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

XML Integrator: Interoperabilidade de Fontes de Dados Heterogêneas, baseada no Mapeamento de Esquemas Conceituais

por

EVERALDO LUIS DARONCO

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Profa. Dra. Cora Helena Francisconi Pinto Ribeiro
Orientadora

Porto Alegre, fevereiro de 2003.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Daronco, Everaldo Luis.
XML Integrator: Interoperabilidade de Fontes de Dados Heterogêneas baseada no Mapeamento de Esquemas Conceituais / por Everaldo Luis Daronco. Porto Alegre: PPGC da UFRGS, 2003.

95 f. : il.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação, Porto Alegre, 2003. Orientador: Ribeiro, Cora Helena Francisconi Pinto.

1. Banco de dados heterogêneos 2. Integração/Mapeameto de Esquemas. 3. Modelo de dados canônico. 4. XML. I. Ribeiro, Cora H.F. Pinto. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitor Adjunto de Pós-Graduação: Prof. Jaime Evaldo Fensterseifer
Diretor do Instituto de Informática: Prof. Philippe Oliver Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

A Rosemari de Ávila Daronco, minha esposa, pelo incentivo, força nas horas difíceis, pela compreensão na ausência em horas de lazer e pelo amor retribuído.

A minha Orientadora, professora Cora Helena Francisconi Pinto Ribeiro, pela dedicação, apoio, confiança e paciência em toda a trajetória deste mestrado.

Ao professor José Palazzo Moreira de Oliveira, pelo incentivo, idéias e apoio na elaboração desta dissertação.

A minha família, pai Isidro (in memorian), mãe Olga (in memorian) e irmãos Fátima, Gilberto, Luci, Lucila, Marli, Sirlei e Elisiane, pelo incentivo que mesmo de longe torcem por meu sucesso.

Aos colegas de Futebol do Instituto de Informática, nas horas de descontração e discussões futebolísticas.

Aos professores Simão Sirineu Toscani e Laira Toscani, pelos referenciais profissionais e por terem sido os responsáveis diretos pelo meu ingresso na carreira acadêmica.

Ao CNPQ, pelo incentivo financeiro na realização desta dissertação.

A todos aqueles que de alguma forma ajudaram para o desenvolvimento deste trabalho.

A Deus, por ter me dado a oportunidade de estudar em um dos institutos mais importante do país.

Sumário

Lista de Abreviaturas

ADO	ActiveX Data Objects
AMOS	Active Mediator Object System
ANSI	American National Standards Institute
ANSI/SPARC	American National Standards Institute/Standards Planning and Requirements Committee
API	Application Program Interface
BD	Banco de Dados
BDROO	Banco de Dados Relacional Orientado a Objetos
CAD	Computer-aided Design
CASE	Computer-aided Software Engineering
CLASSDOC	Classe de Documentos
DB	Database
DOC	Documento
DOM	Document Object Model
DSN	Data Source Name
DTD	Document Type Definition
EC	Esquema Componente
ECG	Esquema Conceitual Global
ECL	Esquema Conceitual Local
EE	Esquema de Exportação
EF	Esquema Federado
EL	Esquema Local
GUI	Graphical User Interface
HERMES	HEterogeneous Reasoning and MEdiator System
HKB	Hybrid Knowledge Bases
HTML	HyperText Markup Language
MC	Modelo de Dados Canônico
MS	Microsoft
MSL	Mediator Specification Language
ODBC	Open Database Connectivity
ODL	Object Description Language
ODMG	Object Database Management Group
OEM	Object Exchange Model
OID	Object Identificator
OLEDB	Object Linking and Embedding Database
OML	Object Manipulation Language
OO	Orientado a Objetos
OQL	Object Query Language
RC	Registro Central
REA	Relação de Equivalência de Atributos
REO	Relação de Equivalência de Objetos
ROL	Relação de Objetos Locais
SAX	Sample API for XML
SBDF	Sistema de Banco de Dados Federado
SBDH	Sistema de Banco de Dados Heterogêneos
SGBD	Sistema Gerenciador de Banco de Dados
SGBDF	Sistema Gerenciador de Banco de Dados Federados

SGBDOO	Sistema Gerenciador de Banco de Dados Orientado a Objetos
SGBDOR	Sistema Gerenciador de Banco de Dados Objeto-Relacional
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
TSIMMIS	The Stanford-IBM Manager of Multiple Information Sources
UdD	Universo de Discurso
URL	Uniform Resource Locator
WSL	Wrapper Specification Language
XML	eXtensible Markup Language
XML-QL	eXtensible Markup Language-Query Language
XQL	XML Query Language

Lista de Figuras

Lista de Tabelas

Resumo

O acesso integrado a informações provenientes de banco de dados autônomos e heterogêneos, localizadas em diferentes ambientes de *hardware* e *software*, vem sendo amplamente pesquisado pela comunidade de banco de dados, com diversas soluções propostas. A maioria delas baseia-se na comparação e na integração ou mapeamento dos esquemas conceituais dos bancos de dados participantes, implementados através de uma camada adicional de *software*, em um nível superior ao dos bancos de dados existentes.

Inicialmente, as metodologias de acesso integrado eram limitadas às informações provenientes de banco de dados. Entretanto, com o crescimento das redes de computadores e, conseqüentemente, com a intensa utilização da Internet, novas fontes de informações passaram a ser utilizadas neste ambiente, tais como fontes de dados semi-estruturadas. Estender o acesso integrado também a esses tipos de informações tornou-se importante.

Este trabalho tem como objetivo propor a utilização de um metamodelo XML como modelo de dados canônico, através do qual é possível obter a representação conceitual dos esquemas de exportação provenientes de bancos de dados relacionais, objeto-relacionais e documentos XML, permitindo, desta forma, o acesso integrado a fontes de dados estruturadas e semi-estruturadas, a partir de metodologias inicialmente voltadas à interoperabilidade de banco de dados heterogêneos. Além do metamodelo apresentado, este trabalho incluiu o desenvolvimento da ferramenta *XML Integrator*, cujo objetivo é fornecer ao usuário mecanismos de apoio ao processo conversão dos esquemas conceituais locais de fontes de dados heterogêneas para o Metamodelo XML, bem como de extração de um esquema conceitual correspondente a um documento XML ou a uma classe de documentos XML. Para isso, a ferramenta utiliza interfaces gráficas, que guiam o usuário através dos diversos passos, desde a seleção da fonte de dados a ser convertida, até a geração do esquema de exportação propriamente dito.

Palavras-chave: Banco de Dados Heterogêneos, Integração/Mapeamento de Esquemas, Modelo de Dados Canônico, XML

TITLE: "XML INTEGRATOR: HETEROGENEOUS DATA SOURCES INTEROPERABILITY BASED ON CONCEPTUAL SCHEMA MAPPING".

Abstract

The integrated access to the information from autonomous and heterogeneous database, placed on different hardware and software environments, has been widely researched by the database community, with several proposed solutions. Most of them are based on comparison, integration, and mapping of conceptual schemas of the participant databases, implemented through an additional layer of software. Methodologies for integrated access were initially limited to structured. However, with the development of computer networks and of the Internet, new sources of information become available in this environment, such as semi-structured data. Extending the integrated access to this type of information has become also important.

This work proposes the use of a XML meta model as the canonical data model for the conceptual representation of relational databases, relational-object, and XML documents, enabling the integrated access to structured and semi structured data sources, through the use of methodologies focused on heterogeneous database interoperability. Besides, this work also included the development of the XML Integrator tool, which supports users in the local conceptual schema conversion (for databases) or extraction (for XML documents or document classes) process.

Keywords: Heterogeneous databases, Schema Integration/Mapping, Canonical Data Model, XML

1 Introdução

O acesso integrado a fontes de dados heterogêneos vem sendo amplamente pesquisado pela comunidade de banco de dados. Muitas das diferentes abordagens e metodologias propostas para permitir o acesso integrado a informações provenientes de banco de dados autônomos e heterogêneos, tais como integração de esquemas globais [BAT 86], sistemas de bancos de dados federados [SHE 90], sistemas *multidatabase* [LIT 90] e sistemas baseados em mediadores [WEI 92], adotam a comparação, integração e/ou mapeamento dos esquemas conceituais dos bancos de dados participantes, após serem estes convertidos para um modelo de dados canônico. O acesso posterior às informações provenientes dos bancos de dados locais é realizado através de recursos fornecidos pelos próprios SGBDs. Com a popularização e disponibilização de informações através da Internet, a comunidade de bancos de dados passou a enfrentar o desafio de estender o acesso integrado também a fontes de dados semi-estruturadas [BAR 99, MAY 99]. Neste contexto, a *Extensible Markup Language* - XML [BRA 2000] vem sendo adotado como padrão de representação e troca de informações semi-estruturadas [ABI 97].

As soluções iniciais para o acesso integrado a informações provenientes de fontes de dados semi-estruturadas e de bancos de dados surgiram a partir de soluções voltadas para o acesso integrado a documentos XML. Em algumas [BAR 99, CRI 2000, VID 2002], a XML é utilizada como suporte ao acesso integrado, tanto a informações semi-estruturadas como àquelas provenientes de bancos de dados, através da conversão da instância acessada no banco de dados em um documento XML. Outras [MAN 2000, CAR 2000, VDO 2001], restringem o uso da XML à interface com o usuário, evitando seu uso no acesso aos dados.

O acesso integrado a fontes de dados semi-estruturadas a partir de metodologias de bancos de dados heterogêneos ainda se encontra em fase de investigação. A possibilidade de representação conceitual destas fontes de dados através do uso de um modelo de dados canônico único surge como uma alternativa para que metodologias de acesso integrado a bancos de dados heterogêneos sejam facilmente estendidas, de forma a também incluir fontes de dados semi-estruturadas, sem comprometer a eficiência do acesso às informações provenientes de bancos de dados. Contudo, os formatos de armazenamento destes arquivos são incompatíveis com os modelos de dados adotados em fontes de dados estruturadas, uma vez que suas estruturas são apenas parcialmente descritas e a representação destes dados pode ser irregular.

O presente trabalho tem como objetivo prover a base para o posterior acesso integrado a dados provenientes de fontes de dados estruturadas e semi-estruturadas, a partir da representação conceitual destas fontes de dados. Para isso, o metamodelo XML é proposto como modelo de dados canônico. Este trabalho também apresenta a ferramenta XML *Integrator*, que tem como objetivo auxiliar o usuário no processo de conversão dos esquemas conceituais locais para o metamodelo XML, bem como no processo de extração de um esquema conceitual de um documento ou classes de documentos XML. Esta solução permite que informações provenientes tanto de bancos de dados como de fontes de dados semi-estruturadas sejam acessadas de forma integrada, a partir de ferramentas para acesso integrado a bancos de dados heterogêneos, sem que os recursos disponíveis nos SGBDs sejam subutilizados. Ao mesmo tempo, possibilita a inclusão de informações semânticas adicionais, referentes também às fontes semi-estruturadas e bancos de dados legados,

possibilitando um melhor entendimento dos diferentes objetos do mundo real representados. A metodologia utilizada, como referência para o desenvolvimento deste trabalho, é baseada na abordagem de mapeamento de esquemas conceituais, proposta por Ribeiro [RIB 95].

O capítulo 2 aborda a representação conceitual de fontes de dados e caracteriza os principais modelos de dados utilizados atualmente em fontes de dados estruturadas e semi-estruturadas.

No capítulo 3, são apresentados os principais conceitos e características de Sistemas de Bancos de Dados Heterogêneos, bem como metodologias para o acesso integrado a informações provenientes de bancos de dados heterogêneos.

O capítulo 4 apresenta as principais soluções disponíveis para acesso integrado a fontes de dados heterogêneas, incluindo uma análise comparativa entre elas, no que diz respeito a aspectos pertinentes a este trabalho.

O capítulo 5 apresenta o metamodelo XML utilizado como modelo canônico para a representação conceitual de fontes de dados estruturadas e semi-estruturadas.

A ferramenta desenvolvida neste trabalho é apresentada no capítulo 6 e a validação da funcionalidade da ferramenta é demonstrada no capítulo 7, através de um estudo de caso.

As conclusões deste trabalho, bem como sugestões de trabalhos futuros, são abordados no capítulo 8.

2 Representação Conceitual de Fontes de Dados

A modelagem conceitual de informações consiste da representação e compreensão do Universo de Discurso - UdD através de uma visão simplificada e direcionada às propriedades relevantes da informação. O principal objetivo da representação conceitual é fornecer uma descrição abstrata de alto nível, independente de tecnologia e das estruturas de armazenamento da informação. Essa descrição é conhecida como o esquema conceitual de uma fonte de dados [ELM 99]. Os esquemas conceituais das fontes de dados são definidos por um modelo de dado, que descreve, de maneira formal, os tipos de informações que estão armazenadas nas fontes de dados [HEU 2000], os relacionamentos entre os dados, a semântica de dados e as regras de consistência [SIL 99].

Os principais benefícios da modelagem conceitual são: maior expressividade, pois há uma maior riqueza de conceitos que expressam um maior número possível de características do UdD; maior simplicidade, pois o esquema é de fácil entendimento; independência entre representação e solução; uso de um modelo formal, onde cada conceito tem uma interpretação única, precisa e bem definida; representação minimalista (ortogonalidade), onde cada característica da realidade tem uma única forma de representação.

As fontes de dados são classificadas em estruturados, semi-estruturados e não estruturados [BUS 99], dependendo das estruturas que as compõe. As fontes estruturadas utilizam um esquema pré-definido, ou seja, todos os dados são definidos através deste esquema. Uma característica deste tipo de fonte de dados é a que as estruturas que modelam os dados não mudam freqüentemente. Exemplos de fontes de dados estruturadas são sistemas gerenciadores de bancos de dados, sistemas de arquivos e sistemas legados.

Fontes de dados semi-estruturados não possuem uma estrutura pré-definida na forma de um esquema explícito [BUN 97]. Assim, cada unidade de dados tem sua própria definição semântica. O somatório das definições semânticas pode ser considerado como um esquema. Contudo, este esquema pode mudar potencialmente a cada vez que um dado é adicionado. Exemplos desse tipo de fonte de dados são páginas *Web* e arquivos XML.

As fontes de dados não-estruturados não apresentam nenhuma estrutura associada aos dados, não existindo um modelo de dados para a representação conceitual deste tipo de fontes de dados. A inclusão deste tipo de fonte de dados no acesso integrado, embora seja também contemplado em algumas metodologias [SUB 2002, HAS 99], está fora do escopo deste trabalho, pois não está baseado na modelagem conceitual da fonte de informação. Exemplos desse tipo de fonte de dados são documentos de texto, imagens e arquivos de música.

2.1 Modelos de Dados para Fontes de Dados Estruturadas

Inicialmente as organizações utilizavam os sistemas de processamento de arquivos para armazenar informações. Estes tipos de sistemas tinham problemas como: inconsistência e redundância de dados, dificuldade de acesso aos dados, isolamento, problemas de integridade, problemas de segurança, entre outros. Para solucionar estes problemas sugiram os sistemas gerenciadores de banco de dados. Os primeiros bancos de

dados utilizavam os modelos de dados hierárquicos e de rede e são comumente chamados de sistemas legados. Estes modelos de dados evoluíram para modelos relacional, orientado a objetos e objeto-relacional [ELM 99, SIL 99].

O modelo relacional [COD 82] é um dos modelos de dados mais utilizados em sistemas gerenciadores de bancos de dados - SGBDs comerciais. Um SGBD que utiliza o modelo de dados relacional é composto de tabelas ou relações. Uma relação *R,* definida sobre *n* conjuntos D_1, D_2, ..., D_n, não disjuntos e de valores atômicos, é um conjunto de *n*-tuplas ordenadas $<d_1, d_2, ..., d_n>$, de tal forma que $d_1 \in D1$, $d_2 \in D_2$, ... $d_n \in D_n$. Esta abordagem é caracterizada por três recursos importantes [COD 82]:

- as estruturas de dados são simples, sendo constituídas de relações formadas por tabelas bidimensionais, cujos elementos são itens de dados;

- oferece uma sólida base para consistência de dados, uma vez que o projeto do banco de dados é apoiado pelo processo de normalização, que elimina anomalias nos dados. Além disso, estados consistentes de um banco de dados podem ser definidos de maneira uniforme e mantidos por regras de integridade;

- permite a manipulação de relações orientadas a conjuntos, através de linguagens não procedurais baseadas na teoria dos conjuntos (álgebra relacional) ou em lógica (cálculo relacional).

Além disto, o modelo relacional possui mecanismos que permitem, entre outros, a definição de permissões de acessos aos dados, segurança, controle de redundâncias, controle de integridade, recuperação a falhas, controle de transações e *stored procedures* [ELM 99]. A linguagem padrão para consultas a bancos de dados relacionais é a *Structured Query Language* - SQL. A SQL é uma linguagem não procedural que possui um conjunto de declarações utilizadas por gerenciadores de banco de dados para acessar os dados. A SQL processa conjuntos de registros, provendo navegação automática através dos dados e permitindo ao usuário a manipulação de tipos complexos de dados. Também, foram incorporadas à SQL outras características e facilidades, tais como: especificação de integridade referencial, definição de domínios, manipulação e definição de esquemas, especificação de restrições de integridade sobre as tabelas, colunas e domínios e criação de visões utilizadas para controlar o acesso de usuários a conteúdos restritos das tabelas. Existem vários dialetos para a SQL, mas os dois principais são o *American National Standards Institute* - ANSI SQL e um padrão adotado em 1992, chamado de SQL-92 ou SQL2 [ELM 99]. Exemplos mais populares de Sistemas Gerenciadores de Banco de Dados Relacional - SGBDR são MS Access, MySQL e SyBase.

O modelo de dados orientado a objetos [CAT 94] surgiu na tentativa de solucionar problemas existentes no desenvolvimento de sistemas complexos, através de uma solução confiável, com baixo custo de desenvolvimento e manutenção, e que possibilitasse a reutilização de código. Esta abordagem propõe uma nova forma de especificar sistemas, usando modelos fundamentados em conceitos do mundo real e combinando estrutura e comportamento em um único esquema. O paradigma da orientação a objetos está baseado no encapsulamento de dados e do código relacionado a um objeto, dentro de uma única estrutura. A idéia principal do encapsulamento é separar, nos objetos, a especificação e a implementação. Um objeto pode ser definido como sendo uma combinação de dados e programas que representam alguma entidade do mundo real [KIM 93]. As interações entre os objetos e o resto do sistema são realizadas através de mensagens. Um objeto tem

associado a ele um conjunto de variáveis e um conjunto de mensagens ou métodos. Os objetos que possuem mesmos atributos e métodos podem ser agrupados, formando uma classe. Um objeto é então criado como uma instância de uma classe, recebendo um identificador interno (*oid*), que o identifica, de forma única, durante sua existência. Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos – SGBDOO utilizam a linguagem *Object Definition Language* - ODL para a definição de esquemas e *Object Manipulation Language* - OML para a manipulação de objetos. Estas linguagens foram definidas pelo comitê *Object Database Management Group* - ODMG, responsável pela padronização de extensões da linguagem C++ e SmallTalk, a fim de definir bibliotecas de classe para aceitar persistência. O padrão ODMG tenta estender o mínimo possível o C++, fornecendo muitas funcionalidades via bibliotecas de classe [SIL 99]. Sistemas gerenciadores de bancos de dados que utilizam o modelo orientado a objetos têm sido utilizados em aplicações complexas, como *Computer-aided design* – CAD (que armazenam informações sobre os projetos de engenharia), *Computer-aided Software Engineering* – CASE (que armazenam dados sobre desenvolvimento de *softwares*) e em bancos de dados multimídia (que incluem imagens, dados espaciais e áudio). Exemplos de SGBDOO são O2, IRIS, GEMSTONE e ORION [ELM 99]. Atualmente, sistemas de bancos de dados orientados a objetos não são ainda utilizados em aplicações comerciais.

O modelo de dados objeto-relacional propicia uma forte integração entre os conceitos relacionais e orientados a objetos. Bancos de dados que adotam este tipo de modelo são chamados de bancos de dados Relacional Orientado a Objetos – BDROO [STO 96]. Estes bancos de dados mantêm relações como formas de persistência para objetos, que são armazenados como tuplas ou valores de colunas. A nova versão da linguagem de consulta, chamada de SQL3, inclui novas características no modelo de dados objeto-relacional, as quais incluem tipos abstratos de dados (TAD), identificadores de objetos, subtipos e herança, objetos complexos, relacionamentos interobjetos, estruturas de controle em consultas e consultas recursivas. Exemplos de SGBDOR populares são Oracle, DB2 e PostgreSQL.

2.2 Modelo de Dados de Fontes de Dados Semi-Estruturados

Segundo [ABI 97, BUN 97], dados semi-estruturados não estão moldados a uma estrutura pré-definida, a qual está implícita nos próprios dados. Assim, cada unidade de dados tem sua própria definição semântica. O somatório das definições semânticas pode ser considerado como um esquema. O esquema pode ser modificado freqüentemente e descreve o estado atual dos dados. Contudo, violações no esquema são permitidas.

Dados semi-estruturados possuem uma estrutura irregular, implícita, parcial, indicativa e flexível. A estrutura irregular diz respeito aos diversos formatos que dados semanticamente similares podem possuir, uma vez que instâncias de dado semi-estruturado podem ser classificadas como sendo de um mesmo tipo e possuir diferentes estruturas, incluindo elementos adicionais ou alguns elementos incompletos. A estrutura implícita é contida nos dados e pode ser extraída pela análise do documento semi-estruturado (*parsing*), onde é possível isolar e estabelecer os relacionamentos entre os dados. A estrutura parcial é aplicada àqueles dados que não apresentam uma estrutura implícita. Uma estrutura indicativa é gerada a partir da análise de um documento e indica um possível tipo de dado, embora sem estabelecer tipos rigorosos para todas as instâncias. Uma estrutura flexível diz respeito às alterações dos tipos de dados para cada instância em um dado instante.

Outra característica encontrada em dados semi-estruturados é a capacidade de representação de objetos complexos, através de uma estrutura aninhada. Assim, um dado semi-estruturado pode conter dados simples, outros documentos, tabelas e outros componentes [BUN 97].

Os dados semi-estruturados não podem ser consultados por uma linguagem padrão como a SQL. Deve-se usar uma linguagem de consulta mais flexível, devido a sua estrutura incompleta. Desta forma, uma linguagem de consulta para dados semi-estruturados deve possuir algumas características que possibilitem uma melhor interação com os dados. Algumas características desejáveis são a navegação entre os dados de um documento e a possibilidade de se realizar consultas em um esquema representativo ou diretamente nos dados.

A utilização do modelo conceitual para a representação de dados semi-estruturados baseia-se na extração do esquema e na integração das informações em uma estrutura de dados mais complexa, para que se possa realizar, de forma mais eficiente, consultas nos dados. Neste contexto, surgiram várias técnicas de extração de dados, que utilizam a exportação das informações para modelos de dados de bancos de dados [LAB 2000, ROS 2000]. Estas técnicas apresentam alguns problemas, tais como manutenção dos dados extraídos, identificação de inconsistências e integração dos esquemas [GUA 2002]. Como pode ser observada, a modelagem conceitual adotada, deve ser adequada à representação de dados que possuem estrutura irregular, não sendo por isso compatível com os modelos de dado relacional, orientado a objeto e objeto-relacional utilizados em fontes de dados estruturadas.

A maioria dos trabalhos disponíveis na literatura, que adotam a representação conceitual de fontes de dados semi-estruturadas [BUN 95, BUN 96, BER 2000], utilizam o modelo de dados *Object Exchange Model* - OEM [ABI 97], voltado para a representação de instâncias de dados semi-estruturados. Desta forma, um esquema OEM descreve uma determinada instância e não um tipo ou classe de um documento [DOR 2000]. O objetivo inicial foi tornar-se um modelo de troca de informações entre sistemas heterogêneos. No modelo OEM, as informações são representadas em forma de um grafo interligado, onde cada nó representa um objeto. Cada objeto OEM é formado por quatro campos:

- o elemento *label*, usado para dar nome aos objetos e também para inferir semânticas usadas no processo de integração;

- o elemento *type*, associado ao tipo do objeto;

- o elemento *value*, que contém uma coleção de literais ou objetos aninhados. Desta forma, é criada uma estrutura de grafo. Porém, neste modelo de representação, os dados não possuem uma estrutura rígida, ou seja, os dados podem possuir atributos multivalorados e tipos diferentes;

- o elemento *object-id*, que identifica o objeto de forma única.

A linguagem de consultas correspondente, utilizada para a recuperação de informações semi-estruturadas, é a LOREL.

Outro modelo de dados, utilizado para representação conceitual de fontes de dados semi-estruturada é o *Ozone Data Model,* proposto por Lahiri *et al* [LAH 99]. Neste

modelo, dados estruturados são representados em um modelo de dados ODMG e os dados semi-estruturados, em um modelo OEM. A integração das informações e o acesso integrado aos dados provenientes de fontes de dados estruturadas e de semi-estruturadas são feitos através de um atributo identificador comum, que representa o mesmo objeto do mundo real.

Outros modelos de dados para a modelagem conceitual de dados semi-estruturados são descritos e analisados em [LAH 99]. Um exemplo é o sistema de gerenciamento de *sites* Strudel, que utiliza um modelo de dados similar ao OEM e uma linguagem de consulta chamada StruSQL.

Em relação aos modelos de dados utilizados para fontes estruturadas, estes modelos apresentam limitações em relação à representação dos esquemas conceituais, incluindo restrições de domínios e relacionamentos entre objetos [BUS 99].

3 Interoperabilidade de Banco de Dados Heterogêneos

Sistemas de bancos de dados heterogêneos são sistemas que permitem o acesso integrado a informações provenientes de fontes de dados heterogêneas e localizadas em diferentes ambientes de *hardware* e *software* [ELM 99].

Usualmente, as metodologias para acesso integrado a banco de dados heterogêneos, nas quais a heterogeneidade ocorre também entre os modelos de dados utilizados localmente, adotam a prática de converter os esquemas conceituais locais para um modelo de dados comum [BAT 86] ou canônico [SHE 90]. Este processo ocorre mediante a tradução dos esquemas conceituais locais para a representação equivalente no modelo de dados canônico. Com isso, a heterogeneidade de representação é eliminada, no que se refere aos modelos de dados locais, facilitando a detecção de relacionamentos semânticos entre os bancos de dados.

3.1 Modelo de Dados Canônico

De acordo com [BUS 99], os modelos de dados apresentam incompatibilidades de representação. Informações semânticas podem ser perdidas quando modelos de dados semanticamente ricos são convertidos para modelos de dados semanticamente pobres. Por exemplo, possivelmente haverá uma perda de informações semânticas quando se converte um esquema conceitual orientado a objetos para o modelo de dados relacional, uma vez que métodos não podem ser representados no modelo de dados relacional. A escolha adequada de um modelo de dados canônico evita este problema. Além disso, possibilita um processo de enriquecimento semântico [HOH 96], desde que o modelo canônico apresente mais recursos de representação do que o modelo local.

Segundo Saltor *et al* [SAL 91], um modelo de dados pode ser avaliado a partir de dois critérios: expressividade e relativismo semântico. Expressividade significa o grau pelo qual o modelo de dados pode representar um conceito do mundo real, incluindo aspectos referentes à estrutura e comportamento. Relativismo semântico é o grau pelo qual um modelo de dados pode representar diferentes conceitos de um mesmo objeto do mundo real. O modelo de dados canônico deve ter uma expressividade semântica igual ou superior à do modelo de dados dos bancos de dados componentes.

3.2 Metodologias de Acesso Integrado a Bancos de Dados Heterogêneos

As metodologias de acesso integrado a BDH são classificadas de acordo com a solução adotada para o acesso integrado às informações. As classificações propostas na literatura apresentam divergências. Muitos autores classificam tais metodologias de acordo com características que envolvem a preservação da autonomia local, a forma de distribuição física dos dados, a heterogeneidade, a forma de integração dos esquemas locais, a arquitetura utilizada e os tipos de componentes de dados suportados. Também existem várias definições e arquiteturas para um mesmo tipo de solução adotada pelas metodologias de acesso integrado. Neste trabalho, a classificação utilizada é a adotada por Elmagarmid [ELM 99], que classifica as soluções existentes para o acesso integrado a banco de dados em integração de esquemas conceituais [BAT 86], sistemas de bancos de

dados federados – SBDF [SHE 90], sistemas *Multidatabase* [LIT 90] e sistemas baseados em mediadores [WEI 92].

3.2.1 Integração de Esquemas Conceituais

Nesta abordagem, descrita inicialmente por Batini [BAT 86], o acesso integrado às informações provenientes de SBDH baseia-se na integração prévia dos esquemas conceituais das fontes de dados, proporcionando ao usuário uma visão única e uniforme das informações acessadas. Os esquemas conceituais das fontes de dados locais são integrados dentro de um único esquema conceitual, chamado de esquema conceitual global – ECG, onde heterogeneidades resultantes de diferenças semânticas entre os esquemas conceituais locais são eliminadas. O ECG resultante do processo de integração é representado usando um modelo de dados canônico – MC, que possui a função de eliminar as diferenças dos modelos de dados adotados pelos bancos de dados locais.

Segundo [ELM 99], uma vantagem desta metodologia é que os usuários têm a visão consistente e uniforme dos dados acessados, não tendo conhecimento de que os dados acessados são heterogêneos e podem estar distribuídos fisicamente em vários locais. Porém, o processo de integração usualmente requer alterações nos esquemas locais, o que está associado à necessidade de revisão das informações dos esquemas conceituais locais dos bancos de dados envolvidos, e pode envolver negociações entre os esquemas conceituais locais, até que o esquema conceitual global, livre de conflitos, possa ser obtido. Além disso, esta metodologia assume que a parcela do UoD, considerada para fins de acesso integrado, é basicamente restrita a um mesmo ambiente do mundo real, sem possibilitar o acesso às informações adicionais, restritas a um banco de dados específico e às aplicações locais. A possibilidade de acesso integrado, por exemplo a dados contábeis e aos dados clínicos referentes a um mesmo objeto do mundo real, não é levada em consideração.

O sistema AURORA [YAN 97] é um exemplo de ferramenta que adota a metodologia de integração de esquemas, suportando sistemas com modelos de dados semelhantes. Por exemplo, dois ou mais sistemas relacionais são integrados a um esquema conceitual global [OZS 99].

3.2.2 Sistemas de Banco de Dados Federados

Os Sistemas de Bancos de Dados Federados [SHE 90] são compostos por vários bancos de dados autônomos, que compartilham a parcela disponibilizada de dados locais com os demais membros da federação, sem com isso comprometer a autonomia local em relação aos seus próprios dados e ao controle de acesso a estes. Este tipo de sistema permite que os bancos de dados tenham maior controle sobre as informações compartilhadas, isto é, maior autonomia de associação entre os bancos de dados independentes em um ambiente cooperativo. Cada banco de dados participante pode operar localmente, limitado aos dados locais, ou globalmente, com a possibilidade de acesso estendida aos dados disponíveis nos demais bancos de dados da federação. Desta forma, o acesso a parcelas disponíveis de dados compartilhados torna-se possível, sem que isso interfira nas aplicações locais, as quais mantêm sua autonomia.

Os sistemas de bancos de dados federados estendem a arquitetura ANSI/SPARC[1], usualmente utilizadas pelos sistema de bancos de dados centralizados, uma vez que esta não é adequada para descrever a arquitetura de sistemas de bancos de dados heterogêneos [SHE 90]. A FIGURA 3.1 apresenta a arquitetura de referência de um SBDF, descrita em cinco camadas.

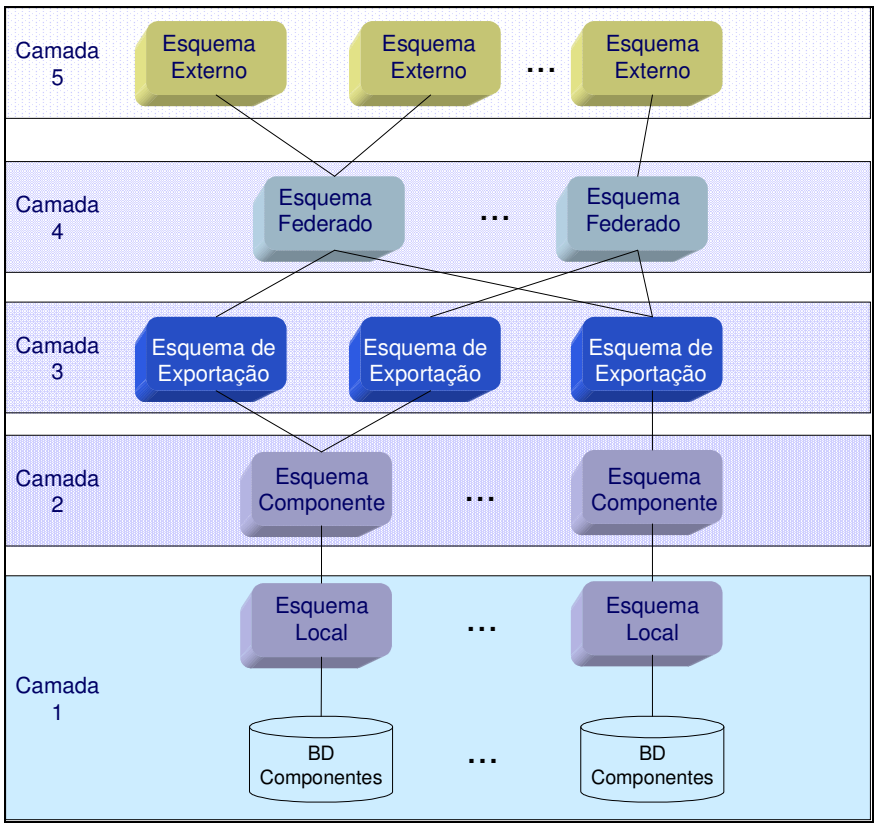


FIGURA 3.1 - Arquitetura de referência de SBDF

A primeira camada refere-se aos esquemas locais, que correspondem ao esquema conceitual local dos bancos de dados componentes, expresso no modelo de dados adotado pelos bancos de dados locais.

A segunda camada define os esquemas componentes, que são derivados do esquema conceitual local, através da tradução destes para um modelo de dados canônico. Os esquemas componentes possuem a finalidade de auxiliar o processo de integração, negociação e eliminação da heterogeneidade dos modelos de dados.

A terceira camada diz respeito aos esquemas de exportação, os quais representam o subconjunto dos esquemas componentes disponíveis à federação. Cada componente participa da federação exportando partes de seu esquema componente via esquema de exportação. O esquema de exportação possibilita maior segurança das informações locais, através da exclusão daquelas informações que não serão compartilhadas com a federação, além de facilitar o controle e o gerenciamento da autonomia local dos componentes.

---

[1] ANSI/SPARC (*American National Standards Institute/Standards Planning and Requirements Committee*) - grupo de estudos que propôs a arquitetura e padrões para o desenvolvimento de sistemas gerenciadores de bancos de dados [OZS 99].

A quarta camada refere-se ao esquema federado, que compreende a integração dos múltiplos esquemas de exportação e representado, também, através do modelo de dados canônico. Sua principal tarefa é auxiliar o tratamento da heterogeneidade lógica (semântica) entre esquemas componentes [KAS 95].

As visões particulares de cada usuário ou aplicação global são definidas pelo esquema externo, o qual constitui a quinta camada da arquitetura de referência.

Um sexto componente, identificado por Elmagarmid *et al* [ELM 99], é o dicionário de dados. Este componente armazena informações sobre:

- os esquemas externos, federados e de exportações;
- os mapeamentos entre os esquemas (esquema federado e externo, esquema federado e seus esquemas de exportação);
- estatísticas e heurísticas para otimizações de consulta;
- informações adicionais sobre esquemas, como: funções para transformação de unidades, formatos, endereços de redes, facilidades de comunicação.

Em sistemas de bancos de dados federados a integração dos esquemas conceituais não é obtida através de um esquema conceitual global, e sim pelas funcionalidades da linguagem de consulta global, que são extensões a uma linguagem de consulta, que manipulam múltiplas fontes de dados e executam transformações necessárias entre elas.

Os sistemas de bancos de dados federados podem ser classificados, dependendo do grau de autonomia e do tipo de integração existente entre os sistemas componentes da federação, em fortemente acoplados e fracamente acoplados.

3.2.2.1 Sistemas de Banco de Dados Fortemente Acoplados

Nos SBDF fortemente acoplados, existe uma autoridade central responsável pela administração da federação, que é o sistema gerenciador de banco de dados federado – SGBDF. O SGBDF possui o controle relativo aos participantes da federação, bem como às informações disponíveis e à localização destas informações. Assim, as consultas e atualizações podem ser realizadas sem que os membros da federação tenham conhecimento dos caminhos de acesso ou da localização física dos dados. Estes tipos de sistemas são chamados de estáticos, pois a criação do esquema federado está interligada à integração de esquemas e raramente são modificados após sua criação. Um exemplo destes sistemas é o HEROS (*HEteRogeneous Object System*) [UCH 99], onde é criada uma federação com acoplamento forte desenvolvimento no Departamento de Informática da PUC-Rio.

Nesta solução, a criação e a manutenção de esquemas federados são tarefas complexas de serem executadas por projetistas e, quando ocorrem mudanças nos esquemas conceituais ou nos esquema de exportação, o processo de integração para cada esquema federado deve ser refeito.

3.2.2.2 Sistemas de Banco de Dados Federados Fracamente Acoplados

Sistemas de Banco de Dados Federados fracamente acoplados são aqueles onde a responsabilidade de criar e manter a federação fica a cargo do usuário, não existindo um controle central da federação. O objetivo destes sistemas é facilitar o compartilhamento dos dados de uma forma dinâmica e flexível. Os membros da federação possuem total liberdade de integrar-se ou retirar-se da federação. Isto é feito, inicialmente, através da disponibilização do esquema de exportação para a federação e através da escolha, dentre os diversos esquemas de exportação disponíveis, daqueles que descrevem os dados que se deseja acessar. O esquema federado é criado a partir de uma interface de usuário, um programa de aplicação ou uma linguagem de consulta global, que possibilitam a importação dos objetos dos esquemas de exportação participantes. Estes sistemas são chamados de dinâmicos, pois os esquemas federados podem ser facilmente criados e removidos, não havendo nenhum sistema ou processo que gerencie ou controle a criação de um esquema federado. Os sistemas fracamente acoplados surgiram para minimizar os problemas associados à criação de um esquema conceitual global. Nestes sistemas, usuários são expostos à idéia de que existem diversas fontes de dados, quebrando, assim, o princípio da transparência de localização. Um exemplo destes tipos de sistema foi desenvolvido por [SOA 99] onde é criada uma federação com acoplamento fraco para bancos de dados, banco de dados geográficos e sistemas legados.

Alguns dos maiores problemas desses sistemas são a complexidade da interface e a necessidade da participação efetiva do usuário na definição ou processo de geração do esquema federado.

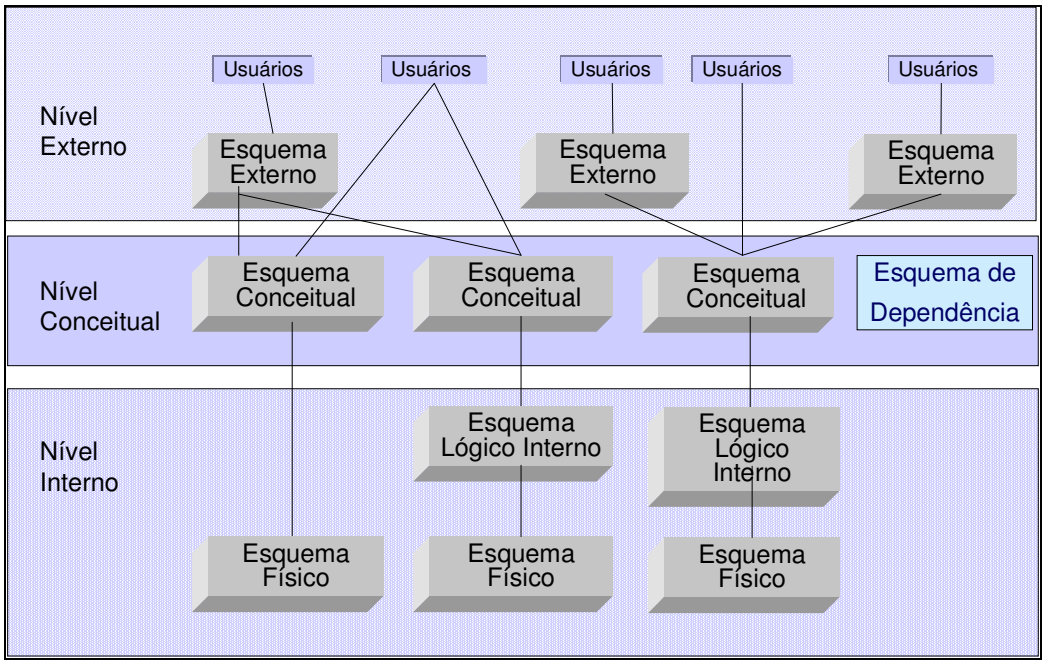


FIGURA 3.2 - Arquitetura de Sistemas *Multidatabase*

3.2.3 Sistemas de Múltiplos Bancos de Dados (*Multidatabase*)

A arquitetura de *multidatabase,* proposta por Litwin [LIT 90], está fortemente baseada na autonomia local dos bancos de dados componentes. Segundo Litwin, os usuários desejam autonomia na definição dos dados e independência na definição de nomes, na reestruturação de dados, na duplicação de dados e na definição de tipos de

valores. Essa autonomia geralmente não é respeitada quando os usuários são forçados a compartilhar informações, utilizando abordagens baseadas em um esquema conceitual global.

A arquitetura de sistemas *multidatabase,* apresentada na FIGURA 3.2, adota a arquitetura de três níveis de esquemas, possuindo os níveis interno, conceitual *multidatabase* e o externo. O nível interno contém os bancos de dados existentes, cada um com seu esquema físico. O nível conceitual *multidatabase* contém os esquemas conceituais dos bancos de dados que desejam compartilhar informações. Esse esquema pode ser tanto o esquema conceitual real do banco de dados, quanto o esquema lógico externo. O esquema conceitual *multidatabase* pode suportar diferentes modelos de dados e também encapsular os dados privados. Os esquemas de dependência também fazem parte deste nível e incluem a definição de dependências entre os grupos de bancos de dados, como equivalências entre atributos e domínios, permitindo ao administrador da base de dados especificar consistências entre os bancos de dados, sem a necessidade de um esquema global. O nível externo contém a construção do esquema externo, que pode corresponder a um ou vários esquemas *multidatadase.* É através dele que os usuários interagem com o sistema.

A idéia principal de sistemas *multidatabase* é o uso de uma linguagem *multidatabase,* que deve fornecer todas as funções de uma linguagem de bancos de dados, permitir que usuários definam um *multidatabase*, além de dar suporte à interoperabilidade entre os componentes.

Uma característica marcante desta abordagem é a preservação da autonomia local, em que os modelos conceituais locais não necessitam ser alterados para a resolução dos diversos tipos de heterogeneidade. Não existe um esquema conceitual global e sim o mapeamento das diversas informações dos distintos bancos de dados componentes. A redundância de dados nos diferentes bancos de dados acessados não causa grande transtorno, como também não é problema a discrepância de nomes, estruturas de dados e tipos de valores [LIT 90].

A arquitetura *multidatabase* não contém mecanismos de negociação para coordenar as informações compartilhadas. Isto permite que usuários e/ou aplicações acessem diretamente dados de diferentes bancos de dados ou através de visões externas (esquema externo). A construção de esquema externo determina a resolução de conflitos semânticos e sintáticos.

Contudo, a implementação deste tipo de solução é complexa. Uma vez que requer a implementação de uma linguagem específica para cada banco de dados participante, incluindo os mapeamentos específicos. Esta abordagem é adotada em sistemas como Pegasus [AHM 91], Carnot [WOE 93] e OMINIBASE [RUS 88].

3.2.4 Sistemas baseados em Mediadores

Os sistemas baseados em mediadores [WEI 92] são responsáveis pela interoperabilidade entre dois ou mais SGBDs e pela interface da aplicação com o usuário final. O acesso aos dados heterogêneos é efetuado através de consultas, que são submetidas aos mediadores que, por sua vez, as transformam em subconsultas a serem enviadas aos SGBDs componentes. As subconsultas geradas pelos mediadores devem ser traduzidas para as linguagens de consulta de cada SGBD componente [VID 97]. Alguns mediadores

podem usar *wrappers*[2] como ferramentas para resolver uma parte específica da conversão de dados. Estes componentes tem a responsabilidade de acessar e converter as informações da fonte de dados. Neste tipo de sistema não existe um sistema ou uma visão global que possa servir para as aplicações. Estes sistemas tendem a ser mais flexíveis, uma vez que os *wrappers* são construídos de forma independente e são utilizados para acessar um único tipo de fonte de dados.

Diferentemente de outras soluções, os sistemas baseados em mediadores podem incluir não somente fontes estruturadas, mas também fontes semi-estruturadas, como arquivos HTML e XML, e fontes não estruturadas, como documentos textos, imagens e vídeo.

Contudo, esses tipos de sistemas não são facilmente implementados e requerem rotinas de acesso independentes para cada fontes de dados existentes e usualmente, requerem o acesso prévio às instâncias das fontes de dados envolvidas para que as informações sejam tratadas e processadas para posteriormente serem entregue aos usuários. Entre as metodologias que adotam o uso de mediadores estão HERMES [SUB 2002] e GARLIC [CAR 95].

3.2.5 Considerações Finais

Vários problemas têm sido encontrados no projeto e implementação de sistemas de integração de informações heterogêneas que fornecem aos usuários uma visão única e uniforme dos dados armazenados em múltiplas fontes de dados. Primeiro, as fontes de dados podem conter modelos e linguagem de dados diferentes, podem conter objetos equivalentes, conflitos de representação e dados complementares, requerendo com isso uma integração das informações obtidas antes de serem apresentadas para o usuário final. As metodologias propostas para a integração de esquemas assumem que a parcela do UD, considerada para fins de acesso integrado é, basicamente, restrita a um mesmo ambiente do mundo real, sem possibilitar o acesso às informações adicionais, relativas a um banco de dados específico e às aplicações locais e não compartilhadas. A possibilidade de acesso integrado, por exemplo, aos dados acadêmicos e aos dados clínicos referentes a um mesmo objeto real, não é levada em consideração. O processo de busca de uma representação comum, iniciado a partir da identificação dos conflitos existentes, tradicionalmente envolve negociação entre os esquemas locais, até que o esquema global, livre de conflitos, possa ser obtido.

Uma abordagem de mediador-tradutor divide a funcionalidade de integração de dados em duas etapas. Os tradutores fornecem acesso aos dados das fontes locais utilizando um modelo de dados canônico e uma linguagem de consulta comum; o mediador fornece uma visão uniforme dos dados em um repositório e apresenta o mesmo problema da integração no que diz respeito à visão e origem da informação.

As diferentes soluções, com exceção de *Multidatabase* adotam um modelo de dados canônico, utilizado para a representação conceitual dos bancos de dados heterogêneos envolvidos no processo. Este tipo de solução é a melhor opção entre as metodologias apresentadas, pois não utiliza um esquema conceitual global, que é substituído pelos

[2] *Wrappers* são módulos de *software* que compõem um mediador e possuem a funcionalidade de resolver problemas relativos a diferenças de expressões de consulta de cada fonte de dados.

mapeamentos entre os esquemas conceituais. Contudo, sua implementação é complexa, pois para membro que é adicionado, os mapeamentos devem ser refeitos.

Dentre as soluções somente os sistemas baseados em mediadores incluem suporte ao acesso integrado a fontes de dados que não sejam estruturadas.

Uma solução que combina algumas características da integração e *multidatabase* é o mapeamento de esquemas conceituais que será apresentada no Capítulo 5 e é adotada neste trabalho. Nesta metodologia, a utilização de atributos comuns às diferentes representações locais da entidade, que possam identificar o objeto de forma única, permanece como uma boa alternativa para o estabelecimento automático de equivalências entre objetos no nível de instância. Na ausência de tais atributos, os critérios a serem utilizados na identificação devem ser definidos pelo projetista.

4 Soluções para o Acesso Integrado a Fontes de Dados Heterogêneas

Diversas soluções para o acesso integrado a bancos de dados heterogêneas [PIR 97, HAS 99, SOA 99, UCH 99] foram propostas. A maioria delas baseia-se na comparação e na integração ou mapeamento dos esquemas conceituais dos bancos de dados participantes. Entretanto, com o crescimento das redes de computadores e, conseqüentemente, com a intensa utilização da Internet, novas fontes de informações passaram a ser utilizadas neste ambiente, tais como fontes de dados semi-estruturadas, surgindo a necessidade de estender o acesso integrado também a estes tipos de fontes de informação.

As soluções iniciais para acesso integrado a dados provenientes de bancos de dados e fontes de dados semi-estruturadas [BAR 99, CRI 2000, VID 2002] partiram de metodologias originalmente desenvolvidas apenas para o acesso integrado de fontes de dados semi-estruturadas, baseadas no uso da XML. Nestas soluções, a XML é utilizada como suporte para o acesso integrado às informações, a partir da conversão de cada instância do banco de dados em um documento XML, para que seja realizada a integração. Contudo, mesmo com a XML apresentando alguns mecanismos presentes em SGBDs, como armazenamento (documento XML), esquemas (DTDs, XML *Schema*), linguagens de consulta (XQL, XML-QL) e interfaces de programação (SAX, DOM) [BOU 2001], sua utilização como forma de acesso a informações provenientes de bancos de dados apresentam limitações de performance. Estas limitações são resultantes da necessidade de conversão da instância acessada em um documento XML, como pré-requisito para o acesso à informação, uma vez que os dados são materializados e integrados para posterior acesso e o resultado é mostrado ao usuário como visão XML. Esta alternativa não faz uso dos recursos de acesso aos dados disponibilizados pelos SGBDs, degradando, assim, o tempo de resposta às consultas.

Outra alternativa apresentada por algumas abordagens, como Agora [MAN 2000], XPERANTO [CAR 2000] e Hera [VDO 2001], consiste em limitar o uso da XML à interface com o usuário, evitando, desta forma, o uso da XML no acesso aos dados. Na primeira abordagem, o acesso integrado as informações das fontes de dados é realizado através da materialização das informações em um banco de dados objeto-relacional centralizado, onde o usuário possui interfaces de consulta, e os resultados são visualizados como documentos XML. Na segunda, o acesso integrado é realizado através de um mediador, que está baseado em um esquema conceitual global, onde as informações provenientes das fontes de dados são convertidas e mostradas ao usuário como uma visão XML.

Nos últimos anos, vários sistemas e protótipos têm sido desenvolvidos para prover acesso integrado a fontes de dados heterogêneos. Alguns desses sistemas dão suporte a fontes de dados estruturadas e semi-estruturadas [MOL 97, LAH 99, RIS 2000]; outros, suportam somente fontes de dados estruturadas [SUB 2002]. A seguir, serão apresentadas algumas características de soluções representativas, com enfoque nos aspectos relevantes a este trabalho, tais como nível de interoperabilidade, metodologia de integração, modelo de dados canônico, utilização de esquema global, tipos de fontes de dados suportadas, parcela de visualização, preservação da autonomia local e forma de visualização das informações pelo usuário.

O *HEterogeneous Reasoning and MEdiator System* - HERMES [SUB 2002] é baseado na teoria *Hybrid Knowledge Bases* – HKB, a qual fornece uma linguagem declarativa para a definição de mediadores, os quais expressam o nível de integração semântica e oferecem transparência no acesso aos dados das diversas fontes de dados. Esta linguagem pode extrair informações de diferentes fontes de dados estruturadas (bancos de dados DBASE, INGRES e PARADOX). O acesso às informações é realizado através de operações específicas para cada fonte de dados acessada. Entretanto, para que o acesso integrado às informações seja realizado, é necessário informar o domínio, isto é, qual o banco de dados, objeto e critérios a serem consultados. O sistema HERMES possui um catálogo (*Yellow Page Server*), usado para manter informações sobre os bancos de dados, incluindo a localização e relação de seus objetos. O sistema utiliza regras de integração que são registradas e armazenadas por uma ferramenta chamada *Information Pooling Toolkit*. Para que a integração seja possível, os atributos identificadores de cada relação nos banco de dados devem ser idênticos. Quanto aos domínios de cada atributo, o sistema possui uma ferramenta chamada *Conflict Resolution Toolkit*, onde são criadas regras para a resolução de conflitos. Os dados são convertidos para um formato definido através de funções (para cada domínio integrado pode ser gerado um tipo de arquivo diferente) e armazenados em padrão de texto. A autonomia local é preservada, pois nenhuma alteração é solicitada para o processo de integração. Uma limitação deste modelo é a implementação para bancos de dados específicos, no caso de fontes de dados estruturadas, como DBASE, INGRES e PARADOX. Outra limitação encontrada é a necessidade de informar o domínio (banco de dados) onde está sendo realizada a consulta, o que pode ser indesejável para o usuário final, bem como a necessidade da utilização de identificadores únicos. O sistema HERMES não utiliza um modelo de dados canônico, pois todas as funcionalidades de resolução de conflitos e de identificação de similaridades estão implementadas em sua linguagem de mediação.

No sistema Ozone [LAH 99], a solução apresentada baseia-se na integração lógica dos esquemas conceituais locais de fontes de dados estruturadas e semi-estruturadas em um esquema conceitual global. Os usuários têm uma visão única, uniforme e transparente das informações acessadas. O modelo de dados utilizado pelo sistema é uma extensão do padrão ODMG para o modelo OEM. Nesta abordagem, fontes de dados estruturadas podem ser tratadas como se fossem fontes de dados semi-estruturadas (se isso for desejado), para permitir a navegação na estrutura dos dados sem conhecimento de sua real estrutura. Para que seja realizado o acesso integrado às informações provenientes de fontes estruturadas e de fontes semi-estruturadas, um atributo identificador comum aos objetos que modelam uma mesma realidade é utilizado e as informações são armazenadas em um banco de dados físico (materialização dos dados). As referências disponíveis deixam claro que a linha de pesquisa dos autores é dedicada a encontrar propriedades estruturais puramente de dados semi-estruturados, sem mostrar como a integração entre os esquemas conceituais é executada.

O sistema TSIMMIS (*The Stanford-IBM Manager of Multiple Information Sources)* [MOL 97] fornece ferramentas para integração de fontes de dados heterogêneas, incluindo bancos de dados relacionais e fontes de dados semi-estruturadas (páginas *web*, arquivos XML). A arquitetura do sistema é baseada em mediadores e utiliza o modelo de objetos OEM, para o qual todos os esquemas conceituais locais são convertidos. O sistema possui uma linguagem de consulta, OEM-QL, que adota a sintaxe da OQL e é baseada nas cláusulas *SELECT-FROM-WHERE*. Todos os resultados das consultas são retornados como um grafo OEM. Um objeto OEM possui uma estrutura de quatro campos: *label* descreve o objeto; *value* representa o valor do objeto; *type* determina o tipo do objeto;

object-id identifica cada objeto de forma única, sendo utilizado para identificar cada fonte de informação. O usuário possui uma visão uniforme dos dados das várias fontes de informação. Uma limitação desta abordagem é que, para efetuar uma consulta, o usuário deve escrever instruções OEM-QL, exigindo conhecimento específico da linguagem.

O *Active Mediator Object Systems* - AMOSII [RIS 2000] consiste em um sistema baseado em mediadores distribuídos que se comunicam através da Internet. A integração dos dados é realizada através de visões, especificadas através de uma linguagem de consulta declarativa, que fornecem transparência de acesso aos dados. O modelo de dados utilizado é uma extensão do modelo de objetos baseado no modelo de dados funcional DAPLEX. As fontes de dados suportadas são banco de dados relacional, objeto-relacional, orientado a objetos, dados semi-estruturados, STEP e outros bancos de dados AMOS. Os conflitos existentes entre entidades que modelam um mesmo objeto do mundo real em diferentes bancos de dados são eliminados através de primitivas de mediação (*mediation primitives*), incluídas na linguagem de consulta utilizada, chamada AMOSQL. Para acessar dados de fontes externas, os mediadores possuem um ou vários *wrappers*. Por exemplo, um *wrapper* pode acessar um banco de dados relacional através de um ODBC, ou simplesmente acessar arquivos XML. Uma limitação deste sistema é que, para efetuar uma consulta, os usuários devem escrever instruções na linguagem nativa, exigindo conhecimento específico da linguagem AMOSQL.

4.1 Análise Comparativa das Metodologias

Nas soluções analisadas, o nível de interoperabilidade é fornecido tanto no nível semântico, que se refere aos vários modelos de dados que podem ser suportados pelas soluções, quanto no nível de comunicação, que se refere à capacidade das soluções de acessar informações localizadas em diferentes plataformas. Além disso, a camada de integração semântica, que se refere aos mecanismos utilizados para a integração, mapeamentos dos esquemas conceituais e resolução de conflitos, é integrada às linguagens de consulta ou encapsulada dentro de uma ferramenta de consulta específica, onde a heterogeneidade semântica dos modelos de dados é tratada.

No que se refere às fontes de dados suportadas pelas soluções analisadas, somente o HERMES não inclui fontes de dados semi-estruturadas no acesso integrado.

Nos Sistemas HERMES, Ozone e AMOSII, as informações (parcela de visualização) são visualizadas de forma única, uniforme e transparente. Estas informações estão limitadas à parcela de dados comuns às fontes de dados envolvidas, não sendo possível o acesso a informações complementares ou a identificação de novos relacionamentos entre as entidades. Estes sistemas utilizam um esquema conceitual global como referência para o acesso integrado. Com isso, o usuário não consegue identificar a origem das informações, o que pode ser crítico em alguns ambientes onde a origem da informação é fundamental (como em ambientes médicos, onde a identificação da fonte de dados correspondente ao exame onde um determinado diagnóstico foi efetuado é de extrema importância).

Ainda, algumas soluções [MOL 97, RIS 2000] fazem uso de ferramentas que exigem do usuário conhecimento de comandos SQL ou de linguagens nativas, o que pode ser indesejável.

Uma característica importante, para todos os sistemas que fornecem acesso integrado, é a preservação da autonomia local. Todos os sistemas estudados preservam a autonomia local das fontes de dados, não exigindo nenhuma alteração para a realização do processo de integração de esquemas.

Os modelos de dados canônicos, utilizados em ferramentas como Ozone e AMOSII, servem apenas para a conversão e armazenamento das fontes de dados suportadas para que, posteriormente, sejam consultadas através da linguagem de consulta nativa (materialização).

Uma limitação identificada em todas as soluções é a de não apresentar nenhum tipo de ferramenta que auxilie o usuário na fase de tradução dos esquemas conceituais locais para o modelo de dados canônico, fazendo com este tenha que informar os dados a serem disponibilizados, através de interfaces complicadas ou por meio da própria linguagem de consulta. Além disso, o acesso integrado baseia-se na existência de um identificador comum e fica limitado à visualização da parcela comum entre as fontes de dados.

A seguir, é apresentado um resumo das principais características das abordagens estudadas (TABELA 4.1).

TABELA 4.1 - Quadro Comparativo entre Abordagens de Acesso Integrado

Soluções/ Características	**HERMES**	**Ozone**	**TSIMMIS**	**AMOS II**
Nível de Interoperabilidade	Semântico e Comunicação	Semântico e Comunicação	Semântico e Comunicação	Semântico e Comunicação
Camada de Integração de Comunicação	Mediador/ *Wrapper*	Não informado	Mediador/ *Wrapper*	Mediador/ *Wrapper* (ODBC)
Camada de Integração Semântica	Linguagem de Mediação	Linguagem de Consultas	MSL	Linguagem de Consulta AMOSQL
Modelo de Dados Canônico	Não utiliza	Extensão ODMG (Materialização)	OEM	OO (Materialização)
Esquema Global	Sim	Sim	Não	Sim
Fontes de dados suportadas	Relacional, OO, Geográficos, Imagens (GIF)	Fontes de dados estruturadas e semi-estruturadas	Relacionais, Semi-estruturado	Relacional, AMOS, Semi-Estruturado, STEP
Parcela de Visualização	Dados Comuns (Identificador Comum)	Dados Comuns (Identificador Comum)	Dados comuns e complementares (Identificador Comum)	Dados Comuns (Identificador Comum)
Preservação da Autonomia Local	Sim	Sim	Sim	Sim
Forma de Visualização	Interface Gráfica e/ou Linguagem de Consulta	Linguagem de Consulta	Linguagem de Consulta	Linguagem de Consulta

5 Proposta de uso do Metamodelo XML como Modelo de Dados Canônico

O acesso integrado a fontes de dados semi-estruturadas a partir de metodologias de acesso integrado a bancos de dados heterogêneos ainda se encontra em fase de investigação [MIL 2000]. Contudo, a possibilidade de representação conceitual de fontes de dados semi-estruturadas viabiliza e simplifica a inclusão do acesso integrado a estas fontes de dados nestas metodologias, onde o acesso é suportado pela representação conceitual dos bancos de dados envolvidos. Neste ambiente, a XML vem sendo adotada como padrão na representação de fontes de dados semi-estruturadas, uma vez que um documento XML incluí a representação tanto da estrutura como dos dados. Entretanto, as linguagens de representação de esquemas para documentos XML apresentam algum tipo de restrição na representação de esquemas conceituais de banco de dados, bem como na representação de domínios, identificação de relacionamentos entre objetos e inclusão de informações semânticas.

Com o propósito de viabilizar o uso da XML como modelo canônico para a representação dos esquemas conceituais de fontes de dados heterogêneos, incluindo banco de dados e documentos XML, este trabalho iniciou com o desenvolvimento de um metamodelo XML, a ser utilizado na representação conceitual dos esquemas de exportação provenientes de bancos de dados relacionais, objeto-relacionais e também de documentos XML, permitindo, desta forma, o acesso integrado a fontes de dados estruturadas e semi-estruturadas a partir de metodologias originalmente voltadas à interoperabilidade de bancos de dados heterogêneos. Cabe salientar que, embora ainda não contemple a representação de métodos, o metamodelo aqui proposto pode ser facilmente estendido com a utilização de entidades externas.

Neste modelo, todos os objetos do mundo real, que se encontram modelados nos esquemas conceituais locais, são representados como entidades. O metamodelo proposto permite a representação de restrições de bancos de dados, que não são representadas em outras soluções que utilizam como base a XML [FLO 99], eliminando assim limitações apontadas pela literatura no uso do XML como modelo de dados canônico. Além disso, permite a inclusão de descrições sobre domínio de atributos e a indicação daqueles atributos que podem ser potenciais identificadores, servindo como suporte ao processo de mapeamentos em nível de instâncias e possibilitando o enriquecimento semântico em relação ao esquema conceitual original [RIB 96]. Algumas características da orientação a objetos encontrados em bancos de dados comerciais, como objetos complexos, também podem ser representadas neste modelo. Desta forma, a expressividade obtida pelo metamodelo XML é equivalente a do modelo de dados orientado a objetos, para a representação das estruturas estáticas, podendo ainda ser estendido, com a utilização de entidades externas, para suportar estruturas dinâmicas (métodos). Cabe salientar que a simplicidade da solução proposta não prejudica o poder de expressividade do metamodelo.

5.1 Metodologia de Mapeamento de Esquemas Conceituais proposta por Ribeiro

A abordagem proposta por Ribeiro [RIB 95], e adotada neste trabalho, incorpora aspectos destas diferentes metodologias de acesso integrado apresentadas na seção 3.2, ao mesmo tempo, eliminando algumas dificuldades apresentadas por elas.

Na metodologia proposta por Ribeiro [RIB 95], o acesso integrado é efetuado a partir do mapeamento dos esquemas conceituais locais exportados, já convertidos para um modelo de dados canônico (eliminando a heterogeneidade dos modelos de dados e permitindo a exclusão da representação de dados protegidos), sem a necessidade de integração de esquemas conceituais locais e das eventuais alterações associadas a este processo. A metodologia também inclui, para cada esquema de exportação criado, a relação dos objetos locais, representados neste modelo conceitual. Esta relação de objetos locais inclui a descrição textual dos objetos que serão disponibilizados a comunidade, além de informações adicionais sobre o universo de discurso, facilitando a identificação posterior de equivalências e conflitos de representação. Desta forma, torna-se possível uma visualização complementar e total das diferentes realidades modeladas nos bancos de dados autônomos, possibilitando ao usuário ter uma visão total e não mais uma visão fragmentada da entidade do mundo real em questão. Esta solução inclui características das diferentes soluções analisadas [RIB 95].

O processo de mapeamento inicia pela comparação dos esquemas de exportação, possibilitando a identificação das equivalências entre os objetos representados. Além disso, também são identificadas e armazenadas informações sobre os atributos equivalentes, atributos identificadores, regras de conversão para a linguagem local ou registro de divergências de representação, utilizadas quando do acesso integrado às informações.

Esta metodologia, além de manter a autonomia local e não requerer modificações das representações locais, apresenta as seguintes vantagens em relação às demais:

- não requer a criação de um esquema conceitual global;
- não requer conhecimento de uma linguagem de consulta específica;
- permite a identificação de equivalências entre representações de uma mesma entidade do mundo real, mesmo que os identificadores adotados localmente não sejam os mesmos;
- permite a identificação de novos relacionamentos entre os objetos do mundo real.

permite a visualização complementar e global dos diferentes enfoques e parcelas da realidade modeladas para uma mesma entidade do mundo real concedendo ao usuário, uma visão global, sem contudo especificar a origem dos dados.

5.2 XML – *Extensible Markup Language*

Nos últimos anos, a XML - *eXtensible Markup Language* [BRA 2000] tornou-se o formato de dados padrão para troca e representação de informação na *Web*. A XML é uma

linguagem de fácil compreensão, de fácil entendimento e baseada em texto. A seguir, é feita uma breve descrição da linguagem XML e de suas principais características.

A linguagem XML descreve uma classe de objetos de dados, chamada documentos XML, e o comportamento de programas que os processam [PIM 99]. O termo objeto de dados é utilizado como generalização das várias formas que podem dar origem a um documento XML, como um arquivo, um registro de um banco de dados ou um conjunto de *bytes* transmitido através de uma rede de computadores.

Um documento XML é constituído de marcações[3] (*markup*) e do texto delimitado por estas marcações (*character data*). Marcações correspondem a *start-tags*, *end-tags*, *empty-element tags*, referências a entidades, comentários, seções CDATA, declarações de tipo de documentos e instruções de processamento. As marcações e o conteúdo entre elas representam elementos XML. As marcações em um documento XML possuem o objetivo de especificar o significado de seu conteúdo. Entretanto, a XML é uma linguagem de descrição que define a estrutura dos documentos. Embora a linguagem XML se proponha a apresentar algumas características semânticas, não está claro como identificar esta semântica em aplicações específicas [DOR 2000].

De acordo com a sintaxe da XML, um documento deve possuir obrigatoriamente um elemento raiz, que contém todos os outros elementos aninhados dentro de si. Na maioria dos casos, este elemento recebe o nome da classe de documento a que pertence.

Uma das principais características, herdadas da SGML (*Standard Generalized Markup Language)*, é a separação do conteúdo (dados) da apresentação (visualização das informações) e a possibilidade de criar um *template* para as marcações dos documentos, cujos elementos e atributos podem ser controlados e validados. Esta característica é alcançada através das linguagens de descrição de esquemas desenvolvidas para a representação de documentos semi-estruturados, como a DTD [BOS 98] e a XML *Schema* [BIR 2001, MIC 2000, THO 2000].

As DTDs (*Document Type Definition)* têm o objetivo de especificar, por meio de uma gramática regular, como os documentos XML são compostos e como seus componentes podem ser aninhados. As DTDs são o que realmente distingue, tanto a XML, como a SGML, das demais linguagens de marcação. Em termos genéricos, DTD é um arquivo separado do restante do documento XML principal e que fornece um conjunto de regras para o qual ele é anexado [PIT 99]. Documentos XML que não apresentam uma DTD associada são considerados bem formados (*well-formated*), mas inválidos, pois não apresentam uma validação com a definição de tipo de documento. As regras de uma DTD podem ser descritas tanto em um arquivo à parte do documento, quanto no cabeçalho do próprio. No entanto, se as regras são escritas no próprio documento, elas são válidas apenas para aquele documento em questão. Ao contrário disso, se as regras forem descritas em um arquivo separado, vários documentos podem utilizá-las para descrever sua estrutura. As regras descritas em uma DTD são capazes de definir as seguintes funcionalidades para as aplicações:

- notificar uma aplicação XML, informando quais nomes e estruturas podem ser utilizadas em um tipo particular de documento e como podem estar aninhadas;

[3] Em uma linguagem de marcação, marcas ou *tags* definem o início e o fim do texto, marcado como unidade ou elemento de informação [BAX 2000].

- listar recursos externos (entidades externas) que podem ser requeridas no documento, bem como declarar recursos internos (entidades internas) que podem fazer parte do documento;

- listar tipos de recursos que não são XML (notações);

- mostrar recursos que não são XML (entidades de dados binários).

O padrão XML *Schema* [BIR 2001] especifica uma classe particular de documentos, assim como a DTD, mas ao contrário dela, XML *Schema* utiliza a sintaxe definida para linguagem XML. Além disso, XML *Schema* permite a utilização de diversos tipos de dados (*string*, *date*, *integer*, *float*, etc), o que não é possível na DTD. XML *Schema* introduz outros conceitos avançados, como tipos complexos, derivação de tipos abstratos, classes de equivalência, unicidade, chaves e referências. Isso aumenta a gama de estruturas que podem ser representadas e permite uma maior flexibilidade [MEL 2000].

Outras linguagens de descrição de esquemas para documentos semi-estruturados são a RDF Schema [BRI 2000], a XDR [LAY 98], a SOX [DAV 99], a Schematron [JEL 2000] e a DSD [KLA 99]. Abordagens como a ERX (*Entity-Relationship for XML*) [PSA 2000] utilizam um modelo conceitual baseado no modelo Entidade-Relacionamento para representação de esquemas conceituais de classes de documentos XML, incluindo os relacionamentos entre eles.

5.3 O Metamodelo XML

O metamodelo XML proposto neste trabalho foi definido a partir da metodologia de acesso integrado a banco de dados heterogêneos proposta por Ribeiro [RIB 95]. Por isso, embora a seqüência deste trabalho implemente uma ferramenta de apoio para a geração de esquemas de exportação, está já contempla a inclusão de informações a serem utilizadas em etapas posteriores do processo de acesso integrado [KAN 2000], também consideradas quando da definição do metamodelo XML e referidas ao longo desta dissertação.

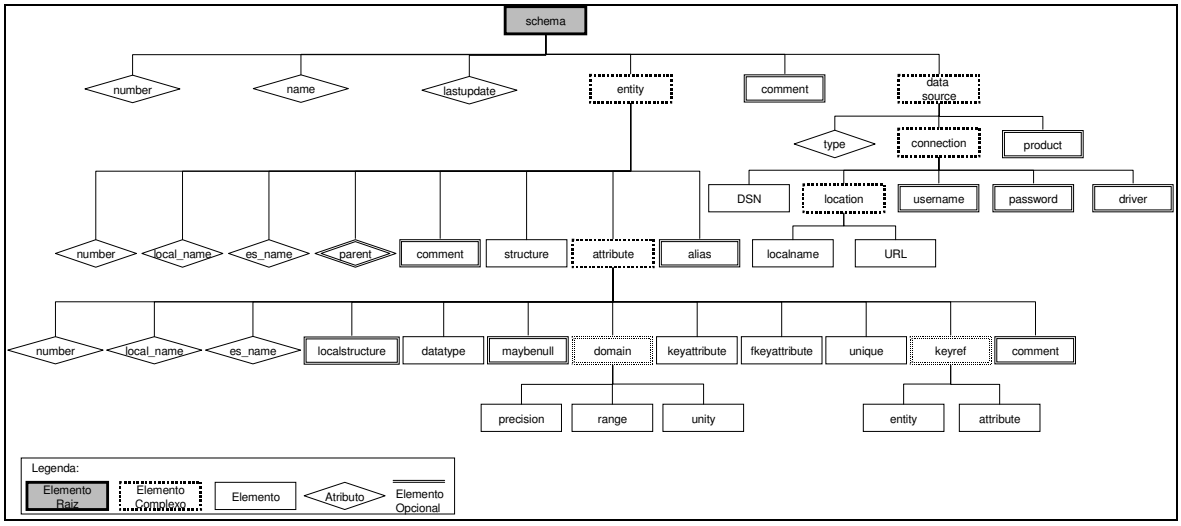


FIGURA 5.1 - Componentes do Metamodelo XML

5.3.1 Componentes do Metamodelo XML

Cada esquema de exportação[4] é representado no metamodelo XML através de um elemento raiz, identificado como "schema" (FIGURA 5.1). Este elemento encapsula todos os componentes da fonte de dados, contendo informações referentes à identificação de cada tipo de fonte de dados e de cada entidade representada. Para fins de identificação, o metamodelo inclui os seguintes componentes:

- *number*: atributo que contém um número inteiro seqüencial, gerado automaticamente, que identifica cada esquema de exportação de forma única;

- *name*: atributo que contém o nome do esquema de exportação, na forma como será visualizado pelo usuário;

- *lastupdate*: atributo que contém a data de geração ou última atualização do esquema de exportação;

- *comment*: elemento que contém a descrição textual da fonte de dados. Esta descrição serve para enriquecer semanticamente o modelo, sendo usada pela ferramenta de acesso integrado [KAN 2000], para interação com o usuário e ajuda no processo de identificação de equivalências.

As informações referentes e necessárias para que as fontes de dados possam ser acessadas são armazenadas através do componente *datasource*. Este componente inclui os seguintes elementos:

- *type:* atributo que indica o tipo da fonte de dados, isto é, banco de dados (DB), documentos XML (DOC) ou classes de documentos XML (ClassDoc);

- *connection:* componente que armazena informações referentes à conexão com a fonte de dados, incluindo:

  - *DSN*[5]*:* elemento que armazena informações sobre a localização e outras informações para conexão, como usuário e senha;

  - *location:* elemento que armazena informações sobre a localização física de uma fonte de dados, incluindo o nome da fonte de dados (*localname*) e o endereço físico, podendo ser um endereço local, um endereço IP ou uma URL (URL). No caso da representação de uma classe de documentos XML, este componente terá tantas instâncias quantos forem o número de documentos;

  - *username:* elemento que armazena o nome do usuário para a conexão com a fonte de dados. Este componente é utilizado para fontes de dados do tipo banco de dados (DB);

---

[4] Esquema de exportação corresponde à parcela do esquema conceitual local, incluindo os dados a serem compartilhados, representada no modelo de dados canônico [RIB 95].

[5] *Data Source Names* – DSN fornecem informações sobre a origem de uma fonte de dados específica.

  - *password*: elemento que armazena a senha para a conexão com a fonte de dados. Este componente é utilizado para fontes de dados do tipo banco de dados (DB);

  - *driver*: elemento que armazena o nome do *driver* para a conexão com a fonte de dados. Este componente é utilizado para fontes de dados do tipo banco de dados (DB).

- *product:* elemento que contém a informação do produto comercial. Este componente é utilizado para fontes de dados do tipo banco de dados (DB).

Cada entidade[6] é representada no "schema" como um componente "entity". Alguns elementos deste componente possuem função definida somente para fontes de dados semi-estruturadas, como por exemplo, o atributo "*parent*", o qual é utilizado para indicar o componente de dado pai em estruturas aninhadas. O componente "entity" é composto por:

- *number*: atributo que contém um número inteiro seqüencial, gerado automaticamente, que identifica uma entidade de forma única;

- *local_name*: atributo que contém a informação do nome local do objeto na fonte de dados;

- *es_name*: atributo que contém a informação do nome da entidade que está sendo modelada no esquema de exportação. Esta informação será visualizada pelo usuário e utilizada para padronização de nomes;

- *parent*: atributo que contém a informação do elemento pai em estruturas aninhadas. Este componente é utilizado para a representação de documentos semi-estruturados que apresentam elementos aninhados;

- *comment*: elemento que contém a descrição textual da entidade. Este componente é utilizado, pela ferramenta de acesso integrado, no processamento de consultas, ajuda no mapeamento de esquemas e pode incluir informações específicas sobre os relacionamentos. Este elemento também está voltado para a inclusão de enriquecimento semântico;

- *structure*: elemento que contém a informação do tipo de estrutura local para a qual entidade está relacionada, podendo ser: Table/Relationship (SGBDR), Class (SGBDOR) ou Element (Documentos);

- *alias*: elemento que contém a informação do apelido de uma entidade, para que este possa ser utilizado em uma linguagem de programação e de consulta;

O componente *attribute* é utilizado para representar cada atributo de uma entidade do esquema de exportação. Assim, para cada atributo de uma entidade deve existir um elemento *attribute* correspondente. Alguns componentes deste elemento são utilizados apenas para a representação de fontes de dados semi-estruturadas, como por exemplo, o

[6] Entidade é um "objeto" do mundo real, concreto (pessoa, carro, casa) ou abstrato (companhia, trabalho, curso universitário), que possui existência independente [ELM 99].

elemento "*localstructure*", o qual indica o tipo do componente de dado em um documento XML. O componente *attribute* é composto por:

- *number*: atributo que contém um número inteiro, gerado automaticamente, que identifica cada atributo de forma única;

- *local_name*: atributo que contém a informação do nome local do atributo na fonte de dados;

- *es_name*: atributo que contém a informação do nome do atributo para o esquema de exportação. Esta informação será visualizada pelo usuário e tem a finalidade de padronizar os nomes dos atributos;

- *localstructure*: elemento que contém a informação do tipo de componente de dado em um documento XML, podendo ser um elemento (E) ou um atributo (A) e é utilizado somente para fontes de dados semi-estruturadas. Esta informação é utilizada pela ferramenta de acesso integrado na montagem de instruções em linguagens de consulta XML;

- *datatype*: elemento que contém a informação do tipo de dado do atributo, utilizado como recurso de enriquecimento semântico. Esta informação é utilizada para a implementação de funções de conversão de dados. Os valores possíveis para este elemento são: *boolean*, *byte*, *integer*, *long*, *currency*, *single*, *double*, *date*, *time*, *text*, *memo* e *empty*;

- *maybenull*: elemento que contém a indicação de que o atributo pode conter valores nulos, utilizado como recurso de enriquecimento semântico;

- *keyattribute*: elemento que indica atributos que fazem parte da chave primária da entidade, usado para fins de enriquecimento semântico. Esta informação é útil para a determinação e identificação de relacionamentos entre as entidades;

- *fkeyattribute*: elemento que indica atributos que são chave estrangeira, usado para a determinação de relacionamentos entre as entidades;

- *unique*: elemento que assinala atributos que são potenciais candidatos à chave primária de uma entidade, servindo também para fins de enriquecimento semântico. Esta informação pode ser utilizada no acesso integrado a diferentes representações de uma mesma entidade do mundo real que possuem diferentes chaves primárias, permitindo a identificação de instâncias equivalentes;

- *comment*: elemento que contém a informação que descreve textualmente um atributo, e usado para enriquecimento semântico. Esta informação é utilizada para auxiliar no processo de identificação de equivalência e pela ferramenta de acesso integrado no processamento de consultas.

O componente "*domain*" é utilizado na representação das informações sobre os domínios dos atributos das entidades. Este elemento inclui recursos de enriquecimento semântico, como especificação de intervalos válidos, unidade adotada e padrões adotados localmente para serem utilizados, juntamente com a informação do tipo de dados, a serem utilizados em funções de conversão de domínios, quando necessário. O elemento "*domain*" é composto por:

- *precision*: elemento que contém a informação do número de casas decimais utilizadas para um atributo;

- *range*: elemento que contém a informação do intervalo válido de valores para um determinado atributo, e usado para enriquecimento semântico;

- *unity*: elemento que contém a informação da unidade de medida adotada por um atributo, e usado para enriquecimento semântico.

Para cada relacionamento existente entre as entidades, é criado um elemento "keyref". Este elemento é definido somente para atributos do tipo "*keyattribute*", "*fkeyattribute*" ou "*unique*". Isso ocorre porque somente atributos chaves, chaves estrangeiras e possíveis identificadores podem fazer parte de um relacionamento. Assim, o elemento "*keyref*" permite o encadeamento das entidades na qual o atributo está representado, possivelmente como uma chave estrangeira. O uso deste recurso é importante para a implementação do acesso integrado às informações pois, através dele, é possível identificar as instâncias equivalentes nas diversas fontes de dados. O componente *keyref* é composto por:

- *entity:* elemento que contém o nome da entidade onde o atributo está presente como chave estrangeira;

- *attribute:* elemento que contém o nome da chave primária da entidade com a qual está relacionado.

5.3.2 Representação Conceitual dos Esquemas de Exportação no Metamodelo XML

As fontes de dados semi-estruturadas, mais especificamente documentos XML, possuem um tratamento diferenciado, devido a grande irregularidade e descrição incompleta dos dados. No metamodelo proposto, um único esquema conceitual pode representar uma classe de documentos XML semanticamente equivalentes. Esta representação única possui os mesmos componentes de fontes de dados estruturadas, acrescidas de informações semânticas, facilitando, desta forma, o processo de identificação de equivalências com as fontes de dados componentes. O esquema de exportação que representa uma classe e/ou um documento XML consiste em uma estrutura aproximada dos objetos, pois pode omitir informações, restringindo-se aos dados compartilhados. Nesta abordagem, a seqüência física dos componentes de dados de uma instância de um documento XML não é relevante, uma vez que o metamodelo XML representa todos os esquemas locais de forma conceitual, sem fazer qualquer referência às estruturas físicas. Desta forma, esta abordagem pode ser utilizada não somente na integração lógica de classes de documentos, mas também na integração física.

No modelo de dados canônico aqui proposto, os componentes originais dos diferentes modelos de dados são representados no esquema de exportação, obedecendo as equivalências identificadas na TABELA 5.1.

TABELA 5.1 - Equivalência entre componentes do metamodelo XML e modelos de dados tradicionais

Metamodelo XML	Modelo Relacional	Modelo Objeto-Relacional	XML
Entidades	Tabelas	classes	elementos complexos
Atributos	Atributos	propriedades	atributos de elementos e elementos *textonly*
Keyattribute	chave primária	identificador da classe	atribuído pelo usuário
Fkeyattribute	chave estrangeira	-	atribuído pelo usuário
Domain	Restrições de domínio	restrições de domínio	análise dos dados para inferência do domínio
Unique	chaves candidatas	chaves candidatas	atribuído pelo usuário
Keyref	-	referências	atribuído pelo usuário
Demais informações, como comentários, algumas restrições de domínio e tipos de dados que não foram identificados, são incluídos pelo usuário.			

5.4 Considerações Finais

Entre as vantagens obtidas pelo uso do metamodelo XML como modelo de dados canônico, podemos citar:

- a representação é de fácil entendimento, uma vez que o documento é organizado de forma hierárquica;

- a representação inclui os aspectos estáticos da orientação a objetos adotados nos bancos de dados comerciais, embora ainda não possibilite a representação de métodos;

- fornece uma visão clara da estrutura do banco de dados ou do documento, independentemente do modelo de dados adotado localmente;

- provê portabilidade e interoperabilidade entre os sistemas componentes, uma vez que os esquemas estão representados em formato textos (ASCII), passíveis de serem lidos por um "*parser*" instalado em qualquer plataforma;

- provê a possibilidade de representação das restrições de dados presentes nos bancos de dados comerciais;

- apresenta flexibilidade de representação, possibilitando a inclusão de novas extensões, informações que se façam necessárias e restrições de dados, provendo assim o enriquecimento semântico;

- facilita a inclusão de fontes semi-estruturadas no acesso integrado a fontes heterogêneas de dados, permitindo comparações de representação, identificação de equivalências e estabelecimento de mapeamentos, uma vez que as fontes de dados semi-estruturadas são representadas, conceitualmente, de forma similar a adotada na representação de bancos de dados;

- permite a utilização de um mesmo modelo conceitual para uma classe de documentos semi-estruturados, conceitualmente equivalentes, mas estruturalmente diferentes;

- pode ser utilizado tanto em metodologias de integração lógica quanto de integração física de dados, uma vez que possibilita o armazenamento das informações provenientes de documentos XML em bancos de dados.

6 *XML Integrator*: a ferramenta desenvolvida

Além de consumir tempo, o processo de conversão de esquemas conceituais locais para um modelo de dados canônico e de detecção de equivalências e conflitos de representação não é trivial e não pode ser totalmente automatizado, requerendo a interação com o usuário [MIL 2001].

A ferramenta XML *Integrator* [DAR 2002], desenvolvida neste trabalho, tem por objetivo a geração de um ambiente que forneça ao usuário mecanismos de apoio ao processo de extração e conversão dos esquemas conceituais locais de fontes de dados heterogêneas[7], incluindo bancos de dados e documentos XML, para o Metamodelo XML.

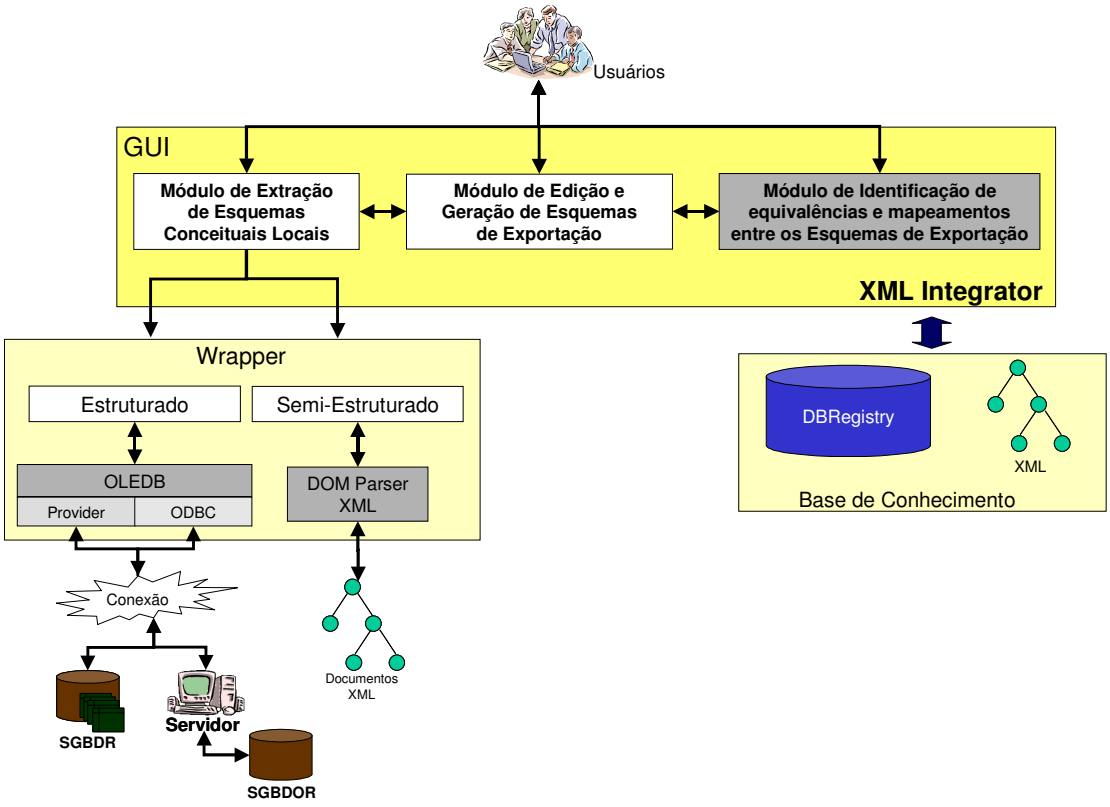


FIGURA 6.1 - Arquitetura do XML Integrator

6.1 Arquitetura da Ferramenta

Na ferramenta XML *Integrator* (FIGURA 6.1), a comunicação com os usuários é feita através de interfaces gráficas (GUI), as quais direcionam o usuário entre os vários processos.

Todas as informações utilizadas pela ferramenta são armazenadas em uma base de conhecimento, composta por um banco de dados relacional, o qual contêm todas as informações internas requeridas pela ferramenta, e por um conjunto de classes de

[7] A versão atual da *XML Integrator* suporta a geração de esquemas de exportações de bancos de dados relacionais, objetos-relacionais e documentos XML.

documentos XML, que armazenam informações sobre os esquemas de exportação, equivalências, conflitos e mapeamentos entre as fontes de dados participantes. Estes documentos poderão ser utilizados posteriormente, para fins de acesso integrado às informações.

A ferramenta XML *Integrator* é divida em três módulos principais:

- o módulo de extração de esquemas conceituais locais, que realiza a importação e extração dos esquemas conceituais das fontes de dados locais suportadas pela ferramenta;

- o módulo de edição e geração de esquema de exportação, que auxilia o usuário na edição, inserção e exclusão de esquemas de exportação;

- o módulo de identificação de equivalências e mapeamentos entre os esquemas gerados, que disponibiliza ao usuário ferramentas que auxiliam a identificação dos objetos equivalentes entre os esquemas conceituais. Este último (sombreado, na FIGURA 6.1), está fora do escopo deste trabalho, tendo sido implementado com funções mínimas.

Além destes componentes, a ferramenta inclui *wrappers,* específicos para cada tipo de fonte de dados, através dos quais realizam o acesso local às informações, convertendo-as em uma representação interna a ferramenta. Cabe salientar que a função destes wrappers fica limitada à extração da estrutura conceitual dos dados locais, e não ao acesso às instâncias destes dados.

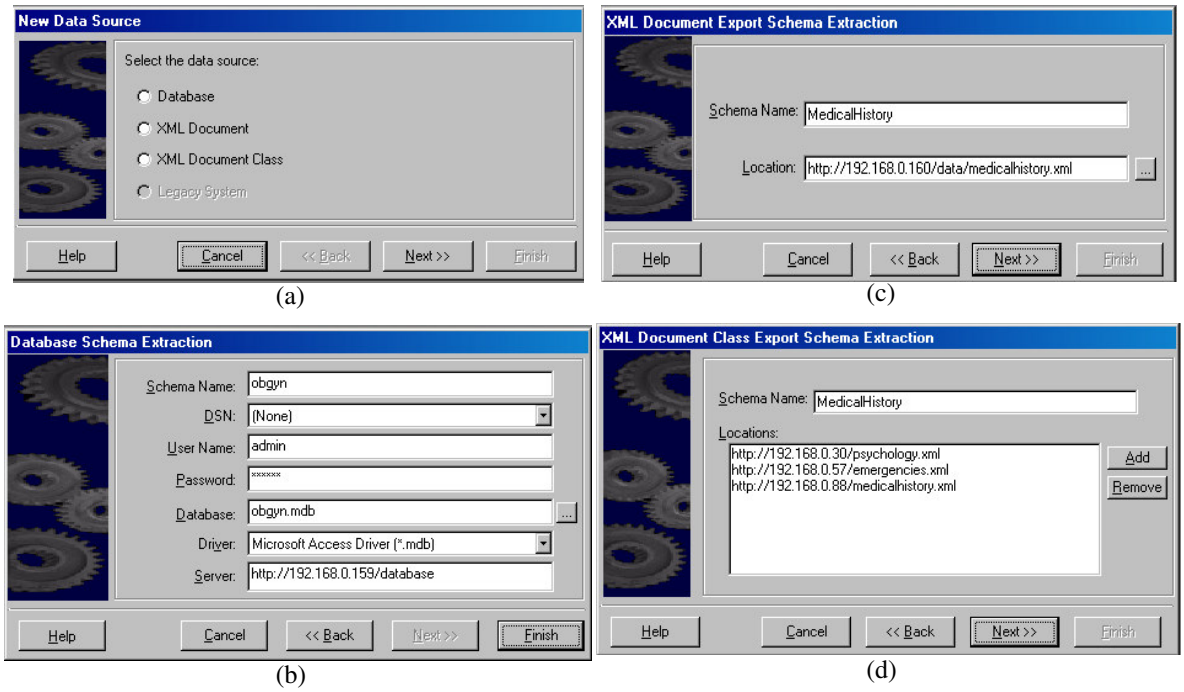


FIGURA 6.2 - Interfaces Iniciais do Módulo de Extração de Esquemas Conceituais

6.1.1 Módulo de Extração de Esquemas Conceituais Locais

O processo de extração de esquemas conceituais foi implementado no XML *Integrator* a partir de interfaces gráficas (FIGURA 6.2), que guiam o usuário durante esta

etapa. Este processo é divido em três etapas distintas: a primeira identifica o tipo de fonte de dados para a extração do esquema conceitual, que pode ser um banco de dados, um documento XML ou uma classe de documentos XML; a segunda etapa consiste da obtenção das informações necessárias à conexão com a fonte de dados; a terceira implementa a extração do esquema conceitual propriamente dito.

O processo de extração do esquema conceitual das fontes de dados é efetuado com o apoio de *wrappers,* específicos para cada tipo de fontes de dados.

6.1.1.1 Extração do Esquema Conceitual Local de Fontes de Dados Estruturadas

Para fontes de dados do tipo banco de dados, foi desenvolvido um *Wrapper* Estruturado (Anexo 6), que efetua a conexão com o banco de dados, extrai e converte as metainformações locais em estruturas utilizadas pela ferramenta. Todas as informações geradas são armazenadas no banco de dados "dbRegistro".

O processo de extração do esquema conceitual de fontes de dados estruturadas inclui as seguintes operações:

- o estabelecimento de uma conexão remota através do *driver* OLEDB[8]. Quando o OLEDB *Provider* não suportar uma fonte de dados, então é necessário um *driver* ODBC para que a conexão com as fontes de dados seja realizada;

- a análise e a importação do dicionário de dados da fonte de dados;

- a aplicação das regras de conversão, específicas para o tipo de fonte de dados.

No processo de extração do esquema conceitual local, são aplicadas as regras de equivalência para os modelos de dados utilizados pelos SGBDs. Após a extração, as informações são convertidas para o esquema de exportação, para que possam ser visualizadas e modificadas pelo usuário.

6.1.1.2 Extração do Esquema Conceitual Local de Fontes de Dados Semi-estruturadas

O *Wrapper* Semi-estruturado (Anexo 7) executa o processo de extração do esquema conceitual local de um único documento ou de uma classe de documentos XML, incluindo as seguintes operações:

- estabelece uma conexão remota através de um DOM *Parser*[9] [DOM 98];

- efetua a análise e aplicação de regras de conversão dos elementos de dados das fontes de dados semi-estruturadas;

Além do *wrapper,* a ferramenta possui um componente interno ao módulo de extração de esquemas conceituais, o qual realiza a otimização dos esquemas obtidos (somente para classes de documentos XML).

---

[8] Interface de baixo nível para bancos de dados Microsoft. A conexão é feita através do ADO (*ActiveX Data Objects*).

[9] A ferramenta utiliza o *Parser* da Microsoft versão 4.

O acesso ao documento XML é efetuado a partir das informações fornecidas pelo usuário. Caso o documento possua uma DTD associada, o usuário pode escolher entre a análise da instância ou da DTD.

Como mencionado anteriormente, a representação do esquema conceitual local de documentos XML está baseada em regras. Tais regras foram adaptadas a partir de diferentes propostas, que tratam da modelagem conceitual de documentos semi-estruturados, voltadas para a conversão de documentos XML em bancos de dados relacionais e orientados a objetos [ABI 97, NES 97, GAR 2000, PSA 2000]. A partir destas referências, para cada componente de dados dos documentos XML, são aplicadas as regras descritas na TABELA 6.1, possibilitando a representação conceitual de documentos e classes de documentos XML no modelo de dados canônico proposto neste trabalho.

O *Wrapper* semi-estruturado foi implementado a partir de um algoritmo recursivo (Anexo 7), onde cada componente de dados do documento XML é analisado individualmente. Inicialmente, é criado o nome do esquema conceitual local, a partir do nome do elemento raiz do documento XML analisado. Após, é verificado se o elemento raiz do documento XML possui componentes do tipo "*textonly*", "*mixed*" ou atributos. Caso possua alguns desses elementos, a primeira entidade do esquema conceitual é criada, recebendo o nome do elemento raiz do documento XML. A seguir, o algoritmo percorre todos os elementos filhos do elemento raiz, verificando a existência de componentes do tipo "*elementonly*" e "*mixed*". Para cada um destes elementos encontrados, uma nova entidade é criada. Dentro de cada um deles, são buscados componentes do tipo atributos e "*textonly*", para os quais são criados atributos para entidade de nível superior. As estruturas aninhadas, que fazem parte dos documentos semi-estruturados e que possuem em sua estrutura atributos e/ou elementos, são representadas como entidades, no esquema de exportação, por meio da aplicação de regras específicas.

Caso o processo de extração do esquema conceitual local seja aplicado a uma classe de documentos XML, este é realizado individualmente para cada documento XML. Após, todos os esquemas conceituais locais são submetidos ao Otimizador de Esquemas, o qual executa a integração destes esquemas, gerando um esquema conceitual local único, que representa a classe de documentos XML. Este módulo também permite a inclusão de novos documentos XML a uma classe de documentos existente, também através do Otimizador de Esquema.

Após o processo de extração do esquema conceitual local das fontes de dados, o esquema obtido é enviado ao módulo de edição e geração de esquemas de exportação para que o usuário possa realizar modificações e/ou incluir novas informações.

TABELA 6.1 - Regras de mapeamento de documentos XML para o Esquema de Exportação

Regra	Ação	Descrição
1	Criação do elemento raiz do Esquema de Exportação	O elemento raiz do metamodelo é criado a partir do elemento raiz do documento XML. Informações adicionais são fornecidas pelo usuário.
2	Criação da primeira entidade	Quando o elemento raiz possui componentes de conteúdo "*textOnly*", "*mixed*" ou atributos, então a primeira entidade criada terá o nome do elemento raiz.
3	Análise de componentes "*elementOnly*"	É criada uma entidade correspondente para todo componente do documento XML do tipo "*elementOnly*".
4	Análise de componentes do tipo "*empty*"	Os componentes do tipo "*empty*" são desconsiderados, podendo receber tratamento de objetos externos em futuras versões do metamodelo.
5	Análise de componentes do tipo "*textOnly*"	É criado um atributo para o componente de nível superior para todo componente do documento XML do tipo "*textOnly*".
6	Análise de componentes do tipo "*mixed*"	É criada uma entidade correspondente para todo componente do documento XML do tipo "*mixed*" (o conteúdo texto do componente é incluído como informação adicional à entidade).
7	Análise dos atributos dos componentes	É criado um atributo para a entidade correspondente, para todo atributo de um componente do documento XML.
8	Identificação de domínios e outras informações	Informações complementares para entidades e atributos e restrições de domínio, como intervalos válidos, precisão numérica, unidades de medidas, comentários e estrutura de dados, podem ser adicionadas através de informações fornecidas pelos usuários. Os tipos de dados dos atributos são identificados através da utilização de funções de reconhecimento de domínio, como "*IsNumeric*", "*IsText*", "*IsBoolean*", etc.

6.1.2 Módulo de Edição e Geração de Esquemas de Exportação

Este módulo é divido em duas etapas: a primeira delas possibilita a edição do esquema de exportação obtido e a segunda fase efetua a geração do esquema de exportação e da relação de objetos locais equivalentes, na forma de documentos XML.

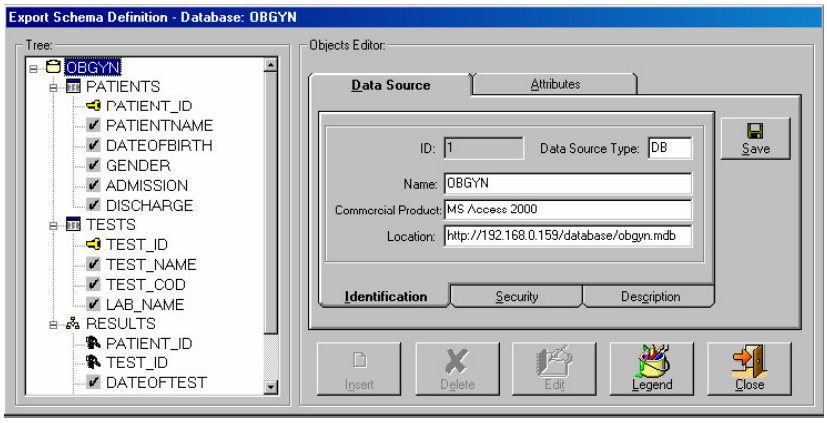


FIGURA 6.3 - Interface de Edição de Esquemas de Exportação

6.1.2.1 Edição do Esquema de Exportação

O processo de edição do esquema de exportação pode ser executado de duas formas:

- através da execução seqüencial do componente de edição interativa de esquemas de exportação;

- diretamente, através da execução do editor XML (somente para esquemas de exportação já gerados).

6.1.2.1.1 Componente de Edição Interativa de Esquemas de Exportação

Neste componente, o processo de edição do esquema de exportação é iniciado logo após a etapa de extração, de forma seqüencial, ou a partir de uma lista dos esquemas de exportação.

O processo de edição de um esquema de exportação inclui a adição de informações semânticas aos componentes do metamodelo (entidades, atributos, relacionamentos, etc), a exclusão de informações que não serão disponibilizadas à comunidade, tanto no nível de entidades como no nível de atributos e a modificação e/ou padronização dos nomes das entidades e atributos.

O esquema de exportação gerado pelo módulo de extração de esquemas conceituais locais é visualizado como uma árvore (FIGURA 6.3), possibilitando que seja estabelecida a hierarquia entre os objetos do esquema de exportação e também para contextualizar o nível onde a edição está ocorrendo, ou seja, para cada elemento da árvore é vinculado um conjunto de operações dependente de contexto, que permitem incluir, alterar e excluir informações dos componentes do esquema de exportação. Por exemplo, se o usuário selecionar a raiz da árvore, o sistema mostra informações a respeito da fonte de dados; caso o usuário selecione uma entidade, o sistema mantém o contexto para que as funcionalidades dos botões inserir, excluir e editar sejam atribuídas ao objeto que está selecionado na árvore, uma vez que existe somente um botão para cada operação. Os ícones associados aos componentes desta árvore indicam o tipo da fonte de dados, o tipo das estruturas internas e as características dos atributos. O significado de cada ícone pode ser visualizado a partir do botão *Legend*, existente na interface de edição (FIGURA 6.4).

O módulo de edição e geração de esquemas de exportação possibilita a inclusão de novas informações, a modificação e a exclusão de informações existentes nos componentes da árvore de esquemas. Esta edição é auxiliada através de interfaces específicas.

As alterações a serem efetuadas referentes ao componente raiz (FIGURA 6.5) possibilitam a edição de informações referentes à fonte de dados, como dados de identificação (nome e tipo da fonte de dados, produto comercial e localização física), dados de segurança (nome, senha entre outras informações necessárias para a realização da conexão com a fonte de dados) e uma descrição textual da fonte de dados.

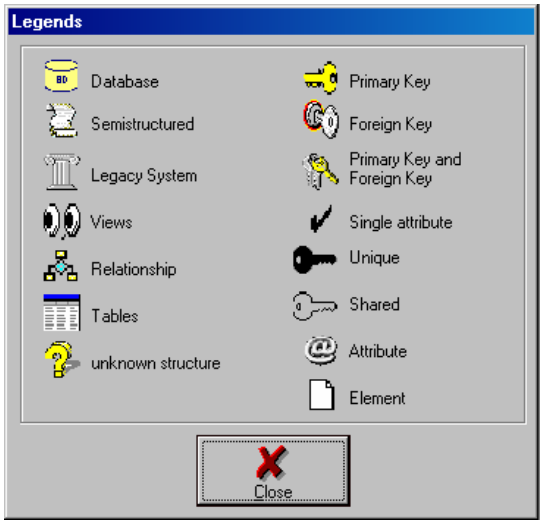
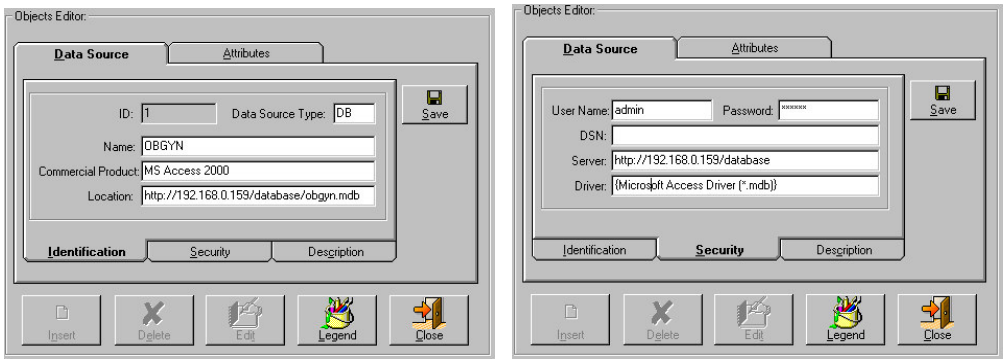
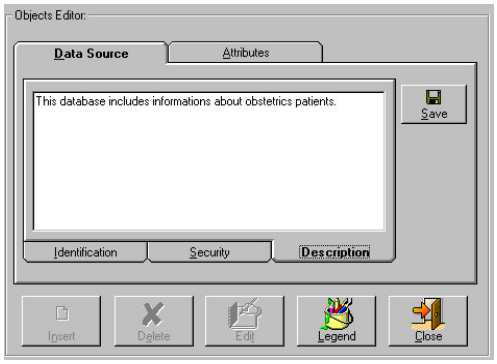


FIGURA 6.4 - Legendas da Árvore de Esquema



(a) (b)



(c)

FIGURA 6.5 - Edição de uma Fonte de Dados

As alterações a serem efetuadas, referentes a entidades, possibilitam a edição de dados relacionados a informações semânticas adicionais e dados de identificação da entidade. Quando o usuário seleciona uma entidade na árvore de esquema é disponibilizada a relação de atributos e habilitado os botões para inserção, edição e exclusão (FIGURA 6.6).

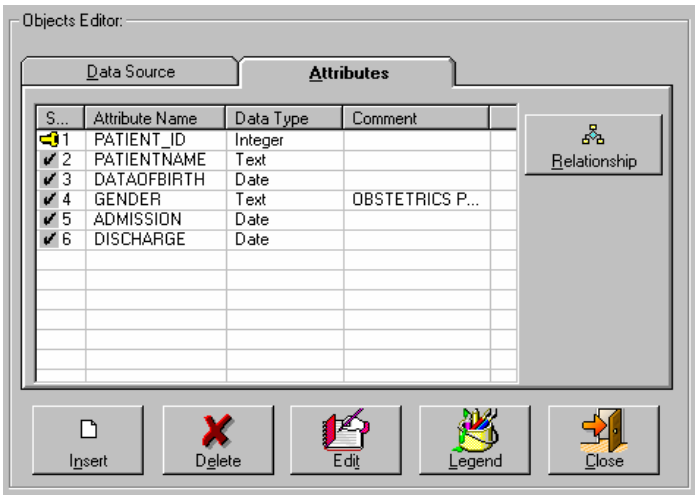
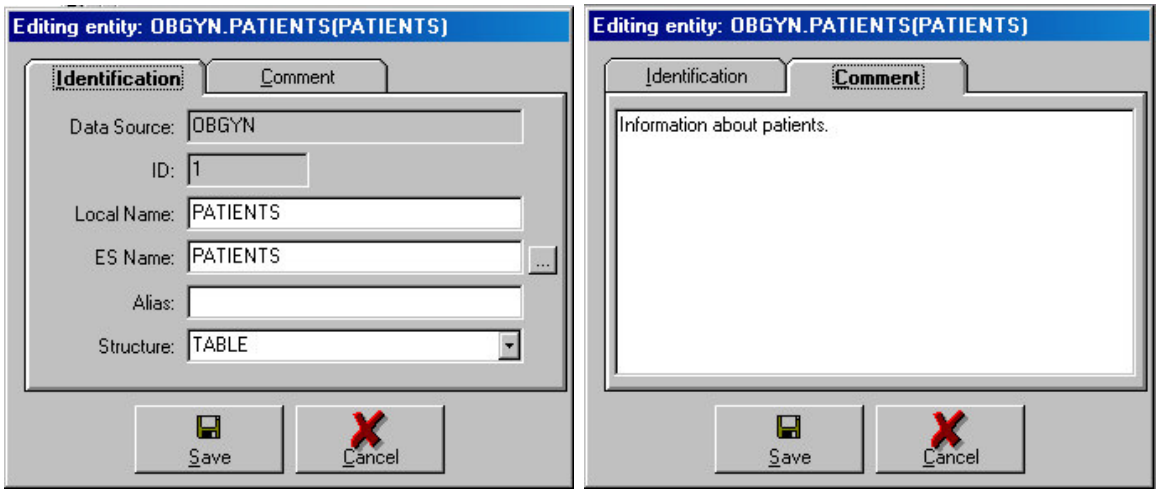


FIGURA 6.6 - Relação de atributos de um entidade

O processo de edição de uma entidade é efetuado através do botão "*edit*" (FIGURA 6.7) e permite atualizações referentes aos dados de identificação e descrição da entidade. O usuário pode alterar o nome local, nos casos em que o nome físico tenha sido alterado, nome da entidade no esquema de exportação, criar um apelido e verificar ou alterar o tipo de estrutura local da entidade na fonte de dados. Para a alteração do nome da entidade no esquema de exportação a interface possui um dicionário de sinônimos que ajuda o usuário a utilizar nomes padronizados. Para inserir uma entidade o usuário deve selecionar uma entidade qualquer, pressionar o botão "*insert*" e completar o formulário. Para apagar uma entidade, o usuário deve selecionar a entidade e pressionar o botão "*delete*", após o sistema emite uma mensagem de alerta, onde o usuário confirma a exclusão da entidade selecionada.

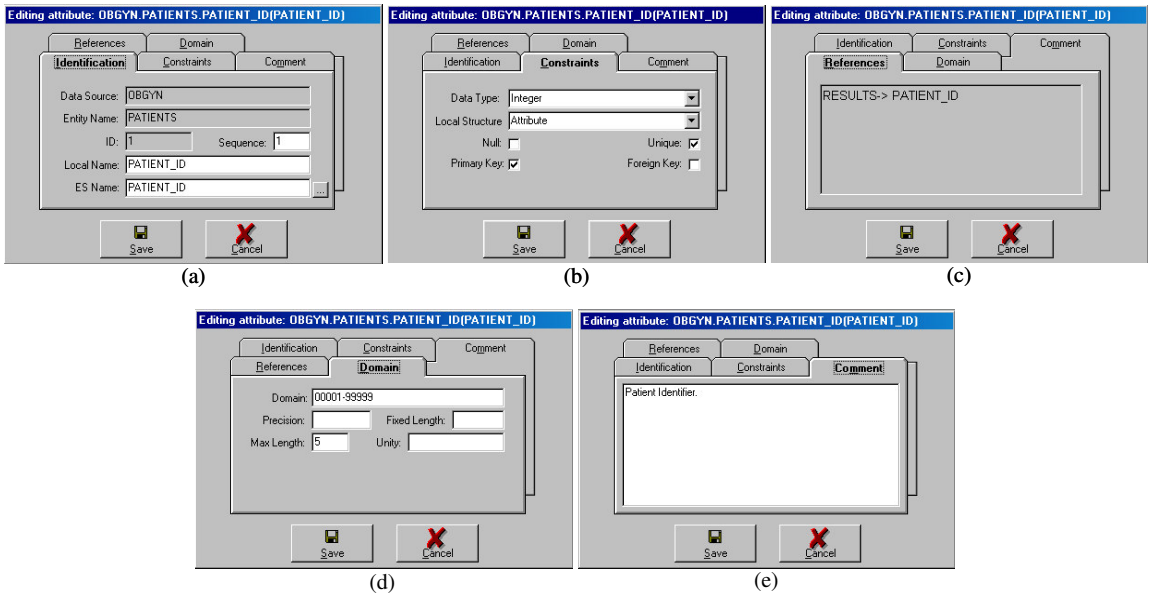
O nome da barra de título da interface de edição de uma entidade permite ao usuário identificar informações referentes a sua origem e é formado pela concatenação do nome da fonte de dados seguido de ponto, nome local da entidade e nome da entidade para o esquema de exportação entre parênteses.

As alterações a serem efetuadas, referentes a atributos, possibilitam a edição de dados relacionados a informações semânticas adicionais, restrições de domínio, referências e dados de identificação. Quando o usuário seleciona um atributo na árvore de esquema, os botões para inserção, edição e exclusão são habilitados.



(a) (b)

FIGURA 6.7 - Edição de Entidades



(a) (b) (c)

(d) (e)

FIGURA 6.8 - Interfaces de Edição de Atributos

O processo de edição dos atributos é efetuado mediante a seleção do atributo na árvore, através do botão "*edit*" da interface de edição (FIGURA 6.8). Esta interface é dividida em cinco componentes. O primeiro armazena informações sobre os dados de identificação do atributo, como nome local, nome para o esquema de exportação, identificador e o número da seqüência de visualização na interface, durante o processamento de consultas. O segundo contém informações sobre restrições de domínio, tais como tipo do dado, tipo de estrutura local, permitir nulos, chave primária, chave estrangeira e atributo de valor único. O terceiro armazena informações de onde o atributo é ou faz parte de uma chave primária. O quarto armazena informações a respeito do domínio do atributo como, por exemplo, faixa de valores permitidos, unidade de medida e tamanho máximo. O quinto refere-se aos comentários sobre o atributo, para fins de auxílio no processamento de consulta e na fase de identificação de equivalências.

O nome da barra de título da janela é formado pela concatenação do nome da fonte de dados seguido de ponto, nome local da entidade à qual pertence o atributo seguido de

ponto, nome local do atributo e nome do atributo no esquema de exportação entre parênteses.

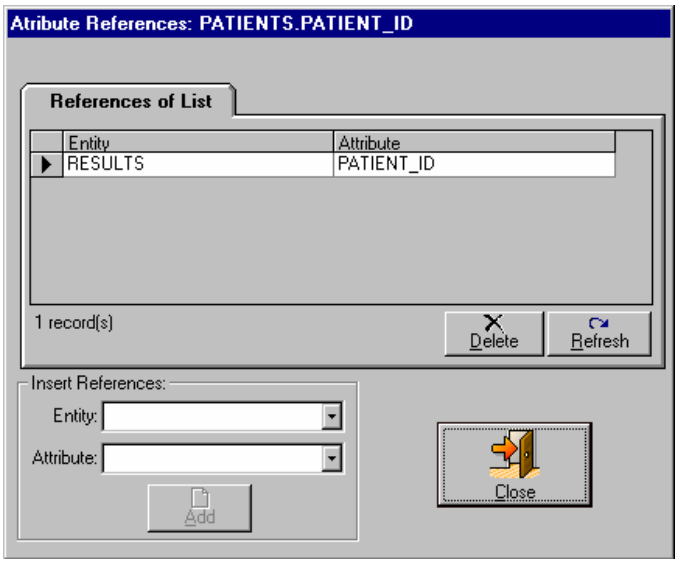


FIGURA 6.9 - Relacionamentos de um atributo

Para inserir um atributo em uma entidade o usuário deve selecionar um atributo qualquer da entidade que receberá o novo atributo, pressionar o botão "*insert*" e completar o formulário. Para apagar um atributo, o usuário deve selecionar o atributo e pressionar o botão "*delete*". Neste caso, o sistema emite uma mensagem de alerta, para que o usuário confirme a exclusão do atributo selecionado.

O processo de edição e inclusão de novos relacionamentos entre as entidades é efetuado através do botão de "*relationship*", habilitado quando da seleção de um atributo do tipo "*keyattribute*", "*fkeyattribute*" ou "*unique*" na árvore de esquema. Novas referências podem ser adicionadas através do quadro "*insert references*".

6.1.2.1.2 Editor XML de Esquemas de Exportação

A ferramenta *XML Integrator* permite que usuários, com conhecimento da sintaxe XML, editem os esquemas de exportação já gerados e armazenados, na forma de documentos XML, através do editor XML (FIGURA 6.10). Neste caso, as informações contidas nesses documentos podem ser diretamente modificadas. No exemplo do documento XML "Anamnese", o conteúdo das *tags* é apresentado em negrito, para facilitar a leitura. O usuário pode, também, imprimir e verificar as propriedades do documento selecionado.

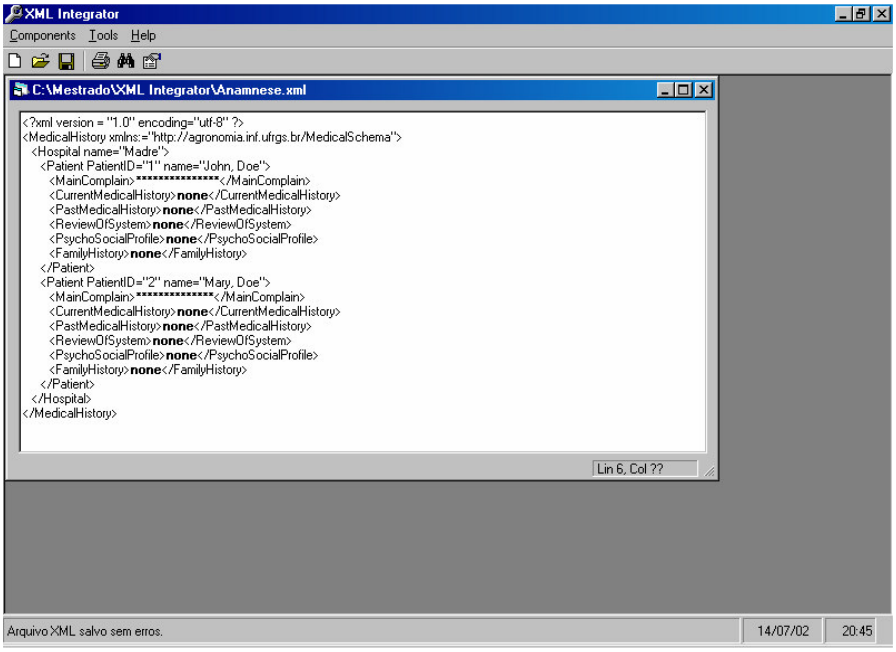


FIGURA 6.10 - Editor XML

6.1.2.2 Geração do Esquema de Exportação

O processo de geração do esquema de exportação é iniciado mediante solicitação do usuário, através do menu de ferramentas do sistema. Este processo inclui as seguintes operações:

- identificação do esquema de exportação editado pelo usuário;
- aprovação do esquema de exportação pelo usuário;
- geração do esquema de exportação definitivo;
- geração da relação dos objetos locais.

Após a aprovação do usuário, o esquema de exportação é verificado, convertido para o metamodelo XML e armazenado como um documento XML (FIGURA 6.11).

A geração do esquema de exportação, está associada à criação da relação de objetos locais, a partir das entidades representadas neste esquema. Conseqüentemente, cada esquema de exportação tem a si associado um catálogo de objetos disponíveis [RIB 95]. A relação dos objetos locais contém a descrição textual de cada entidade nos esquemas de exportação gerados, fornecendo informações semânticas adicionais que são utilizadas no mapeamento dos esquemas conceituais.

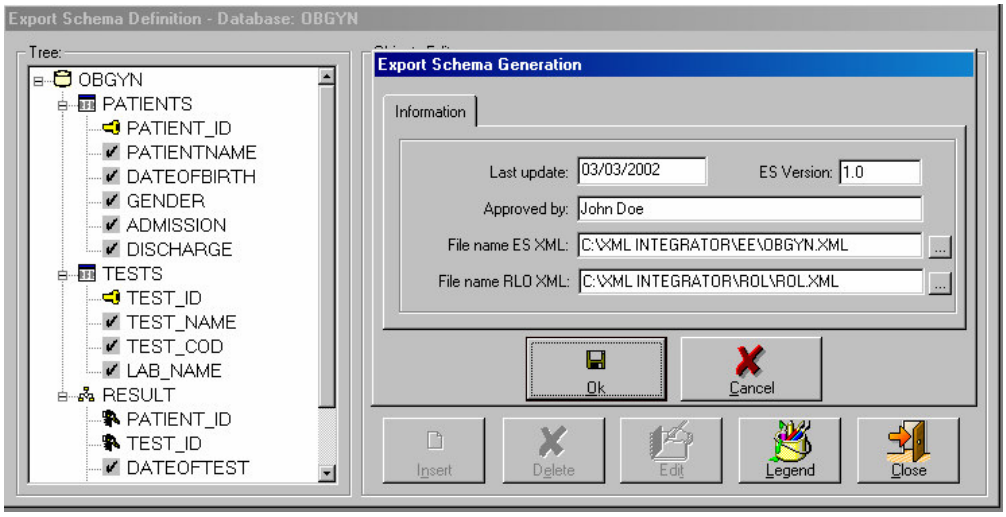


FIGURA 6.11 - Geração do Esquema de Exportação

Na ferramenta XML *Integrator*, a relação dos objetos locais é armazenada como um documento XML, contendo informações do esquema de exportação (número) seguido das informações das entidades (nome, estrutura e comentários). Assim, um único documento armazena as relações de objetos locais dos esquemas de exportação participantes. No processo de geração do esquema de exportação, a ferramenta verifica a existência da relação de objetos locais. Caso exista, a lista dos objetos do esquema de exportação é então adicionada na relação. Caso contrário, a relação dos objetos locais é criada, incluindo os objetos locais do esquema de exportação. A seguir, a relação dos objetos locais é mostrada ao usuário para que este possa fazer modificações que se façam necessárias (FIGURA 6.12).

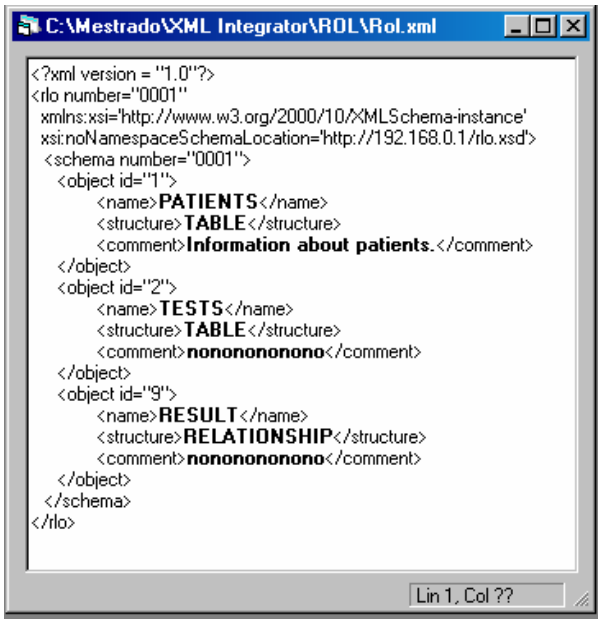


FIGURA 6.12 - Relação dos Objetos Locais

6.1.3 Módulo de Identificação de Equivalência e Mapeamentos entre os Esquemas de Exportação

A identificação de equivalências e possíveis conflitos de representação entre os esquemas conceituais locais das fontes de dados participantes é um processo adotado pela maioria das metodologias de integração. Este processo não pode ser completamente automatizado, uma vez que os esquemas locais podem ser modelados por diferentes projetistas, o que pode resultar em diferenças sintáticas e semânticas entre as modelagens adotadas.

A ferramenta aqui desenvolvida possui um módulo que tem como objetivo auxiliar o usuário no processo de identificação de equivalências e mapeamentos entre os esquemas de exportação. Este módulo é composto por uma máquina de inferência, um componente de edição de equivalências e um módulo de testes de correspondência. A máquina de inferência realiza o processo de comparação entre os esquemas de exportação e disponibiliza uma lista de prováveis equivalências ou conflitos entre um par de esquemas de exportação previamente escolhido pelo usuário. O processo de identificação de equivalências utiliza os nomes de exportação dos objetos para fins de comparação, uma vez que são armazenados de forma padronizada na base de conhecimento. Futuramente, a ferramenta poderá incluir outros métodos, como *thesaurus*, inteligência artificial e análise léxica e semântica. Identificados os objetos equivalentes, o processo continua, comparando somente os atributos que são qualificados como identificadores ou únicos. O componente de edição permite ao usuário editar, incluir e excluir equivalências identificadas. O componente de teste de equivalência utiliza-se de uma máquina de consulta que verifica as equivalências e identifica a natureza dos relacionamentos. A intervenção do usuário é requisitada em todos os componentes deste módulo, sendo realizado através de uma interface gráfica (FIGURA 6.13).

Após aprovação do usuário, a ferramenta armazena na base de conhecimento as informações referentes às equivalências, conflitos e diferenças de representação, bem como procedimentos para identificação de instâncias equivalentes, rotinas de mapeamento de representações equivalentes, conversão de domínio para os padrões estabelecidos localmente. A heterogeneidade de domínio pode ser solucionada através de funções de conversão, as quais são armazenadas como entidades externas ao documento XML que representa o metamodelo.

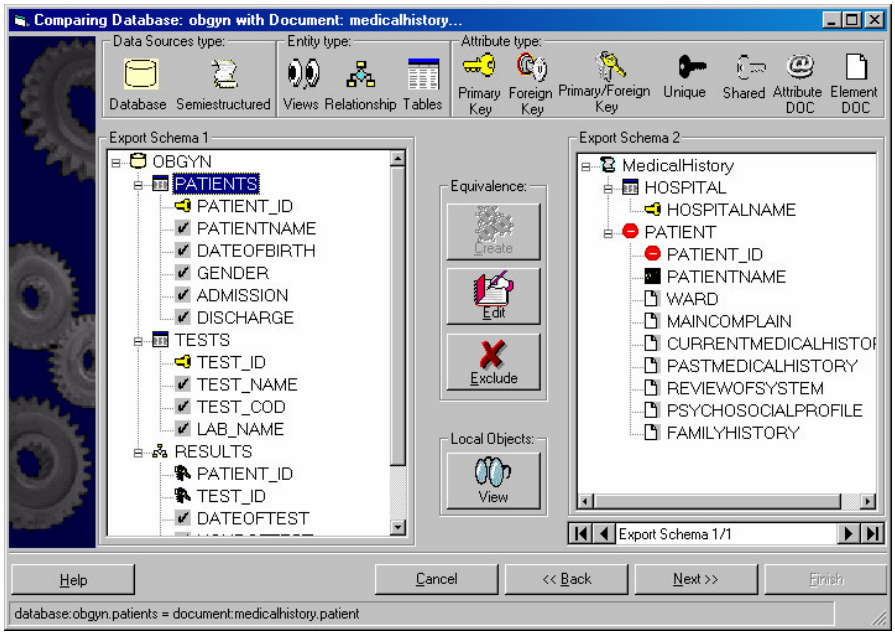


FIGURA 6.13 - Interface de Edição de Equivalências

O processo de identificação de equivalências e mapeamentos entre os esquemas de exportação está fora dos objetivos deste trabalho. Por esta razão, foram implementadas apenas funções básicas do processo. Este módulo servirá de base para o desenvolvimento futuro de uma nova versão da *XML Integrator*.

6.1.4 Base de Conhecimento

A base de conhecimento é responsável pelo armazenamento das informações referentes aos esquemas conceituais locais, esquemas de exportação, mapeamentos, equivalências e conflitos.

Os componentes da base de conhecimento (FIGURA 6.14) são armazenados em um banco de dados relacional[10], chamado "dbRegistro", e como documentos XML.

O banco de dados "dbRegistro" (Anexo 1) inclui estruturas temporárias que armazenam informações sobre os esquemas conceituais locais importados pelos *wrappers*, esquemas de exportação gerados, e metainformações incluídas pelos usuários através das interfaces de edição de esquema. O banco de dados contém, ainda, de forma permanente, o dicionário de sinônimos – DS e informações sobre os usuários, incluídas pelo administrador do sistema. O DS armazena informações referentes a palavras que possuem um mesmo significado, sendo utilizado para facilitar o processo de identificação de equivalência entre as fontes de dados.

As classes de documentos XML, que armazenam informações para o acesso físico às fontes de dados, os esquemas de exportação gerados e informações de equivalências, mapeamentos e conflitos entre estes esquemas de exportação, estão representadas em XML *Schema* [BIR 2001].

[10] O SGBD Microsoft Access© 97 foi utilizado para a implementação do banco de dados relacional.

Inicialmente, o módulo de extração de esquemas conceituais realiza a extração do esquema conceitual local de cada fonte de dados, as quais são temporariamente armazenadas no "dbRegistro", na forma de tabela. Estas informações são então enviadas ao módulo de edição e geração do esquema de exportação, para que o usuário efetue a edição do esquema exportação. O esquema de exportação final (Anexo 2), editado e aprovado pelo usuário, juntamente com a relação dos objetos locais (Anexo 3) correspondentes, são armazenados na forma de documentos XML válidos e bem formados. As equivalências entre os objetos (Anexo 4) e entre os atributos (Anexo 5) dos esquemas de exportação, identificadas e incluídas pelo usuário através do módulo de identificação de equivalências e mapeamentos, também são armazenadas na forma de documentos XML válidos e bem formados.

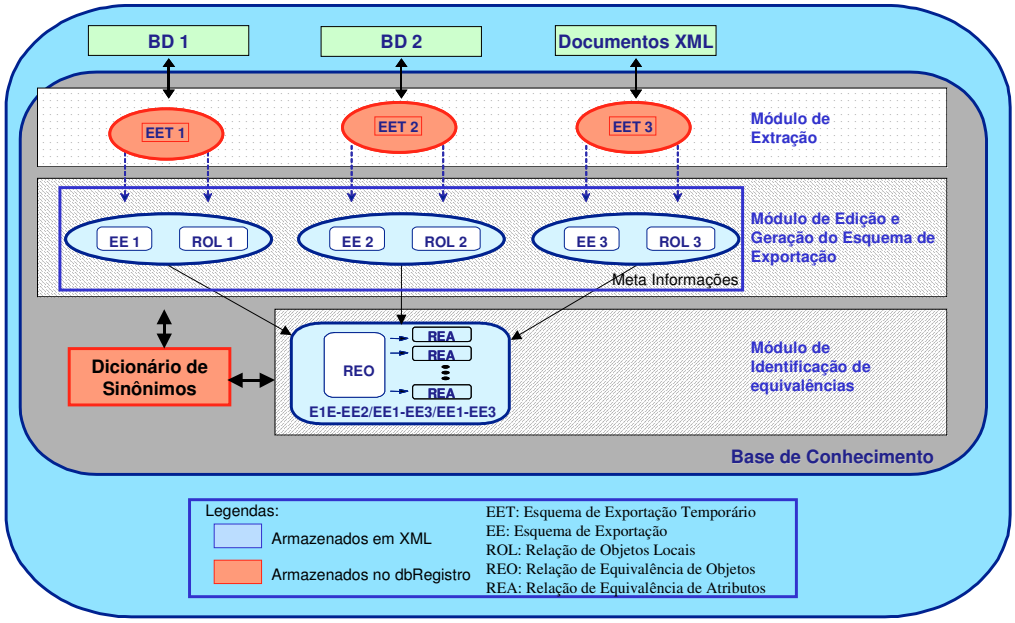


FIGURA 6.14 - Componentes da Base de Conhecimento

7 Gerando Esquemas de Exportação de Fontes de Dados Estruturadas e Semi-estruturadas: Estudo de Caso

A ferramenta XML *Integrator* foi validada através de sua aplicação em um ambiente hospitalar, onde as informações armazenadas em um banco de dados relacional e as informações provenientes de documentos XML foram representadas conceitualmente no metamodelo XML, de forma a permitir o posterior acesso integrado a estas informações a partir de uma ferramenta [KAN 2000] já existente, voltada para o acesso integrado a banco de dados heterogêneos. Esta ferramenta também foi desenvolvida a partir da metodologia proposta por Ribeiro [RIB 95], baseada na comparação e mapeamento dos esquemas de exportação das fontes de dados participantes.

7.1 O Ambiente Modelado

A ferramenta XML *Integrator* foi utilizada para a geração do esquema de exportação do banco de dados relacional "Obgyn", implementado em MS Access e contendo informações referentes aos exames laboratoriais de pacientes hospitalizados no setor de obstetrícia e ginecologia (FIGURA 7.1) e de fichas de anamnese dos pacientes baixados nos diferentes setores do hospital, armazenadas como documentos XML (FIGURA 7.2).

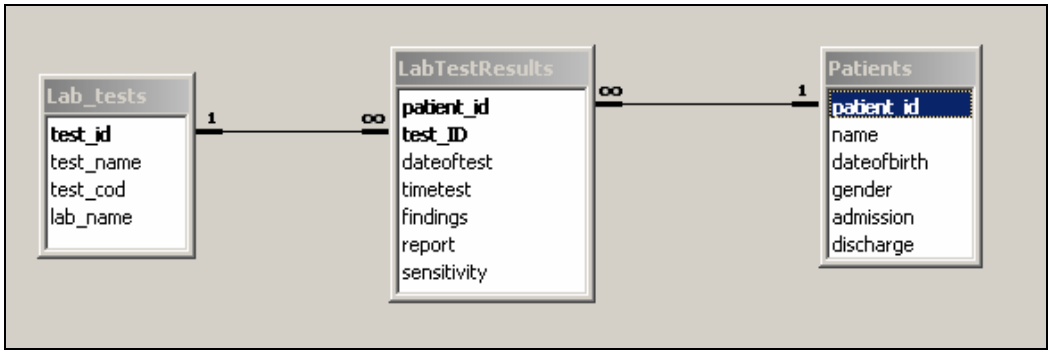


FIGURA 7.1 - Modelo Lógico do Banco de Dados Obgyn

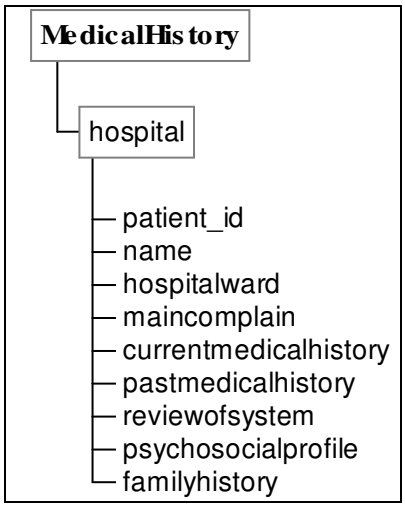


FIGURA 7.2 - Hierarquia dos elementos do Documento XML *MedicalHistory*

7.2 O Processo de Extração de Esquemas Conceituais Locais de Fontes de Dados

O processo de extração de um esquema conceitual local de uma fonte de dados é iniciado mediante a indicação, por parte do usuário, do tipo de fonte de dados, para que a ferramenta guie o usuário, passo a passo, até a geração do esquema de exportação. A seguir são descritos os processos de extração dos esquemas conceituais locais para fontes de dados estruturados e semi-estruturados.

7.2.1 O Processo de Extração de Esquemas Conceituais Locais de Fontes de Dados Estruturados

O banco de dados relacional "Obgyn", utilizado como exemplo, é composto pelas tabelas "PATIENTS", "LAB_TESTS" e "LABTESTRESULTS", contendo respectivamente informações sobre os pacientes, exames laboratoriais e resultados de exames. A última tabela ("LABTESTRESULTS") representa o relacionamento entre as duas primeiras tabelas ("PATIENTS" e "LAB_TESTS").

Inicialmente, o módulo de extração de esquemas conceituais locais conecta-se com a base de dados "Obgyn", mediante a disponibilização das informações pelo usuário. Após, o módulo realiza a importação das informações constantes no dicionário de dados para que sejam aplicadas as regras de mapeamento definidas. Aplicando as regras de mapeamento, no banco de dados "Obgyn", obtém-se como resultado três entidades, chamadas "PATIENTS", "LAB_TESTS" e "LABTESTRESULTS". A última é qualificada como um relacionamento, pois sua chave primária é composta de atributos que são chaves estrangeiras. No exemplo, a tabela "PATIENT" contém o atributo "PATIENT_ID", o qual está presente como chave estrangeira na tabela "LABTESTRESULTS", com um atributo de mesmo nome. Esta informação é capturada e representada no esquema de exportação através de uma característica do atributo, conforme FIGURA 7.3. Desta forma, os atributos do tipo chave primária possuem a lista de todas as entidades em que eles são chaves estrangeiras.

O resultado da extração do esquema conceitual local é mostrado ao usuário através do módulo de edição e geração do esquema de exportação para que ele possa efetuar a edição das informações extraídas.

De acordo com a FIGURA 7.4, as entidades “PATIENTS”, “LAB_TESTS” e “LABTESTRESULTS” são renomeadas para “PATIENTS”, “TESTS” e “RESULTS”, respectivamente.

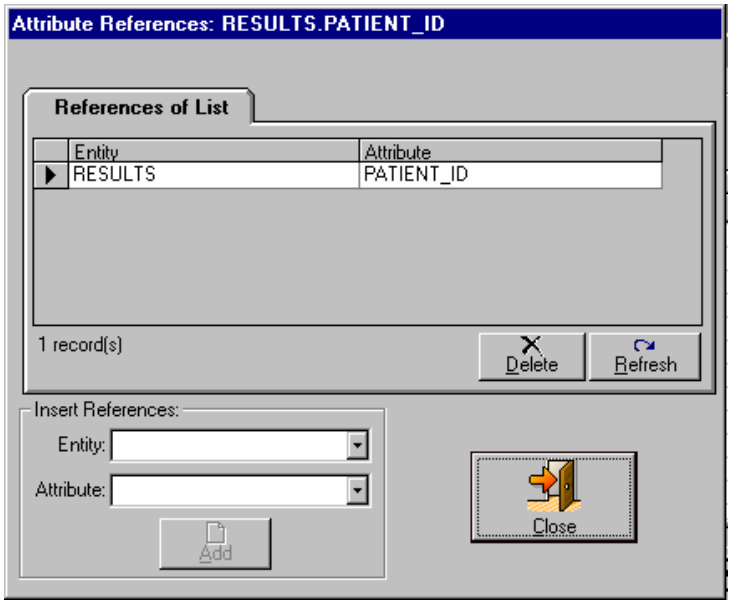


FIGURA 7.3 - Relacionamentos entre entidades

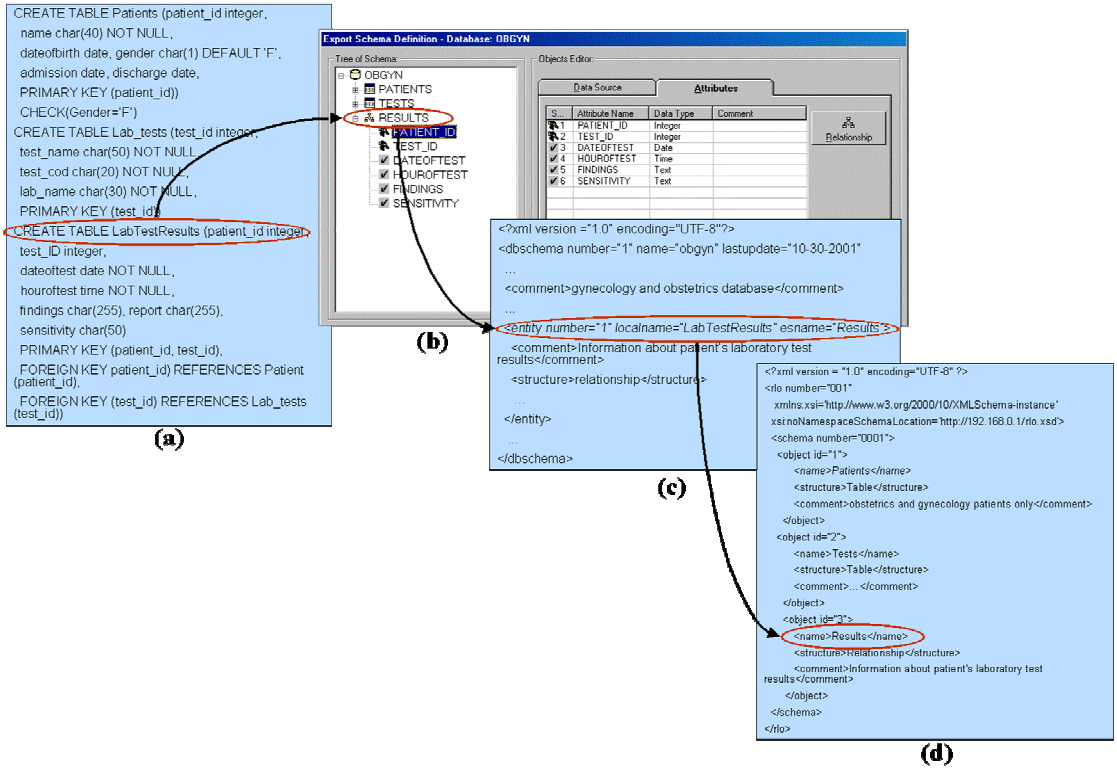


FIGURA 7.4 - (a) Esquema Conceitual Local (b) Interface de Edição de Esquema (c) Esquema de Exportação (d) Relação dos Objetos Locais

Após a edição das informações extraídas, ocorre a geração do esquema de exportação. Feito isso, a ferramenta solicita ao usuário uma confirmação, conforme FIGURA 7.5, onde consta informações referentes ao usuário que aprovou, data e versão e nomes físicos para armazenamento do esquema de exportação e o nome do arquivo XML que armazena a relação dos objetos locais. A relação dos objetos locais contém informações relevantes para cada entidade do esquema de exportação. Um fragmento do esquema de exportação gerado e da relação de objetos é mostrado na FIGURA 7.4. A versão completa do esquema de exportação e a relação dos objetos locais do banco de dados "Obgyn" estão disponíveis no Anexo 8.

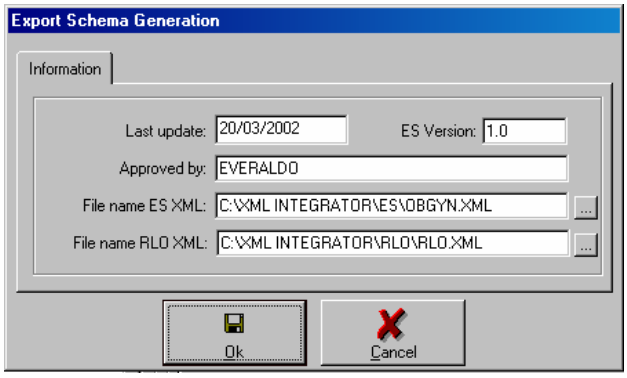


FIGURA 7.5 - Interface de confirmação da geração do EE para uma fonte de dados estruturada

7.2.2 O Processo de Extração do Esquema Conceitual Local de Documentos XML

O documento XML "MedicalHistory", utilizado como exemplo, contém informações sobre as fichas médicas de paciente de um hospital e é formado pelo elemento "hospital", o qual contém outros elementos que armazenam informações de cada paciente, como por exemplo, número de identificação, nome, histórico familiar, entre outros.

Inicialmente, o módulo de extração de esquemas conceituais locais faz a importação do documento XML "MedicalHistory", mediante a disponibilização das informações pelo usuário. Em seguida, o módulo faz a aplicação das regras de conversão definidas na seção 6.1.1. Estas regras são aplicadas a todos os componentes de dados do documento XML. Este processo é realizado pelo *wrapper* semi-estruturado, o qual recebe como entrada um documento XML e fornece como saída o esquema conceitual local do documento XML analisado.

O elemento raiz "schema" correspondente ao documento XML "MedicalHistory" é criado a partir aplicação a regra número 1. O atributo "name" recebe o nome do elemento raiz do documento XML e o elemento "comment" recebe as informações disponibilizada pelo usuário através da interface de edição de esquemas. O elemento "datasource" é criado a partir do elemento raiz. O atributo "type" e o elemento "product" recebem os valores "DOC" e "XML", respectivamente. As informações sobre a conexão são armazenadas no elemento "connection", o qual é composto pelo elemento "location", que por sua vez é composto pelos elementos "url" e "localname", os quais recebem os valores "http://192.168.0.160/data/" e "medicalhistory.xml", respectivamente. No caso de classes de documentos XML, para cada documento é associado um elemento "location".

Os elementos "hospital" e "patient" são identificados como entidades, a partir da aplicação da regra 3. Analisando o elemento "hospital", o atributo "name" é adicionado à

entidade pela aplicação da regra 7. Informações sobre o tipo de domínio e informações complementares são obtidas através da regra 8. Analisando o elemento “patient”, são identificados três atributos e seis elementos do tipo “textOnly”. Os três atributos “patientID”, “name” e “hospitalward” são adicionados como atributos da entidade “patient”, através da aplicação da regra 5. Os elementos “maincomplain”, “currentmedicalhistory”, “postmedicalhistory”, “reviewofsystem”, “psychosocialprofile” e “familyhistory” são adicionados à entidade “patient” pelo emprego da regra 7. A regra 8 é aplicada a todos os atributos da entidade para identificar informações sobre tipos de domínios e informações adicionais.

```
<dbschema number=”1” name=”MedicalHistory”
          lastupdate=”01-30-2002” version=”1.0”>
    …
 <comment>XML Document – Medical History</comment>
 <datasource type=”DOC”>
  <product>XML</product>
  <connection>
    <location>
      <localname>historymedical.xml</localname>
      <url>http://198.162.0.160/data/<url>
    </location>
  </connection>
 </datasource>
 ...
</dbschema>
```

```
...
<entity number=“1” localname= “Hospital”
          esname=“Hospital”>
 <comment>...</comment>
 <structure>DOC</structure>
 <attribute number= “1” localname= “name”
              esname=“namepatient”>
  <localstructure>A</localstructure >
  <datatype>text</datatype>
  <maybenull>false</maybenull>
  <keyAttribute>true</keyattribute>
  <fkeyattribute>false</keyattribute>
  <unique>true</unique>
  <comment>...</comment>
 </attribute>
 ...
</entity>
...
```

```
<entity number=“2” localname= “Patient” esname= “Patient”
          parent=”Hospital”>
 <comment>...</comment>
 <structure>DOC</structure>
 <attribute number=“1”  localname=“patientID”  esname= “patientID”>
  <localstructure>A</localstructure >
  <datatype>integer</datatype>
  <maybenull>false</maybenull>
  <keyAttribute>true</keyattribute>
  <fkeyattribute>false</keyattribute>
  <unique>true</unique>
  <comment>...</comment>
 </attribute>
  ...
</entity>
```

FIGURA 7.6 - Fragmento do Esquema de Exportação do documento XML “*MedicalHistory*”

O elemento “Hospital” possui o elemento “patient” aninhado. Na representação conceitual, este elemento é qualificado como uma entidade. A metainformação “parent” da entidade “patient” recebe o valor “Hospital”, o qual indica elemento pai. A FIGURA 7.6 mostra fragmentos do esquema de exportação do documento XML “*medicalhistory*”.

O resultado da extração do esquema conceitual local é mostrado ao usuário através do módulo de edição e geração do esquema de exportação para que ele possa efetuar a edição das informações extraídas, conforme FIGURA 7.7.

Após a edição das informações do esquema de exportação, o próximo passo é a geração do esquema de exportação e da relação dos objetos locais. Este processo é idêntico ao processo de geração de esquema de exportação de fontes de dados estruturadas. O anexo 9 apresenta o esquema de exportação e a relação dos objetos locais correspondente para o documento XML “MedicalHistory”.

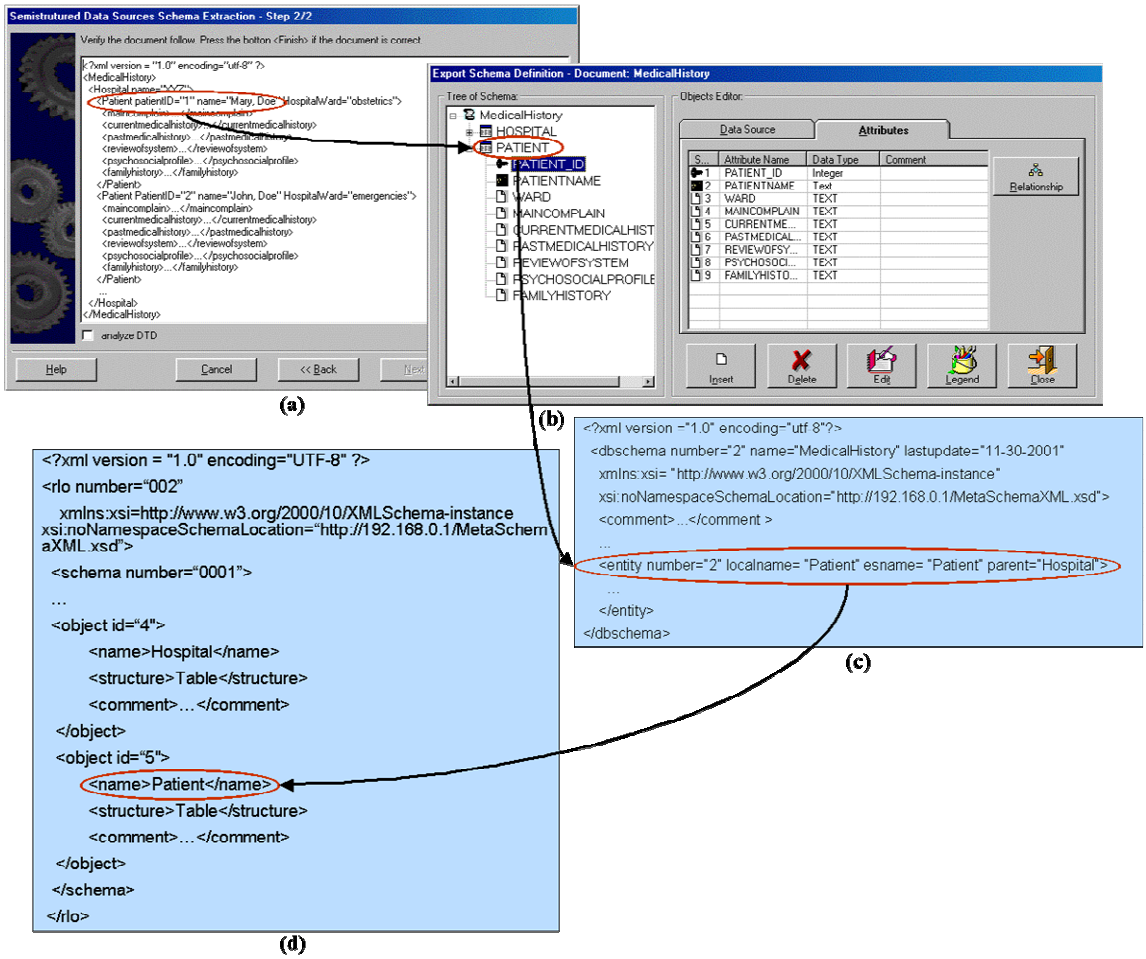


FIGURA 7.7 - (a) Documento XML (b) Interface de Edição de Esquemas (c) Esquema de Exportação (d) ROL

7.3 O processo de Geração da Relação de Equivalência de Objetos e Atributos

Na ferramenta *XML Integrator* o processo de geração de equivalências entre os objetos dos esquemas de exportação não está completamente implementado. No estudo de caso em questão, a ferramenta apresentou resultado adequado, pois o processo de identificação leva em consideração somente o nome de exportação dos objetos para verificar as equivalências. Como os nomes dos objetos são padronizados, a ferramenta conseguiu identificar todos os objetos equivalentes.

```
<?xml version = "1.0" encoding="UTF-8" ?>
<roe number="001" federation="001" creation="10-01-2001"
  xmlns:xsi=http://www.w3.org/2000/10/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="MetaSchemaXML.xsd">
  <object id="1">
        <baseentity>rlo001.Patients</baseentity>
        <externalentity>rlo002.Patient</externalentity >
        <equivalence>equal</equivalence >
        <constraint>…</constraint>
    </object>
</roe>
```

(a)

```
<?xml version = "1.0" encoding="UTF-8" ?>
<rae number="001" federation="001" creation="10-01-2001"
  xmlns:xsi=http://www.w3.org/2000/10/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="MetaSchemaXML.xsd">
  <object id="1" baseentity="dlo001.patients" baseattribute="patient_ID">
        <equivalence number="1">
          <externalentity>rlo002.patient</externalentity >
          <externalattribute>patientID</externalattribute>
          <type>equal</type>
        </equivalence>
  </object>
  <object id="2" baseentity="rlo001.patients" baseattribute = "name">
        <equivalence number="1">
          <externalentity>rlo002.patient</externalentity >
          <externalattribute>name</externalattribute>
          <type>equal</type>
        </equivalence>
  </object>
  <object id="3" baseentity="rlo001.results" baseattribute = "patient_ID">
        <equivalence number="1">
          <externalentity>rlo002.patient</externalentity >
          <externalattribute>patientID</externalattribute>
          <type>equal</type>
        </equivalence>
  </object>
</rae>
```

(b)

FIGURA 7.8 - (a) Relação de Equivalência de Objetos (b) Relação de Equivalência de Atributos

O esquema de exportação do banco de dados "obgyn" é utilizado como base, para fins de exemplificar o processo de geração das tabelas de equivalência. Realizando o processo de análise e comparação com o esquema de exportação do documento XML "MedicalHistory", a ferramenta apresenta o resultado ao usuário. A relação de equivalência de objetos e a relação de equivalência de atributos (FIGURA 7.8) são geradas após a verificação e aprovação do usuário.

8 Conclusões

Este trabalho propõe uma solução para o acesso integrado a fontes de dados heterogêneas, incluindo banco de dados e documentos XML, a partir de metodologias para o acesso integrado a banco de dados heterogêneos, baseadas na comparação dos esquemas conceituais.

Para isso, propõe-se o uso de um Metamodelo XML como modelo de dados canônico para a representação de fontes de dados, o qual inclui mecanismos de enriquecimento semântico, que possibilitam a inclusão da expressividade requerida para a representação conceitual, tanto de fontes de dados estruturadas, como de semi-estruturadas, de forma compatível com a utilizada para bancos de dados. Embora a maioria das soluções disponíveis adote o modelo de dados orientado a objetos como modelo de dados canônico, devido a sua capacidade de expressividade, o metamodelo XML adotado neste trabalho, embora mais simples, consegue capturar todas as características importantes dos modelos de dados utilizados comercialmente, além de possibilitar a representação conceitual de documentos semi-estruturados.

A representação de esquemas conceituais em um modelo e dados canônico requer a conversão dos esquemas conceituais locais e a extração do esquema concietual de fontes de dados semi-estruturadas.

Embora a conversão do esquema conceitual para o modelo de dados canônico não possa ser totalmente automatizado, a existência de uma ferramenta de apoio pode facilitar enormemente este processo. Com este objetivo, foi desenvolvida nesta dissertação a ferramenta *XML Integrator* [DAR 2002] de apoio à conversão e extração de esquemas conceituais locais para o modelo de dados canônico proposto. Em relação a outras soluções, a ferramenta aqui desenvolvida auxilia o usuário de forma amigável e interativa, sem a necessidade do conhecimento de uma linguagem de consulta ou de programação específica para a realização do processo de extração dos esquemas conceituais locais e geração dos esquemas de exportação.

Entre as vantagens da utilização do modelo de dados canônico proposto, podemos citar:

- a representação é de fácil entendimento, uma vez que é organizada de forma hierárquica, similar ao modelo relacional, possibilitando a representação de aspectos da orientação a objetos, tais como objetos complexos, adotados nos bancos de dados comerciais;

- uma visão clara da estrutura do banco de dados ou documento, independentemente do modelo de dados adotado localmente;

- a portabilidade e interoperabilidade entre os sistemas componentes, uma vez que os esquemas estão representados em formato textos (ASCII), passíveis de serem lidos por um *parser* instalado em qualquer plataforma;

- a possibilidade de representação das restrições de dados presentes nos bancos de dados;

- a flexibilidade de representação, possibilitando a inclusão de novas extensões, informações que se façam necessárias e restrições de dados;

- a possibilidade de modelagem conceitual de fontes de dados semi-estruturadas, facilitando a inclusão destas fontes de dados em metodologias de acesso integrado a fontes heterogêneas de dados, baseado na comparação de esquemas conceituais dos banco de dados participantes;

- a utilização de um único modelo conceitual para uma classe de documentos semi-estruturados conceitualmente equivalentes, mas estruturalmente diferentes.

Outra contribuição consiste na possibilidade de utilização desta abordagem em processos de integração física, bem como na área de *Data Mining,* uma vez que possibilita que informações provenientes de documentos XML sejam armazenados em um banco de dados.

Em relação a outras soluções, a ferramenta aqui desenvolvida apresenta as seguintes vantagens:

- as equivalências são identificadas e armazenadas sem necessidade de um esquema conceitual global;

- os esquemas de exportação são representados em um modelo canônico simples e semanticamente rico, compatível com a representação conceitual de documentos XML e bancos de dados comerciais;

- os mapeamentos entre diferentes representações são identificados a partir da comparação dos esquemas de exportação, sem a necessidade de implementar uma linguagem específica, como em sistemas *multidatabases*;

- para a identificação e mapeamentos entre os esquemas, não há necessidade de acesso aos dados das fontes de dados, como em sistemas baseados em mediadores.

A solução aqui proposta apresenta limitações como a de não inclui a representação de métodos dos modelos de dados orientados-a-objeto e a representação de dados textuais. A primeira restrição pode ser solucionada através da inclusão de entidades externas ao metamodelo XML proposto. A segunda, pode ser solucionada através da construção de um *wrapper* específico para dados não estruturados. Estas extensões podem ser incluídas em trabalhos futuros.

A usabilidade da XML *Integrator* encontra-se ainda em fase de estudo e testes, ficando algumas sugestões a serem tratadas em novas versões da ferramenta:

- aperfeiçoamento das interfaces do protótipo desenvolvido, de forma a incluir novas funcionalidades;

- implementação do módulo de identificação de equivalências, incluindo também funções de conversão de domínio.

- integração da *XML Integrator*, já completo, com a ferramenta desenvolvida por Kantorski [KAN 2000].

Anexo 1 Dicionário de Dados dbRegistro

O dicionário de dados possui as especificações para as diferentes entidades da ferramenta apresentado no capítulo 6.

A tabela "tbEsquema" é utilizada para armazenar, de forma temporária, as informações do esquema de exportação durante o processo de extração do esquema conceitual local.

TABELA: tbEsquema		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador do esquema
TipoEsquema	Texto	DB, Doc, ClassDoc
NomeEsquema	Texto	Nome para o esquema
NomeFisico	Texto	Nome físico do esquema
ProdutoComercial	Texto	Nome do produto comercial para banco de dados
Path	Texto	Localização da fonte de dados
Uid	Texto	Nome do usuário para conexão
Pwd	Texto	Senha para conexão
DNS	Texto	Arquivo DNS para conexão
Server	Texto	Local no servidor para conexão
Driver	Texto	*Driver* utilizado para a conexão
Versão	Texto	Versão do EE
ArquivoEE	Texto	Localização do arquivo XML que contém as informações do esquema de exportação da fonte de dados
ArquivoDOL	Texto	Localização do arquivo XML que contém as informações da relação dos objetos locais do esquema de exportação
DataUltimaAtualizacao	Data	Data da última atualização do esquema de exportação
AprovadoPor	Texto	Nome do usuário que aprovou e gerou o esquema de exportação
Descrição	Texto	Descrição textual da fonte de dados

A tabela "tbEntidadesTemp" é utilizada para armazenar, de forma temporária, as informações das entidades que são identificados pelo *wrapper* semi-estruturado durante o processo de extração do esquema conceitual de documentos XML.

TABELA: tbEntidadesTemp		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador
NomeEntidade	Texto	Nome da entidade

A tabela “tbAtributosTemp” é utilizada para armazenar, de forma temporária, as informações dos atributos que são identificados pelo *wrapper* semi-estruturado durante o processo de extração do esquema conceitual de documentos XML.

TABELA: tbAtributoTemp		
Nome Coluna	**Tipo**	**Descrição**
IDEntidade	Inteiro longo	Entidade a que pertence
NomeAtributo	Texto	Nome do atributo
Tipo	Texto	Tipo de dado
Estrutura	Texto	Estrutura original do atributo (em casos de DOC ELEMENT ou ATRIBUTE)
Exemplo	Texto	Exemplo de um conteúdo do atributo

A tabela “tbPrimaryKey” é utilizada para armazenar, de forma temporária, as informações das chaves primárias durante o processo de extração do esquema conceitual local de fontes de dados estruturadas.

TABELA: tb tbPrimaryKey		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador
ID_DB	Inteiro longo	Indica a fonte de dados
table_name	Texto	Nome da tabela
Ordinal	Inteiro	Número do índice
column_name	Texto	Nome da coluna que é chave primária

A tabela “tbForeignKey” é utilizada para armazenar, de forma temporária, as informações das chaves estrangeiras (relacionamentos entre tabelas) durante o processo de extração do esquema conceitual local de fontes de dados estruturadas.

TABELA: tb tbForeignKey		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador
ID_DB	Inteiro longo	Indica a fonte de dados
pk_table_name	Texto	Nome da tabela que contém a chave primária
pk_column_name	Texto	Nome da coluna que é chave primária
fk_table_name	Texto	Nome da tabela em que a chave primária é chave estrangeira
fk_column_name	Texto	Nome da coluna na tabela de chave estrangeira

A tabela “tbEntidades” é utilizada para armazenar, de forma temporária, as informações das entidades criadas durante o processo de extração do esquema conceitual local.

TABELA: tbEntidades		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identifica cada entidade
CodigoEsquema	Inteiro longo	Esquema a que pertence a entidade
NomeLocal	Texto	Nome local da entidade
NomeEE	Texto	Nome para o esquema de exportação
Alias	Texto	Apelido usado para facilitar programação
Estrutura	Texto	Tabela; Classe; Relacionamento ou elemento
Comentários	Texto	Descrição textual da entidade

A tabela “tbAtributos” é utilizada para armazenar, de forma temporária, as informações dos atributos de cada entidade criada durante o processo de extração do esquema conceitual local.

TABELA: tbAtributos		
Nome Coluna	**Tipo**	**Descrição**
IDAtributo	Inteiro longo	Identificador único do atributo
IDEntidade	Inteiro longo	Entidade a que pertence o atributo
NomeLocal	Texto	Nome do atributo na fonte de dados
NomeEE	Texto	Nome do atributo no esquema de exportação
Seqüência	Byte	Número seqüencial que indica a ordem em que o atributo será visualizado pelo usuário
EstruturaLocal	Texto	Estrutura original do atributo (em casos de DOC ELEMENT ou ATRIBUTE)
Tipo	Texto	Tipo de dado do atributo
MayBeNull	Boolean	Indica se o atributo pode conter valores nulos
PK	Boolean	Indica se o atributo é chave primária
FK	Boolean	Indica se o atributo é chave estrangeira
Comentário	Texto	Descrição textual do atributo
Unique	Boolean	É chave candidata?
Domínio_Range	Texto	Faixa de domínio
Domínio_Unidade	Texto	Unidade do domínio
Domínio_Precisão	Texto	Casas decimais
Domínio_TamanhoFixo	Texto	Quantidade de caracteres
Domínio_TamanhoMaximo	Texto	Tamanho máximo
Exemplo	Texto	Exemplo de um conteúdo do atributo

A tabela “tbKeyRef” é utilizada para armazenar, de forma temporária, as informações sobre os relacionamentos entre as entidades do esquema de exportação durante o processo de edição do esquema de exportação.

TABELA: tb tbKeyRef		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador
ID_DB	Inteiro longo	Indica a fonte de dados
pk_entidade	Inteiro longo	Indica a entidade que contém a chave primária
pk_atributo	Inteiro longo	Indica o atributo que é chave primária na tabela
fk_entidade	Inteiro longo	Indica a tabela de chave estrangeira
fk_atributo	Inteiro longo	Indica o atributo que é chave estrangeira

A tabela “tbEquivalenciasObjetos” é utilizada para armazenar, de forma temporária, as informações de equivalências entre os objetos identificados como equivalentes, durante o processo de identificação de equivalências e mapeamentos entre os esquemas de exportação.

TABELA: tbEquivalenciaObjetos		
Nome Coluna	**Tipo**	**Descrição**
ObjetoLocal	Inteiro longo	Indica a entidade local
ObjetoExterno	Inteiro longo	Indica a entidade externa
Equivalência	Texto	Tipo de equivalência
Aceitar	Boolean	Indica se a equivalência foi aceita pelo usuário.

A tabela “tbEquivalenciasAtributos” é utilizada para armazenar, de forma temporária, as informações de equivalências entre atributos daqueles objetos identificados como equivalentes, durante o processo de identificação de equivalências e mapeamentos entre os esquemas de exportação.

TABELA: tbEquivalenciaAtributos		
Nome Coluna	**Tipo**	**Descrição**
ObjetoLocal	Inteiro longo	Indica entidade local
ObjetoExterno	Inteiro longo	Indica a entidade externa
AtributoLocal	Inteiro longo	Indica o atributo local da entidade local
AtributoExterno	Inteiro longo	Indica o atributo externo da entidade externa
Equivalência	Texto	Tipo de equivalência
Aceitar	Boolean	Indica se a equivalência foi aceita pelo usuário.

A tabela “tbSinonimos” é utilizada para armazenar, de forma permanente, as informações a respeito das correspondências entre os nomes dos objetos (atributos e entidades) dos esquemas de exportação. Estas informações são atualizadas durante o processo de edição dos esquemas de exportação e também utilizadas no processo de identificação de equivalência e mapeamentos entre esquemas de exportação.

TABELA: tb tbSinonimos		
Nome Coluna	**Tipo**	**Descrição**
ID	Inteiro longo	Identificador
Nome	Texto	Nome
Sinônimo	Texto	Sinônimo para o nome

Anexo 2 XML Schema do Metamodelo XML

A seguir é apresentado o código em XML *Schema* utilizado para descrever o documento que armazena, de forma permanente, o metamodelo XML proposto neste trabalho, correspondente ao esquema de exportação gerado pelo módulo de edição e geração do esquema de exportação da ferramenta XML *Integrator*.

```
<?xml version="1.0"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
                  targetNamespace="http://192.168.0.1/metaschema">
                  xmlns:"http://192.168.0.1/metaschema"
 <xsd:element name = "dbschema"  type = "tSchema"  minOccurs = "1" maxOccurs = "*"/>

 <xsd:complexType name = "tSchema" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = "number" type = "xsd:integer" use = "required"/>
        <xsd:attribute name = "name" type = "xsd:string"  use = "required"/>
        <xsd:attribute name = "lastupdate" type = "xsd:date"  use = "required"/>
        <xsd:element name = "comment" type = "xsd:string"/>
        <xsd:element name = "aprovadopor" type = "xsd:string"/>
        <xsd:element name = "DataSource" type = "tDataSource"
                                 minOccurs="1" maxOccurs="1"/>
        <xsd:element name = "entity" type = "tEntity" minOccurs = "1" maxOccurs = "*"
                                 nullable = "false"/>
    </xsd:sequence>
 </xsd:complexType>

<xsd:complexType name = "tDataSource" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = "type"  type = "xsd:string"  use = "required"/>
        <xsd:element name = "connection"  type = "tConnection"
                                minOccurs = "1" maxOccurs = "*"  nullable = "false"/>
        <xsd:element name = "product" type = "xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
 </xsd:complexType>

<xsd:complexType name = "tConnection" content = "elementOnly">
    <xsd:sequence>
        <xsd:element name = "dsn" type = "xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name = "location" type = "tLocation" minOccurs="1" maxOccurs="*"/>
        <xsd:element name = "username" type = "xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name = "password" type = "xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name = "driver" type = "xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
 </xsd:complexType>

<xsd:complexType name = "tLocation" content = "elementOnly">
    <xsd:sequence>
        <xsd:element name = "localname" type = "xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name = "url" type = " xsd:string " minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tEntity" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = "number"  type = "xsd:integer"  use = "required"/>
```

```
        <xsd:attribute name = "localname" type = "xsd:string" use = "required"/>
        <xsd:attribute name = "esname" type = "xsd:string" use = "required"/>
        <xsd:attribute name = "parent" type = "xsd:string"/>
        <xsd:element name = "structure"  type = "xsd:string"/>
        <xsd:element name = "alias"  type = "xsd:string"/>
        <xsd:element name = "comment" type = "xsd:string"/>
        <xsd:element name = "attribute"  type = "tAttribute" minOccurs = "1" maxOccurs = "*"
                                    nullable = "false"/>
        <xsd:element name = "alias" type = "xsd:string"/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tAttribute" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = "number"  type = "xsd:integer"  use = "required"/>
        <xsd:attribute name = "localname" type = "xsd:string" use = "required"/>
        <xsd:attribute name = "esname" type = "xsd:string" use = "required"/>
        <xsd:attribute name = "sequence" type = "xsd:string" />
        <xsd:element name = "localstructure"  type = "xsd:string"/>
        <xsd:element name = "datatype"  type = "xsd:string"/>
        <xsd:element name = "maybenull"  type = "xsd:boolean"/>
        <xsd:element name = "unique" type = "xsd:boolean"/>
        <xsd:element name = "keyAttribute"  type = "xsd:boolean"/>
        <xsd:element name = "fkeyAttribute" type = "xsd:boolean"/>
        <xsd:element name = "comment"  type = "xsd:string"/>
        <xsd:element name = "domain"  type="tDomain" minOccurs="0" maxOccurs="1" />
        <xsd:element name = "keyref" type = "tKeyRef" minOccurs = "0" maxOccurs = "*" />
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tKeyRef" content = "elementOnly">
    <xsd:sequence>
        <xsd:element name = "entity"  type = "xsd:string"/>
        <xsd:element name = "attribute"  type = "xsd:string"/>
    </xsd:sequence>
 </complexType>

 <xsd:complexType name = "tDomain" content = "elementOnly">
    <xsd:sequence>
        <xsd:element name = "precision" type = "xsd:integer"/>
        <xsd:element name = "range" type = "xsd:string"/>
        <xsd:element name = "unit" type = "xsd:string"/>
        <xsd:element name = "fixedLength"  type = "xsd:boolean"/>
        <xsd:element name = "maxLength" type = "xsd:integer"/>
    </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

Anexo 3 XML Schema da Relação de Objetos Locais

A seguir é apresentado o código em XML *Schema* utilizado no documento que armazena, de forma permanente, informações sobre os objetos locais de cada esquema de exportação. Este documento é criado juntamente com o esquema de exportação através do módulo de edição e geração de esquemas de exportação da ferramenta XML *Integrator*.

```
<?xml version="1.0"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
                  targetNamespace="http://192.168.0.1/metaschema">
                  Xmlns:"http://192.168.0.1/metaschema"

 <xsd:element name = “rlo”  type = “tRlo” minOccurs = “1” maxOccurs = “*”/>

 <xsd:complexType name = "tRlo" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “number” type = “xsd:integer” use = “requered”/>
        <xsd:element name =  “schema” type = “tSchema” minOccurs = “1” maxOccurs = “*”
                                  nullable = “false”/>>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tScherma" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “number” type = “xsd:integer” use = “requered”/>
        <xsd:element name =  “object” type = “tObj” minOccurs = “1” maxOccurs = “*”
                                  nullable = “false”/>>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tObj" content = "elementOnly">
    <xsd:sequence>
         <xsd:attribute name = “ID”         type = “integer”  use = “required”/>
         <xsd:element name = “name”    type = “string”    use = “required”/>
         <xsd:element name = “structure” type = “string” use = “required”/>
         <xsd:element name = “comment” type = “string” use = “optional”/>
    </xsd:sequence>
 </xsd:complexType>

</schema>
```

Anexo 4 XML Schema da Relação de Equivalência de Objetos

A seguir é apresentado o código em XML *Schema* para o documento que armazena, de forma permanente, as informações sobre as equivalências entre os esquemas de exportação das fontes de dados participantes da federação. Este documento é criado a partir do módulo de identificação de equivalências e mapeamentos entre esquemas conceituais da ferramenta XML *Integrator*.

```
<?xml version="1.0"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
                  targetNamespace="http://192.168.0.1/metaschemaxml">
                  Xmlns:"http://192.168.0.1/metaschemaxml"

 <element name = “roe”  type = “tRoe” minOccurs = “1” maxOccurs = “*”/>

 <xsd:complexType name = "tRoe" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “number” type = “xsd:integer” use = “requered”/>
        <xsd:attribute name = “federation” type = “xsd:integer” use = “requered”/>
        <xsd:attribute name = “creation” type = “xsd:date” use = “requered”/>
        <xsd:element name =  “object” type = “tObj” minOccurs = “1” maxOccurs = “*”
                                nullable = “false”/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tObj" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “ID” type = “integer”  use = “required”/>
        <xsd:element name = “baseentity” type = “string”/>
        <xsd:element name = “externalentity” type = “string”/>
        <xsd:element name = “equivalence” type = “string”/>
        <xsd:element name = “constraint” type = “string”/>
    </xsd:sequence>
 </xsd:complexType>

</Schema>
```

Anexo 5 XML Schema da Relação de Equivalência de Atributos

A seguir é apresentado o código em XML *Schema* do documento que armazena, de forma permanente, as informações sobre as equivalências de representação entre os atributos dos objetos dos esquemas de exportação. Este documento é criado a partir do módulo de identificação de equivalências e mapeamentos entre esquemas conceituais da ferramenta XML *Integrator*.

```
<?xml version="1.0"?>
<xsd:schema
      xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
                  targetNamespace="http://192.168.0.1/metaschemaxml">
                  Xmlns:"http://192.168.0.1/metaschemaxml"

 <element name = “roe”  type = “tRae” minOccurs = “1” maxOccurs = “*”/>

 <xsd:complexType name = "tRae" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “number” type = “xsd:integer” use = “requered”/>
        <xsd:attribute name = “federation” type = “xsd:integer” use = “requered”/>
        <xsd:attribute name = “creation” type = “xsd:date” use = “requered”/>
        <xsd:element name =  “object” type = “tObj” minOccurs = “1” maxOccurs = “*”
                                nullable = “false”/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tObj" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “ID” type = “integer”  use = “required”/>
        <xsd:attribute name = “baseentity” type = “string”  use = “required”/>
        <xsd:attribute name = “baseattribute” type = “string” use = “required”/>
        <xsd:element name = “equivalence” type = “tEquivalence” minOccurs=“1”
                                maxOccurs=“*”  nullable = “false”/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name = "tEquivalence" content = "elementOnly">
    <xsd:sequence>
        <xsd:attribute name = “number” type = “integer”  use = “required”/>
        <xsd:element name = “externalentity” type = “string”/>
        <xsd:element name = “externalattribute” type = “string”/>
        <xsd:element name = “type” type = “string”/>
    </xsd:sequence>
 </xsd:complexType>
</Schema>
```

Anexo 6 Código do *Wrapper* Estruturado

A seguir é apresentado o código em Visual Basic do *wrapper* desenvolvido para a extração do esquema conceitual local das fontes de dados estruturadas.

```
Private Sub btnConcluir_Click()
‘Código para fazer a conexão com o banco de dados
   Dim sConnect    As String
   Dim sADOConnect As String
   Dim sDSN        As String

   frmPrincipal.StatusBar.Panels(1).Text = "Wait Database connecting ..."

   If cboDSNList.ListIndex > 0 Then
      sDSN = "Data Source=" & cboDSNList.Text & ";"
   Else
      sConnect = sConnect & "Driver=" & cboDrivers.Text & ";"
      sConnect = sConnect & "Server=" & txtServer.Text & ";"
   End If

   sConnect = sConnect & ";User Id=" & txtUID.Text & ";"
   sConnect = sConnect & "Password=" & txtPWD.Text & ";"

   If Len(txtDatabase.Text) > 0 Then
      sConnect = sConnect & "Database=" & txtDatabase.Text & ";"
   End If

   'sADOConnect = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=" & txtDatabase.Text &
sConnect

'Conexão ADO:
   Set gConnection = New ADODB.Connection
   gConnection.Open sADOConnect

   frmPrincipal.StatusBar.Panels(1).Text = "Connect Ok... Wait object databases importing"
   ImportaBancoDeDados
   Unload frmNovoComponente
   Unload Me
   frmDefinicaoEE.Show
End Sub

Private Sub ImportaBancoDeDados()
‘Código que faz a importação do dicionário de dados de um banco de dados
  Dim codObj      As Long
  Dim criterio    As String
  Dim seqAttr     As Integer
  Dim strCriterio As String
  Dim qdf         As DAO.QueryDef

  If EventFlag = conNewEsquema Then

    With tbEsquema
      .AddNew
      setCodigoEsquema (!ID)
      !nomeEsquema = frmImportaEsquemaBD.txtNomeEsquema
      !Path = frmImportaEsquemaBD.txtDatabase
      !Uid = frmImportaEsquemaBD.txtUID
```

```
    !Pwd = frmImportaEsquemaBD.txtPWD
    !DSN = frmImportaEsquemaBD.cboDSNList
    !Server = frmImportaEsquemaBD.txtServer
    !Driver = frmImportaEsquemaBD.cboDrivers
    !Descricao = Null
    !TipoEsquema = "BD"
    .Update
  End With
ElseIf EventFlag = conOpenEsquema Then
  setCodigoEsquema (Val(frmOpenComponente.listEsquemas.SelectedItem.Text))
Else
  MsgBox "Error"
  Exit Sub
End If

Set rstSchematable = gConnection.OpenSchema(adSchemaTables)
Set rstSchemaColumns = gConnection.OpenSchema(adSchemaColumns)
Set rstSchemaForeignKey = gConnection.OpenSchema(adSchemaForeignKeys)
Set rstSchemaIndexes = gConnection.OpenSchema(adSchemaIndexes)
Set rstSchemaPrimaryKey = gConnection.OpenSchema(adSchemaPrimaryKeys)

'adiciona registros na tabela de chaves estrangeiras
If Not rstSchemaForeignKey.EOF Then
  rstSchemaForeignKey.MoveFirst

  Do Until rstSchemaForeignKey.EOF
    tbFK.AddNew
    tbFK!ID_DB = getCodigoEsquema
    tbFK!PK_TABLE_NAME = rstSchemaForeignKey!PK_TABLE_NAME
    tbFK!PK_COLUMN_NAME = rstSchemaForeignKey!PK_COLUMN_NAME
    tbFK!fk_table_name = rstSchemaForeignKey!fk_table_name
    tbFK!fk_column_name = rstSchemaForeignKey!fk_column_name
    tbFK.Update
    rstSchemaForeignKey.MoveNext
  Loop
End If

'adiciona registros na tabela de chaves primaria
If Not rstSchemaPrimaryKey.EOF Then
  rstSchemaPrimaryKey.MoveFirst

  Do Until rstSchemaPrimaryKey.EOF
    tbPK.AddNew
    tbPK!ID_DB = getCodigoEsquema
    tbPK!TABLE_NAME = rstSchemaPrimaryKey!TABLE_NAME
    tbPK!COLUMN_NAME = rstSchemaPrimaryKey!COLUMN_NAME
    tbPK!ORDINAL = rstSchemaPrimaryKey!ORDINAL
    tbPK.Update
    rstSchemaPrimaryKey.MoveNext
  Loop
End If

'adiciona registros nas tabelas de Entidades e Colunas
Do Until rstSchematable.EOF
    tbEntidades.AddNew
    codObj = tbEntidades!ID
    tbEntidades!CodigoEsquema = getCodigoEsquema
    tbEntidades!nomeLocal = rstSchematable!TABLE_NAME
    tbEntidades!nomeEE = rstSchematable!TABLE_NAME
    tbEntidades!Estrutura = StructuredType(rstSchematable!TABLE_TYPE)
```

```
    tbEntidades!Comentarios = rstSchematable!Description
    tbEntidades.Update

    rstSchemaColumns.MoveFirst
    seqAttr = 0
    Do Until rstSchemaColumns.EOF
      If rstSchemaColumns!TABLE_NAME = rstSchematable!TABLE_NAME Then
        tbAtributos.AddNew
        tbAtributos!IDEntidade = codObj
        tbAtributos!nomeLocal = rstSchemaColumns!COLUMN_NAME
        tbAtributos!nomeEE = rstSchemaColumns!COLUMN_NAME
        tbAtributos!EstruturaLocal = "A"
        seqAttr = seqAttr + 1
        tbAtributos!Sequencia = seqAttr
        tbAtributos!Tipo = FieldType(rstSchemaColumns!DATA_TYPE)
        tbAtributos!MayBeNull = rstSchemaColumns!IS_NULLABLE

        '**** verifica se o atributo atual é chave primária
        tbAtributos!PK = False
        With tbPK
          .Requery
          If Not .EOF Then
            .MoveFirst
            strCriterio = "ID_DB =" & getCodigoEsquema & " AND TABLE_NAME = " & Chr(39) &
CStr(rstSchematable!TABLE_NAME) & Chr(39) & " AND COLUMN_NAME = " & Chr(39) &
CStr(rstSchemaColumns!COLUMN_NAME) & Chr(39)
            .FindFirst strCriterio
            If Not .NoMatch Then
              tbAtributos!PK = True
            End If
            .MoveFirst
          End If
         End With

        '**** verifica se o atributo atual é chave estrangeira
        tbAtributos!FK = False
        With tbFK
          .Requery
          If Not .EOF Then
            .MoveFirst
            strCriterio = "ID_DB =" & getCodigoEsquema & " AND FK_TABLE_NAME = " & Chr(39) &
CStr(rstSchematable!TABLE_NAME) & Chr(39) & " AND FK_COLUMN_NAME = " & Chr(39) &
CStr(rstSchemaColumns!COLUMN_NAME) & Chr(39)
            .FindFirst strCriterio
            If Not .NoMatch Then
              tbAtributos!FK = True
            End If
            .MoveFirst
          End If
        End With

        tbAtributos!Comentario = rstSchemaColumns!Description
        tbAtributos!Unique = False
        tbAtributos!Dominio_Range = Null
        tbAtributos!Dominio_Unidade = Null
        tbAtributos!Dominio_Precisao = rstSchemaColumns!NUMERIC_SCALE
        tbAtributos!Dominio_TamanhoFixo = rstSchemaColumns!NUMERIC_PRECISION
        tbAtributos!Dominio_TamanhoMaximo =
rstSchemaColumns!CHARACTER_MAXIMUM_LENGTH
        tbAtributos.Update
```

```
      End If

      rstSchemaColumns.MoveNext
    Loop
    rstSchematable.MoveNext
  Loop

  'Adiciona referencias das entidades em tbKeyRef
  Set qdf = BaseRegistro.QueryDefs("AdicionaKeyRefs")
  qdf!CodEsq = getCodigoEsquema
  qdf.Execute

  frmPrincipal.StatusBar.Panels(1).Text = "Databases importing finish with success."

End Sub

Function FieldType(intType As Integer) As String
'retorna o tipo do dado conforme DataTypeEnum

   Select Case intType
      Case adBoolean
         FieldType = "Boolean"
      Case adUnsignedTinyInt, adBinary, adTinyInt
         FieldType = "Byte"
      Case adSmallInt, adInteger, _
          adUnsignedInt, adUnsignedSmallInt
         FieldType = "Integer"
      Case adBigInt, adUnsignedBigInt
         FieldType = "Long"
      Case adCurrency
         FieldType = "Currency"
      Case adSingle, adDecimal, adNumeric
         FieldType = "Single"
      Case adDouble
         FieldType = "Double"
      Case adDate, adDBDate
         FieldType = "Date"
      Case adDBTime
         FieldType = "Time"
      Case adDBTimeStamp
         FieldType = "Date/Time"
      Case adVarWChar, adChar, adWChar
         FieldType = "Text"
      Case adLongVarWChar
         FieldType = "Memo"
      Case adGUID
         FieldType = "GUID"
      Case adEmpty
         FieldType = "Empty"
      Case adVariant
         FieldType = "Variant"
      Case Else
         FieldType = intType  '"undefined"
   End Select
```

Anexo 7 Código do *Wrapper* Semi-Estruturado

A seguir é apresentado o código em Visual Basic do *wrapper* aplicado às fontes de dados semi-estruturadas, o qual estabelece o acesso ao documento XML e extrai seu esquema conceitual.

```
Private Sub extraiEsquemaDoc()
‘realiza a analise e extração do esquema de uma instância de um documento XML
  Dim objRoot   As IXMLDOMElement
  Dim i, j      As Integer
  Dim cod_ent   As Long
  Dim strSQL, strCriterio As String

  frmPrincipal.StatusBar.Panels(1).Text = "Wait... document analyzing!"
  ent_count = 0
  Set objRoot = XMLParser.documentElement

  'cria a primeira entidade com o nome do elemento raiz
  i = XMLParser.getElementsByTagName("*").length
  frmPrincipal.ProgressBar.Visible = True
  frmPrincipal.ProgressBar.Max = i + 1
  frmPrincipal.ProgressBar.Value = 1

  ReDim entidades(i + 1)
  'coluna 1 = Nro da Entidade
  'coluna 2 = Nome do Atributo
  'coluna 3 = estrutura original (elemento ou atributo)
  'coluna 4 = conteúdo de exemplo
  ReDim atributos(i + 1, 2)

  cod_ent = criaEntidade(objRoot.nodeName)
  If cod_ent = 0 Then
    MsgBox "Error creating first entity (root element): " & objRoot.nodeName, vbInformation
  End If

  'variáveis de controle da recursão
  depth = 1
  depthList(depth) = cod_ent

  i = verificaNode(objRoot)

  frmPrincipal.ProgressBar.Max = ent_count + 1
  frmPrincipal.ProgressBar.Value = 1
  frmPrincipal.StatusBar.Panels(1).Text = "Wait... verifying and optimizing entities identifiers!"

  'elimina entidades e atributos duplicados através de em uma 'instrução SQL.
  ‘Logo após realiza um laço para a 'criação do esquema com suas respectivas entidades

  If Not criaEsquema() Then
    MsgBox “The schema is not created. Verify....!", vbCritical
    Exit Sub
  End If

  'conjunto de entidades
  strSQL = "SELECT DISTINCT tbEntidadesTemp.NomeEntidade FROM tbEntidadesTemp"
  Set tbEnt = BaseRegistro.OpenRecordset(strSQL)
```

```
  'conjunto de entidades com atributos
  strSQL = "SELECT DISTINCT tbEntidadesTemp.NomeEntidade, tbAtributosTemp.NomeAtributo,
tbAtributosTemp.Estrutura " & _
           "FROM tbEntidadesTemp, tbAtributosTemp Where tbEntidadesTemp.ID =
tbAtributosTemp.IDEntidade"
  Set tbAttr = BaseRegistro.OpenRecordset(strSQL)

  Do While Not tbEnt.EOF
    tbAttr.MoveFirst
    strCriterio = "NomeEntidade = " & Chr(34) & tbEnt!nomeEntidade & Chr(34)
    tbAttr.FindFirst strCriterio
    If Not tbAttr.NoMatch Then
       seq = 0
       tbEntidades.AddNew
         cod_ent = tbEntidades!ID
         tbEntidades!CodigoEsquema = getCodigoEsquema()
         tbEntidades!nomeLocal = tbEnt!nomeEntidade
         tbEntidades!nomeEE = tbEnt!nomeEntidade
         tbEntidades!Estrutura = "TABELA"
       tbEntidades.Update
    End If
    Do While Not tbAttr.NoMatch
      tbAtributos.AddNew
       tbAtributos!IDEntidade = cod_ent
       seq = seq + 1
       tbAtributos!Sequencia = seq
       tbAtributos!nomeLocal = tbAttr!nomeAtributo
       tbAtributos!nomeEE = tbAttr!nomeAtributo
       tbAtributos!Tipo = "Text"
       tbAtributos!EstruturaLocal = tbAttr!Estrutura
      tbAtributos.Update
      tbAttr.FindNext strCriterio
    Loop
    tbEnt.MoveNext
  Loop

  frmPrincipal.ProgressBar.Visible = False
  frmPrincipal.StatusBar.Panels(1).Text = "Done. Schema create with sucess."

End Sub

Function verificaNode(nodeAux)
  Dim objChild    As IXMLDOMNode
  Dim attrib      As IXMLDOMAttribute
  Dim i, j        As Integer
  Dim flag        As Boolean
  Dim current     As Integer
  Dim cod_ent     As Long

  For current = 0 To (nodeAux.childNodes.length - 1)

    frmPrincipal.ProgressBar.Value = frmPrincipal.ProgressBar.Value + 1
    Set objChild = nodeAux.childNodes.Item(current)

    If objChild.nodeType = NODE_ELEMENT Then
      'na recursão se o elemento possuir atributos, deverá ser
      'criado um atributo ref para a entidade pai(depth-1)
      If (objChild.Attributes.length > 0) Then
        cod_ent = criaEntidade(objChild.nodeName)
        If cod_ent > 0 Then
```

```
          For i = 0 To objChild.Attributes.length - 1
            Set attrib = objChild.Attributes.Item(i)
            If criaAtributos(cod_ent, attrib.Name, "A", attrib.Text) = False Then
              MsgBox "Error creating attribute of the entity: " & objChild.nodeName, vbInformation
            End If
          Next
        Else
          MsgBox "Error creating entity: " & objChild.nodeName, vbInformation
        End If
      End If

      If (objChild.hasChildNodes) Then
        If objChild.childNodes.nextNode.nodeType <> NODE_TEXT Then
          If objChild.Attributes.length = 0 Then
            cod_ent = criaEntidade(objChild.nodeName)
            If cod_ent = 0 Then
              MsgBox "Error creating entity: " & objChild.nodeName, vbInformation
            End If
          End If
          'controle da recursão
          depth = depth + 1
          depthList(depth) = cod_ent
          i = verificaNode(objChild)
          depth = depth - 1
        Else
          If Not criaAtributos(depthList(depth), objChild.nodeName, "E",
objChild.childNodes.nextNode.Text) Then
            MsgBox "Error creating attribute of the entity: " & objChild.parentNode.nodeName,
vbInformation
          End If
        End If
      Else
        If Not criaAtributos(depthList(depth), objChild.nodeName, "E", objChild.Text) Then
          MsgBox "Error creating attribute of the entity: " & objChild.parentNode.nodeName,
vbInformation
        End If
      End If
    End If
  Next

End Function

Function criaEsquema() As Boolean
‘adiciona um esquema novo no banco de dados dbRegistry
  criaEsquema = False
  tbEsquema.AddNew
    setCodigoEsquema (tbEsquema!ID)
    tbEsquema!nomeEsquema = frmImportaEsquemaDoc.txtNomeEsquema
    tbEsquema!Path = frmImportaEsquemaDoc.txtlocalizacao
    tbEsquema!TipoEsquema = "DOC"
  tbEsquema.Update
  criaEsquema = True

End Function

Function criaEntidade(nomeEntidade As String) As Integer
‘adiciona uma nova entidade ao esquema criado
  criaEntidade = 0
  tbEnt.AddNew
    tbEnt!nomeEntidade = nomeEntidade
```

```
   criaEntidade = tbEnt!ID
  tbEnt.Update
  seq = 0
  ent_count = ent_count + 1

End Function

Function criaAtributos(nroEntidade, nomeAtributo As String, Estrutura As String, Dado As
String) As Boolean
‘adiciona uma novo atributo a uma entidade
 Static attr_count As Integer

 tbAttr.AddNew
  tbAttr!IDEntidade = nroEntidade
  tbAttr!nomeAtributo = nomeAtributo
  tbAttr!Tipo = "Text"
  tbAttr!Estrutura = Estrutura
  tbAttr!Exemplo = Dado
 tbAttr.Update

 attr_count = attr_count + 1
 criaAtributos = True

End Function
```

Anexo 8 Estudo de Caso: Esquema de exportação e Relação de Objetos Locais do banco de dados “obgyn”

A seguir é apresentado o documentos XML utilizado para o armazenamento do esquema de exportação do banco de dados “Obgyn” utilizado como exemplo no capítulo 7.

```
<?xml version = "1.0" encoding="ASCII" ?>
<dbschema number="1" name="OBGYN" lastupdate="10/10/01" version="1.0"
 xmlns:xsi='http://www.w3.org/2000/10/XMLSchema-instance'
 xsi:noNamespaceSchemaLocation=' http://192.168.0.1/ MetaSchemaXML.xsd '>
 <comment>This database includes information about obstetrics patients. </comment>
 <aprovadopor>Everaldo</aprovadopor>
 <datasource type="DB">
  <product>MS Access 2000</product>
  <connection>
   <location>
    <localname>obgyn.mdb</localname>
    <url>http://192.168.0.159/database/obgyn.mdb</url>
   </location>
  </connection>
 </datasource>
 <entity number="1" localname="PATIENTS" esname="PATIENTS" parent="">
      <structure>TABLE</structure>
      <alias>P</alias>
      <attribute number="1" localname="PATIENT_ID" esname="PATIENT_ID" sequence="1">
       <localstructure>A</localstructure>
       <datatype>Integer</datatype>
       <maybenull>Falso</maybenull>
       <unique>Verdadeiro</unique>
       <keyattribute>Verdadeiro</keyattribute>
       <fkeyattribute>Falso</fkeyattribute>
       <comment>Identifier of the patients.</comment>
       <domain>
        <range>00001-99999</range>
        <maxlength>5</maxlength>
       </domain>
       </attribute>
      <attribute number="2" localname="NAME" esname="PATIENTNAME" sequence="2">
       <localstructure>A</localstructure>
       <datatype>Text</datatype>
       <maybenull>Falso</maybenull>
       <unique>Falso</unique>
       <keyattribute>Falso</keyattribute>
       <fkeyattribute>Falso</fkeyattribute>
       <comment></comment>
       </attribute>
      <attribute number="3" localname="DATAOFBIRTH" esname="DATAOFBIRTH"
sequence="3">
       <localstructure>A</localstructure>
       <datatype>Date</datatype>
       <maybenull>Verdadeiro</maybenull>
       <unique>Falso</unique>
       <keyattribute>Falso</keyattribute>
       <fkeyattribute>Falso</fkeyattribute>
       <comment></comment>
      </attribute>
      <attribute number="4" localname="GENDER" esname="GENDER" sequence="4">
```

```
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <comment>OBSTETRICS PATIENTS ONLY GENDER FEMALE</comment>
        <domain>
          <range>F</range>
          <fixedlength>1</fixedlength>
          <maxlength>1</maxlength>
        </domain>
        </attribute>
      <attribute number="5" localname="ADMISSION" esname="ADMISSION" sequence="5">
        <localstructure>A</localstructure>
        <datatype>Date</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <comment></comment>
      </attribute>
      <attribute number="62" localname="DISCHARGE" esname="DISCHARGE" sequence="6">
        <localstructure>A</localstructure>
        <datatype>Date</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
  </entity>
  <entity number="2" localname="LAB_TESTS" esname="TESTS" parent="">
      <structure>TABLE</structure>
      <alias>T</alias>
      <attribute number="6" localname="TEST_ID" esname="TEST_ID" sequence="1">
        <localstructure>A</localstructure>
        <datatype>Integer</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Verdadeiro</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="7" localname="TEST_NAME" esname="TEST_NAME" sequence="2">
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <domain>
          <maxlength>50</maxlength>
        </domain>
        </attribute>
      <attribute number="8" localname="TEST_COD" esname="TEST_COD" sequence="3">
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
```

```
        <domain>
          <maxlength>20</maxlength>
        </domain>
        </attribute>
       <attribute number="9" localname="LAB_NAME" esname="LAB_NAME" sequence="4">
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <domain>
          <maxlength>30</maxlength>
        </domain>
       </attribute>
  </entity>
  <entity number="9" localname="LABTESTRESULTS" esname="RESULTS" parent="">
       <structure>RELATIONSHIP</structure>
       <alias>R</alias>
       <attribute number="18" localname="PATIENT_ID" esname="PATIENT_ID" sequence="1">
        <localstructure>A</localstructure>
        <datatype>Integer</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Verdadeiro</keyattribute>
        <fkeyattribute>Verdadeiro</fkeyattribute>
       </attribute>
       <attribute number="19" localname="TEST_ID" esname="TEST_ID" sequence="2">
        <localstructure>A</localstructure>
        <datatype>Integer</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Verdadeiro</keyattribute>
        <fkeyattribute>Verdadeiro</fkeyattribute>
       </attribute>
       <attribute number="20" localname="DATEOFTEST" esname="DATEOFTEST"
sequence="3">
        <localstructure>A</localstructure>
        <datatype>Date</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
       </attribute>
       <attribute number="21" localname="HOUROFTEST" esname="HOUROFTEST"
sequence="4">
        <localstructure>A</localstructure>
        <datatype>Time</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <comment></comment>
       </attribute>
       <attribute number="22" localname="FINDINGS" esname="FINDINGS" sequence="5">
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
```

```
        <fkeyattribute>Falso</fkeyattribute>
        <domain>
          <maxlength>255</maxlength>
        </domain>
      </attribute>
      <attribute number="23" localname="SENSITIVITY" esname="SENSITIVITY" sequence="6">
        <localstructure>A</localstructure>
        <datatype>Text</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
        <domain>
          <maxlength>50</maxlength>
        </domain>
      </attribute>
  </entity>
</dbschema>
```

A seguir é apresentado o documento XML utilizado para o armazenamento da relação dos objetos locais do banco de dados “Obgyn” utilizado como exemplo no capítulo 7.

```
<?xml version = "1.0" encoding="ASCII" ?>
<rlo number="001"
  xmlns:xsi='http://www.w3.org/2000/10/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='http://192.168.0.1/rlo.xsd'>
   <schema number=”0001”>
    <object id="1">
         <name>PATIENTS</name>
         <structure>TABLE</structure>
    </object>
    <object id="2">
         <name>TESTS</name>
         <structure>TABLE</structure>
    </object>
    <object id="9">
         <name>RESULTS</name>
         <structure>RELATIONSHIP</structure>
    </object>
   </schema>
</rlo>
```

Anexo 9 Estudo de Caso: Esquema de Exportação e Relação de Objetos Locais do Documento XML "MedicalHistory"

A seguir é apresentado o documento XML para o armazenamento do esquema de exportação do documento XML "MedicalHistory" utilizado como exemplo no capítulo 7.

```
<?xml version = "1.0" encoding="ASCII" ?>
<dbschema number="2" name="MedicalHistory" lastupdate="10/10/01" version="1.0"
 xmlns:xsi='http://www.w3.org/2000/10/XMLSchema-instance'
 xsi:noNamespaceSchemaLocation='http://192.168.0.1/metaschema.xsd'>
 <aprovadopor>Everaldo</aprovadopor>
 <datasource type="DOC">
  <product>XML</product>
  <connection>
   <location>
    <localname>medicalhistory.xml</localname>
    <url>http://192.168.0.160/data/medicalhistory.xml</url>
   </location>
  </connection>
 </datasource>
 <entity number="12" localname="HOSPITAL" esname="HOSPITAL" parent="">
     <structure>TABLE</structure>
     <attribute number="51" localname="NAME" esname="HOSPITALNAME" sequence="1">
      <localstructure>A</localstructure>
      <datatype>Text</datatype>
      <maybenull>Falso</maybenull>
      <unique>Falso</unique>
      <keyattribute>Verdadeiro</keyattribute>
      <fkeyattribute>Falso</fkeyattribute>
     </attribute>
 </entity>
 <entity number="13" localname="PATIENT" esname="PATIENT" parent="HOSPITAL">
     <structure>TABLE</structure>
     <attribute number="52" localname="PATIENTID" esname="PATIENT_ID" sequence="1">
      <localstructure>A</localstructure>          <datatype>Integer</datatype>
      <maybenull>Falso</maybenull>
      <unique>Verdadeiro</unique>
      <keyattribute>Falso</keyattribute>
      <fkeyattribute>Falso</fkeyattribute>
     </attribute>
     <attribute number="55" localname="NAME" esname="NAME" sequence="2">
      <localstructure>A</localstructure>
      <datatype>Text</datatype>
      <maybenull>Falso</maybenull>
      <unique>Falso</unique>
      <keyattribute>Falso</keyattribute>
      <fkeyattribute>Falso</fkeyattribute>
     </attribute>
     <attribute number="56" localname="MAINCOMPLAIN" esname="MAINCOMPLAIN"
sequence="4">
      <localstructure>E</localstructure>
      <datatype>TEXT</datatype>
      <maybenull>Verdadeiro</maybenull>
      <unique>Falso</unique>
      <keyattribute>Falso</keyattribute>
```

```
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="57" localname="CURRENTMEDICALHISTORY"
esname="CURRENTMEDICALHISTORY" sequence="5">
        <localstructure>E</localstructure>
        <datatype>TEXT</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="58" localname="PASTMEDICALHISTORY"
esname="PASTMEDICALHISTORY" sequence="6">
        <localstructure>E</localstructure>
        <datatype>TEXT</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="59" localname="REVIEWOFSYSTEM" esname="REVIEWOFSYSTEM"
sequence="7">
        <localstructure>E</localstructure>
        <datatype>TEXT</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="60" localname="PSYCHOSOCIALPROFILE"
esname="PSYCHOSOCIALPROFILE" sequence="8">
        <localstructure>E</localstructure>          <datatype>TEXT</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="61" localname="FAMILYHISTORY" esname="FAMILYHISTORY"
sequence="9">
        <localstructure>E</localstructure>
        <datatype>TEXT</datatype>
        <maybenull>Verdadeiro</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
      <attribute number="64" localname="WARD" esname="WARD" sequence="3">
        <localstructure>E</localstructure>
        <datatype>TEXT</datatype>
        <maybenull>Falso</maybenull>
        <unique>Falso</unique>
        <keyattribute>Falso</keyattribute>
        <fkeyattribute>Falso</fkeyattribute>
      </attribute>
  </entity>
</dbschema>
```

A seguir é apresentado o documento XML para o armazenamento da relação de objetos locais do documento XML “MedicalHistory” utilizado como exemplo no capítulo 7.

```
<?xml version = "1.0" encoding="ASCII" ?>
<rlo number="001"
  xmlns:xsi='http://www.w3.org/2000/10/XMLSchema-instance'
  xsi:noNamespaceSchemaLocation='http://192.168.0.1/rlo.xsd'>
   <schema number=”0001”>
    <object id="1">
         <name>PATIENTS</name>
         <structure>TABLE</structure>
    </object>
    <object id="2">
         <name>TESTS</name>
         <structure>TABLE</structure>
    </object>
    <object id="3">
         <name>RESULTS</name>
         <structure>RELATIONSHIP</structure>
    </object>
   </schema>
   <schema number=”0002”>
    <object id="4">
         <name>HOSPITAL</name>
         <structure>TABLE</structure>
    </object>
    <object id="5">
         <name>PATIENT</name>
         <structure>TABLE</structure>
    </object>
  </schema>
</rlo>
```

Bibliografia

[ABI 97] ABITEBOUL, S. Querying semi-structured data. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, ICDT, 6., Delphi, Greece. **Database Theory:** proceedings. Berlin:Springer-Verlag, 1997. p.1-18. (Lecture Notes in Computer Science, v.1186).

[AHM 91] AHMED, R. et al. The Pegasus heterogenous multidatabase system. **IEEE Computer**, Los Alamitos, v.24, n.12, p.19-27, Dec. 1991.

[BAR 99] BARU, C.K. et al. XML-Based Information Mediation with MIX. **SIGMOD Record**, New York, v.28, n.2, June 1999. Trabalho apresentado na ACM SIGMOD International Conference On Management of Data, 1999.

[BAT 86] BATINI, C.; LENZERINI, M.; NAVATHE, S. A Comparative Analysis of Methodologies for Database Schema Integration. **ACM Computing Surveys**, New York, v.18, n.4, p.323-364, Dec.1986.

[BAX 2000] BAX, M.P. **Introdução às Linguagens de Marcas**. 2000. Disponível em: <http://www.paradigma.com.br/XML/introxml.html>. Acesso em: ago. 2000.

[BER 2000] BERNSTEIN, P.A.; LEVY, A.Y.; POTTINGER, R. A Vision for Management of Complex Models. **SIGMOD Record**, New York, v.29, n.4, p.55-63, 2000.

[BRI 92] BRIGHT, M.; HURSON, A.; PAKZAD, S. A taxonomy and current issues in multidatabase systems. **Computer**, New York, v.25, n.3, p.50-62, 1992.

[BIR 2001] BIRON, P. V.; MALHOTRA, A. **XML-Schema Part 2: Datatypes**. W3C, May 2001. Disponível em: <http://www.w3.org/TR/xmlschema-2/>. Acesso em: jan. 2002.

[BOS 98] BOSAK, J. et al. **W3C XML Specification DTD.** W3C, June 1998. Disponível em: <http://www.w3.org/XML/1998/06/xmlspec-report>. Acesso em: jan. 2002.

[BOU 2001] BOURRET, R. **XML and Databases**. Disponível em: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>. Acesso em: set. 2001.

[BRA 2000] BRAY, T. et al. **Extensible Markup Language (XML) 1.0**. W3C, October 2000. Disponível em: <http://www.w3.org/TR/2000/REC-xml-20001006>. Acesso em: jan. 2002.

[BRI 2000] BRICKLEY, D.; GUHA, R. V. (Ed.). **Resource Description Framework (RDF) Schema Specification 1.0**. W3C, March 2000. Disponível em: <http://www.w3.org/TR/2000/cr-rdf-schema-20000327>. Acesso em: jan. 2002.

[BUN 95] BUNEMAN, P. et al. Programming Constructs for Unstructured Data. In: INTERNATIONAL WORKSHOP ON DATABASE PROGRAMMING LANGUAGES, DBPL, 1995. **Database Programming Languages**: proceedings. Berlin: Springer-Verlag, [1995].

[BUN 96] BUNEMAN P. et al. A query Language and Optimization Techniques for Unstructured Data. **SIGMOD Record**, New York, v.25, n.2, p.505-516, June, 1996. Trabalho apresentado na 6ª ACM SIGMOD International Conference on Management of Data, 1996, Montreal.

[BUN 97] BUNEMAN, P. Tutorial: Semi-structured data. In: ACM SIGACT-SIGMOD-SIGART SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS, PODS, 16.,1997. **Proceedings...** [S.l.:s.n.], 1997.

[BUS 99] BUSSE, S. et al. **Federated Information Systems:** Concepts, Terminology and Architectures. Berlin:Technische Universitat Berlin, 1999. (Technical Report 99-9).

[CAR 95] CAREY, M.J.et al. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In: WORKSHOP ON RESEARCH ISSUES IN DATA ENGINEERING-DISTRIBUTED OBJECT MANAGEMENT, RIDE-DOM'95, 5., 1995. **Proceedings…** [S.l.:s.n.], 1995.

[CAR 2000] CAREY, M.J. et al. XPERANTO: Publishing Object-Relational Data as XML. In: INTERNATIONAL WORKSHOP ON THE WEB AND DATABASES, 2000. **Proceedings…** [S.l.:s.n.], 2000.

[CAT 94] CATTELL, R. G. G. **Object Data Management:** Object-Oriented and Extended Relational Database Systems. Rev. Ed. Reading, MA:Addison-Wesley, 1994. 82p.

[CHA 91] CHATTERJEE, A.; SEGEV, A. Data Manipulation in Heterogeneous Databases. **SIGMOD Record**, New York, v.20, n.4, p.64-68, Dec. 1991.

[COD 82] CODD, E.F. Relational Databases: A Practical Foundation for Productivity. **Communications of the ACM**, New York, v.25, n.2, p.109-117, Feb. 1982.

[CRI 2000] CRISTOPHIDES, V. et al. On wrapping query languages and efficient XML integration. **SIGMOD Record**, New York, v.29, n.2, p.141-142, May 2000. Trabalho apresentado na 18ª ACM SIGMOD International Conference On Management of Data, 2000, Dallas.

[DAR 2002] DARONCO, E.L.; RIBEIRO, C.H.F.P. XML Integrator: extending heterogeneous databases interoperability to XML documents. In: EDBT, 2002, Prague. **XML-Based Data Management.** [S.l.:s.n.], 2002. p. 152-161.

[DAV 99] DAVIDSON, A. et al. **Schema for Object-Oriented XML 2.0**. W3C, July 1999. Disponível em: <http://www.w3.org/TR/note-SOX>. Acesso em: jan. 2002.

[DOM 98] DOCUMENT Object Model (DOM) Level 1 Specification 1998. Disponível em: <http://www.w3.org/TR/PR-DOM-Level-1/>. Acesso em: jan. 2002.

[DOR 2000] DORNELLES, C. **Extração de dados Semi-estruturados com base em uma ontologia**. 2000. 88p. Dissertação (Mestrado em ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[ELM 99] ELMAGARMID, A.; RUSENKIEWICZ, M.; SHETH, A. (Ed.). **Management of Heterogeneous and Autonomous Database Systems**. San Francisco: Morgan Kaufmann, 1999. 413p.

[FLO 99] FLORESCU, D.; KOSSMANN, D. Storing and Querying XML Data using an RDMBS. **Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**, [S.l.], v.22, n.3, p.27-34, 1999.

[GUA 2002] GUANDELINE, E. L. T. **Integração Materializada na Web:** um estudo de caso. 2002. 90p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática – UFRGS, Porto Alegre.

[GAR 2000] GAROFALAKIS, M. et al. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. **SIGMOD Record**, New York, v.29, n.2, p.165-176, May 2000. Trabalho apresentado na 18ª ACM SIGMOD International Conference On Management of Data, 2000, Dallas.

[HAS 99] HAAS, L.M. et al. **Transforming Heterogeneous Data with Database Middleware:** beyond Integration. [S.l.:s.n.],1999

[HEU 2000] HEUSER, C.A. **Projeto de Banco de Dados**. 3. ed. Porto Alegre: Sagra Luzzatto, 2000.

[HOH 96] HOHENSTEIN, U.; PLESSER,V. Semantic Enrichment: a First Step to provide Database Interoperability. In: WORKSHOP FÖDERIERTE DATENBANKEN, Magdeburg, pp. 3-17, 1996.

[JEL 2000] JELLIFFE, R. **Schematron.** May 2000. Disponível em: <http://www.ascc.net/xml/resource/schematron/>. Acesso em: jan. 2002.

[KAN 2000] KANTORSKI, G. Z.; RIBEIRO, C. H. F. P. Heterogeneous Database Interoperability the WWW. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 15., 2000, João Pessoa. **Anais...** João Pessoa: PUCRS, 2000.

[KAS 95] KASHYAP, V; SHETH, A. Semantic and Schematic Similarities between Database Objects: A Context-based approach. **VLDB Journal**, Heidelberg, v.5, n.4, p.276-304, 1995.

[KIM 93] KIM, W. Object Oriented Database systems: promises, reality and future. In: VLDB, 19., 1993, Dublin, Ireland **Proceedings…** San Francisco: Morgan Kaufmann, 1993. p.676-692.

[KLA 99] KLARLUND, N.; MOLLER, A.; SCHWATZBACH, M. I. **Document Structure Description 1.0.** 1999. Disponível em: <http://www.brics.dk/DSD/>. Acesso em: jan. 2002.

[LAB 2000] LABRINIDIS, A.; ROUSSOPOULOS, N. Webview Materialization. In: THE ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2000, Dallas, Texas, USA. **Proceedings...** [S.l.:s.n.], 1997.

[LAH 99] LAHIRI, T.; ABITEBOUL, S. WIDOM, J. Ozone: Integrating Structured and Semistructured Data. In: WORKSHOP ON DATABASE PROGRAMMING LANGUAGES, 1999. **Proceedings...** [S.l.:s.n.], 1999.

[LAY 98] LAYMAN, A.; JUNG, E. et al. **XML-Data**. W3C, January 1998. Disponível em: <http://www.w3.org/TR/1998/NOTE-XML-data>. Acesso em: jan. 2002.

[LIN 2000] LIN, H.; RISCH, T.; KATCHAOUNOV, T. Object-Oriented Mediator Queries to XML Data. In: INTL. CONF. ON WEB INFORMATION

SYSTEMS ENGINEERING, 1., 2000, Hong Kong. **Proceedings…** [Hong Kong: Steering Commitee of Wise Conference], 2000. v.2.

[LIT 90] LITWIN, W.; MARK, L.; ROUSSOPOULOS, N. Interoperability of multiple autonomous databases. **ACM Computing Surveys**, New York, v.22, n.3, 1990.

[MAN 2000] MANOLESCU, I. et al. Agora: Living with XML and Relational. In: INTERNACIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 26., 2000. **Proceedings…** [S.l.:s.n.], 2000.

[MAY 99] MAY, W. et al. A Unified Framework for Wrapping, Mediating, Restructuring Information from the Web. In: ER, 1999**,** Paris. **International Workshop on the World-Wide Web and Conceptual Modeling**. [S.l:s.n], 1999. p.307-320.

[MEL 2000] MELLO, R.S.; DORNELES, C.F.; KADE, A.; BRAGANHOLO, V.P.; HEUSER, C. Dados Semi-Estruturados. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 15., 2000, João Pessoa. **Anais...** João Pessoa:PUCRS, 2000.

[MIC 2000] MICROSOFT. **XML Schema Developer's Guide**. May 2000. Disponível em: <http://msdn.microsoft.com/xml/xmlguide/schema-overview.asp>. Acesso em: jan. 2002.

[MIL 2000] MILLER, R.J.; HAAS, L.M.; HERNÁNDEZ, M.A. Schema Mapping as Query Discovery. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 26., 2000. **Proceedings...** [S.l.:s.n.], 2000.

[MIL 2001] MILLER, R.J. et al. The Clio Project: Managing Heterogeneity. **ACM SIGMOD**, New York, v.30, n.1, p.78-83, Mar 2001.

[MOL 97] GARCIA-MOLINA, H. et al. The TSIMMIS Approach to Mediation: Data Models and Language. **Journal of Intelligent Information Systems**, [S.l.], v.8, n.2, p.117-132, 1997.

[NES 97] NESTOROV, S.; ABITEBOUL, S.; MOTWANI, R. Inferring structure in semistructured data. In: WORKSHOP ON MANAGEMENT OF SEMI-STRUCTURED DATA, 1997. **Proceedings …** [S.l.:s.n.], 1997.

[OZS 99] ÖZSU, M.T.; VALDURIEZ, P. **Principles of Distributed Database Systems.** 2nd ed. New Jersey: Prentice-Hall, 1999. 666p.

[PIM 99] PIMENTEL, M. da G.C.; TEIXEIRA, C.A.C.; PINTO, C. DA C. Hiperdocumentos Estruturados na WWW: teoria e prática. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 19., 1999. **Anais...** [S.l.:s.n.], 1999.

[PIR 97] PIRES, P. de F. **Himpar,** uma Arquitetura para Interoperabilidade de Objetos Distribuídos. 1997. 114p. Dissertação (Mestrado em Ciência da Computação) - PPGE da UFRJ, Rio de Janeiro.

[PIT 99] PITTS-MOULTIS, N.; KIRK, K. **XML Black Book**. São Paulo: Makron Books, 1999.

[PSA 2000] PSAILA, G. ERX: a Conceptual Model for XML Documents. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2000, Villa Olmo, Italy. **Proceedings …**[S.l.:s.n], 2000.

[RIB 95] RIBEIRO, Cora Helena Francisconi Pinto. **Banco de Dados Heterogêneos: Mapeamento dos Esquemas Conceituais em um Modelo Orientado a Objetos**. 1995. 165p. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[RIB 96] RIBEIRO, C.H.F.P.; OLIVEIRA, J.P.M. de. Multidatabase Interoperability Through Conceptual Schema Mapping. in a Object Oriented Model. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING SYSTEMS, PDCS, 9., 1996. **Proceedings…** Raleigh:Isca, 1996. v.2, p.647-652.

[RIS 2000] RISCH, T.; JOSIFOVSKI, V.; KATCHAOUNOV, T. **AMOS II Concepts.** Disponível em: <www.dis.uu.se/~udbl/amos/doc/amos_concepts.html>. Acesso em: out. 2000.

[ROS 2000] ROSA, M. et al. Materializing the Web. In: CONFERENCE ON COOPERATIVE INFORMATION SYSTEMS, 2000. **Proceedings...** [S.l.:s.n.], 2000. Disponível em: <ftp://ftp.dis.uniroma1.it/pub/iocchi/publications/web-coopis98.ps.gz>. Acesso em: jan.2002.

[RUS 88] RUSINKIEWICZ, M.E. et al. OMNIBASE: Design and implementation of a multidatabase system. **Distributed Processing Technical Commitee Newsletter**, [S.l.], v.10, n.2, p.20-28, Nov. 1988.

[SAL 91] SALTOR, F.; CASTELLANOS, M.; GARCIA-MOLINA, H.. Suitability of data models as canonical models for federated databases. **ACM Computing Surveys,** New York, v.22, n.4, 1991.

[SHE 90] SHETH, A; LARSON, J. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. **ACM Computing Surveys**, New York, v.22, n.3, 1990.

[SIL 99] SILBERSCHATZ, A.; KORTH, H.F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Makron Books, 1999.

[SOA 99] SOARES, H.R.; MEDEIROS, C.B. Integrando Sistemas Legados a Banco de Dados Heterogêneos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 14., 1999, Florianópolis. **Anais …** [S.l.:s.n], 1999.

[SPA 91] SPACCAPIETRA, S.; PARENT, C. Conflicts and Correspondence Assertions in Interoperable Databases. **SIGMOD Record,** Semantic Issues in Multidatabase Systems, New York, v.20, n.4, p.49-54, 1991.

[STO 96] STONENBRAKER, M.; MOORE, D. **Object Relational DBMS – The next great wave**. San Francisco: Morgan Kaufmann, 1996.

[SUB 2002] SUBRAHMANIAN, V.S. et al. HERMES: a heterogeneous reasoning and mediator system. University of Maryland, 2002. Disponível em: <http://www.cs.umd.edu/projects/hermes/overview/paper/index.html>. Acesso em: jan. 2002.

[SUC 98] SUCIU, D. Semistructured Data and XML. In: THE INTERNATIONAL CONFERENCE ON FOUNDATIONS OF DATA ORGANIZATION AND ALGORITHMS, FODO, 1998. **Proceedings…** [S.l.:s.n], 1998.

[THI 2000] THIRAN, P. et al. CASE Support for the Development of Federated Information Systems. In: INTERNATIONAL WORKSHOP ON

ENGINEERING FEDERATED INFORMATION SYSTEMS, EFIS, 3., 2000. **Proceedings…** [S.l.:s.n.], 2000.

[THO 2000] THOMPSON, H.S. et al. (Ed.). **XML-Schema Part 1: Structures**. W3C, April 2000. Disponível em: <http://www.w3.org/TR/xmlschema-1/>. Acesso em: jan. 2002.

[UCH 99] UCHOA, E.M.A.; MELO, R.N. Integração de Sistemas de Banco de Dados Heterogêneos usando Frameworks. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 14., 1999, Florianópolis. **Anais...** Florianópolis:UFSC, 1999.

[VDO 2001] VDOVJAK, R.; HOUBEN, G. RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources. In: INTERNATIONAL WORKSHOP ON INFORMATION INTEGRATION ON THE WEB – TECHNOLOGIES AND APPLICATIONS, WIIW, 2001. **Proceedings…** [S.l.:s.n.], 2001.

[VID 97] VIDAL, V.M.P.; LÓSCIO, B.F. Especificação de mediadores para acesso e atualização de múltiplas bases de dados. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 12., 1997. **Anais...** Fortaleza:UFC, 1997.

[VID 2002] VIDAL, V.M.P.; VILAS BOAS, R.M. de F. Uma abordagem Top-Down para a Geração das Correspondências entre XML Schemas Semânticos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 17., 2002, Gramado. **Anais...** Gramado:UFRGS, 2002.

[VIS 97] VISSER, P.R.S. et al. An Analysis of Ontology Mismatches: Heterogeneity versus Interoperability. In: SPRING SYMPOSIUM ON ONTOLOGICAL ENGINEERING, AAAI, 1997. **Proceedings…** [S.l.:s.n.], 1997.

[WEI 92] WIEDERHOLD, G. Mediators in the Architecture of Future Information System. **COMPUTER,** New York, v.25, n.3, p.38-49, Mar. 1992.

[WOE 93] WOELK, D. et al. Using Carnot for interprise information integration. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED INFORMATION SYSTEMS, 2., 1993. **Proceedings…** [S.l.:s.n.], 1993.