a álgebra envolvida nestas transformações, sugere classes padrão para transformações, um algoritmo para testes e a construção de um esquema mínimo representativo do documento gerado. Sugere-se a utilização de *DSSSL*[6].

WADLER (1999) discute a *XSLT*[7], uma linguagem para transformar documentos *XML* em outros documentos, do mesmo tipo ou não. O elemento chave

---

[6] *Document Style Semantics and Specification Language*. ISO-1994.

[7] *XML Stylesheet Transformations. W3C 1999.*

desta linguagem é uma sub-linguagem de padrões, que é utilizada para comparações e seleções (*matching and selections*). Discute a interação da linguagem com *Xpath* e *Xpointer,* padrões do *W3C* na tecnologia *XML* para seleção de caminhos e expressões. Discute uma notação para a semântica e padrões de comparação.

ABITEBOUL e outros (1997) discutem a formalização de uma linguagem de consulta para dados semi-estruturados denominada *LOREL*. A linguagem é apresentada como uma extensão da *OQL*, do modelo de dados da *ODMG*[8]. A linguagem foi criada como meio de consulta do projeto *LORE*[9]. O texto discute a incompatibilidade das representações semi-estruturadas com o *framework* das bases de dados convencionais. O protótipo do projeto *LORE* fornece uma *interface* para o usuário, tanto em modo texto quanto em modo gráfico, via *HTML*[10], na qual pode-se expressar consultas usando *LOREL*. As consultas são espressas digitando-se os comandos da linguagem. Este trabalho, em especial, tem sido bastante referenciado por outros autores.

FERNANDEZ, SIMEÓN, WADLER e outros (2000) discutem as características essenciais das linguagens de consulta *XML*. Comparam-se quatro linguagens: *XML-QL*, *YATL* , *LOREL*, e *XQL*, que embora desenvolvidas independentemente possuem similaridades. Discute a formação de consultas (cláusulas padrão, filtro e construtora) e a realização de *nested queries*. O trabalho apresenta um esquema e realiza consultas neste esquema utilizando as quatro linguagens, demonstrando ao mesmo tempo sua sintaxe e seu poder de recuperação. Discute a transformação de documentos devido a agrupamentos e a realização de consultas em fontes de dados diferentes

BROWN, FUCHS, ROBIE e WADLER (2001) apresentam um modelo para descrição formal de estruturas de um *schema*. O trabalho modela somente a

---

[8] *Object Data Manegement Group*, http://www.odmg.org

[9] http://www-db.stanford.edu/lore

[10] *Hypertext Markup Language*

primeira parte da especificação de esquemas do *W3C,* correspondente às *structures*, não atingindo por exemplo *constraints*, mas fornece uma boa visão de normalização.

CERI e outros (1999), apresentam uma linguagem visual denominada *XML-GL* (de *XML Graphical Language*). Esta linguagem foi desenvolvida para efetuar consultas no modelo *XML-GDM*. O modelo de dados *XML-GDM* (de *XML Graphical Data Model*) é formado por três conceitos: objetos, relacionamentos e propriedades. Este modelo objetiva representar a estrutura e o conteúdo de um documento. As consultas podem ser realizadas em um documento ou em um conjunto de documentos, sendo o resultado da consulta a construção de um novo documento. Discute brevemente as transformações necessárias no documento para permitir a correspondência entre a representação gráfica e a *query XML-GL*. As consultas são apresentadas em forma gráfica com uma representação formada por duas figuras, dispostas lado a lado e separadas por uma linha vertical. A figura esquerda corresponde ao elemento solicitado no documento (a consulta), com informações de localização do documento e predicados, e a direita corresponde a uma representação do *DTD*[11] do novo documento construído pela consulta (em forma de um grafo *XML-GDM*). A representação gráfica é exibida na Figura 6.

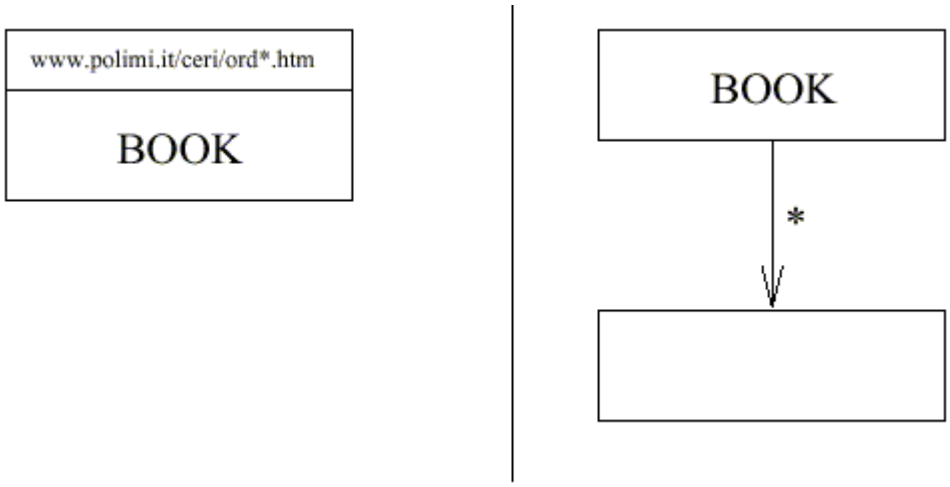

Figura 6 – Representação de uma consulta em *XML-GL*.

[11] *Document Type Definition*

BONIFATI e LEE (2000), analisam com grande detalhamento uma dúzia de representações de esquemas e linguagens de consulta *XML*. A cada momento são realizadas comparações quanto à possiblidade de execução de consultas propostas em cada uma das linguagens analisadas. Diversos aspectos interessantes são abordados, tais como herança e consultas aninhadas. O poder expressivo de seis linguagens é comparado, classificado e expresso em uma representação gráfica, apresentada na Figura 7.

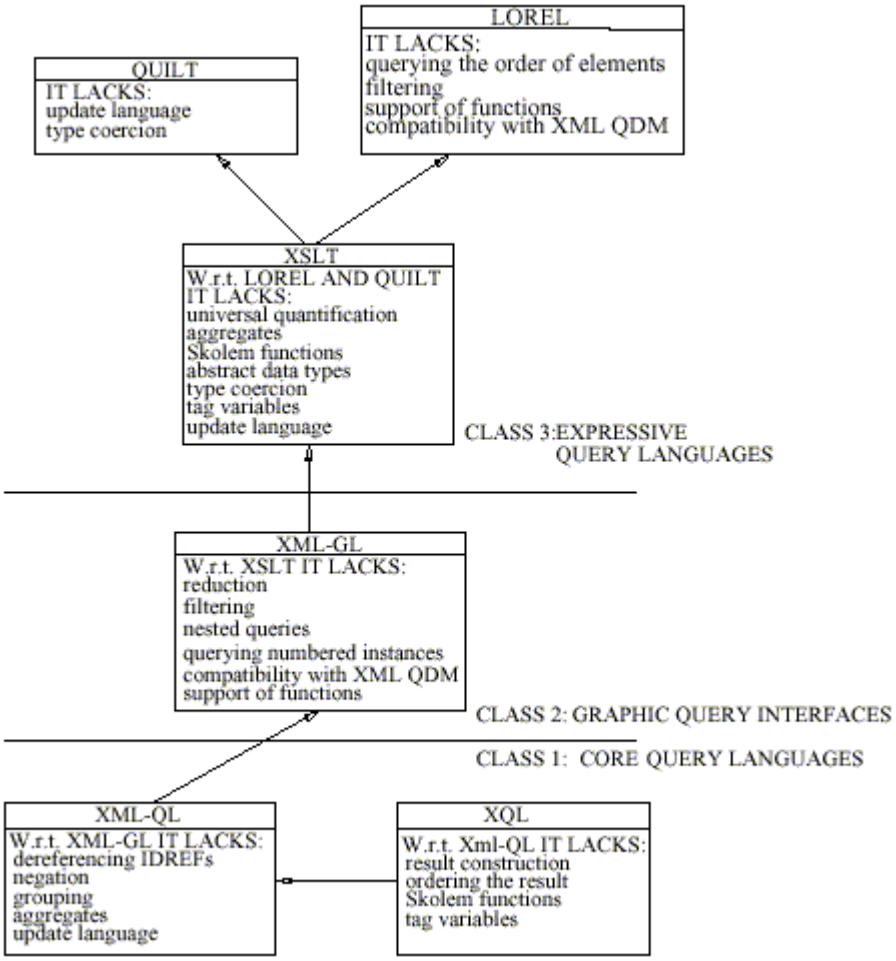

Figura 7 – Poder expressivo de seis linguagens de consulta *XML* de acordo com BONIFATI e LEE.

Este trabalho é uma extensão de um trabalho anterior de BONIFATI e CERI (2000), que compara cinco linguagens de consulta. Estes trabalhos foram realizados antes da publicação do primeiro *draft* para a *XML-Query language* pelo *W3C* (a proposta do *W3C* é bem recente: junho/2001).

COHEN e outros (1999) demonstram como formular consultas em documentos *XML* por meio da extração de informações a partir de suas descrições de arquivos *DTD*, no projeto denominado *EquiX* (*Easy queriyng in XML databases*). As consultas são formuladas em um navegador, por meio de uma interface gráfica que representa a hierarquia do documento. Embora de interface interessante (exibida na Figura 8) o modelo proposto pode exigir uma extensa navegação no documento bem como conhecimento de seus detalhes para a formulação da consulta.

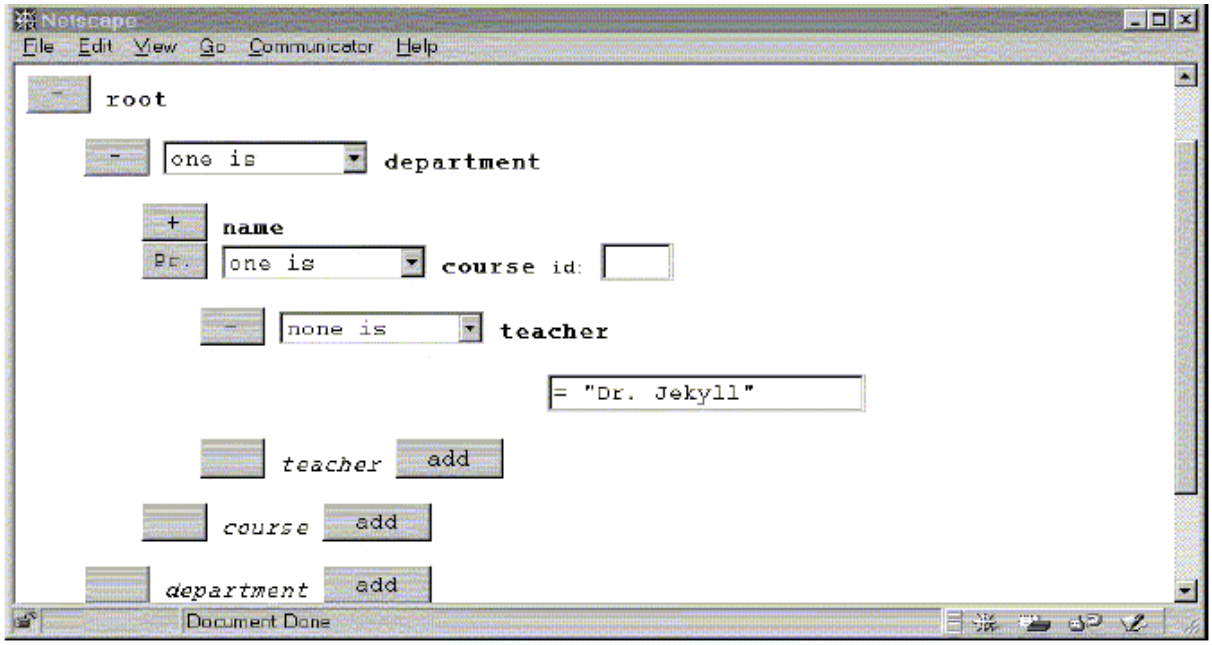

Figura 8 – A *Interface* de consulta do projeto *EquiX*.

MILO, SUCIU e VIANU (2000) sugerem um *framework* para garantir a integridade de transformações efetuadas em documentos por meio de linguagens tais como *XSLT* e *XQL*[12]. São analisadas características de transformações, baseadas em árvores e corretude quanto às definições presentes nos arquivos *DTD*.

2.1 CONCLUSÕES

Dos trabalhos pesquisados, grande parte das pesquisas concentram-se no poder expressivo das linguagens de consulta como ABITEBOUL e outros (1997),

[12] *XML Query Language*

BONIFATI e LEE (2000) e BONIFATI e CERI (2000), suas características essenciais, por exemplo FERNADEZ e outros (2000) e problemas envolvidos na transformação de documentos, em especial MURATA (1997) e MILO e outros (2000).

Todas convergem para a extração de consultas levando-se em consideração a representação semântica do conteúdo, que é obtida em arquivos de descrição de conteúdo do tipo *DTD* ou *Schema*. A maioria delas preocupa-se somente com documentos *XML*, inexistindo conexão com bases de dados, exceção feita à LOREL, de ABITEBOUL e outros (1997), que armazena os documentos em base própria.

Em geral são gerados novos documentos *XML* e/ou sua nova descrição, por exemplo um novo arquivo *DTD*, mas existem exceções, como demonstrado por BISKUP e EMBLEY (2001).

Nas pesquisas que utilizam mecanismos gráficos para a formulação de consultas, notamos que é necessário um certo conhecimento do conteúdo a ser pesquisado. como no caso do *EquiX*, de COHEN e outros (1999), ou elas apresentam a consulta em uma interface pouco usual, e que não apresenta todas as entidades envolvidas ao mesmo tempo, como no caso da *XML-GL*, de CERI e outros (2000). Ou ainda, possuem características excelentes mas exigem a digitação de comandos de consulta, o que não é trivial para usuários, abordagem de ABITEBOUL e outros no projeto *LORE* com a linguagem *LOREL* (1997).

Na pesquisa realizada não foi localizada nenhuma interface visual de consultas que permita ao mesmo tempo visualizar todo o conteúdo da base, uma representação em um modelo de grande poder expressivo, como o *ERC+*, e ainda que trabalhe com mais de uma fonte de dados.

Este trabalho pretende explorar esta lacuna.

3 INTEGRAÇÃO DE ESQUEMAS

A integração é uma das problemáticas existentes no estudo de bancos de dados e corresponde ao ato de analisar um conjunto de esquemas e extrair deles um esquema conceitual comum que represente as entidades comuns.

Para sua representação utiliza-se uma representação gráfica baseada em um modelo que tenha suficiente força semântica para representá-lo, tal como o modelo Entidade Relacionamento Complexo.

O motivo principal é a ausência semântica na implementação em modelos relacionais, mais comum, que não permite por exemplo representar atributos complexos/multivalorados. A Figura 9, extraída de ABITEBOUL (1995) representa uma possível visão de uma base de dados sobre filmes, na qual considera-se que os atores são atributos do atributo título de um filme (que possui ainda um atributo diretor).

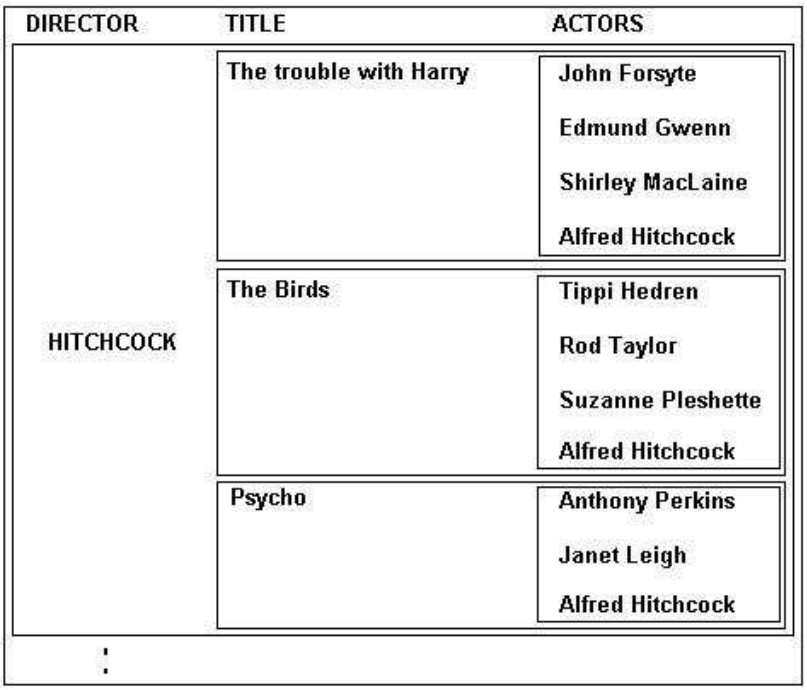

Figura 9 – Atributos que não podem ser diretamente representados em uma base relacional.

O problema surge do fato de que nos modelos relacionais à representação da base para a figura acima corresponde à criação de três tabelas,

ligadas por um atributo comum, conforme exibido na Figura 10. Desta maneira, ao criarmos uma nova entidade de certa forma perdemos a noção de atributo.

Figura 10 – Atributo complexo/multivalorado representado no modelo relacional.

3.1 O MODELO ENTIDADE RELACIONAMENTO COMPLEXO (*ERC+*)

O Modelo Entidade Relacionamento Complexo, ou abreviadamente *ERC+*, utiliza diagramas Entidade Relacionamento para representar o esquema conceitual, que é independente do gerenciador de bases de dados utilizado, e é uma estensão do Modelo *ER*.

O Modelo *ER* foi originalmente proposto por CHEN (1976) como uma forma de facilitar o desenho de bases de dados. Embora este modelo seja suficiente para a representação da maioria das aplicações administrativas convencionais, o crescimento da aplicação da informática às diversas áreas do conhecimento e indústrias trouxe consigo a necessidade de melhoria do poder semântico, visando a representação de situações mais complexas, como por exemplo na área de *CAD/CAM*[13] sugerido por PARENT e SCCAPAPIETRA (1989).

Surgiu então o Modelo *ERC+* que possui as mesmas características do Modelo *ER* e agrega os conceitos de especialização, generalização e composição, demonstrados por SCCAPAPIETA e outros (1995).

O Modelo *ERC+* dá suporte a dois outros tipos de ligações entre as entidades: *is-a*[14] e *maybe-a*[15]. O tipo *is-a* é utilizado para especificar que a população de um tipo entidade ES é uma subclasse, ou especialização, de um outro tipo de entidade EG; portanto EG contém os elementos de ES. O tipo *maybe-a* é utilizado

[13] Desenho e manufatura auxiliados por computador.

[14] É um

[15] Pode ser um

para especificar que é possível que, dada a existências dos tipos entidade E1 e E2, é possível que E1 contenha objetos de E2 e vice-versa [16].

A Figura 11, extraída de PARENT e SCCAPAPIETRA (1989), é um exemplo de representação de um diagrama *ERC+* .

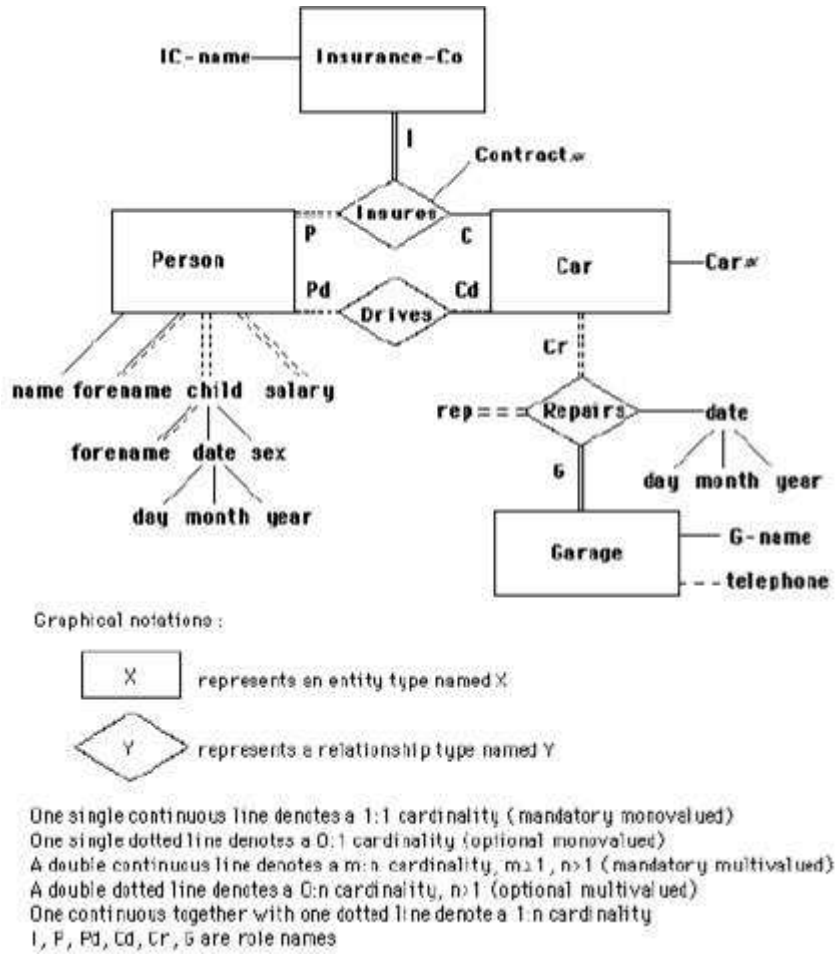

Figura 11 – Exemplo de modelo *ERC+* .

Existem ferramentas que trabalham com o modelo *ERC+*, implementando-o totalmente, tal como a *SUPER* (Figura 12), citada por SCCAPAPIETA e outros (1995). Esta ferramenta permite a formulação de consultas em ambiente gráfico, consistindo na seleção de objetos e formulação de predicados. Porém, existe a limitação de que a ferramenta não oferece suporte a bases de dados relacionais, fortemente difundidas no mercado, inexistindo assim suporte ao legado (GUEIBER, 2001).

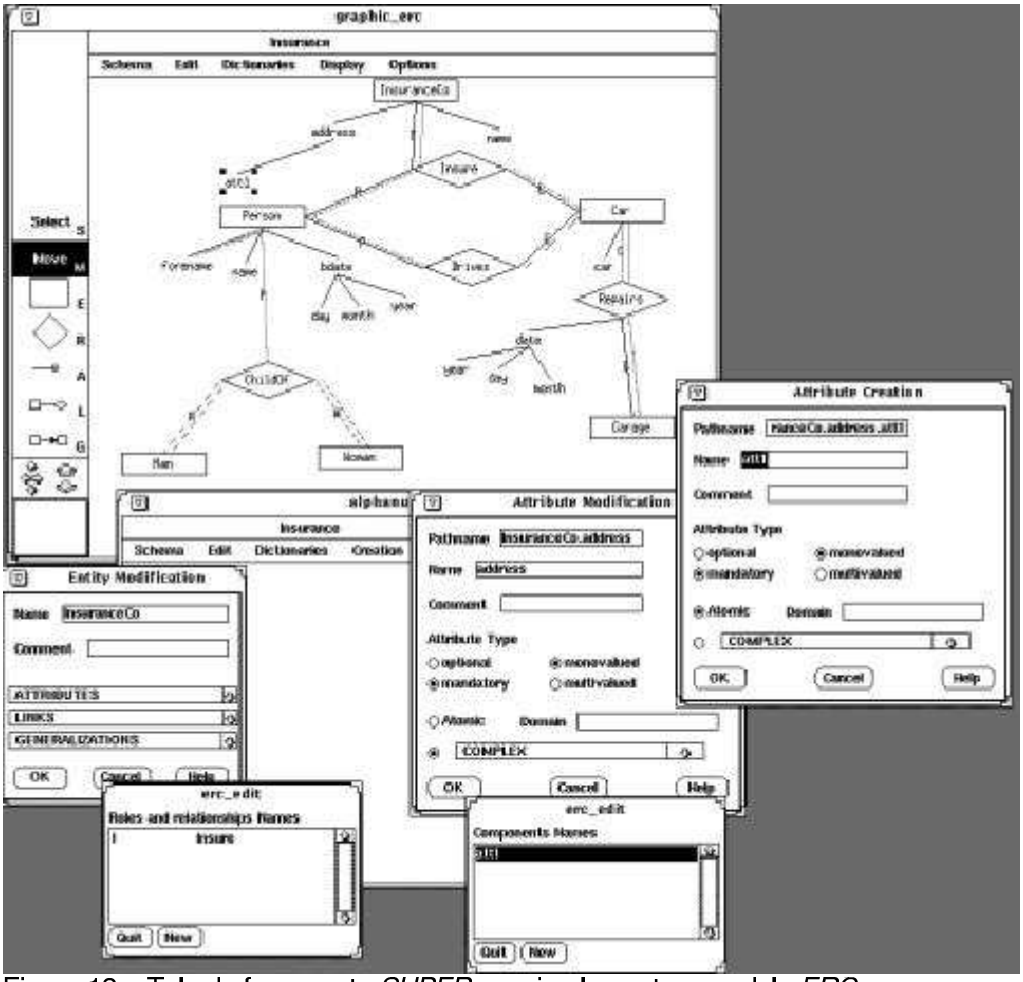

Figura 12 – Tela da ferramenta *SUPER*, que implementa o modelo *ERC+*.

Nossa proposta neste trabalho utiliza a ferramanta desenvolvida no Sistema *VIQUEN* para representar graficamente um modelo *ERC+* e interagir com o mesmo. O mapeamento *ERC+* / bases integradas será realizado utilizando descrições de documentos em *XML*, de maneira que possamos executar consultas às bases legadas, heterogêneas, independentemente de seu suporte a atributos complexos/multivalorados.

4 A LINGUAGEM DE MARCAÇÃO *XML*

A *SGML*, abreviatura de *Standardized General Markup Language*, linguagem de marcação padronizada de uso geral, foi originalmente desenvolvida por GOLDFARB, MOSHER, e LORIE na IBM, em1969, como um meio de estruturar documentos legais. A *SGML* foi introduzida em aplicações comerciais de maneira mais generalizada em 1986, e, em 1990, foi adaptada (reduzida) para poder ser utilizada na *WEB*, nascendo a *HTML – HyperText Markup Language*, linguagem de marcação de hipertextos. A linguagem *XML* teve seu primeiro rascunho (*draft*) publicado pelo *W3C* em 1996, sendo sua recomendação de 1998. *XML* é um acrônimo para *Extensible Markup Language*, linguagem de marcação extensível .

4.1 CONCEITOS DE *XML*

Conforme HAROLD (1999), a *XML* foi criada para suprir necessidades de representação dos usuários, pois o padrão *HTML* evolui de forma lenta, criando margem a extensões (*tags* específicas de cada navegador), e também devido à impossibilidade da diferenciação entre dados e metadados, ou seja a impossiblidade de descrever o conteúdo de uma página ou documento, em *HTML*, bem como para facilitar a construção de mecanismos de busca e a especificação de coleções de páginas relacionadas. É um padrão da *WEB*, criada por uma recomendação do *W3C* (a versão 1.0 é de fevereiro de 1998).

A linguagem *XML* especifica dados em um formato legível por humanos, similar ao *HTML*. É compreensível pelas máquinas por meio da utilização de *parsers*, estruturada (hierárquica), verificável, podendo ser verificada quanto a corretude e integridade e extensível. Sendo uma meta linguagem, pode ser utilizada para definir outros domínios, ou linguagens específicas de um determinado domínio (matemática, química, biologia, etc), podendo também ser utilizada para definição de outras linguagens, como exemplifica MEGGINSON (1998), geralmente por meio do

uso de arquivos *DTD.* Os arquivos *DTD* são equivalentes à notação *BNF*[16] das linguagens de programação.

A linguagem *XML* é independente de mídia de apresentação (navegador, aplicativo, telefone celular, etc.), pois descreve dados e não apresentação visual, é interoperável, sendo independente tanto de plataforma quanto de fornecedor, já que *XML* é, basicamente, um arquivo texto.

BOSAK (1998) considera a *XML* essencial para a transmissão de dados do servidor para o navegador, no caso de aplicações Internet; e de aplicações para aplicações e de máquinas para máquinas, no caso de programas, processos e bases de dados.

Resumidamente as especificações do *W3C* para este padrão compreendem:

- *XML* – a linguagem em si;
- *XLL* (*XLink* e *XPointer*) – para ligação entre documentos e partes de documentos, com poder semântico maior do que as ligações *HTML*;
- *XSL* (*Extensible Style Language*) e *XSLT* (*Extensible Style Language Trasnformations*) - permitem a transformação de um documento em outro, para fins de apresentação ou intercâmbio;
- *DOM* (*Document Object Model*) – estrutura de *APIs*[17] criadas com a finalidade de possibilitar a navegação em documentos *XML*;
- *RDF* (*Resource Description Framework*) – descrição de propriedades dos documentos; semelhante a um *DTD* mas com maior semântica.
- *XHTML* (*Extensible HTML*) - *HTML* versão 4 com sintaxe *XML*;
- *XML Namespaces* – para resolver conflitos de nomes;
- *XML Schema* – para especificação de tipos de dados, subclasses e melhores modelos de conteúdos;
- *XML Query* – permite exprimir consultas de forma semelhante à utilizada na linguagem *SQL*. Recomendação do *W3C* de junho de 2001[18].

---

[16] *Bakus-Naür Form*

Para HAROLD (1999), quando comparada à *HTML*, a *XML* possui os seguintes benefícios:

- Extensões específicas para cada domínio;
- Vocabulários específicos para cada domínio;
- Dados estruturados;
- Complementa e extende a *HTML*;
- Documentos autodescritivos eliminam necessidade de conhecimento prévio do modelo;
- Metadados / coleções;
- Importante para mecanismos automáticos de busca;

Do ponto de vista do desenvolvedor, dentre as principais facilidades estão as atualizações localizadas, já que torna-se possível atualizar somente uma parte da página ou documento, dado que sua estrutura é conhecida e seus elementos são identificáveis, a apresentação independente da estrutura, que facilita a difusão do documento entre diferentes dispositivos de apresentação (vídeo, aparelho de telefone celular, etc), e a possiblidade de oferecer uma visão ao usuário dependente dos dados, isto é, específica para cada usuário. Esta última possibilidade é de especial interesse, tanto na problemática de bancos de dados quanto na IHC[19].

Do ponto de vista científico SIMON ST.LAURENT[20] aponta as seguintes aplicações:

- Reutilização de protocolos, partes e ferramentas, com a vantagem do controle de documentos bem formados;
- Facilidades para indexação e busca;
- Pesquisa distribuída, utilizando um formato comum de colaboração;
- Apresentação sofisticada, incluindo documentos separados mas "ligados" uns aos outros.
- Documentos em forma de dados:

---

[17] *Application Programming Interfaces.*

[18] http://www.w3c.org/TR/2001/WD-query-semantics-20010607.html.

[19] Interface Humano Computado.

[20] *Inside XML DTDs: Scientific and Technical.*

- Catálogos e especificações;
- Artigos e livros;
- Notação e cálculo: por exemplo *MathML* (*Mathematical Markup Language*) para matemática, *CML* (*Chemical Markup Language*) para química e *AIML* (*Astronomical Instruments Markup Language*) para astronomia;
- Hierarquias.

Dentre as possibilidades de aplicações, destacam-se os agentes inteligentes, ordens de compra, objetos com métodos e atributos, especificados em formato comum; meta-conteúdo de web sites, *GUI*[21], catálogo com partes interativas, formato de armazenamento persistente e intercâmbio eletrônico de dados (*EDI*).

Em relação às áreas de aplicação, BOSAK (1997) escreveu:

> "As aplicações que dirigirão a aceitação do *XML* serão àquelas que não podem ser realizadas dentro da limitação do *HTML*, e podem ser divididas em 4 categorias:
>
> 1. Aplicações que requeiram do cliente web a mediação entre duas ou mais bases de dados heterogêneas;
> 2. Aplicações que tentem distribuir porções significativas da carga de processamento do *web server* para o *web client*;
> 3. Aplicações que requeiram que o cliente "web" apresente diferentes visões dos mesmos dados para diferentes usuários;
> 4. Aplicações nas quais agentes inteligentes procurem adaptar a informaão descoberta para diferentes usuários individualmente."

4.2 DIFERENÇAS ENTRE *HTML* E *XML*

Embora ambas sejam linguagens de marcação, *HTML* e *XML* diferem quanto à forma com que podem ser construídas, a finalidade de suas marcações e a sua utilização final. Como a *XML* é uma metalinguagem que permite descrever outros domínios a própria HTML poderia ser descrita naquela linguagem (HAROLD, 1999).

Quando comparada à *HTML*, a linguagem *XML* tem como benefícios as extensões e vocabulários específicos para cada domínio, os dados estruturados, o

[21] Interface Gráfica com o Usuário.

fato de ser autodescritiva (não é necessário conhecimento anterior sobre a estrutura dos dados), metadados, coleções, e o fato de ser altamente relevante para mecanismos de busca.

As principais diferenças entre *XML* e *HTML* estão exibidas no Quadro 1, a seguir:

HTML	***XML***
Para exibição somente	Para representação da estrutura dos dados
Pode revelar a estrutura do documento	Exibe a estrutura do documento
Sem conhecimento sobre os dados	Independente da apresentação
Linguagem fechada, com elementos definidos pelo *W3C*	Linguagem aberta, permite definição de seus próprios elementos
Curva de aprendizagem pequena	Curva de aprendizagem em degraus, dependendo da ferramenta desejada
Case insensitive	*Case sensitive*
Espaços em branco ignorados	Espaços em branco significativos, dentro de um contexto
Indiferente à "má formação" dos documentos	Somente aceita documentos "bem formados". Todas as *tags* abertas devem, obrigatoriamente, ser fechadas

Quadro 1 – Principais diferenças entre as linguagens de marcação *HTML* e *XML*

Segundo MEGGINSON (1998), quanto à marcação, seus mecanismos são idênticos: declara-se uma marca (ou *TAG*) que corresponde a determinado atributo ou ação (*HTML*), ou a um elemento ou atributo de um elemento (*XML*). Porém, na *HTML* as marcações são definidas pelo *W3C* em suas recomendações, enquanto para a *XML* são definidas marcações básicas que permitem criar outras, à vontade (necessidade) do usuário. Além disto, em geral os *parsers* para a linguagem *HTML* não se preocupam com o balanceamento das *TAGs*, o que é obrigatório em *XML*.

A Figura 13 exemplifica a diferença entre as linguagens de marcação *HTML* e *XML*, e ao mesmo tempo exibe um problema comum em páginas *HTML*, que não impede que as mesmas sejam exibidas, que corresponde a um problema de formação em documentos *XML*: a *TAG* correspondente à marcação do atributo itálico (<I>) para apresentação do endereço eletrônico não está corretamente fechada (deveria ser </I>).

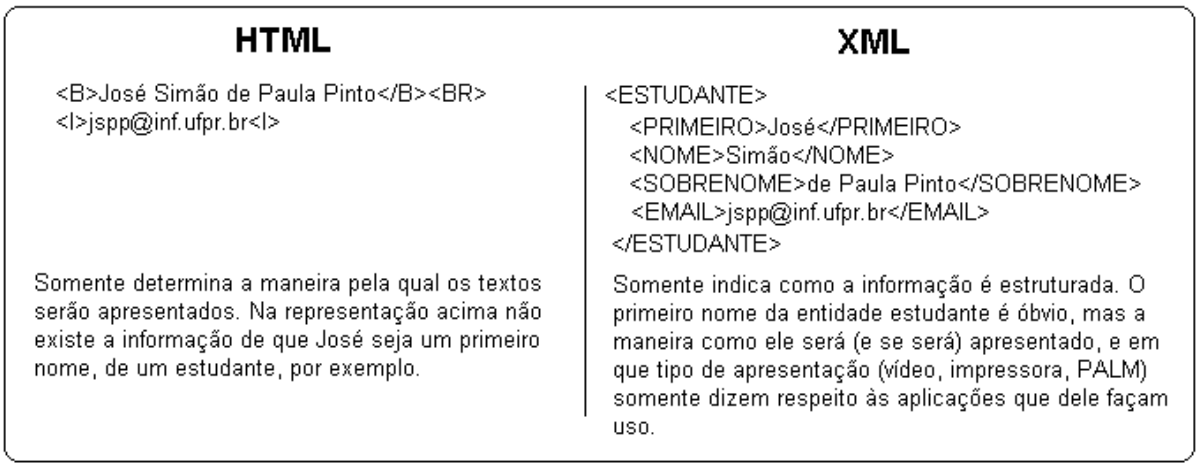

Figura 13 - Diferenças entre as linguagens de marcação *HTML* e *XML*

Utilizando–se a notação *DOM*, criada para navegação em documentos, a notação exibida para o exemplo *XML* da figura seria: *doc.estudante[posição].nome[0].primeiro[0].value*.

4.3 COMPOSIÇÃO DE UM DOCUMENTO *XML*

Um documento *XML* consiste de duas partes (HAROLD, 1999 e *W3C*): o prólogo (*PROLOG*), no qual são declarados os nomes de elementos, atributos e as regras para validar as marcações, e a instância do documento (*DOCUMENT INSTANCE*), que possui dados com marcação, um elemento raíz (*ROOT*), do qual todos os outros elementos são filhos.

A organização do documento é hierárquica, comumente adotando-se uma estrutura em árvore para sua manipulação. Este conceito é exibido na Figura 14.

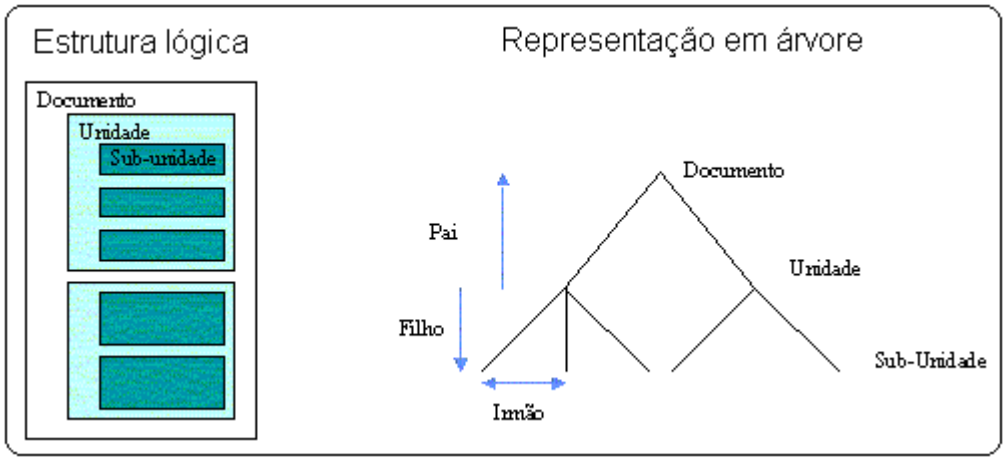

Figura 14 – Exibição da estrutura de um documento *XML* e sua representação em uma estrutura de árvore.

4.3.1 O PRÓLOGO (*PROLOG*) DE UM DOCUMENTO *XML*

Na primeira linha do prólogo encontraremos as declarações que indicam o início do documento e informações necessárias para sua validação, conforme a sintaxe `<?XML Version Encoding RequiredMarkupDeclaration ?>`, na qual `<?XML` é um marcador de início, obrigatório, "`Version`" refere-se à versão da recomendação do *W3C*, `Encoding` representa o conjunto de caracteres a ser utilizado e `RequiredMarkupDeclaration` determina se o documento poderá ser validado somente com as informações internas ou não (é necessário ou não um arquivo *DTD* ou *Schema* ?) e `?>` é um terminador da declaracão, conforme exemplificado por HAROLD (1999) e MEGGINSON (1998).

- Exemplo:

<?XML version="1.0" ENCODING= "UTF-8" STANDALONE="no"?>

Na segunda linha do documento encontraremos informações referentes ao documento em si, tais como seu tipo, o nome e localização do documento que contém suas regras de validação, ou as próprias regras, seguindo a sintaxe `<!DOCTYPE RootElement SYSTEM ExternalDeclarations [InternalDeclarations] >`, na qual `<!DOCTYPE` representa o tipo de documento, `RootElement` seu elemento raíz e `SYSTEM` indica, se for o caso, documentos agregados à sua validação. Caso existam declarações internas, elas estarão declaradas na seqüência e contidas entre colchetes (`[ ]`).

- Exemplo:

<!DOCTYPE InstrumentControl PUBLIC "-//Code588//DTD Instrument Control//EN" "http://someplace.nasa.gov/xml-vocabs/aiml.dtd" >

4.3.2 CONTEÚDO DE UMA INSTÂNCIA DE DOCUMENTO *XML*

Após as declarações do prólogo teremos a instância do documento, a qual é formada por um conjunto de elementos.

Um *ELEMENT* (elemento), é um nome conceitual, conforme explica MEGGINSON (1998); enquanto *TAG* ele especifica uma situação especial de marcação. Por exemplo, as *TAGs* de início *<FONT>* e de fim *</FONT>* especificam

uma determinada condição. O texto inserido entre a ocorrência das TAGs de início e fim é chamado de *CONTENT* (conteúdo).

De acordo com HAROLD (1999) e MEGGINSON (1998), os elementos possuem atributos. Um *ATRIBUTE* é um nome de algo que qualifica um elemento. Embora alguns atributos possam não ter um valor específico (podem ser nulos), a maioria irá assumir um valor discreto, podendo possuir um conjunto infinito de valores.

Os atributos sempre ocorrem dentro da *TAG* inicial. Por exemplo para atribuir à fonte de caracteres o tamanho 3 faríamos *<FONT SIZE="3">*..conteúdo...*</FONT>.* Os valores dos atributos sempre serão indicados entre aspas, independentemente de eles serem numéricos ou carcateres.

As tabelas Tabela 1 (*W3C*) e Tabela 2 (HAROLD, 1999) a seguir representam as regras de notação correspondentes às regras de *constraints* e de cardinalidade, aplicáveis em documentos do tipo *DTD*, existentes nos mecanismos formais e na linguagem *SQL* para bancos de dados relacionais.

TAG	**FINALIDADE**
#REQUIRED	O atributo deve ser especificado, obrigatoriamente
#IMPLIED	O atributo pode ser especificado; não é obrigatório
"default"	Valor padrão (*default*) para o atributo, se desconhecido
#FIXED	Somente um valor permitido

Tabela 1 - Marcações para controle de nulos em documentos *XML*

NOTAÇÃO	**FINALIDADE**
()	Para representar uma expressão
A?	A ou nada (A é opcional, mas existe somente um)
A+	Um ou mais A's (pelo menos um)
A*	Zero ou mais A's (A é opcional, mas podem ocorrer muitos A's)
(A | B | C)	Qualquer um: A or B or C (mas somente um)
(A, B, C)	Primeiro A, seguido por B, seguido por C (todos, e nesta ordem)
(A & B)	Ambos, A e B, em qualquer ordem
-- texto --	Comentários (usa-se <!-- texto --> quando dentro do arquivo *XML*)

Tabela 2 - Notações para controle de cardinalidade em documentos *XML*

4.3.3 REGRAS DE FORMAÇÃO

Conforme HAROLD (1999), para que um documento *XML* possa ser verificado automaticamente, quanto à sua corretude e validade devemos respeitar dois conceitos durante sua construção: *weel formed* (bem formado) e *valid* (válido).

O conceito de documento bem formado refere-se ao balanceamento das *TAGs* utlizadas em sua formação. Um documento bem formado possui todas as seqüências das marcações completas, no que se refere à abertura e fechamento, bem como seu correto aninhamento na estrutura (uma declaração não se sobrepõe à outra). A Figura 15 exibe este conceito.

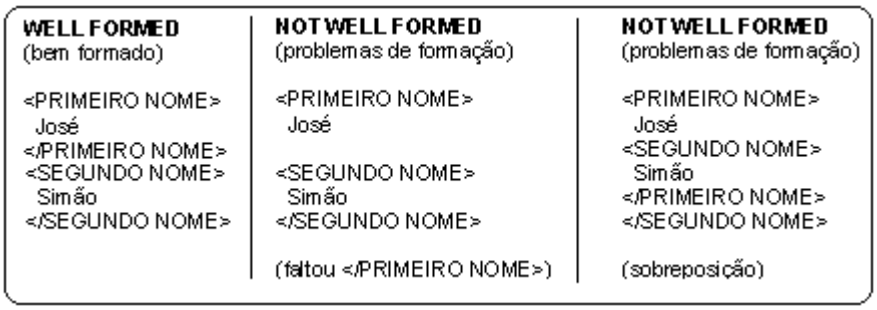

Figura 15 – O conceito de documento bem formado (*well formed*)

De acordo com a documentação do *W3C*, o conceito de validade do documento está ligado à sua especificação formal. Embora um documento possa ser bem formado, quanto à sua constituição, para garantir sua validade deveremos verificar se todos os elementos constantes em sua definição estão presentes na instância que estivermos analisando, e se todos os elementos presentes na instância realmente fazem parte daquele tipo de documento. A definição das características do documento poderá estar presente embutida no próprio documento, no prólogo, ou estar presente em um arquivo de definição externo, do tipo *DTD* ou *Schema*.

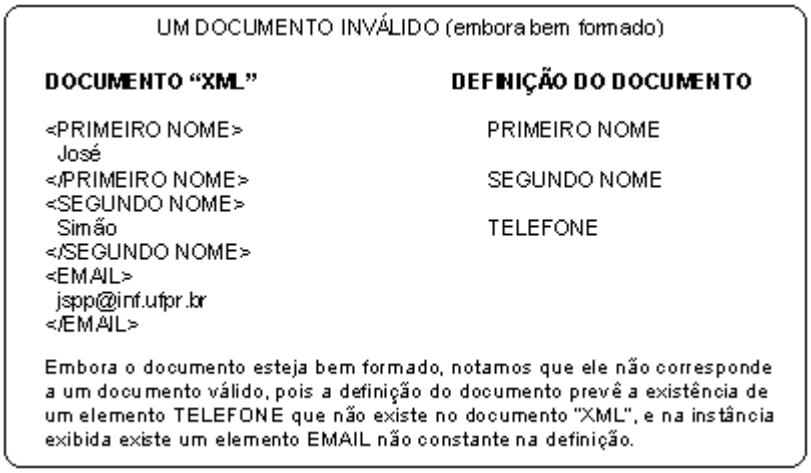

Figura 16 – O conceito de um documento válido (*valid*).

De qualquer maneira, os documentos *XML* devem seguir as seguintes regras de formação:

- Documentos iniciam-se por **<?xml version='1.0' ?>**;
- Todos os valores de atributos devem estar entre aspas, por exemplo: <*TAG* nome="valor">;
- Os espaços em branco e quebras de linha encontrados dentro do conteúdo (*CONTENT*) são significativos e não serão ignorados;
- Todas as *TAGs* de início devem possuir *TAGs* de finalização (estrutura balanceada). Exemplo: *<TAG>...</TAG>*;
- As *TAGs* devem ser encadeadas corretamente, ou seja, os elementos deverão estar corretamente aninhados. Exemplo: <A><B>…</B></A>;
- As *TAGs* que não possuírem conteúdo deverão ser indicadas; utilizar a notação "</">. Por exemplo, se uma pessoa não possui um telefone celular a marcação correspondente seria "<celular/>";
- Os nomes são sensíveis a maiúsculas e minúsculas (*CASE SENSITIVE*). Portanto, <NOME> não é igual a <Nome>;
- Palavras chave (reservadas), tais como *DOCTYPE* e *ENTITY*, serão escritas em maiúsculas;
- Caso utilizados, os comentários serão inseridos entre "<!--" e " -->", por exemplo: <--Comentário -->.

4.4 TIPOS DE PROCESSADORES (*PARSERS*)

O parser é um interpretador de documentos (HAROLD, 1999), que efetua regras de validação e aplica os mecanismos de transformação dos documentos. Pelas regras do *W3C*, após passar pelo *parser*, um documento poderá ser classificado como:

- Bem formado (*WELL FORMED*), quando seus elementos estiverem propriamente aninhados (*NESTED*) e marcados (*MARKEDUP*);
- Válido (*VALID*), quando, em adição a ser bem formado, o documento está de acordo com o respectivo *DTD*, interno ou externo. Todos os elementos, atributos e entidades usados no documento são encontrados no *DTD* (ou *Schema*) e nenhum elemento presente no documento deixa de estar presente em sua definição.

Em um documento, as declarações estão presentes no formato <! … > ou <!...[<! … >]>, seguindo um pequeno conjunto de *TAGs* pré-definidas, que compreendem a sintaxe *XML* a ser utilizada nas marcações, de maneira que possa ser interpretada pelo *parser* ou pela aplicação que fará uso do documento XML, além de todas as marcações criadas pelo usuário, cuja descrição esteja presente no próprio documento ou em arquivos de definição.

As definições de marcação padrão da *XML* estão representadas na Tabela 3, a seguir (*W3C*).

TAG	**FINALIDADE**
<!DOCTYPE … >	Definição do tipo de documento
<![CDATA[…]]>	Existência de dados tipo caracter
<!ENTITY … >	Declaração de entidades
<!NOTATION … >	Declaração de notação
<!ELEMENT … >	Declaração de elemento
<!ATTLIST … >	Declaração de atributos
<![INCLUDE[…]]> *e* *<![IGNORE[…]]>*	Declarações epeciais

Tabela 3 - Declarações da sintaxe *XML*

Conforme sua funcionalidade e atuação, poderemos ter quatro diferentes tipos de *parsers*.

4.4.1 TIPO 1 - *NON-VALIDATING PARSER*

Este primeiro tipo de *parser* não verifica o documento contra o *DTD*. Ele somente testa se o documento é bem formado ou não. É relativamente pequeno, de maneira que pode ser colocado em um *applet*[22], caso necessário.

4.4.2 TIPO 2 - *VALIDATING PARSER*

Este *parser* verifica se o documento está de acordo com o estabelecido no *DTD* (independentemente de se é interno ou externo) ou *Schema*. É relativamente grande, e portanto mais adequado para ser inserido em um navegador. Atualmente os navegadores que possuem *parsers* deste tipo embutidos em seu código são o *Microsoft Internet Explorer*, versão 5 em diante[23], o *Netscape Communicator*, versão 6 em diante, o *OPERA* e o *AMAYA*. Este último é oferecido pelo próprio *W3C* e é capaz de apresentar documentos *XML* que possuam componentes gráficos, tais como as marcações voltadas para química (*CML*), que possuem informações sobre o "desenho" da molécula da substância representada no documento.

4.4.3 TIPO 3 - *TREE-BASED PARSING*

Conforme HAROLD (1999), constituem *parsers* deste terceiro tipo aqueles baseados no *DOM – Document Object Model*.

Mais complexo, este *parser* possibilita uma visão centrada no documento. É desaconselhável para documentos muito grandes, pois necessita muito espaço e utiliza intensivamente a memória; durante seu funcionamento uma instância completa do documento ficará disponível na memória de trabalho. É extremamente útil para navegação nos elementos que constituem o documento, busca e alteração. Atua basicamente transformando o documento lido em uma estrutura de árvore.

---

[22] Mini-aplicativo escrito em linguagem *JAVA*

[23] A versão 4 identifica documentos *XML* mas necessita de atualizações no *parser* para processar funcionalidades recentes.

O acesso aos elementos do documento, e seus atributos, é efetuado por operações relativamente simples, correspondentes a chamadas a suas *APIs*, conforme exemplificado na Figura 17.

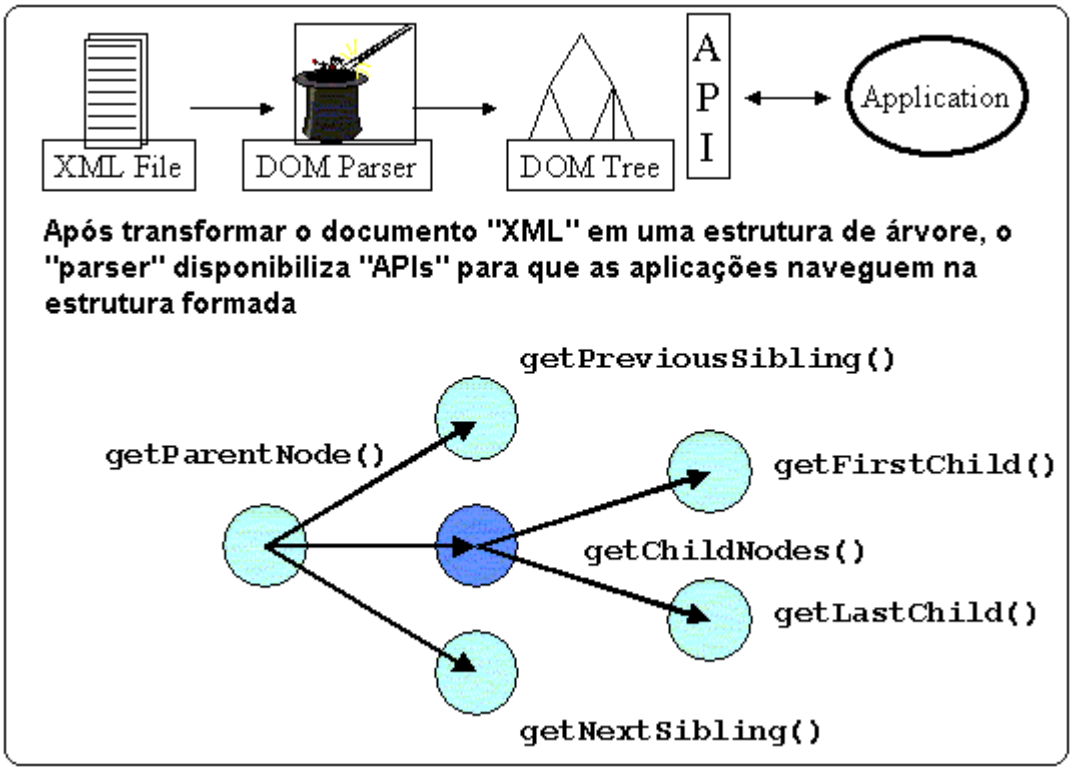

Figura 17 – A estrutura hierárquica e operações possíveis no modelo *DOM*.

4.4.4 TIPO 4 - *EVENT-BASED PARSING*

Neste *parser* a visão está centrada nos dados.

A tarefa do *parser* é ler um documento de definição (*DTD*) e, após, o documento *XML.* Para cada novo elemento encontrado será gerado um evento. Cada evento é disparado, independente se o elemento é encontrado no documento original. Cabe à aplicação decidir se vai responder àquele evento ou não.

Este tipo de *parser* não utiliza árvore e portanto requer menos memória e é mais rápido. É o tipo ideal para transferência de dados. Os *parsers* baseados nas *APIs* da *SUN Microsystems* entituladas *SAX*[24] são exemplos de *parsers event-based*.

[24] *Simple API for XML*

Para o propósito deste trabalho será utilizado um *parser* do tipo 4, baseado em eventos.

4.5 DEFINIÇÕES DO DOCUMENTO: *DTD* E *SCHEMA*

Para que um documento possa ser validado por um *parser*, é necessário que este conheça as definiçoes do documento, que poderão estar presentes no próprio documento, quando ele não é muito extenso, ou em arquivo separado, conforme especificações do *W3C*. Quando em separado, as definições poderão seguir o modelo *DTD*, de definição do tipo de documento, ou *Schema*, mais recente e de expressividade maior.

A *XML* dispõe dos seguintes mecanismos e arquivos para efetuar a descrição e a validação de documentos, de acordo com HAROLD (1999):

- *Document Type Definitions (DTD)*;
- *Document Content Descriptions (DCD)*;
- *XML-Data Reduced (XDR)*;
- *XML Schema Definition (XSD)*.

HAROLD (1999) considera ainda a inciativa *Biztalk*[25], que é um repositório de esquemas criado pela *Microsoft Corporation*, de base aberta, na qual pode-se encontrar descrições prontas de modelos de documentos para praticamente todas as áreas do conhecimento, bem como vocabulários.

4.5.1 EXEMPLOS EXTRAÍDOS A PARTIR DA BASE DO SISTEMA *VIQUEN*

Na Figura 18 podemos observar a descrição dos elementos constituintes, correspondentes às tabelas da base de dados utilizada no protótipo do Sistema *VIQUEN*. Para simplificação do exemplo foram obtidos somente elementos do tipo texto (representação *#PCDATA*). Esta figura exibe basicamente a especificação dos campos encontrados nas tabelas, e suas seqüências.

---

[25] *http://www.biztalk.org*

```
<!ELEMENT TesteXML (Agricultor_Arrenda,Agricultor_Cultiva,Agricultor_Fisica,
                    Agricultor_Gleba,Agricultor_Juridica,Agricultor_Lavoura,
                    Agricultor_Possui,Agricultor_Produtor,
                    Agricultor_Propriedade,Agricultor_Proprietario,
                    Agricultor_REGISTRO_IMOVEL)+>
    <!ELEMENT Agricultor_Arrenda (RegistroRural,NrInscricao,DataInicio, Duracao)>
        <!ELEMENT RegistroRural (#PCDATA)>
        <!ELEMENT NrInscricao (#PCDATA)>
        <!ELEMENT DataInicio (#PCDATA)>
        <!ELEMENT Duracao (#PCDATA)>
    <!ELEMENT Agricultor_Cultiva (IdCultiva,RegistroRural,NrInscricao,Ano)>
        <!ELEMENT IdCultiva (#PCDATA)>
        <!ELEMENT RegistroRural (#PCDATA)>
        <!ELEMENT NrInscricao (#PCDATA)>
        <!ELEMENT Ano (#PCDATA)> ...
```

Figura 18 – Fragmento de arquivo *DTD* para a base de dados proposta, sem utilizar a notação de pontos.

A Figura 19 exibe uma representação de documento *XML* contendo dados obtidos a partir da base em questão. Podemos notar o aninhamento dos dados, correspondente ao significado lógico e físico de que um elemento "contém" outros elementos. Por exemplo na mesma figura vê-se que o elemento AGRICULTOR_ARRENDA contém (ou é formado por) REGISTRORURAL, NRINSCRICAO, DATAINICIO, DURACAO.

```
<?xml version="1.0" ?>
- <TesteXML>
  - <Agricultor_Arrenda>
      <RegistroRural>8444</RegistroRural>
      <NrInscricao>1340</NrInscricao>
      <DataInicio>23/08/00</DataInicio>
      <Duracao>36</Duracao>
    </Agricultor_Arrenda>
  + <Agricultor_Arrenda>
  - <Agricultor_Cultiva>
      <IdCultiva>100</IdCultiva>
      <RegistroRural>9802</RegistroRural>
      <NrInscricao>1340</NrInscricao>
      <Ano>1999</Ano>
    </Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  + <Agricultor_Cultiva>
  - <Agricultor_Fisica>
      <Identificador>2</Identificador>
```

Figura 19 – Fragmento de arquivo *XML* para a base de dados proposta utilizando *DTD* sem a notação de pontos.

Para esta representação podem ocorrer problemas de duplicidade no nome de elementos, de maneira que a representação mais eficiente é a chamada

representação “com pontos”, na qual os elementos são referenciados em relação aos elementos aos quais estão ligados, como exibido na Figura 20.

```
<!ELEMENT TesteXML (Agricultor_Arrenda,Agricultor_Cultiva,Agricultor_Fisica,
                    Agricultor_Gleba,Agricultor_Juridica,Agricultor_Lavoura,
                    Agricultor_Possui,Agricultor_Produtor,
                    Agricultor_Propriedade,Agricultor_Proprietario,
                    Agricultor_REGISTRO_IMOVEL)+>
    <!ELEMENT TesteXML.Agricultor_Arrenda
                   (TesteXML.Agricultor_Arrenda.RegistroRural,
                    TesteXML.Agricultor_Arrenda.NrInscricao,
                    TesteXML.Agricultor_Arrenda.DataInicio,
                    TesteXML.Agricultor_Arrenda.Duracao)>
              <!ELEMENT
                 TesteXML.Agricultor_Arrenda.RegistroRural (#PCDATA)>
              <!ELEMENT
                 TesteXML.Agricultor_Arrenda.NrInscricao (#PCDATA)>
              <!ELEMENT
                 TesteXML.Agricultor_Arrenda.DataInicio (#PCDATA)>
              <!ELEMENT
                 TesteXML.Agricultor_Arrenda.Duracao (#PCDATA)>
    <!ELEMENT TesteXML.Agricultor_Cultiva
                   (TesteXML.Agricultor_Cultiva.IdCultiva, ...
```

Figura 20 – Fragmento de arquivo *DTD* para a base de dados proposta, utilizando notação com pontos.

Embora esta representação seja melhorada ela não garante que durante uma integração, por exemplo, não surjam conflitos de nomes, de maneira que foi criado o mecanismo *namespace* que garante uma informação da origem dos dados. Esta característica será explorada em nosso protótipo de forma a possibilitar o “conhecimento” sobre a base origem da base integrada, possibilitando assim enviar a consulta para o local de origem correto.

Uma representação de um documento *XML* seguindo a notação pontuada é exibida na Figura 21.

```
<?xml version="1.0" ?>
<!DOCTYPE TesteXML SYSTEM "XML_Agricultura2.dtd">
- <TesteXML>
  - <TesteXML.Agricultor_Arrenda>
      <TesteXML.Agricultor_Arrenda.RegistroRural>8444</TesteXML.Agricultor_Arrenda.RegistroRural>
      <TesteXML.Agricultor_Arrenda.NrInscricao>1340</TesteXML.Agricultor_Arrenda.NrInscricao>
      <TesteXML.Agricultor_Arrenda.DataInicio>23/08/00</TesteXML.Agricultor_Arrenda.DataInicio>
      <TesteXML.Agricultor_Arrenda.Duracao>36</TesteXML.Agricultor_Arrenda.Duracao>
    </TesteXML.Agricultor_Arrenda>
  + <TesteXML.Agricultor_Arrenda>
  - <TesteXML.Agricultor_Cultiva>
      <TesteXML.Agricultor_Cultiva.IdCultiva>100</TesteXML.Agricultor_Cultiva.IdCultiva>
      <TesteXML.Agricultor_Cultiva.RegistroRural>9802</TesteXML.Agricultor_Cultiva.RegistroRural>
      <TesteXML.Agricultor_Cultiva.NrInscricao>1340</TesteXML.Agricultor_Cultiva.NrInscricao>
      <TesteXML.Agricultor_Cultiva.Ano>1999</TesteXML.Agricultor_Cultiva.Ano>
    </TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  + <TesteXML.Agricultor_Cultiva>
  - <TesteXML.Agricultor_Fisica>
      <TesteXML.Agricultor_Fisica.Identificador>2</TesteXML.Agricultor_Fisica.Identificador>
      <TesteXML.Agricultor_Fisica.CPF>233.199.111-01</TesteXML.Agricultor_Fisica.CPF>
```

Figura 21 – Fragmento de arquivo *XML* para a base de dados proposta utilizando notação com pontos.

Embora nos exemplos anteriores tenha sido usado somente um tipo que corresponde a uma seqüência de caracteres, para simplificação dos exemplos, podemos efetuar o mapeamento de diversos tipos de dados diferentes, conforme exemplificado na Tabela 4.

TIPO *XML*	TIPO NA BASE DE DADOS
bin.base64	*binary, image, varbinary, varchar*
Boolean	*bit*
char	*char*
datetime	*datetime*
int	*int*
r4	*real*
r8	*decimal, float, money, numeric*
string	*nchar, ntext, nvarchar, text*
ui1	*tinyint*
ui8	*timestamp*
uuid	*uniqueidentifier*

Tabela 4 - Mapeamento de tipos *XML* para tipos padrão de bases de dados.

4.6 LINGUAGENS DE TRANSFORMAÇÃO *XSL* E *XSLT*

A denominação *XSL* é um acrônimo para *XML Stylesheet Language*, enquanto *XSLT* é a *XML Stylesheet Language Transformations*. Ambas atuam como folhas de estilo, semelhantes às *CSS*[26] dos documentos *HMTL.* A *XSL* está para a *XML* assim como a *CSS* está para a *HTML*, mas possue um poder de processamento maior.

Conforme MEGGINSON (1998) e HAROLD (1999), podemos utilizar as folhas de estilo *CSS* para trabalhar com *XML*, conforme mostrado no Quadro 2.

	CSS	*XSL*
Uso com *HTML*	Sim	Sim
Uso com *XML*	Não	Sim
Linguagem de transformação ?	Não	Sim
Segue a sintaxe...	"CSS"	*XML*

Quadro 2 - Comparação entre *CSS* e *XSL* para controlar a apresentação de documentos *XML*

Enquanto a *XSL* basicamente preocupa-se em transformar a apresentação dos documentos, embora possa ser utilizada para algumas tarefas maiores, a *XSLT* foi especificamente criada para transformar documentos *XML* em outros documentos *XML* e foi desenhada para ser utilizada como parte da *XSL*. Inclui um vocabulário *XML* para especificar formatação. A *XSL* especifica o estilo para um documento *XML* e utiliza *XSLT* para descrever como um documento pode ser transformado em outro. Porém, a *XSLT* também foi desenhada para funcionamento independente da *XSL*, mas não foi especificada para ser uma linguagem de transformação de uso geral, somente para os tipos de transformação que são necessários quando utilizada como parte da *XSL*.

De acordo com HAROLD (1999) a sintaxe de ambas é relativamente complexa, mas sua utilização garante uma flexibilidade enorme para a tecnologia. Seu uso basicamente tem sido norteado para a transformação de um documento *XML* em um documento *HTML* apresentável em um navegador padrão.

---

[26] *Cascade Stylesheets*

O documento exibido na Figura 22 é utilizado como exemplo de transformação. Notar que o documento não necessita de um arquivo de definição externo, pois a definição de suas características está incluída no prólogo do próprio documento.

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl= "http://www.w3.org/1999/XSL/Transform">
 <xsl:template match = "/">
<HTML xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <BODY>
<!-- demonstracao de visualizacao. //-->
<TABLE border="2">
<CAPTION class="titulo">Informacões cadastrais</CAPTION>
<THEAD class="cabecalho">
   <TD>Nome</TD>
   <TD>Endereço</TD>
   <TD>Cidade</TD>
   <TD>Estado</TD>
</THEAD>
<xsl:for-each select="database/registro">
   <TR> <TD class="cell">
      <xsl:value-of select="Nome"/>
   </TD> <TD class="cell">
      <xsl:value-of select="Endereco"/>
   </TD> <TD class="cell">
      <xsl:value-of select="Cidade"/>
   </TD> <TD class="cell">
      <xsl:value-of select="Estado"/>
   </TD></TR>
</xsl:for-each>
</TABLE>
</BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```

A aplicação típica de um arquivo de definições "XSL" é a transformação de um documento "XML" em um documento de apresentação "HTML", embora também possamos gerar uma apresentação para um telefone celular, por exemplo.
As definições ao lado produzem a apresentação mostrada abaixo.

Informacões cadastrais

Nome	Endereço	Cidade	Estado
José Simão	Reinaldo Tha, 353	Curitiba	PR

Figura 22 – Transformação de um documento *XML* em um documento *HTML* por meio da utilização de transformações *XSL*

Visto que esta linguagem é adequada a transformações de documentos, ela também pode ser utilizada quando se efetua a leitura de um documento *XML* como um meio de extrair dele as informações desejadas. Podemos, por exemplo, gerar um *script SQL* a partir de um docuemento *XML* (TURAU, 1999).

4.7 CONSULTAS POR MEIO DA *XML QUERY LANGUAGE* (*XQUERY*)

Recentemente[27] o *W3C* disponibilizou a primeria versão de uma recomendação descrevendo uma linguagem de consulta para *XML*, baseada na linguagem de consulta estruturada, *SQL,* padrão em bancos de dados que seguem a

[27] A recomendação foi disponibilizada na Internet em 27/agosto/2001.

álgebra relacional, na OQL[28], utilizada em sistemas orientados a objetos, e na *NRA*[29], álgebra relacional aninhada, utilizada em bancos de dados objeto-relacionais. A linguagem foi denominada *XQuery*.

A *XQuery* possui um conjunto de regras, que formam seu núcleo (*core*) e formam sua álgebra, garantindo que as operações possam ser bem definidas e também que possa ser utilizado um mecanismo de otimização de consultas.

A semântica é definida tanto para a forma estática, na forma de regras de inferência para tipos, quanto dinâmica (ou operacional) que apresenta as regras de inferência para valores.

4.7.1 A NOTAÇÃO *XQUERY*

A seguir temos representada a notação para a linguagem *XQuery*, conforme especificações do documento *XQuery 1.0 Formal Semantics* do W3C.

A simbologia utilizada segue a seguinte regra:

```
E       expressão
l       nome local (NCName)
a       qualificador de nome
n:l     nome qualificado com namespace n e nome local l
$v      variáveis
$dot    variável que contém o nó atual
        $roots  variável que contém uma seqüência de nós raíz de
        documentos
T       tipo
```

O nó atual: é sempre indicado por um ".":

```
[[ . ]] e corresponde a $dot
```

O nó raíz: indicado por uma barra "/":

```
[[ / ]] e corresponde a $roots
```

Navegação: no exemplo a seguir, "a" é um *NameSet*, ou elemento.

```
[[ /a ]] equivale a
         [[ $root/a ]]
```

Navegação recursiva:

```
[[ E//a ]] equivale a
           [[ descendent-or-self([[E]])/a ]]

[[ //a ]] equivale a
          [[ descendent-or-self($root)/a ]]
```

---

[28] *Object Query Language*

[29] *Nested Relational Algebra*

Acesso a nós:

```
[[ E/NODE() ]] equivale a
               for $v1 in [[ E ]] return children($v1)
```

Acesso a texto:

```
[[ E/TEXT() ]] equivale a
               string([[ E ]])
```

Deferência:

```
[[E => a]] equivale a
             for $v1 in [[ E ]] return
               for $v2 in deref($v1) return
                  typeswitch ($v2) as $v3
                    case ELEMENT a (AnyComplexType) return $v3
             default return ()

[[E => @a]] equivale a
             for $v1 in [[ E ]] return
               for $v2 in deref($v1) return
                  typeswitch ($v2) as $v3
                    case ATTRIBUTE a (AnyComplexType) return $v3
             default return ()
```

Predicados:

```
[[ E1[i] ]] equivale a [[ E1[i to i] ]]
```

(A partir de uma expressão booleana com o parâmetro, retorna somente os nós para os quais a expressão é verdadeira).

```
[[ E1[i1 to i2] ]] equivale a
                    for $v in index([[ E1 ]]) return
                      if ([[ $v/fst/data() >= i1 ]] and [[
                             $v/fst/data() <= i2 ]])
                      then [[ $v/snd/deref() ]] else ()
```

(Este predicado recebe um inteiro como parâmetro e retorna os nós cujo índice na seqüência possui valor valor igual ao inteiro informado).

```
[[ E1[E2] ]] equivale a
             for $v in [[ E1 ]] return
                 let $dot := $v return
               if [[ E2 ]] then $v else ()
```

(Um predicado em faixa seleciona nós cujo índice esteja entre os valores apontados na faixa fornecida pela expressão).

Pais (“parent”):

```
[[ E/.. ]] equivale a parent([[ E ]])
```

Ordenação:

```
[[ E sortby E1 ASCENDING, ..., En ASCENDING ]]
```

Operadores booleanos:

```
[[ E1 = E2 ]] equivale a
               [[ not(empty(for $v1 in [[ E1 ]] return
                  for $v2 in [[ E2 ]] return
                  if eq($v1,$v2) then $v1 else ())) ]]

[[ E1 == E2 ]] equivale a
                [[ not(empty(for $v1 in [[ E1 ]] return
                   for $v2 in [[ E2 ]] return
                   if nodeeq($v1 = $v2) then $v1 else ())) ]]
```

```
[[ E1 < E2 ]] equivale a
              [[ not(empty(for $v1 in [[E1]] return
                 for $v2 in [[E2]] return
                 if lt($v1, $v2) then $v1 else ())) ]] ]]

[[ E1 <= E2 ]] equivale a
               [[ not(empty(for $v1 in [[E1]] return
                  for $v2 in [[E2]] return
                  if lteq($v1, $v2) then $v1 else ())) ]]

[[ E1 >= E2 ]] equivale
               [[ not(empty(for $v1 in [[E1]] return
                  for $v2 in [[E2]] return
                  if gteq($v1, $v2) then $v1 else ())) ]]

[[ E1 > E2 ]] equivale a
              [[ not(empty(for $v1 in [[E1]] return
                 for $v2 in [[E2]] return
                 if gt($v1, $v2) then $v1 else ())) ]]

[[ E1 != E2 ]] equivale a
               [[ not(empty(for $v1 in [[E1]] return
                  for $v2 in [[E2]] return
                  if neq($v1, $v2) then $v1 else ())) ]]
```

Função conjunto vazio:

```
[[ empty(E) ]] equivale a
               [[ E ]] e isto retorna ()
```

União de conjuntos:

```
[[ E1 UNION E2 ]] equivale a
                  distinct-node([[E1]], [[E2]])
```

Intersecção de conjuntos:

```
[[ E1 INTERSECT E2 ]] equivale a
                      for $v1 in [[ E1 ]] return
                       if not((for $v2 in [[ E2 ]] return
                          if node_equal($v2,$v1) then $v2
                          else ()) = ()) then $v1
                      else ()

[[ E1 EXCEPT E2 ]] equivale a
                   for $v1 in [[ E1 ]] return
                    if ((for $v2 in [[ E2 ]] return
                       if node_equal($v2,$v1) then $v2
                       else ()) = ()) then $v1
                    else ()
```

Funções “antes e depois”:

(retornam os elementos de E1 que ocorrem antes ou depois, conforme o operador, de E2).

```
[[ E1 BEFORE E2 ]] equivale a
                   for $v1 in [[ E1 ]] return
                    if not((for $v2 in [[ E2 ]] return
                    if before($v2,$v1) then $v2 else ()) = ())
                       then $v1
                    else ()

[[ E1 AFTER E2 ]] equivale a
                  for $v1 in [[ E1 ]] return
```

```
                    if not((for $v2 in [[ E2 ]] return
                    if after($v1,$v2) then $v2 else ()) = ())
                       then $v1
                    else ()
```

Quantificadores:

```
[[ some $v in E1 satisfies E2 ]] equivaale a
                                  not((for $v in [[ E1 ]] return
                                   if [[ E2 ]] then $v
                                   else ()) = ())

[[ every $v in E1 satisfies E2 ]] equivale a
                                  (for $v in [[ E1 ]] return
                                    if not([[ E2 ]]) then $v
                                    else ()) = ()
```

Operador de seleção:

```
[[ typeswitch (E0) as $v
     case $v1 : T1 return E1
     ...
     case $vn : Tn return En ]]
É equivalente a
  typeswitch ([[ E0 ]]) as $v
    case T1 return [[ E1 ]]
    ...
    case Tn return [[ En ]]
    default return ERROR
```

Promoção de tipos (não muda o tipo original):

```
[[ CAST AS Type (E) ]] equivale a
                       CAST AS Type ([[ E ]])
```

Troca de tipos (muda o tipo original. Pode gerar uma exceção):

```
[[ (TREAT AS Type) E ]] equivale a
                         typeswitch ([[ E ]]) as $v
                           case Type return $v
                           default return ERROR
```

Teste de tipo de dado:

```
[[ E INSTANCEOF Type ]] equivale a
                         typeswitch ([[ E ]]) as $v
                           case Type return true
                           default return false
```

Operadores de agregação:

```
function sum((Integer | Float)* $x)
  {
     if ($x = ())
     then 0
     else head($x) + sum(tail($x))
  }

function count(AnyType $x) returns Integer
  {
     if ($x = ())
     then 0
     else 1 + count(tail($x))
  }

function avg((Integer | Float)* $x) returns (Integer | Float)
  {
```

```
      sum($x) div count($x)
  }

function get_max((Integer | Float)* $x, (Integer | Float) $y)
returns (Integer | Float)
  {
     if $x = ()
     then $y
     else
       let $first := tail($x) return
       if ($first > $y)
       then get_max(tail($x),$first)
       else get_max(tail($x),$y)
  }

function max((Integer | Float)* $x) returns (Integer | Float)
  {
     get_max($x, -Infinite)
  }

function get_min((Integer | Float)* $x, (Integer | Float) $y)
returns (Integer | Float)
  {
     if $x = ()
     then $y
     else
       let $first := tail($x) return
       if ($first < $y)
       then get_min(tail($x),$first)
       else get_min(tail($x),$y)
  }

function min((Integer | Float)* $x) returns (Integer | Float)
  {
     get_min($x, +Infinite)
  }
```

4.8 EXEMPLO DE USO DA LINGUAGEM *XQUERY* (*W3C*)

O exemplo fornecido pelo *W3C* para compreensão da linguagem *XQuery* está baseado nos arquivos mostrados nas figuras Figura 23 e Figura 24 representadas a seguir:

SCHEMA

```
<xs:group name="Bib">
  <xs:element name="bib">
    <xs:complexType>
      <xs:group ref="Book" minOccurs="0"
                maxOccurs="unbounded"/>
    </xs:complexType>
  </xs:element>
</xs:group>
<xs:group name="Book">
  <xs:element name="book">
    <xs:complexType>
      <xs:attribute
         name="year" type="xs:integer"/>
      <xs:attribute
         name="isbn" type="xs:string"/>
      <xs:element
         name="title" type="xs:string"/>
      <xs:element
         name="author"type="xs:string"
                maxOccurs="unbounded"/>
    </xs:complexType>
  </xs:element>
</xs:group>
```

Instância de documento

```
<bib>
  <book year="1999" isbn="1-55860-622-X">
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
  </book>
  <book year="2001" isbn="1-XXXXX-YYY-Z">
    <title>XML Query</title>
    <author>Fernandez</author>
    <author>Suciu</author>
  </book>
</bib>
```

Figura 23 – Documento de amostra do *W3C* para demonstração da linguagem *XQuery*

Na representação da Figura 23 notamos a declaração de um *Schema* que agrupa livros (*book*) em entidades *Bib*. Ao lado do esquema é exibida uma instância de documento. A notação na forma *XQuery*, bem como a criação de uma variável (*$book0*) para conter livros, e uma para conter a base (*$bib0*) é apresentada na Figura 24.

Definições em XQuery

```
TYPE Bib = ELEMENT bib (Book*)
TYPE Book =
  ELEMENT book
    ( ATTRIBUTE year  (xs:integer) &
      ATTRIBUTE isbn  (xs:string)
      ELEMENT   title (xs:string),
      (ELEMENT   author(xs:string))+
    )
LET $bib0 :=
<bib>
  <book year="1999" isbn="1-55860-622-X">
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
  </book>
  <book year="2001" isbn="1-XXXXX-YYY-Z">
    <title>XML Query</title>
    <author>Fernandez</author>
    <author>Suciu</author>
  </book>),
</bib> : Bib
RETURN ...
```

```
LET $book0 :=
    <book
       year="1999"
       isbn="1-55860-622-X">
       <title>
         Data on the Web
       </title>
       <author>
         Abiteboul
       </author>
       <author>
         Buneman
       </author>
       <author>
         Suciu
       </author>
      </book> : Book
RETURN ...
```

Figura 24 – Notação em *XQuery* e criação de variáveis.

A partir destas figuras representativas do conteúdo da base e sua estrutura serão realizadas algumas operações a título de exemplo.

4.8.1 PROJEÇÃO

```
$bib0/book/author
Produz ==>(<author>Abiteboul</author>, <author>Buneman</author>,
           <author>Suciu</author>, <author>Fernandez</author>,
           <author>Suciu</author>)
          : (ELEMENT author (xs:string))*
```

(retorna todos os autores presentes na variável $bib0)
(`: (ELEMENT author (xs:string))*` corresponde ao tipo de elemento retornado)

4.8.2 PROJEÇÃO DE ATRIBUTOS

```
$book0/@year
Produz ==> ATTRIBUTE year "1999"
           : ATTRIBUTE year (xs:string)
```

4.8.3 PROJEÇÃO DE VALORES DE ATRIBUTOS

```
$book0/@year/data()
Produz ==> 1999
           : xs:integer
```

4.8.4 PROJEÇÃO DE ATRIBUTOS MULTIVALORADOS

```
$book0/author/data()
Produz ==> ("Abiteboul", "Buneman", "Suciu")
           : xs:string+
```

4.8.5 ITERAÇÃO

Permite a navegação pelo conjunto de dados e sua transformação em um conjunto diferente.

```
FOR $b IN $bib0/book RETURN
      <book> { $b/author, $b/title } </book>
Produz ==> (<book>
              <author>Abiteboul</author>
              <author>Buneman</author>
              <author>Suciu</author>
              <title>Data on the Web</author>
            </book>,
            <book>
              <author>Fernandez</author>
              <author>Suciu</author>
              <title>XML Query</author>
            </book>)
           : (ELEMENT book(
              (ELEMENT author(xs:string))+,
                ELEMENT title(xs:string))
           )*
```

(O código acima retida os elementos “year” e “isbn” do conjunto de dados)

4.8.6 SELEÇÃO

```
FOR $b IN $bib0/book
    WHERE $b/@year/data() <= 2000 RETURN
      $b
Produz ==> <book year="1999" isbn="1-55860-622-X">
             <title>Data on the Web</title>
             <author>Abiteboul</author>
             <author>Buneman</author>
             <author>Suciu</author>
           </book>
           : Book*
```

(Seleciona livros cujo ano de publicação seja igual ou inferior a 2000)

Em geral uma expressão na forma

```
where e1 return e2
```

é convertida para a forma

```
if e1 then e2 else ()
```

na qual e_1 e e_2 são expressões. O indicador `()` é uma expressão que representa uma uma seqüência vazia.

Assim, a expressão de consulta anterior (ano <=2000) ficaria:

```
FOR $b IN $bib0/book RETURN
  IF $b/@year/data() <= 2000 THEN $b ELSE ()
```

Que produz o mesmo resultado que a anterior

A expressão seguinte seleciona livros em que algum do autores chama-se "Buneman", e equivale a um "like %" da linguagem "SQL":

```
FOR $b IN $bib0/book
    WHERE SOME $a IN $b/author SATISFIES $a/data() = "Buneman"
RETURN
      $b
Produz ==> <book year="1999" isbn="1-55860-622-X">
             <title>Data on the Web</title>
             <author>Abiteboul</author>
             <author>Buneman</author>
             <author>Suciu</author>
           </book>
           : Book*
```

Uma consulta para livros em que o nome do autor seja "Buneman", como abaixo, produz um resultado vazio (pois o nome do autor, para o exemplo, é multivalorado):

```
FOR $b IN $bib0/book
    WHERE EVERY $a IN $b/author SATISFIES $a/data() = "Buneman"
          RETURN $b
Produz ==> ()
           : Book*
```

4.8.7 JUNÇÕES (“JOINS”)

Para exemplificar as junções definimos um segundo elemento, correspondente aos comentários a respeito dos livros, exibido na figura a seguir:

```
TYPE Reviews  =
  ELEMENT reviews (
    (ELEMENT book (
      ELEMENT title  (xs:string),
      ELEMENT review (xs:string))
    )*
  )
LET $review0 :=
  <reviews>
    <book>
      <title>XML Query</title>
      <review>A darn fine book.</review>
    </book>,
    <book>
      <title>Data on the Web</title>
      <review>This is great!</review>
    </book>
  </review> : Reviews
RETURN ...
```

Figura 25 – Exemplo para demonstrações de junções (“joins”) na linguagem “XQuery”

Para efetuarmos a junção, produzindo uma saída que contém livros, seus autores e os comentários, fazemos:

```
FOR $b IN $bib0/book, $r IN $review0/book
    WHERE $b/title/data() = $r/title/data() RETURN
      <book>{ $b/title, $b/author, $r/review }</book>
Produz ==> (<book>
              <title>Data on the Web</title>
              <author>Abiteboul</author>
              <author>Buneman</author>
              <author>Suciu</author>
              <review>A darn fine book.</review>
            </book>,
            <book>
             <title>XML Query</title>
             <author>Fernandez</author>
             <author>Suciu</author>
             <review>This is great!</review>
            </book>)
           : (ELEMENT book(
              ELEMENT title  (xs:string),
              (ELEMENT author (xs:string))+,
               ELEMENT review (xs:string))
           )*
```

4.8.8 REFERÊNCIAS PARA NÓS DE DOCUMENTOS

A linguagem suporta referências para nós, portanto cada objeto possui seu próprio identificador, ou seja, seu *oid*[30]. Isto impossibilita comparações diretas entre os elementos:

```
LET $a1 := <author>Suciu</author>,
$a2 := <author>Suciu</author>
RETURN $a1 == $a2
Produz ==> false
           : xs:boolean
```

4.8.9 REESTRUTURAÇÃO E REAGRUPAMENTO DOS NÓS DE UM DOCUMENTO

Podemos reestruturar / reconstruir um documento como mostrado a seguir, no qual o documento exemplo será reconstruído apresentando cada autor de um livro separadamente, em conjunto com o título do livro:

```
FOR $a IN distinct-value($bib0/book/author/data()) RETURN
      <biblio>
        <author>{ $a }</author>
        { FOR $b IN $bib0/book, $a2 IN $b/author/data()
          WHERE $a = $a2 RETURN
            $b/title
        }
      </biblio>
Produz ==> (<biblio>
             <author>Abiteboul</author>
             <title>Data on the Web</title>
            </biblio>,
            <biblio>
             <author>Buneman</author>
             <title>Data on the Web</title>
            </biblio>,
            <biblio>
             <author>Suciu</author>
             <title>Data on the Web</title>
             <title>XML Query</title>
            </biblio>,
            <biblio>
             <author>Fernandez</author>
             <title>XML Query</title>
            </biblio>)
           : (ELEMENT biblio (
              ELEMENT author (xs:string),
              (ELEMENT title  (xs:string))*)
           )*
```

(Esta expressão equivale a um *self-join* de livros e autores)

[30] *Unique Object Identificator*

4.8.10 ORDENAÇÃO E RECUPERAÇÃO ORDENADA

Podemos ordenar os elementos do exemplo:

```
index($book0/author)
Produz ==> (<q:pair><q:fst>1</q:fst>
            <q:snd><q:ref><author>Abiteboul</author>
              </q:ref></q:snd></q:pair>,
            <q:pair><q:fst>2</q:fst>
            <q:snd><q:ref><author>Buneman</author>
              </q:ref></q:snd></q:pair>
            <q:pair><q:fst>3</q:fst>
            <q:snd><q:ref><author>Suciu</author>
              </q:ref></q:snd></q:pair>)
           : (ELEMENT q:pair(
              ELEMENT q:fst (xs:integer),
              ELEMENT q:snd
              (REFERENCE
               (ELEMENT author (xs:string)))))+
```

E posteriormente recuperá-lo

4.8.11

```
                                title>
              <review>This is pretty good too!</review>
             </book>)
           : (ELEMENT book (
              ELEMENT title (xs:string),
               ELEMENT review (xs:string))
           )*
```

4.8.12 AGREGAÇÃO

```
FOR $b IN $bib0/book
       WHERE count($b/author) > 2 RETURN
             $b
Produz ==> <book year="1999" isbn="1-55860-622-X">
             <title>Data on the Web</title>
             <author>Abiteboul</author>
             <author>Buneman</author>
             <author>Suciu</author>
            </book>
           : Book*
```

(Esta expressão seleciona livros que tenham mais de dois autores)

4.8.13 FUNÇÕES

Podemos imaginar uma consulta na qual queremos recuperar livros em que “Buneman” não é autor:

```
FOR $b IN $bib0/book
    WHERE EVERY $a IN $b/author SATISFIES
          NOT($a/data() = "Buneman") RETURN $b
Produz ==> <book year="2001" isbn="1-XXXXX-YYY-Z">
             <title>XML Query</title>
             <author>Fernandez</author>
             <author>Suciu</author>
           </book>
           : Book*
```

Esta consulta pode ser expressada de forma mais genérica com a criação de uma função:

```
DEFINE FUNCTION notauthor
       (xs:string $s, Book $b) RETURNS xs:boolean
         {
          EVERY $a IN $b/author SATISFIES NOT($a/data() = $s)
         }
```

A aplicação da função:

```
FOR $b IN bib0/book
    WHERE notauthor("Buneman", $b) RETURN $b
Produz ==> <book year="2001" isbn="1-XXXXX-YYY-Z">
             <title>XML Query</title>
             <author>Fernandez</author>
             <author>Suciu</author>
           </book>
           : Book*
```

Conjunto de funções de uso geral: várias funções foram definidas para facilitar a utilização da linguagem. Alguns exemplos são:

- `attributes(Expr)`: retorna atributos de um elemento;
- `children(Expr)`: retorna os filhos de um elemento;
- `comment(Expr)`: para comentários;
- `name(Expr)` ; Retorna a *TAG* do nome do elemento;
- `namespace-uri(Expr)`: extrai a localização de um elemento;
- `parent(Expr)`: retorna os pais de um elemento.

Notamos então que a aplicação dos mecanismos *XQuery* permite pesquisar a ocorrência de atributos e entidades em documentos, os quais mantém a cada momento uma identidade com sua origem. A linguagem é suficientemente

poderosa para permitir quaisquer *queries*[31] necessárias, bem como permite a transformação de um tipo de documento em outro. Esta última característica é altamente desejável, pois permite o mapeamento de entidades compostas para entidades simples, e vice-versa.

[31] Consultas

5 O PROTÓTIPO X-VIQUEN

Após efetuada a integração de bases de dados surge a necessidade de consultas à base integrada. Em geral a consulta é formulada a partir de um modelo que possui um poder de representação grande, e torna-se necessário mapear os conceitos selecionados para as bases originais.

A abordagem adotada no Sistema *VIQUEN* (GUEIBER, 2001), foi a de efetuar a conexão com um banco de dados relacional, mapear os objetos encontrados no banco para o modelo *ERC+* e apresentá-los em um ambiente gráfico, disponibilizando ao usuário uma ferramenta de consulta visual.

Para a transformação dos objetos vindos do *RDBMS*[32] para objetos *ERC+* o projeto fez uso de cursores, uma implementação do banco de dados *ORACLE*©, utilizado no protótipo. A Figura 26 resume o funcionamento do Sistema *VIQUEN*.

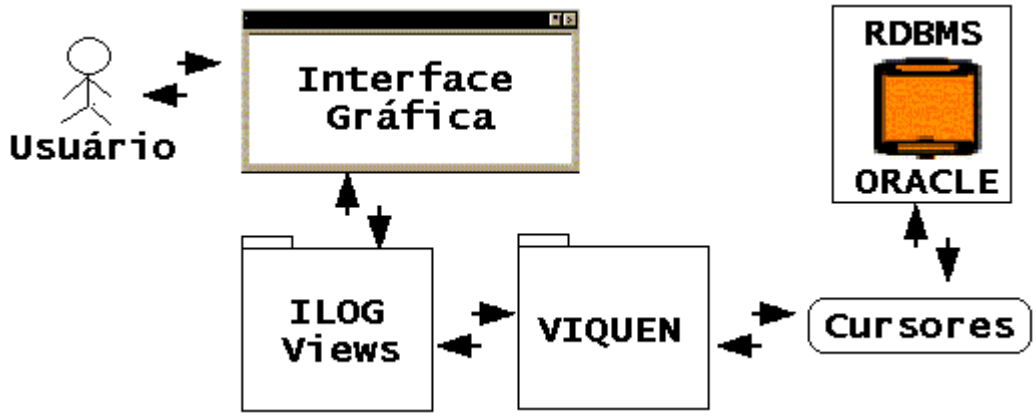

Figura 26 - Diagrama funcional do protótipo do Sistema *VIQUEN*

O operador *R-Join*, da álgebra *ERC+*, por exemplo, é implementado pelo mapeamento de um conjunto de relações para uma determinada entidade. As relações são extraídas da base de dados usando-se cursores, que implementam também relações de união por igualdade entre as relações, conforme mostrado a seguir:

```
select cursor( select R1.*, cursor( select R2.*, ..., cursor( select Ri.*, from
Ri where J(Ri-1, Ri)) Ri from R2 where J(R1, R2))R2, ..., cursor( select Rk.* from
Rk where J(R1, Rk))Rk from R1) Z
```

[32] *Relational Database Manegement System*

Nesta operação, Ri são relações, J é o operador de junção por igualdade e Z é o novo objeto criado. Após sua execução a consulta permite retornar atributos complexos multivalorados de uma base relacional.

A abordagem adotada no Sistema *VIQUEN* permitiu atingir o objetivo proposto, de produzir um esquema a partir de uma base relacional utilizando o modelo *ERC+* com bastante sucesso, mas limitou a ferramenta ao banco de dados *ORACLE*©.

Neste trabalho efetuamos o mapeamento dos objetos exibidos no esquema por meio de documentos *XML* que descrevem os objetos originais e seus locais de origem, eliminando a necessidade de cursores no *RDBMS* e possibilitando a realização de consultas em mais de uma base de dados.

Conforme podemos notar na representação anteriormente mostrada no operador *R-Join*, existe um "aninhamento" dos resultados, que são agrupados em cursores. A abordagem deste trabalho substituirá os cursores por documentos *XML*, mantendo o aninhamento adequado à representação semântica pretendida.

5.1 EXTRAÇÃO DO ESQUEMA DA BASE DE DADOS

ANDERSON (1994) sugere uma metodologia para extrair esquemas de bases relacionais, mas afirma que somente o dicionário de dados não é suficiente para identificar todos os tipos *ER* envolvidos, e devem ser estudadas também as declarações de manipulação da base.

Em seu método, a partir da criação de um diagrama de conexões, aplicam-se uma série de regras de transformações, que são aplicadas em diagramas de conexão construídos a partir da pesquisa das chaves e suas ligações. Portanto, necessariamente deverá existir integridade referencial declarada / implementada na base em questão.

A patir da aplicação das regras teremos três derivações, a serem refinadas posteriormente: *id-independent*, que geram entidades; *id-dependent*,

subdividida naquelas que possuem *links* para mais de uma ligação (*relationship*) ou uma única ligação (tipo dependente ou entidade fraca).

O autor introduz o conceito de *dependent type*, utilizado na generalização de atributos multivalorados e entidade fraca durante o processo de tradução. Estes tipos serão separados no processo de refinamento.

Na seqüência serão identificadas as ligações entre nós que correspondem a somente um objeto, e que formarão as propriedades, a dois objetos, com dependência de inclusão, que gerarão relações do tipo *is-a* e ligações a dois objetos que possuam valores comuns (*overlapping*) que irão gerar relacionamentos *may-be-a*. Tais proposições são definidas por SPACCAPIETA e outros (1995) conforme mostrado na Figura 27.

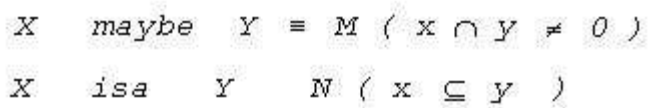

Figura 27 - Proposições *is-a* e *maybe-a* do modelo *ERC+*

Regras de refinamento são aplicadas sobre o resultado a partir dos dois grupos prévios para gerar a tradução final.

Neste trabalho deixaremos o processo de tradução a cargo dos mecanismos já implementados na interface do Sistema *VIQUEN* (GUEIBER, 2001), embora possamos previamente pesquisar equivalências que levem a transformações de esquemas, como sugerido por McBRIEN (1997), desde que a definição do documento de integração assim o exija. Estas equivalências seriam pesquisadas nos documentos *XML* envolvidos, a partir de suas definições.

5.2 ATRIBUTOS COMPLEXOS / MULTIVALORADOS

Dada uma base de dados hipotética, que representa filmes, seus autores e os atores participantes, poderíamos ter uma representação de sua organização do como mostrado na Figura 9, já citada (página 25).

Naquela representação podemos notar a dificuldade existente para o armazenamento em bases de dados relacionais do atributo *Actors*, pois ele está

especificamente ligado ao atributo *Title*, e pode ocorrer um número indeterminado de vezes.

Ao mesmo tempo notamos que uma representação hierárquica é capaz de expressar corretamente uma pesquisa sobre um diretor, seus filmes e os atores que dele participaram.

Considerando a forma hierárquica de um documento XML poderíamos possuir a seguinte representação:

```
<MOVIES>
...
 <DIRECTOR nome="Hitchcock">
  <TITLE nome="The trouble with Harry">
    <ACTOR>"John Forsyte"</ACTOR>
    <ACTOR>"Edmund Gwenn"</ACTOR>
    <ACTOR>"Shirley MacLaine"</ACTOR>
    <ACTOR>"Alfred Hitchcock"</ACTOR>
  </TITLE>
  <TITLE nome="The Birds">
    <ACTOR>"Tippi Hedren"</ACTOR>
    <ACTOR>"Rod Taylor"</ACTOR>
    <ACTOR>"Suzanne Pleshette"</ACTOR>
    <ACTOR>"Alfred Hitchcock"</ACTOR>
  </TITLE>
  <TITLE nome="Psycho">
    <ACTOR>"Anthony Perkins"</ACTOR>
    <ACTOR>"Janet Leigh"</ACTOR>
    <ACTOR>"Alfred Hitchcock"</ACTOR>
  </TITLE>
 </DIRECTOR>
</MOVIES>
```

Uma vez especificado o documento em *XML*, neste formato podemos realizar as pesquisas por meio de mecanismos oferecidos pelas extensões, tais como *XQuery*, ou mesmo a partir de transformações realizadas com o par *XSL/XSLT*.

Supondo que dispomos de um documento contendo os dados integrados, em *XML-QL*, por exemplo, poderíamos formular a seguinte consulta TURAU (1999):

```
WHERE
 <MOVIES>
  <DIRECTOR>
   <TITLE>
    <nome>$P</nome>
    <ACTOR>"Tippi Hedren"</ACTOR>
   </TITLE>
  </DIRECTOR>
 </MOVIES>
```

```
CONSTUCT $P
```

Esta consulta retornaria o nome dos filmes em que o ator "*Tippi Hedren*" participou (neste caso retornaria *The Birds*).

Opcionalmente, poderíamos representar a consulta utilizando *XQL,* conforme demonstrado por ERDMANN e STUDER (2001):

```
//MOVIES/TITLE/nome [ACTOR/nome="Tippi Hedren"]
```

E como resposta obteríamos o mesmo resultado.

Considerando que documentos *XML* podem ser transformados em outros documentos, utilizando os operadores da linguagem de transformação XSL ou XSLT, podemos recuperar dados em um novo documento, formado a partir da pesquisa:

```
<xsl:template match="ACTOR">
 <xsl:apply-templates/>
 <xsl:for-each select="nome">
  <xsl:choose>
   <xsl:when test="Tippi Hedren">
    ...
   </xsl:when>
  </xsl:choose>
 </xsl:for-each>
```

Mas, devemos considerar também o caso em que não dispomos, ou não queremos dispor, do documento *XML*. Neste caso nossa intenção é a de realizar a consulta correspondente ao subesquema selecionado na ferramenta gráfica diretamente na base de origem.

A abordagem deste trabalho para esta tarefa requer conhecimento sobre os dados. Este conhecimento é fornecido por arquivos de definição de conteúdo de documentos, do tipo *DTD* ou *Schema*, os quais possuem informações sobre os elementos (nome, atributos), relacionamentos e, em nosso caso, obrigatoriamente, sua localização.

A identificação da origem de cada um dos atributos existentes no esquema integrado será realizada numerando-se cada partícula formadora de cada elemento, de maneira a mantermos a identificação da origem da entidade ou atributo, conforme sugerido por ANUTARYA e outros (2000).

A ferramenta então, após gerar a consulta do subesquema, é capaz de localizar nos arquivos de definição originais de onde veio aquela parcela dos dados.

As transformações serão efetuadas aplicando-se mecanismos da linguagem XSL, tal como sugerido por CHAWATHE e outros (1999).

5.3 FUNCIONAMENTO

Para a geração de um documento *XML* a partir de uma base de dados utilizamos as *APIs* descritas no projeto *db2xml*[33] por TURAU (1999). Aquelas funções permitem a conexão com bases de dados relacionais e a geração de documentos *XML* de conteúdo e de descrição de dados (*DTD*, *XML*). A Figura 28 exibe a tela gráfica da ferramenta *db2xml*, embora possamos utilizar somente suas *APIs*.

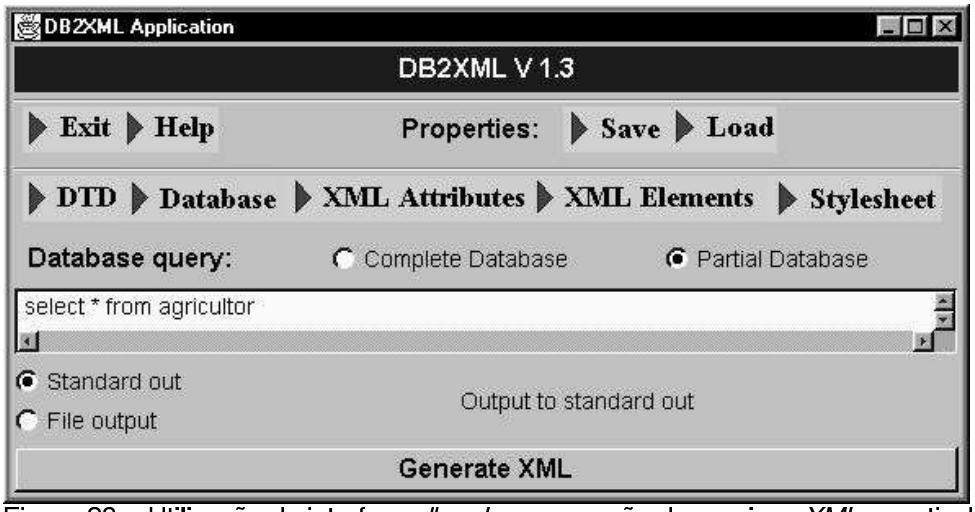

Figura 28 – Utilização da interface *dbxml* para geração de arquivos *XML* a partir de bases relacionais.

Para efetuar a construção de arquivos XSL e a validação dos modelos criados foi utilizada uma ferramenta comercial em versão de avaliação, *XmlSpy*[34].

A partir dos documentos *XML* gerados, para cada um deles é realizado seu mapeamento para um documento de integração, que mantém conhecimento sobre a origem dos documentos parciais, conforme exibido na Figura 29.

[33] Disponível gratuitamente em http://www.informatik.fh-wiesbaden.de/~turau

[34] http://www.xmlspy.com

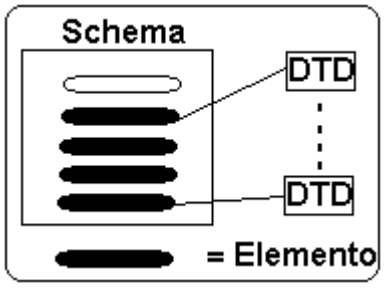

Figura 29 - Um documento de integração possui conhecimento sobre os documentos integrados.

O protótipo utiliza então seu conhecimento dos elementos (vindo dos arquivos de definição de documentos) quando necessita referenciar-se a eles em uma consulta.

5.3.1 EXEMPLO DE DOCUMENTO

A sequir temos a representação de um fragmento de documento gerado a partir de uma consulta à base contendo as descrições originalmente utilizadas no protótipo do Sistema *VIQUEN*. A numeração não faz parte do arquivo e foi incluída somente par fins da explicação a seguir.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <baseDadosXML URL="jdbc:odbc:sql_k6_agricultura">
3      <tabela0 QUERY="select p.identificador,p.nome,
4        f.data_nascimento,f.CPF,f.RG,p.endereco,p.Cidade,
5        p.UF from fisica f, proprietario pwhere
6        p.identificador=
7        f.identificador">
8     <registro0>
9        <identificador ACCESS="RW"
10         NAME="identificador" NULLABLE="false"
11         AUTOINCREMENT="false" TYPE="int"
12         PRECISION="10"
13         >2</identificador>
14       <nome CASESENSITIVE="false"
15         ACCESS="RW" MAXLEN="40" NAME="nome"
16         NULLABLE="false" TYPE="varchar">
17         <![CDATA[Mario Carneiro]]></nome>
18      ...
19    </registro0>
```

A linha 1 contém a definição básica do prólogo de um documento *XML*, conforme já comentado à página 34 deste trabalho.

A linha 2 contém a identificaão da base de dados utilizada (baseDadosXML) e a string utilizada para a conexão com a mesma, neste caso um mapeamento *JDBC / ODBC* (jdbc:odbc:sql_k6_agricultura).

As linhas 3 a 7 descrevem a consulta (*query*) aplicada ao banco de dados mencionado na linha 2.

A linha 8 inicia um elemento de nome *registro0*, que findará na linha 19 deste exemplo (a linha 18 representa as linhas intermediárias, não exibidas).

A linha 9 inicia a descrição do elemento de nome identificador o qual está contido dentro do elemento já mencionado *registro0*. Ainda na linha 9 notamos que o elemento de nome *identificador* possui um atributo de nome *ACCESS*, cujo valor é *RW*. Isto significa que o acesso a este elemento no banco de dados mencionado na linha 2 é para leitura (R=*read*) e gravação (W=*write*). Nas linhas 10, 11 e 12 seguintes os atributos representados informarão que o elemento *identificador* possui na base de dados o nome de identificador (*NAME="identificador"*), não aceita nulos (*NULLABLE="false"*), que ele não é um campo de auto-incremento (*AUTOINCREMENT="false"*), que seu tipo de dados aceita número inteiros (*TYPE="int"*) com precisão de 10 dígitos (*PRECISION="10"*).

Finalmente a linha 13 informa o valor do elemento *identificador*, neste caso 2, e termina a marcação do elemento (com a *TAG* </identificador>).

Na linha 14 temos o início da representação do elemento de nome *nome*, representação esta que vai até a linha 16. Este elemento possui como atributos não ser sensível a maiúsculas e minúsculas (*CASESENSITIVE="false"*), permite acesso para leitura e gravação (*ACCESS="RW"*), aceita no máximo 40 caracteres (*MAXLEN="40"*), seu nome original no banco de dados é nome (*NAME="nome"*), não aceita nulos (*NULLABLE="false"*), e seu tipo de dados aceita caracteres, ocupando somente os espaços efetivamente preenchidos (TYPE="varchar").

Na linha 17 temos finalmente o valor do elemento *nome*, que é *Mario Carneiro*. Devemos notar que a inclusão do nome entre as marcações da *TAG CDATA* (entre *<![CDATA[* e *]]>*) significa que trata-se de uma seqüência de

caracteres. Nesta mesma linha 17 temos o fechamento do elemento, com a marcação *</nome>*.

A linha 18 foi incluída para representar a existência de mais dados, que não são exibidos. A linha 19 termina a estrutura *registro0*, que corresponde a uma tupla da tabela referenciada na consulta da linha 3.

Como observação final, devemos notar a existência, para todos os elementos, do nome original do atributo na tabela do banco de dados. Isto permite que o documento *XML* gerado possa ser descrito com uma nomenclatura dirente da original, seja por possuir um poder semântico maior, seja por facilitar o processo de integração. Podemos, pois, utilizar os nomes dados à visão integrada, sem prejuízo do mapeamento para os elementos originais.

5.4 FORMAÇÃO DA REPRESENTAÇÃO GRÁFICA

A representação gráfica do esquema das descrições de documentos obtidas é realizada por meio da utilização de um *event based parser* (descrito na página 41 deste trabalho).

O *parser* comunica-se com o protótipo do Sistema *VIQUEN*, agora transformado em um conjunto de *APIs*, enviando chamadas a funções de criação de objetos na tela para cada novo elemento, atributo de elemento ou relacionamento encontrado nos documentos de descrição, passando como parâmetros o conteúdo encontrado. A Figura 30 demonstra uma visão geral do processo. Adotou-se a terminologia *X-VIQUEN* de maneira a referenciar as etapas desenvolvidas no presente trabalho, diferenciando-as das etapas presentes no protótipo do trabalho original de Ezequiel Gueiber, o Sistema *VIQUEN* (GUEIBER, 2001).

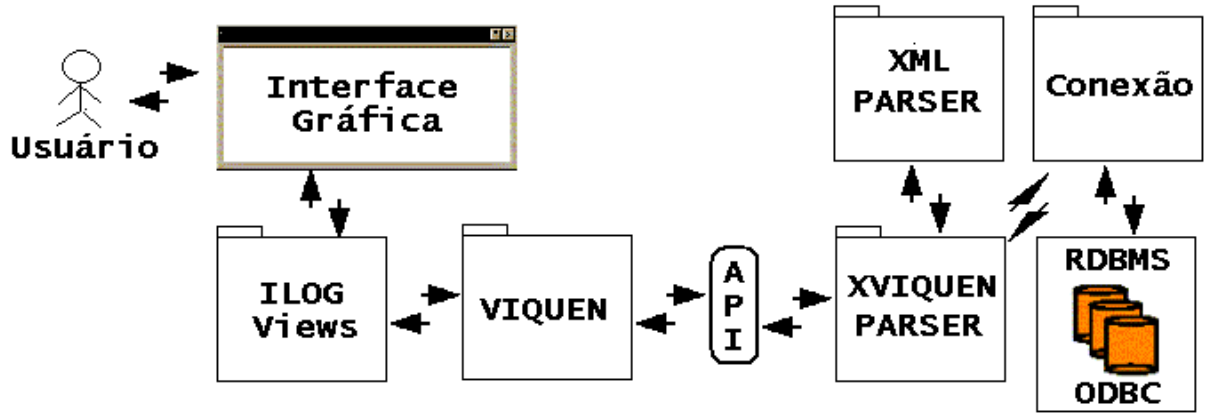

Figura 30 - Visão geral do protótipo *X-VIQUEN*

Dada a possiblidade de transformação de documentos *XML* em outros documentos do mesmo tipo mas com características diferentes, podemos agir nos documentos de maneira a melhorar as visões deles obtidas.

A organização dos documentos permite a criação de atributos de tal maneira que no esquema integrado temos algo semelhante a:

```
<!DOCTYPE ...
 <!ELEMENT ...
  <!ATTLIST ...
   DTDOriginal ... #REQUIRED
<!ELEMENT ...
  <!ATTLIST ...
   DTDOriginal ... #REQUIRED
```

E em cada um dos DTDs quer descrevem bases a serem integradas teremos:

```
<!DOCTYPE ...
 <!ELEMENT ...
  <!ATTLIST ...
   DatabaseOriginal ... #REQUIRED
   NomeOriginal ...
   TipoDadosOriginal ...
```

O procedimento seguinte consiste em efetuar *queries* sequenciadas às bases de origem, de maneira a obter os dados necessários à resposta da consulta.

Então, depois de lido o documento pelo *parser*, e efetuadas as chamadas às *APIs* do Sistema *VIQUEN*, teremos a representação gráfica do modelo, exibido a seguir, na Figura **Erro! Argumento de opção desconhecido.**.

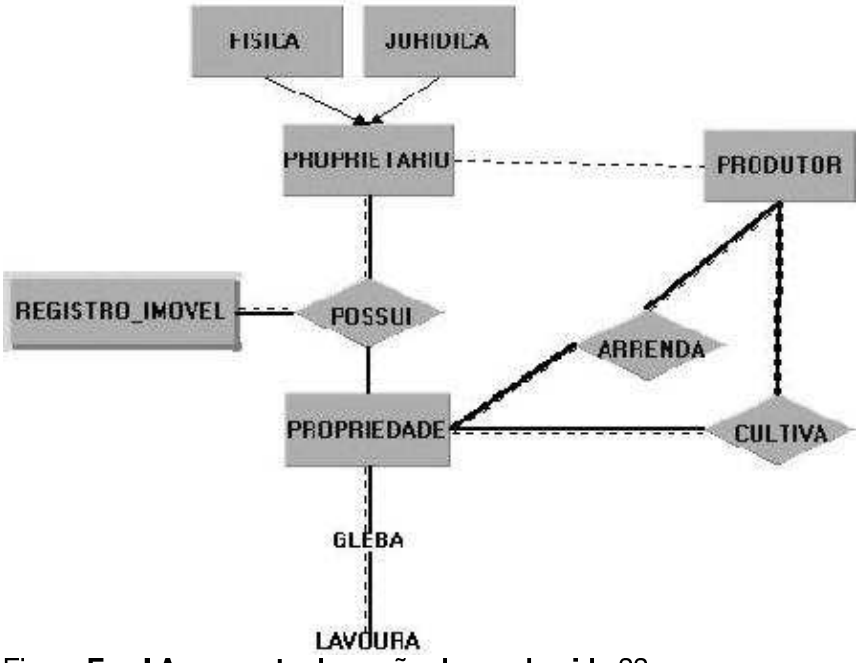

Figura **Erro! Argumento de opção desconhecido.**??

5.5

?? ??**Erro! Nenhum nome foi dado ao indicador.**??

??

?? ??**Erro! Nenhuma seqüência foi especificada.**??

nte à visão desejada pelo usuário, conforme mostrado na Figura 32.

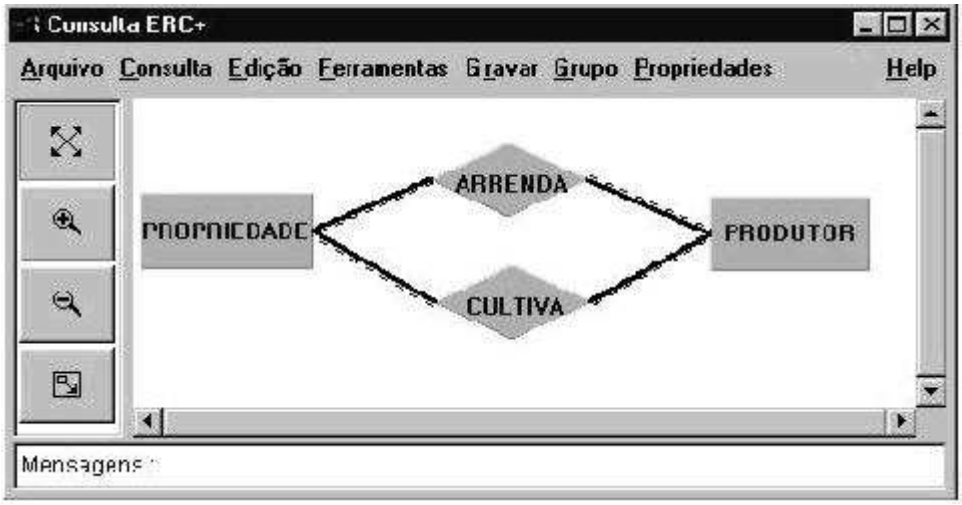

Figura 32 – Subesquema gerado pela seleção de objetos pelo usuário.

A cada momento em que o usuário clica em um objeto é chamada uma *API* do protótipo *X-VIQUEN*, a qual é responsável pela "descoberta" do objeto original e pela construção de um documento *XML* que permita efetuar o mapeamento das características desejadas.

Uma vez estabelecido o documento correspondente ao subesquema solicitado pelo usuário, o protótipo *X-VIQUEN* inicia a formação das consultas necessárias à recuperação dos objetos que serão utilizados para a construção dos elementos solicitados, conforme exibido na Figura 33.

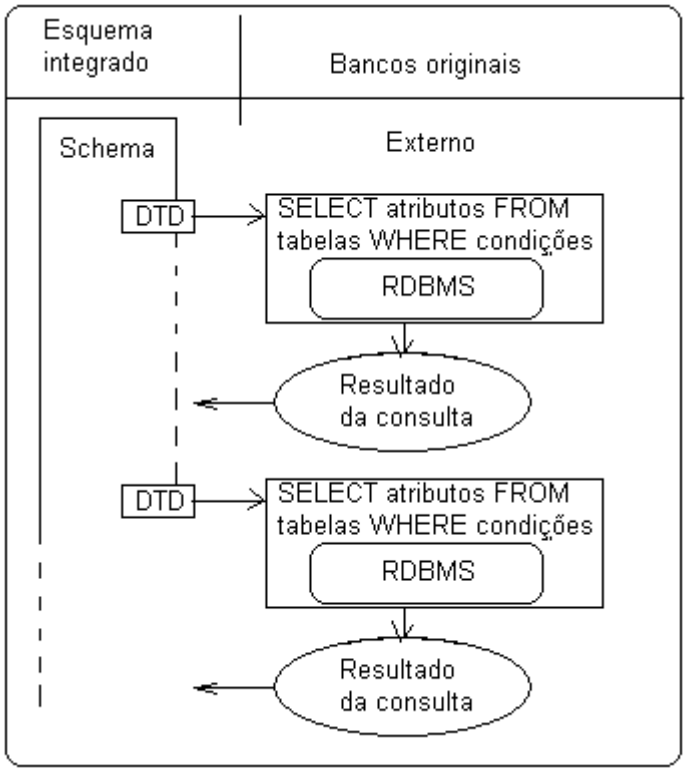

Figura 33 – Execução de consultas às bases originais pelo protótipo *X-VIQUEN*.

Dado que podemos modificar a forma de um documento *XML*, em princípio podemos efetuar as consultas em qualquer base de dados para a qual uma consulta possa ser realizada utilizando-se SQL. Os testes do protótipo, porém restringiram-se à base criada no protótipo do Sistema *VIQUEN*.

Os elementos formadores do documento podem ser pesquisados individualmente, se for o caso, e depois são reunidos de maneira a formar um elemento complexo. Isto permite a consulta de atributos representados como

complexos ou multivalorados no modelo *ERC+* em bases de dados que não permitem este tipo de representação, tais como as relacionais, mais difundidas atualmente. Pode-se esperar que com o correto mapeamento também seja possível consultar bases hierárquicas e objeto-relacionais e orientadas a objetos, porém estas possibilidades não foram exploradas neste protótipo.

Após efetuar as consultas aos bancos de dados orgininais, a próxima tarefa do *X-VIQUEN* é receber o conjunto de dados correspondente a cada uma das consultas efetuadas. Não importa a ordem em que a consulta foi formulada ou a seqüência de retorno, no caso de consultas múltiplas, pois os dados serão reorganizados de acordo com definições já existentes. Ao receber um conjunto de dados o protótipo, por meio dos arquivos de definição, "sabe" exatamente de que entidade eles fazem parte e coloca-os no local adequado seguindo as regras descritas nestes arquivos de definição de tipos de documentos, conforme exibido na

.

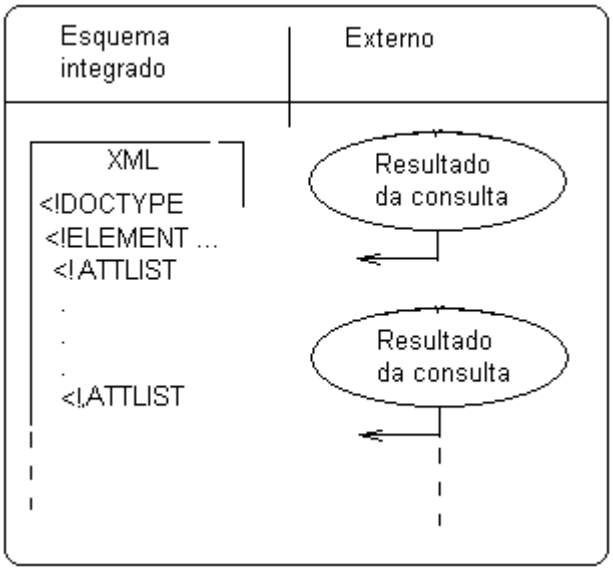

Figura 34 – Retorno dos conjuntos de dados e seu mapeamento.

Todo o mapeamento de atributos é realizado pelo protótipo *X-VIQUEN* seguindo as regras de definição existentes nos documentos respectivos em relação ao resultado de cada consulta. Por exemplo, digamos que uma regra defina que uma entidade *Pessoa* possua um atributo *Telefone*, de cardinalidade tal que permita desde sua não existência até sua existência multivalorada.

No primeiro caso, quando não for encontrado o elemento, teremos a representação que segue:

```
<PESSOA>
...
<TELEFONE/>
...
</PESSOA>
```

Nesta representação, seguindo a sintaxe *XML*, indicamos que Telefone é um atributo vazio (por meio da barra "/").

No segundo caso, supondo que foram encontrados três telefones para uma determinada pessoa, teríamos a seguinte representação:

```
<PESSOA>
...
<TELEFONE>
       111-1111
</TELEFONE>
<TELEFONE>
       222-2222
</TELEFONE>
<TELEFONE>
       333-3333
</TELEFONE>
...
</PESSOA>
```

Da mesma forma, uma definição de documento na qual *Pessoa* possui um atributo *Endereço*, formado pelos elementos *Rua* e *Número*, poderia ser perfeitamente representada pelo *X-VIQUEN*, usando por exemplo a representação a seguir:

```
<PESSOA>
...
<ENDEREÇO>
<RUA>
       Nome da rua
</RUA>
<NÚMERO>
       000001
</NÚMERO>
</ENDEREÇO>
...
</PESSOA>
```

Como se pode verificar, por meio do mapeamento para arquivos *XML* soluciona-se o problema da consulta e construção de atributos multivalorados e de

atributos complexos, sem a utilização de cursores, inexistentes na maioria das implementações disponíveis no mercado.

5.6 EXIBIÇÃO DOS RESULTADOS DA CONSULTA

Finalmente o *X-VIQUEN* invoca as *APIs*, modificadas, do Sistema *VIQUEN*, passando os parâmetros adequados, de maneira a possibilitar ao usuário a visualização dos resultados de sua consulta.

A Figura 35 apresenta uma visão das chamadas às *APIs* do Sistema *VIQUEN* de maneira a produzir uma representação gráfica.

Figura 35 - O *parser X-VIQUEN* chamando *APIs* do Sistema *VIQUEN*.

Após receber os parâmetros enviados pelo parser do módulo *X-VIQUEN* o Sistema *VIQUEN* irá construir uma representação gráfica por meio de chamadas às *APIs* do *ILOG VIEWS*, conforme exibido na Figura 36.

Figura 36 – Exibição dos resultados da consulta no Sistema *VIQUEN*.

Este modelo de representação foi idealizado para permitir a visualização de toda a estrutura da consulta gerada, permitindo também a navegação por seus componentes. Porém, outros modelos de representação de conteúdo poderiam ter sido utilizados, como por exemplo a representação para atributos complexos sugerida por ERWIG (2000), exibida na Figura 37.

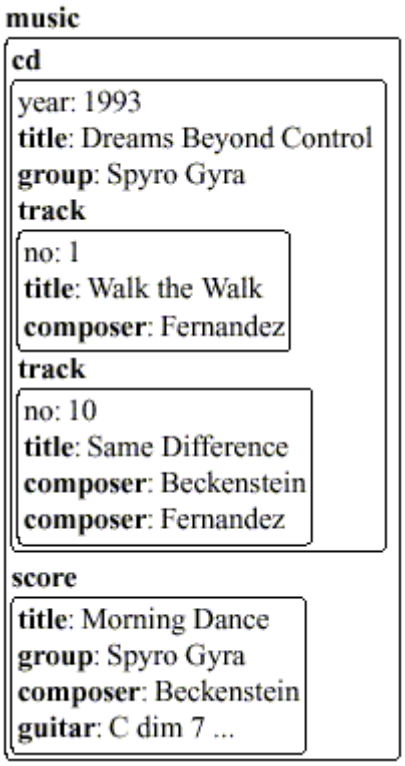

Figura 37 - Representação aninhada para atributos complexos sugerida por ERWIG(2000).

Esta representação, embora visualmente agradável e explicativa, poderia não ser prática para grandes conjuntos de resultados.

6 CONCLUSÕES

O grande poder da *XML* vem de sua flexibilidade, desde que qualquer um pode construir um documento que descreve a estrutura (*DTD*) de um outro documento (*XML*), na forma desejada (ERDMANN, 2001). Conforme TURAU (1999), os *DTDs* incluem suporte para descrever os tipos de dados suportados pela linguagem *SQL*, então eles serão utilizados para descrevermos o conteúdo da base original, ou seja, mapear seus componentes.

Utilizando-se o mapeamento dos elementos da base original em *DTDs* e mantendo-se informações de sua origem, pudemos alcançar os resultados pretendidos no início do trabalho, ou seja, o mapeamento de atributos complexos e multivalorados advindos de um processo de integração, no qual foi utilizado um modelo *ERC+* para consultas em bases relacionais, evitando-se a utilização de cursores.

O mapeamento do *X-VIQUEN* permite ainda a formulação de consultas e o mapeamento de atributos para bases não relacionais, por meio da formulação de novos documentos *XML*, possiblitando assim a consulta a bases heterogêneas.

Em relação ao Sistema *VIQUEN* a X-VIQUEN adiciona as seguintes vantagens:

- Mapeamento de atributos sem a utilização de cursores;
- Possiblidade de mapeamento / consulta a mais de uma base simultaneamente;
- Não ser restrita a um único fabricante de *RDBMS*.

Quanto à utlização da linguagem de marcação *XML* e suas variantes, o protótipo demonstrou sua grande versatilidade e poder de descrição de dados, que a habilitam para ser utilizada como ferramenta de apoio em processos de integração.

6.1 A ABORDAGEM *X-VIQUEN VERSUS* OS MODELOS PESQUISADOS

Em relação aos trabalhos pesquisados e comentados no segundo capítulo deste trabalho (Página 19), o *X-VIQUEN* oferece como vantagem o fato de usufruir do ambiente gráfico possibilitado pela ferramenta desenvolvida no protótipo do Sistema *VIQUEN* e o uso de *XML* para referenciar os bancos integrados.

O ambiente gráfico utilizado no protótipo, por ser montado a partir de uma representação do esquema baseada no Modelo *ERC+*, possui um grande poder semântico, possibilitando sua utilização por usuários leigos (Figura **Erro! Argumento de opção desconhecido.**, página 14). Em contraste, ABITEBOUL e outros (1997), com seu projeto *LORE* e sua linguagem *LOREL* utilizam interface em *HTML*. Neste projeto o usuário deve digitar os comandos, o que obriga-o a conhecer a linguagem (proprietária) bem como conteúdo a ser pesquisado. CERI e outros (1999) apresentam a *XML-GL*, uma linguagem gráfica para consultas, de excelente usabilidade (Figura 6, página 14), mas que exige a construção de um modelo *XML-GDM*, proprietário, para descrição dos dados. A desvantagem desta abordagem de CERI e seus colaboradores é que ela não representa toda a base para o usuário. Uma grande inspiração é que cada objeto guarda informações a respeito de sua base original, modalidade esta empregada no *X-VIQUEN*. Quanto à abordagem do Sistema *EquiX*, de COHEN e outros (1999), o projeto utiliza-se de navegação fácil e intuitiva (Figura 8, página 23), mas que exige conhecimento do conteúdo para efetuar a consulta, bem como torna a navegação trabalhosa.

O Sistema *X-VIQUEN* também possui como vantagem o tratamento de atributos complexos e multivalorados utilizando arquivos *XML*, e seu mapeamento para bancos de dados que seguem a álgebra relacional, tratamento este baseado em uma visão integrada construída a partir do Modelo *ERC+*.

6.2 TRABALHOS FUTUROS

Este trabalho, assim como o Sistema *VIQUEN*, utilizou-se do conjunto de componentes *ILOG VIEWS*©, da *ILOG*. Uma sugestão de continuidade do trabalho é sua implementação em uma linguagem não proprietária e multi-ambiente, tal como *C*, *C++* ou *JAVA*, de maneira a poder disponibilizá-lo para aqueles que têm interesse, e que eventualmente possua maior integração com *XML* (como *JAVA*).

Dado o grande poder de expressão da linguagem *XML*, pode-se presumir que a partir de estudos de mapeamento uma possiblidade de expansão é a construção de rotinas de acesso, na forma de *interfaces*, para diversos modelos de bases de dados (hierárquica, objeto-relacional,...) e mesmo para sistemas de arquivos, expandindo assim as fornteiras deste protótipo e as possiblidades de uso em casos de integração de bancos de dados.

Também podemos imaginar uma ferramenta de integração gráfica, que permita ao usuário ao mesmo tempo três visões e uma caixa de operadores: uma visão para o esquema do primeiro SGBD; uma visão para o segundo; uma caixa de ferramentas que contenha os operadores de integração; e uma terceira visão contendo o esquema integrado. Sendo uma continuidade deste trabalho, poderíamos cahmar tal ferramenta de X-VIQUEN++. Uma sugestão de tela está exibida na Figura 38.

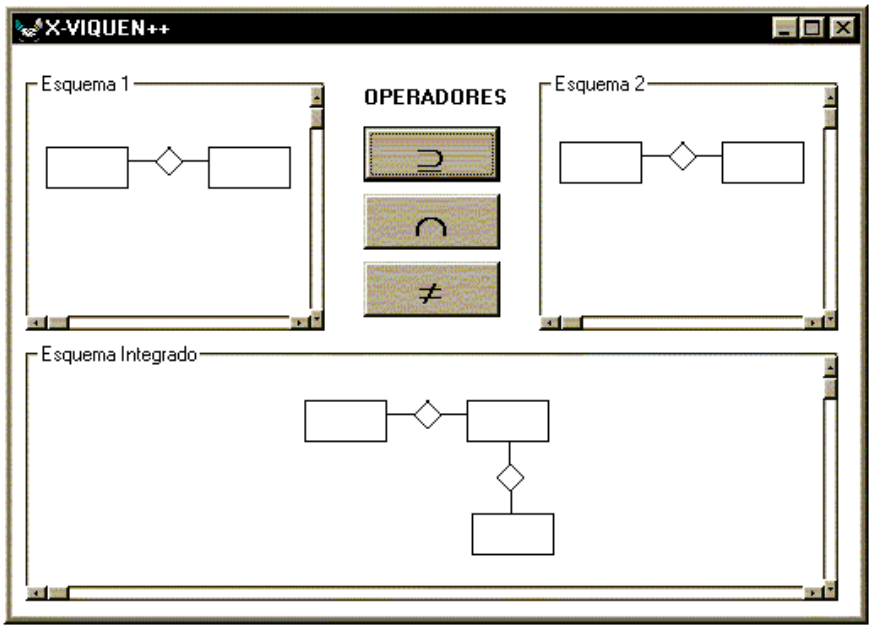

Figura 38 - Tela sugerida para o protótipo do sistema *X-VIQUEN++*.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ABITEBOUL, Serge, HULL, Richard, VIANU, Victor. **Foundations of databases**. Addison-Wesley, 1995.

ABITEBOUL, Serge, QUASS, Dallan, Mchugh, Jason, WIDOM, Jennifer, WIENER, Janet L. **The Lorel: Query Language for Semistructured Data**. In Journal of Digital Libraries, volume 1:1, 1997.

ANDERSON, Martin. **Extracting an entity relationship schema from a relational database through reverse engineering**. In Proceedings of the 13th. Intl. Conference on ER Approach, Pericles Loucopoulos, editor, 403-419, Manchester, UK, December 1994.

ANUTARYA, Chutiporn, WUWONGSE, Vilas, AKAMA, Kiyoshi, NANTAJEEWARAWAT, Ekawit. **A foundation for XML document database: DTD modeling**. Proceedings of 1 st International Conference on Electronic Commerce and Web Technologies (EC-Web 2000), London, UK.

BATINI, C., LENZERINI, M. **A comparative analysis of methodologies for database schema integration**. ACM Computing Surveys, Vol. 18, Nº 4, December 1986.

BISKUP, Joachim, EMBLEY, David W. **Extracting information from heterogeneous information sources using ontologically specified target views**. http://www.deg.byu.edu/papers, 2001. Acessado em 7 / setembro / 2001.

BONIFATI, Angela, CERI, Stefano. **Comparative analisys of five XML query languages**. ACM SIGMOD Record , 29(1), 2000.

BONIFATI, Angela, LEE, Dongwon. **Technical survey of XML schema and query languages**. ACM SIGMOD Record, 29(3), pp. 76-87, September 2000.

BOSAK, John. **XML, Java, and the future of the WEB**. Communications of ACM, 1997.

_____. **Media-independent publishing: four myths about XML**. IEEE Computer society. Volume 31, Number 10, October 1998. http://computer.org/computer/co1998/rx120abs.htm.

BROWN, Allen, FUCHS Matthew, ROBIE, Jonathan, WADLER, Philip. **MSL: a model for W3C XML Schema**. WWW10, May 1-5, 2001, Hong Kong.

BROWN, Allen, FUCHS, Matthew, ROBIE, Jonathan, WADLER, Philip. **MSL. A model for W3C XML schema**. WWW10, may 1-5, 2001. Hong Kong.

CERI, S., COMAI, S., DAMIANI, E., FRATERNALI P., PARABOSCHI, S., TANCA, L. **XML-GL: a graphical language for querying and restructuring XML documents**. In Proc. of the Int. World Wide Web Conference, Canada, 1999.

CHAWATHE, Sudarshan S. **Describing and manipulating XML data**. Bulletin of the IEEE Technical Committee on Data Engineering, Vol. 22(3):pp. 3--9, 1999.

CHEN, P. P. **The entity-relationship model: toward a unified view of data**. ACM Transactions on Database Systems 1:1 pp 9-36, 1976.

COHEN, Sara, KANZA, Yaron, KOGAN, Yakov, NUTT, Werner, SAGIV, Hiehoshua, SEREBRENICK, Alexander. **EquiX: easy querying in XML databases**. In Proceedings of the ACM SIGMOD Workshop on The Web and Databases (WebDB99), 1999.

CONNOLLY, Thomas, BEGG, Carolyn, STRACHAN, Anne. **Database systems**. A practical approach to design, implementation and manegement. Addison-Wesley, 1996.

ERDMANN, Michael, STUDER, Rudi. **How to structure and access XML documents with ontologies**. Data & Knowledge Engineering 36 (2001) 317-335.

ERWIG, Martin. **A Visual Language for XML**. In 16th IEEE Symp. on Visual Languages, pages 47--54, 2000.

_____. XML queries and transformations for end users. In XML 2000.

FERNANDEZ, Mary, SIMEÓN, Jérome, WADLER, Philip e outros. **XML query languages: experiences and exemplars.** http://www-db.research.bell-labs.com/user/simeon/xquery.ps. Acessado em 7 / Setembro / 2001.

GUEIBER, Ezequiel. **VIQUEN – Um ambiente interativo para consulta visual e extração de esquemas**. Porgrama de pós-graduaão em informática. Universidade Federal do Paraná. 2001. Dissertação de mestrado.

HAROLD, Rusty Elliotte. **XML bible.** IDG Books worldwide, 1999.

KLARLUND, Nils, MOLLER, Anders, SCHWARTZBACH, Michael **I. DSD: a schema language for XML.** FMSP 2000. Portland, Oregon.

KLAS, Wolfgang, FISCHER, Gisela, ABERE, Karl. **Integrating relational and object-oriented database systems using a metaclass concept.** Journal of Systems Integration, Vol.4 No.4, 1994, Kluwer Academic Publisher.

MAROKOWITZ, Victor M., SHOSHANI, Arie. **Representing extended entity-relationship structures in relational databases: a modular approach.** ACM transactions on database systems, vol. 17, nº 3, september 1992, pág. 423-164.

McBRIEN, Peter, POULOVASSILIS, Alexandra. **A formal framework for ER schema transformation.** In Proceedings of ER'97, volume 1331 of LNCS, pages 408--421, 1997.

MEGGINSON, David. **Structuring XML documents.** Prentice Hall PTR, 1998.

MILO, Tova, SUCIU, Dan, VIANU, Victor. **Typechecking for XML transformers.** In PODS 00, May 2000.

MURATA, Makoto. **Transformation of documents and schemas by patterns and contextual conditions**. Proceedings of the Third International Workshop on Principles of Document Processing (PODP 96), pages 153--169, Heidelberg, 1997.

PARENT, C., SPACCAPIETRA, S. **About entities, complex objects and object oriented data models**. Falkenberg and Lindgreen (eds): Information Systems Concepts: An In-depth Analysis. North Holland, 1989.

SMITH, Diane. C. P., SMITH, John Miles. **Database abstractions: aggregation and generalization.** ACM transactions on database systems, 2(2):105—133, June, 1977.

SPACCAPIETRA, S., PARENT, C., SUNYE, M., YETONGNON, K., LEVA, A. D. **ERC+: an object + relationship paradigm for database applications**. Readings in object-oriented systems, D. Rine (Ed.), IEEE Press, 1995.

TURAU, Volker. **Making legacy data acessible for XML applications**. FH Wiesbaden, University of Applied Sciences, Department of Computer Science. http://www.informatik.fh-wiesbaden.de/~turau, 1999 e http://citeseer.nj.nec.com/turau99making.htm. Acessado em 7 / setembro / 2001.

WADLER, Philip. **A formal semantics of patterns in XSLT**. Proceedings of Markup Technologies, Philadelphia, 1999.

WORLD WIDE WEB CONSORTIUM (W3C). **A schema language for XML**. http://www.w3c.org/XML/schema.HTML. Acessado em 7 / Setembro / 2001.

_____. **Extensible markup language**. http://www.w3c.org/XML. Acessado em 7 / Setembro / 2001.

ANEXOS

Arquivos *XML* correspondentes à base gerada no Sistema *VIQUEN* e utilizados como modelo para o *X-VIQUEN*, e suas representações *DTD*.

(Propriedade - *DTD* interno)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:23:28 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen [
    <!ELEMENT xmlViquen (propriedade0)>
    <!ATTLIST xmlViquen
    URL CDATA #REQUIRED
>
    <!ELEMENT propriedade0 (registroPropriedade0)*>
    <!ATTLIST propriedade0
    QUERY CDATA #REQUIRED
>
    <!ELEMENT registroPropriedade0 (registroPropriedade0.NrInscricao, registroPropriedade0.Nome,
registroPropriedade0.Local)>
    <!ELEMENT registroPropriedade0.NrInscricao (#PCDATA)>
    <!ATTLIST registroPropriedade0.NrInscricao
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroPropriedade0.Nome (#PCDATA)>
    <!ATTLIST registroPropriedade0.Nome
    TYPE CDATA #FIXED "varchar"
        NAME CDATA #FIXED "Nome"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        CASESENSITIVE (true | false) #FIXED "false"
        MAXLEN CDATA #FIXED "40"
>
    <!ELEMENT registroPropriedade0.Local (#PCDATA)>
    <!ATTLIST registroPropriedade0.Local
    TYPE CDATA #FIXED "varchar"
        NAME CDATA #FIXED "Local"
        NULLABLE (true | false) #FIXED "true"
        ISNULL (true | false) #IMPLIED
        CASESENSITIVE (true | false) #FIXED "false"
        MAXLEN CDATA #FIXED "40"
>
]>
<xmlViquen URL="jdbc:odbc:sqlxml">
    <propriedade0 QUERY="select * from propriedade">
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1029</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Serra do Mar]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Bom Retiro]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1230</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Tapajos]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Rodovia do Calcario]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1340</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Campo Alto]]></registroPropriedade0.Nome>
```

```
            <registroPropriedade0.Local><![CDATA[Pinhão]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1420</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Nascente do Sol]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Imbituva]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>2245</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Olho d Agua]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Ponta Grossa]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>2330</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Tibagi]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Ponta Grossa]]></registroPropriedade0.Local>
        </registroPropriedade0>
    </propriedade0>
</xmlViquen>
```

(Propriedade - *DTD* externo)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:19:13 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<xmlViquen URL="jdbc:odbc:sqlxml">
    <propriedade0 QUERY="select * from propriedade">
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1029</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Serra do Mar]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Bom Retiro]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1230</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Tapajos]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Rodovia do Calcario]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1340</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Campo Alto]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Pinhão]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>1420</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Nascente do Sol]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Imbituva]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>2245</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Olho d Agua]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Ponta Grossa]]></registroPropriedade0.Local>
        </registroPropriedade0>
        <registroPropriedade0>
            <registroPropriedade0.NrInscricao>2330</registroPropriedade0.NrInscricao>
            <registroPropriedade0.Nome><![CDATA[Fazenda Tibagi]]></registroPropriedade0.Nome>
            <registroPropriedade0.Local><![CDATA[Ponta Grossa]]></registroPropriedade0.Local>
        </registroPropriedade0>
    </propriedade0>
</xmlViquen>
```

(Propriedade - *DTD*)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
```

```
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:19:13 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen SYSTEM "adtd.dtd">
 <!ENTITY % types '(int|varchar)'>
 <!ENTITY % bool '(true|false)'>

 <!ELEMENT xmlViquen (propriedade0)>
  <!ATTLIST xmlViquen URL CDATA #REQUIRED>
 <!ELEMENT propriedade0 (registroPropriedade0)*>
  <!ATTLIST propriedade0
        QUERY CDATA #REQUIRED
  >
 <!ELEMENT registroPropriedade0 (registroPropriedade0.NrInscricao, registroPropriedade0.Nome,
registroPropriedade0.Local)>
 <!ELEMENT registroPropriedade0.NrInscricao (#PCDATA)>
  <!ATTLIST registroPropriedade0.NrInscricao
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroPropriedade0.Nome (#PCDATA)>
  <!ATTLIST registroPropriedade0.Nome
        TYPE %types; #FIXED "varchar"
        NAME CDATA #FIXED "Nome"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        CASESENSITIVE %bool; #FIXED "false"
        MAXLEN CDATA #FIXED "40"
 >
 <!ELEMENT registroPropriedade0.Local (#PCDATA)>
  <!ATTLIST registroPropriedade0.Local
        TYPE %types; #FIXED "varchar"
        NAME CDATA #FIXED "Local"
        NULLABLE %bool; #FIXED "true"
        ISNULL CDATA #IMPLIED
        CASESENSITIVE %bool; #FIXED "false"
        MAXLEN CDATA #FIXED "40"
 >
```

(Gleba - *DTD* Interno)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:24:55 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen [
    <!ELEMENT xmlViquen (gleba0)>
    <!ATTLIST xmlViquen
    URL CDATA #REQUIRED
>
    <!ELEMENT gleba0 (registroGleba0)*>
    <!ATTLIST gleba0
    QUERY CDATA #REQUIRED
>
    <!ELEMENT registroGleba0 (registroGleba0.NrInscricao, registroGleba0.Numero, registroGleba0.Area)>
    <!ELEMENT registroGleba0.NrInscricao (#PCDATA)>
    <!ATTLIST registroGleba0.NrInscricao
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
```

```
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroGleba0.Numero (#PCDATA)>
    <!ATTLIST registroGleba0.Numero
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "Numero"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroGleba0.Area (#PCDATA)>
    <!ATTLIST registroGleba0.Area
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "Area"
        NULLABLE (true | false) #FIXED "true"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
]>
<xmlViquen URL="jdbc:odbc:sqlxml">
    <gleba0 QUERY="select * from gleba">
        <registroGleba0>
            <registroGleba0.NrInscricao>1029</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>120000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1029</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>100000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>160000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>12</registroGleba0.Numero>
            <registroGleba0.Area>140000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1340</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>20000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1340</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>80000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1420</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>50000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1420</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2245</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
```

```
            <registroGleba0.Area>250000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2245</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2330</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2330</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>135000</registroGleba0.Area>
        </registroGleba0>
    </gleba0>
</xmlViquen>
```

(Gleba - *DTD* Externo)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:25:35 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<xmlViquen URL="jdbc:odbc:sqlxml">
    <gleba0 QUERY="select * from gleba">
        <registroGleba0>
            <registroGleba0.NrInscricao>1029</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>120000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1029</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>100000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>160000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1230</registroGleba0.NrInscricao>
            <registroGleba0.Numero>12</registroGleba0.Numero>
            <registroGleba0.Area>140000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1340</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>20000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1340</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>80000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1420</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
```

```
            <registroGleba0.Area>50000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>1420</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2245</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>250000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2245</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2330</registroGleba0.NrInscricao>
            <registroGleba0.Numero>10</registroGleba0.Numero>
            <registroGleba0.Area>150000</registroGleba0.Area>
        </registroGleba0>
        <registroGleba0>
            <registroGleba0.NrInscricao>2330</registroGleba0.NrInscricao>
            <registroGleba0.Numero>11</registroGleba0.Numero>
            <registroGleba0.Area>135000</registroGleba0.Area>
        </registroGleba0>
    </gleba0>
</xmlViquen>
```

(Gleba - *DTD*)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:25:35 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen SYSTEM "adtd.dtd">
 <!ENTITY % types '(int)'>
 <!ENTITY % bool '(true|false)'>

 <!ELEMENT xmlViquen (gleba0)>
  <!ATTLIST xmlViquen URL CDATA #REQUIRED>
 <!ELEMENT gleba0 (registroGleba0)*>
  <!ATTLIST gleba0
        QUERY CDATA #REQUIRED
  >
 <!ELEMENT registroGleba0 (registroGleba0.NrInscricao, registroGleba0.Numero, registroGleba0.Area)>
 <!ELEMENT registroGleba0.NrInscricao (#PCDATA)>
  <!ATTLIST registroGleba0.NrInscricao
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroGleba0.Numero (#PCDATA)>
  <!ATTLIST registroGleba0.Numero
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "Numero"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroGleba0.Area (#PCDATA)>
  <!ATTLIST registroGleba0.Area
        TYPE %types; #FIXED "int"
```

```
        NAME CDATA #FIXED "Area"
        NULLABLE %bool; #FIXED "true"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
```

(Lavoura - *DTD* Interno)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:29:11 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen [
    <!ELEMENT xmlViquen (lavoura0)>
    <!ATTLIST xmlViquen
    URL CDATA #REQUIRED
>
    <!ELEMENT lavoura0 (registroLavoura0)*>
    <!ATTLIST lavoura0
    QUERY CDATA #REQUIRED
>
    <!ELEMENT registroLavoura0 (registroLavoura0.NrInscricao, registroLavoura0.Numero, registroLavoura0.Ano,
registroLavoura0.Safra, registroLavoura0.IdCultura, registroLavoura0.QtdeProduzida)>
    <!ELEMENT registroLavoura0.NrInscricao (#PCDATA)>
    <!ATTLIST registroLavoura0.NrInscricao
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroLavoura0.Numero (#PCDATA)>
    <!ATTLIST registroLavoura0.Numero
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "Numero"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroLavoura0.Ano (#PCDATA)>
    <!ATTLIST registroLavoura0.Ano
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "Ano"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
    <!ELEMENT registroLavoura0.Safra (#PCDATA)>
    <!ATTLIST registroLavoura0.Safra
    TYPE CDATA #FIXED "varchar"
        NAME CDATA #FIXED "Safra"
        NULLABLE (true | false) #FIXED "false"
        ISNULL (true | false) #IMPLIED
        CASESENSITIVE (true | false) #FIXED "false"
        MAXLEN CDATA #FIXED "10"
>
    <!ELEMENT registroLavoura0.IdCultura (#PCDATA)>
    <!ATTLIST registroLavoura0.IdCultura
    TYPE CDATA #FIXED "smallint"
        NAME CDATA #FIXED "IdCultura"
        NULLABLE (true | false) #FIXED "true"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "5"
>
```

```
    <!ELEMENT registroLavoura0.QtdeProduzida (#PCDATA)>
    <!ATTLIST registroLavoura0.QtdeProduzida
    TYPE CDATA #FIXED "int"
        NAME CDATA #FIXED "QtdeProduzida"
        NULLABLE (true | false) #FIXED "true"
        ISNULL (true | false) #IMPLIED
        PRECISION CDATA #FIXED "10"
>
]>
<xmlViquen URL="jdbc:odbc:sqlxml">
    <lavoura0 QUERY="select * from lavoura">
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>50</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verao]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>200</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>180</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>225</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>270</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>250</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>12</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>100</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>1999</registroLavoura0.Ano>
```

```
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>100</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>110</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>130</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>50</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>230</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>110</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>160</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
```

```
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>90</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>1998</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>90</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>1998</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>170</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>60</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>190</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>1998</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>60</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>1998</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>30</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>180</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
    </lavoura0>
</xmlViquen>
```

(Lavoura - *DTD* Externo)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:27:31 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<xmlViquen URL="jdbc:odbc:sqlxml">
    <lavoura0 QUERY="select * from lavoura">
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>50</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verao]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>200</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1029</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>180</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>10</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>225</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>270</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>11</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>250</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
            <registroLavoura0.NrInscricao>1230</registroLavoura0.NrInscricao>
            <registroLavoura0.Numero>12</registroLavoura0.Numero>
            <registroLavoura0.Ano>2000</registroLavoura0.Ano>
            <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
            <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
            <registroLavoura0.QtdeProduzida>100</registroLavoura0.QtdeProduzida>
        </registroLavoura0>
        <registroLavoura0>
```

```
        <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>100</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>110</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1340</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>130</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>50</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>1420</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>230</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>10</registroLavoura0.Numero>
        <registroLavoura0.Ano>2000</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>110</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
    <registroLavoura0>
        <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
        <registroLavoura0.Numero>11</registroLavoura0.Numero>
        <registroLavoura0.Ano>1999</registroLavoura0.Ano>
        <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
        <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
        <registroLavoura0.QtdeProduzida>160</registroLavoura0.QtdeProduzida>
    </registroLavoura0>
```

```
<registroLavoura0>
    <registroLavoura0.NrInscricao>2245</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>11</registroLavoura0.Numero>
    <registroLavoura0.Ano>2000</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>90</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>10</registroLavoura0.Numero>
    <registroLavoura0.Ano>1998</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>90</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>10</registroLavoura0.Numero>
    <registroLavoura0.Ano>1998</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>170</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>10</registroLavoura0.Numero>
    <registroLavoura0.Ano>2000</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>60</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>10</registroLavoura0.Numero>
    <registroLavoura0.Ano>2000</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>190</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>11</registroLavoura0.Numero>
    <registroLavoura0.Ano>1998</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>2</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>60</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>11</registroLavoura0.Numero>
    <registroLavoura0.Ano>1998</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>120</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>11</registroLavoura0.Numero>
    <registroLavoura0.Ano>2000</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Inverno]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>3</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>30</registroLavoura0.QtdeProduzida>
</registroLavoura0>
<registroLavoura0>
    <registroLavoura0.NrInscricao>2330</registroLavoura0.NrInscricao>
    <registroLavoura0.Numero>11</registroLavoura0.Numero>
    <registroLavoura0.Ano>2000</registroLavoura0.Ano>
    <registroLavoura0.Safra><![CDATA[Verão]]></registroLavoura0.Safra>
    <registroLavoura0.IdCultura>1</registroLavoura0.IdCultura>
    <registroLavoura0.QtdeProduzida>180</registroLavoura0.QtdeProduzida>
```

```
        </registroLavoura0>
    </lavoura0>
</xmlViquen>
```

(Lavoura - *DTD*)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://sanfrancisco/xml/db2xml/test.xsl" type="text/xsl"?>
<!--
  Generated with DB2XML version 1.3
  http://www.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html
  Database: jdbc:odbc:sqlxml
  Date: Tue Aug 14 10:27:31 GMT-03:00 2001
  Driver: JDBC-ODBC Bridge (SQLSRV32.DLL) 1.2001 (2000.80.0100)
  Database system: Microsoft SQL Server  08.00.0100
-->
<!DOCTYPE xmlViquen SYSTEM "adtd.dtd">
 <!ENTITY % types '(int|varchar|smallint)'>
 <!ENTITY % bool '(true|false)'>

 <!ELEMENT xmlViquen (lavoura0)>
  <!ATTLIST xmlViquen URL CDATA #REQUIRED>
 <!ELEMENT lavoura0 (registroLavoura0)*>
  <!ATTLIST lavoura0
        QUERY CDATA #REQUIRED
  >
 <!ELEMENT registroLavoura0 (registroLavoura0.NrInscricao, registroLavoura0.Numero, registroLavoura0.Ano,
registroLavoura0.Safra, registroLavoura0.IdCultura, registroLavoura0.QtdeProduzida)>
 <!ELEMENT registroLavoura0.NrInscricao (#PCDATA)>
  <!ATTLIST registroLavoura0.NrInscricao
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "NrInscricao"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroLavoura0.Numero (#PCDATA)>
  <!ATTLIST registroLavoura0.Numero
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "Numero"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroLavoura0.Ano (#PCDATA)>
  <!ATTLIST registroLavoura0.Ano
        TYPE %types; #FIXED "int"
        NAME CDATA #FIXED "Ano"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
 >
 <!ELEMENT registroLavoura0.Safra (#PCDATA)>
  <!ATTLIST registroLavoura0.Safra
        TYPE %types; #FIXED "varchar"
        NAME CDATA #FIXED "Safra"
        NULLABLE %bool; #FIXED "false"
        ISNULL CDATA #IMPLIED
        CASESENSITIVE %bool; #FIXED "false"
        MAXLEN CDATA #FIXED "10"
 >
 <!ELEMENT registroLavoura0.IdCultura (#PCDATA)>
  <!ATTLIST registroLavoura0.IdCultura
        TYPE %types; #FIXED "smallint"
        NAME CDATA #FIXED "IdCultura"
        NULLABLE %bool; #FIXED "true"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "5"
 >
 <!ELEMENT registroLavoura0.QtdeProduzida (#PCDATA)>
  <!ATTLIST registroLavoura0.QtdeProduzida
        TYPE %types; #FIXED "int"
```

```
        NAME CDATA #FIXED "QtdeProduzida"
        NULLABLE %bool; #FIXED "true"
        ISNULL CDATA #IMPLIED
        PRECISION CDATA #FIXED "10"
>
```