KEMMEL DA SILVA SCOPIM

J-SCHEMAS INTEGRATOR - UMA FERRAMENTA PARA INTEGRAÇÃO DE ESQUEMAS DE BANCOS DE DADOS HETEROGÊNEOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática pelo Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Marcos Sfair Sunye

CURITIBA

2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Kemmel da Silva Scopim, avaliamos o trabalho intitulado, "*J-Schemas Integrator –uma ferramenta para integração de esquemas de bancos de dados heterogêneos*", cuja defesa foi realizada no dia 27 de outubro de 2003, às quatorze horas, no Anfiteatro A do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 27 de outubro de 2003.

Prof. Dr. Marcos Sfair Sunye
DINF/UFPR – Orientador

Prof. Dr. Carlos Alberto Heuser
UFRG - Membro Externo

Profª. Dra. Laura Sanchez Garcia
DINF/UFPR - Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

TERMO DE APROVAÇÃO

KEMMEL DA SILVA SCOPIM

J-SCHEMAS INTEGRATOR – UMA FERRAMENTA PARA INTEGRAÇÃO DE ESQUEMAS DE BANCOS DE DADOS HETEROGÊNEOS

Dissertação aprovada como requisito à obtenção do grau de Mestre em Informática pelo Programa de Pós-Graduação em Informática da Universidade Federal do Paraná pela seguinte banca examinadora:

Orientador: Prof. Dr. Marcos Sfair Sunye
Departamento de Informática, UFPR

Prof. Dr. Carlos Alberto Heuser
Departamento de Informática, UFRG

Profa. Dra. Laura Sanchez Garcia
Departamento de Informática, UFPR

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

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

Agradecimentos

Ao Professor **Marcos S. Sunye** por sua confiança no trabalho desenvolvido, que além de me orientar brilhantemente, tornou-se um grande amigo pelos ensinamentos transmitidos.

Aos meus pais, **José Scopim e Tereza da Graças Silva Scopim**, a meu irmão **Igor Silva Scopim** e a minha namorada **Leila Cristina Silva** pelo incentivo e motivação que me ajudaram a concluir este trabalho.

Ao Programa de **Pós-Graduação em Informática da UFPR** pela excelente equipe de professores disponíveis em seu departamento e pela condição em oferecer um programa de mestrado de boa qualidade a sociedade.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho...

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

CASE - Computer aided software engineering

CDF - Channel Definition Format

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

DOM - Document Object Model

DSSSL - Dynamic Style Semantics and Specification Language

DTD – Document type definitions

ERC+ - (Modelo) Entidade Relacionamento Complexo

HTML - HyperText Markup Language

MathML - Mathematical Markup Language

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

RDF - Resource Description Framework; linguagem para escrita de metadados

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

SGBD – Sistema Gerenciador de Bases de Dados

SGML - Standard Generalized Markup Language (pré-web)

TAG – Marca; marcação; declaração de comando ou atributo

W3C – World Wide Web Consortium

XHTML - Extensible HyperText Markup Language

XLink, XPointer - Extensible Link e Extensible Pointer Languages

XLL - Extensible Link Language

XML – Extensible Markup Language

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

XSL - Extensible Style Language

XSLT- Extensible Stylesheet Language Transformations

Resumo

Atualmente, várias organizações e companhias utilizam diversos sistemas de bancos de dados para gerenciar grande quantidade de seus dados. Entretanto, esses numerosos sistemas de banco de dados heterogêneos foram projetados para rodarem isoladamente e para não cooperarem entre si. Prover interoperabilidade entre esses bancos de dados é importante para o sucesso das organizações, nas quais ganhos de produtividade serão obtidos se esses sistemas puderem ser integrados e permitirem uma visão unificada dos dados. A integração de esquemas de bancos de dados heterogêneos pode ser definida como um processo que, através de uma entrada de um conjunto de esquemas de banco de dados, produz como saída, uma descrição unificada dos esquemas iniciais, chamado esquema integrado e a descrição da informação de mapeamento entre o esquema integrado e os esquemas iniciais. Essa dissertação de mestrado consiste na implementação de uma ferramenta cujo objetivo seja auxiliar e facilitar o processo de integração de esquemas de banco de dados. A ferramenta visual é responsável por importar esquemas de banco de dados, facilitar a identificação dos objetos conflitantes entre esquemas e pelo processo de integração e geração do esquema integrado e da informação de mapeamento entre o esquema integrado e os esquemas iniciais.

Palavras chaves: ferramenta para integração de esquemas de banco de dados distribuídos, geração da informação de mapeamento entre os esquemas, integração de esquemas.

Abstract

In today's modern organizations is common to find several different databases being used to accomplish their operational data management functions. These numerous heterogeneous databases systems were designed to run in isolation and they do not cooperate with each other. Providing interoperability among these databases is important to the successful operation of the organization. Improvements in productivity will be gained if the systems can be integrated – that is, made to cooperate with each other – to support global applications accessing multiple databases. This work is dedicated to the implementation of a tool to help developers in the task of integration of heterogeneous database schemas. The database schema integration can be defined as the process that, through an input of a set of databases schemas, produces, as the output, an unified description of the initials schemas, called integrated schema and the information mapping that supports the access to the data stored in the initial databases schemas from the integrated schema. The tool is responsible for importing/exporting schemas from/to database, facilitate the identification of conflicting objects between schemas and for the process of integration and generation of the integrated schema and the mapping information between the integrated schema and the initial schemas.

Keywords: Tool for Database Schemas Integration, information mapping between schemas, integrated schema.

1 Introdução

Várias organizações usam diversos bancos de dados para realizar suas funções de gerenciamento de dados no dia a dia. Tipicamente, estes bancos de dados são heterogêneos, pois eles armazenam diferentes modelos de dados, representam os dados de uma maneira diferente, rodam em diferentes plataformas de hardware e normalmente são gerenciados por softwares diferentes.

Esta diversidade de sistemas de bancos de dados pode ser tornar um grande problema quando existe a necessidade de acessar todos os dados de fora das organizações ou até mesmo de dentro das próprias organizações. A replicação dos dados é um dos principais problemas enfrentados, pois dados que representam o mesmo domínio de informação são armazenados em locais diferentes, dificultando a manutenção dos mesmos e desperdiçando espaço de armazenamento em disco. Outro grande problema pode se tornar comum quando sistemas de informação necessitam acessar informações em vários destes bancos de dados.

Pesquisas no gerenciamento de banco de dados heterogêneos têm dado grande ênfase no desenvolvimento de mecanismos que provêem acesso unificado as informações localizadas em vários bancos de dados distribuídos, preservando a autonomia dos mesmos. Ou seja, a construção de uma visão unificada, uma interface homogênea, dos dados sem a necessidade de realizar alterações nos bancos de dados existentes, preservando sua integridade e os investimentos iniciais realizados na criação destes sistemas.

Está se tornando cada vez mais importante desenvolver mecanismos que permitam interoperabilidade entre esses bancos de dados. Uma aproximação para prover tal interoperabilidade é definir um ou mais esquemas[2] que representem uma visão coerente dos bancos de dados em questão. O processo para geração desses esquemas é conhecido como integração de esquemas segundo BATINI (1986). A integração de esquemas é utilizada para prover interoperabilidade entre os banco de dados e envolve a resolução de conflitos semânticos, descritivos e estruturais conforme descritos por BATINI (1984), entre os esquemas dos bancos de dados a serem integrados.

A integração de esquemas é um dos meios utilizados para resolver o problema de interoperabilidade entre as várias fontes de dados unificando os esquemas e criando um único

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

esquema federado conforme SHELT (1990). Esse esquema unificado pode oferecer flexibilidade e eficiência para acessar várias fontes de dados heterogêneas.

Como a integração de esquemas permite uma integração de banco de dados heterogêneos, vários modelos de dados podem ser utilizados para representar os esquemas a serem integrados. Para facilitar a obtenção da especificação do esquema integrado segundo RAM (1999) é recomendável representar todos os esquemas das bases de dados a serem integradas em um único modelo de dados, mesmo levando-se em consideração que as bases de dados iniciais foram baseadas em modelos de dados diferentes.

O modelo de dados ERC+ - Entidade Relacionamento Complexo Estendido de SPACCAPIETRA (1975), possui algumas características interessantes que podem auxiliar no processo de integração de bancos de dados. Este modelo, que é uma extensão do modelo ER[3] tradicional, expressa de uma forma mais completa a relação entre objetos com o mesmo significado no mundo real e sua representação no banco de dados.

O processo de integração de esquemas de banco de dados pode ser definido como uma atividade que, através de uma entrada de um conjunto de esquemas de banco de dados, já representado em um mesmo modelo de dados, produz como saída, uma descrição unificada dos esquemas iniciais chamadas de esquema integrado e a informação de mapeamento entre o esquema integrado e os esquemas iniciais.

Apesar de existirem várias metodologias para o processo de integração de esquema e conseqüentemente a obtenção da visão unificada, BATINI (1986) identifica 3 principais etapas encontradas nas metodologias analisadas em seu trabalho: etapa de pré-integração; identificação das correspondências entre os esquemas; geração do esquema integrado e da informação do mapeamento entre os esquemas.

Na fase de pré-integração, todas as bases de dados iniciais devem ser modeladas usando um modelo de dados comum. Na fase de identificação das correspondências entre os esquemas, devem ser identificados e categorizados todos os objetos relacionados e conflitantes entre as bases de dados a serem integradas.

Para o processo de geração do esquema integrado é necessário também gerar a informação de mapeamento dos objetos entre o esquema integrado e os objetos nos esquemas iniciais.

---

[3] ER – Entidade Relacionamento

Pode-se também numa fase posterior implementar o esquema integrado. Nesta fase o esquema integrado e as informações de mapeamento entre os esquemas iniciais e o esquema integrado são utilizados para implementar uma interface homogênea permitindo que consultas sejam realizadas e redirecionadas para as bases de dados iniciais. Isto é realizado graças às informações de mapeamento existente entre o esquema integrado e os esquemas dos bancos de dados utilizados no processo de integração.

1.1 Justificativa da Dissertação

A grande profusão de SGBDs baseados no modelo relacional existente no mercado, e a eventual necessidade da integração de esquemas neles construídos, criam a demanda por ferramentas que facilitem e auxiliem a visualização e o processo de integração de esquemas. Hoje em dia, existem várias ferramentas que auxiliam a visualização e modelagem de um único esquema. Porém há uma carência de ferramentas que auxiliem e facilitem o processo de geração de esquemas integrados.

Sem a utilização de uma ferramenta que auxilie o processo de integração de esquemas, este procedimento se torna um processo inteiramente manual e custoso. Além do que, toda a inteligência do processo de integração, fica a cargo dos responsáveis pela tarefa de integração.

Por outro lado, o poder semântico do modelo relacional deixa a desejar, seja qual for o método de integração adotado. (BATINI e LENZERINI, 1986 e KLAS e outros, 1994).

1.2 Objetivos e Contribuição da Dissertação

O objetivo desta dissertação é implementar uma ferramenta de apoio para visualização e participação no processo de integração de esquemas de bancos de dados heterogêneos.

A ferramenta J-Schemas Integrador (objeto de implementação deste trabalho) possibilita ao usuário (integrador) as seguintes tarefas:

- Visualizar os esquemas de banco de dados.
- Identificar os conflitos existentes entre os esquemas a serem integrados.
- Gerar o esquema integrado a partir dos esquemas informados e dos conflitos apontados pelo usuário no editor visual da ferramenta.

- Gerar a informação de mapeamento entre o esquema gerado e os esquemas iniciais num formato padrão.

Também foi objetivo desta dissertação implementar uma arquitetura flexível, para que futuros trabalhos pudessem utilizar e complementar as tarefas desenvolvidas pela ferramenta J-Schemas Integrator.

1.3 Organização da Dissertação

Esta dissertação encontra-se organizada em cinco capítulos.

No capítulo 2 serão mostrados alguns trabalhos relacionados a ferramentas de integração de banco de dados.

No capítulo 3 será explicado o modelo de dados ERC+ que combina características dos modelos de dados semânticos tradicionais com capacidades de orientação a objetos. Este modelo será utilizado pelo editor para a representação visual dos esquemas iniciais a serem integrados. O esquema integrado resultante do processo de integração também estará representado por este modelo.

O capítulo 4 descreve todo o processo de integração juntamente com as fases que compreendem este processo e mostra como a ferramenta desenvolvida encaixa-se em cada uma das fases.

- *Fase de pré-integração*, na qual todos os esquemas dos bancos de dados iniciais devem ser modelados utilizando um mesmo modelo de dados comum.
- *Fase de identificação das correspondências e dos conflitos*, na qual o usuário deve identificar e classificar os conflitos existentes entre os esquemas das bases de dados iniciais.
- *Fase de geração do esquema integrado*, a qual provê a geração de uma descrição unificada dos esquemas iniciais (esquema integrado) e a descrição das informações de mapeamento entre o esquema integrado e os esquemas iniciais.

O capítulo 5 descreve o funcionamento da ferramenta J-Schemas Integrator bem como sua arquitetura e ilustrará quais foram as tecnologias utilizadas para a implementação da ferramenta. Este capítulo mostrará também as atuais limitações da ferramenta.

Finalizando, são apresentadas as considerações, conclusões e sugestões de possíveis trabalhos futuros.

No Apêndice 1 deste trabalho é detalhado a linguagem XML[4] que é utilizado largamente durante a implementação da ferramenta.

[4] Extensible Markup Language

2 Trabalhos Relacionados

Vários projetos para sistemas de bancos de dados integrados têm sido desenvolvidos através dos anos, tanto ferramentas comerciais quanto protótipos acadêmicos. Tais sistemas são limitados pela funcionalidade ou pela especificidade das ferramentas.

2.1 Ferramentas para integração de banco de dados distribuídos homogêneo

Produtos como SQLServer da Microsoft, Oracle, Ingres/Star e DataJoiner da IBM são exemplos de ferramentas comerciais. Esses produtos, na maioria das vezes, limitam-se apenas a conectarem-se os bancos de dados e prover interoperabilidade entre eles. Porém estas ferramentas carecem de processos metodológicos para geração de uma visão unificada dos bancos de dados (esquema integrado) resolvendo os possíveis conflitos existentes entre as bases de dados interligadas. Algumas ferramentas oferecem soluções para alguns tipos de conflitos que podem existir entre os objetos, mas não possibilitam a flexibilização da integração de outros tipos não suportados por estas ferramentas.

- O SQLServer proporciona duas linguagens para múltiplos bancos de dados: *Transac-SQL* e *Visual Query Language* (VQL). Através destas linguagens é possível qualificar o nome de um relacionamento com o nome de um banco de dados ou definir visões, especificar procedimentos ("*queries*") entre os vários bancos de dados e manipular dependências.

- A Oracle possui uma linguagem para múltiplos bancos de dados chamada SQL*PLUS. Esta linguagem oferece possibilidade de qualificar uma relação com um banco de dados, definir sinônimos para relações ou banco de dados e definir procedimentos entre os bancos de dados envolvidos. A Oracle possui também um gerenciador de bancos de dados distribuídos SQL*STAR que permite operações múltiplas que serão avaliadas pelos bancos de dados distribuídos.

- A Ingres/Star é uma camada de software para acesso transparente ao sistema distribuído Ingres. Permite a definição de um esquema múltiplo externo e virtual através de banco de dados Ingres ou não. Porem, o Ingres/Star não permite que sejam

formulados procedimentos nos bancos de dados locais, somente através do esquema externo.

Segundo (GARDARIN; GANNOUNI; FINANCE, 1995) estes produtos podem ser tidos como sistemas de bancos de dados distribuídos homogêneos, suportando conexões com SGBDs[5] externos através de *gateways*. Além do que, estas ferramentas não têm um processo embutida para facilitar a geração de esquemas integrados resolvendo conflitos existentes nas bases de dados. O objetivo destas ferramentas é facilitar a interoperabilidade entre os banco de dados.

2.2 Ferramentas para integração de dados baseado em integração de XML

Existe um outro conjunto de ferramentas que fazem a integração da informação contida em várias fontes de dados, através do XML. Estas ferramentas, como é o caso das ferramentas *Nimble* de DRAPER(2000) e *MetaMatrix*, funcionam como uma plataforma de software baseado na integração de XML retornado pelo banco de dados em tempo de execução. Isto é, estas ferramentas comunicam-se com as fontes de dados, mapeiam o resultado da consulta para um formato XML e em tempo de execução é realiza a integração de todas as consultas realizadas nas farias fontes de dados através da integração dos XMLs retornados por cada consulta individual.

Este tipo de ferramenta não se preocupa em criar uma visão unificada das fontes de dados em questão. Elas simplesmente fazem uma integração dos dados retornados pelos diversos bancos de dados envolvidos no processo.

2.3 Ferramentas para integração de dados baseado em agentes

Outra estratégia de integração de dados pode ser a integração de dados baseado em agentes. O projeto CoopBDH de CAVINATO(2001) tem como objetivo principal desenvolver um modelo de cooperação que dê suporte à integração de bancos de dados heterogêneos utilizando componentes de um sistema *multibanco*, numa abordagem de integração baseada na utilização de esquemas globais e na descentralização do conhecimento sobre a integração. O modelo de cooperação será implementado numa arquitetura de agentes, através de agentes de bancos de dados cooperativos, sendo que cada agente representa um

banco de dados que faz parte do sistema *multibanco*. Tal modelo deve permitir a distribuição das estruturas de integração geralmente utilizadas, concentradas nos mediadores, definindo uma arquitetura de mediação baseada em componentes cooperativos.

2.4 Ferramentas para integração de esquemas de banco de dados

O projeto AutoMed de JASPER (2003) que ainda está em andamento é um projeto que pretende fazer uma integração de dados baseada no uso de seqüências reversíveis de transformações de esquemas. Visões podem ser geradas a partir dessas seqüências. Dentre os objetivos do projeto AutoMed estão:

- Investigar como o framework teórico que eles propuseram para transformações de esquemas e integração baseada num modelo de grafo pode ser praticamente aplicada e escalada para problemas de integração de banco de dados real.
- Investigar quantas funcionalidades no processamento global das consultas podem ser geradas automaticamente dando disponibilidade em transformações de duas vias entre esquemas iniciais e esquema integrado.
- Investigar como as técnicas de computação podem ser aplicadas para evoluir os esquemas e melhorar a otimizações nas consultas globais feitas para o esquema integrado quando disparados para as bases heterogêneas.

Outro projeto de integração de esquema em andamento é o UNIBASE (2003): "*A System for Unifying Heterogeneous DataBases*". O Sistema UNIBASE é um conjunto de ferramentas e metodologias para integração e acesso a banco de dados heterogêneos. O Sistema UNIBASE pode ser dividido em dois sub-sistemas: o *Unifier* e o *Acessor*. O sub-sistema Unifier é um conjunto de ferramentas e metodologias que facilitam a integração de banco de dados. O sub-sistema *Acessor* permite que usuários submetam consultas para as bases de dados relacionadas usando uma linguagem de consultas padrão.

[5] Sistema Gerenciador de Banco de Dados

3 O modelo de dados Entidade Relacionamento Complexo (ERC+)

O Modelo ER - Entidade Relacionamento - foi originalmente proposto por CHEN (1976) como uma forma de facilitar o desenho de bases de dados. Embora este modelo seja suficiente para a representação da maioria das aplicações administrativas convencionais, o crescimento da aplicação da informática às diversas áreas do conhecimento e indústrias trouxe consigo a necessidade de melhoria do poder semântico, visando a representação de situações mais complexas, como por exemplo na área de CAD/CAM[6] sugerido por PARENT e SCCAPAPIETRA (1989).

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

O modelo ERC+ - Entidade Relacionamento Complexo Estendido - é um modelo especificamente projetado para suportar modelagem de objetos complexos. Seu objetivo é representar com mais fidelidade objetos do mundo real. É utilizado o conceito de tipo de entidade para descrever objetos e o conceito de tipo de relacionamento para representar as relações entre os objetos.

3.1 Conceitos

Um **tipo de entidade - TE** - é definido pelo seu nome, seu esquema (que é o conjunto de estrutura de seus atributos), sua entidade genérica, o conjunto de tipo de entidades que estão em conjunção, e sua população, que é o conjunto de ocorrências (entidades) com seus identificadores de objetos (*oids*) e valores.

Um **tipo de relacionamento - TA** é definido pelo seu nome, o conjunto de tipo de entidades aos quais está conectado, com a descrição das características dos *links* (papel e cardinalidade), o conjunto de estruturas de seus atributos (que constituem seu esquema), e o conjunto de suas ocorrências.

Um **atributo** é definido por um objeto ao qual está relacionado sua estrutura e seus valores para cada ocorrência do seu objeto.

Um **Domínio** define o conjunto de todos os possíveis valores para um atributo, uma entidade, ou um tipo de relacionamento. Valores podem ser tanto atômicos, como Brasil ou

[6] Desenho e manufatura auxiliado por computador.

2000, ou complexos, isto é, composto por outros valores. Um valor complexo é um conjunto de pares <nome do atributo,v> onde v é ou um valor ou um conjunto de valores.

Uma **identificação do objeto - *oid*** - é associada a cada entidade. Um objeto do mundo real, apesar de sua complexidade, pode ser modelado como uma ocorrência de um tipo de entidade simples.

Um tipo de entidade é representado por um retângulo enquanto que um tipo de relacionamento é representado por um losango. Ambos com o nome que a representam em seu interior. A cardinalidade é representada da seguinte maneira: uma simples linha contínua identifica uma relação obrigatória e monovalorada (1:1); uma simples linha tracejada representa uma relação opcional e monovalorada (0:1); uma linha tracejada dupla representa uma relação opcional e multivalorada (0:n) e uma linha tracejada juntamente com uma linha contínua representa uma relação multivalorada e obrigatória (1:n) conforme figura 1.

Vários (dois ou mais) tipos de entidades podem ser unidos por um tipo de relacionamento. Como tipo de entidades, um tipo de relacionamento pode ter atributos. Tipos de relacionamentos podem ser cíclicos (um tipo de relacionamento pode unir duas, ou várias vezes, o mesmo tipo de entidade, cada vez com um papel diferente).

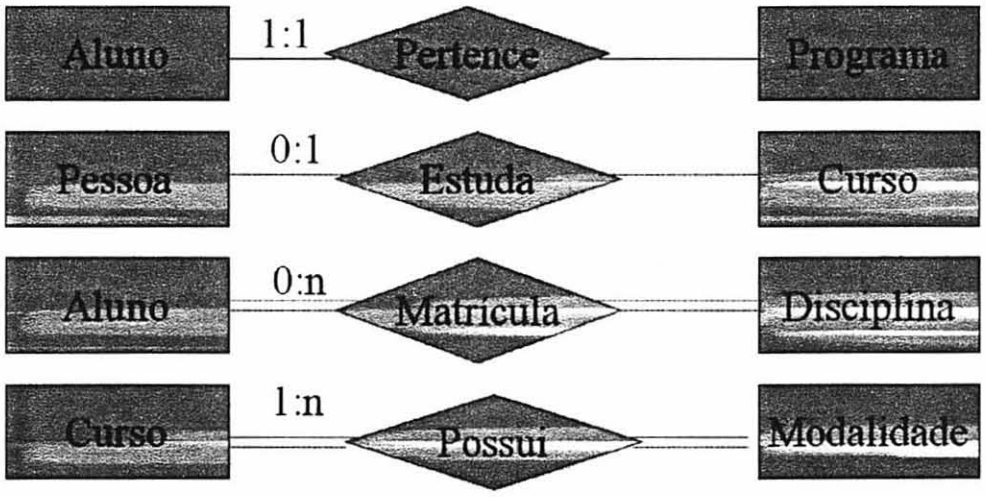

Figura 1Cardinalidade do modelo ERC+

Além disso, o modelo ERC+ suporta dois outros tipos de *links* entre tipos de entidades: o **link *is-a***[7] e o **link *may-be-a***[8]. Ambos são usados para relacionar dois tipos de entidades que tem alguma representação estendida do mesmo objeto do mundo real, cada link com um diferente ponto de visão. Entidades representando o mesmo objeto do mundo real compartilham o mesmo *oid*, que é associado com um objeto do mundo real.

O clássico link *is-a* (ou link de generalização) especifica que a população do tipo de entidade ES é uma subclasse (especialização) de um outro tipo de entidade EG (generalização). Portanto, cada *oid* contido em ES é também um *oid* de EG. Este link é representado no diagrama ERC+ por uma flecha: ES → EG. Ciclos não são permitidos usando este tipo de link.

O link *may-be-a* (ou link de conjunção) entre dois tipos de entidades E1 e E2 é usado para especificar que algumas (possivelmente todas) entidades de E1 descrevem o que os mesmos objetos do mundo real nas entidades E2 fazem, e vice e versa, isto é, a população de E1 e E2 podem compartilhar *oids*. O link *may-be-a* é graficamente representado por uma linha tracejada entre E1 e E2: E1 - - - E2.

3.2 Comparação entre o modelo ERC+ e o modelo de Venn

Os links *is-a* e *may-be-a* permitem-nos modelar duas situações, conforme mostrado nas figuras 2 e 3, que podemos levantar entre os conjuntos de oids de dois tipos de entidades E1 e E2:

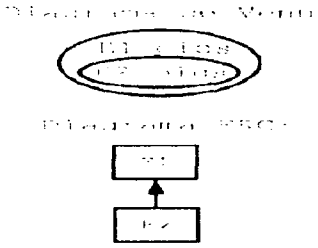

Figura 2 Os oids de E2 estão incluídos em E1.

Isto é um link is-a

[7] É um
[8] Pode ser um

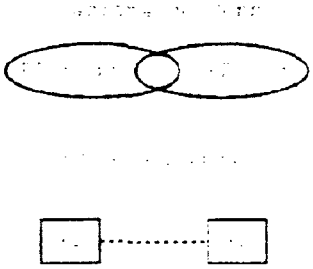

Figura 3 E1 e E2 compartilham alguns oids.

Isto é um link may-be-a

3.3 A Ferramenta SUPER

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

3.4 Utilização do Modelo ERC+ dentro da Ferramenta J-Schemas Integrator

Nossa proposta neste trabalho foi utilizar o modelo ERC+ para representar graficamente os esquemas de bancos de dados a serem integrados e o esquema integrado gerado. As informações descrevendo os objetos contidos em um esquema ERC+ foram representadas através de um documento XML, o qual será descrito no capítulo 5 referente a ferramenta J-Schemas Integrator.

4 Integração de esquemas de base de dados

A integração de esquemas de base de dados é o processo que leva como entrada um conjunto de esquemas de bases de dados e produz como a saída uma única descrição unificada dos esquemas de entrada chamado esquema integrado conforme representado pela figura 4. Também é gerado como saída, o mapeamento de informações associadas que suportam acesso integrado aos dados existentes nos esquemas de entrada através do esquema integrado. A integração da base de dados é usada também no processo de reengenharia de sistemas legados. BATINI e LENZERINI (1986) fizeram uma comparação das principais metodologias de desenvolvimento e observaram que as principais etapas na geração do esquema integrado são:

a) Pré-integração: onde os esquemas de entrada são transformados para se tornarem mais homogêneos (tanto sintaticamente e semanticamente);

b) Identificação de correspondência: dirigida à identificação e à descrição de relacionamentos de interesquemas;

c) Integração: a etapa final que resolve conflitos de interesquemas e unifica itens correspondentes em um esquema integrado.

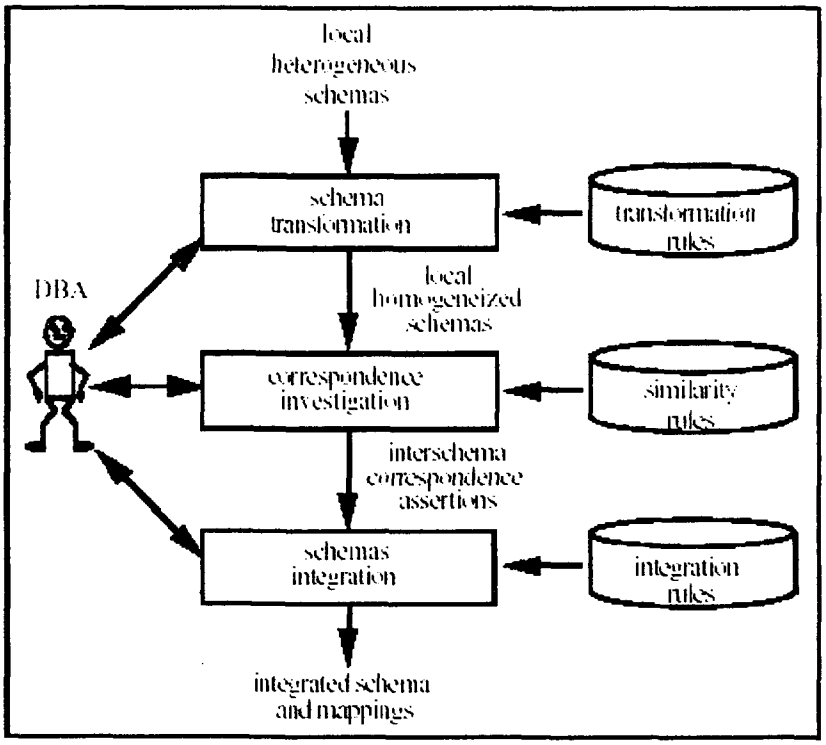

Figura 4 O processo global de integração, SPACCAPIETRA (98)

O objetivo geral da integração de esquemas é integrar uma organização com diferentes propósitos ou sistemas de base de dados e percepções do usuário, permitindo o acesso global a um recurso organizacional integrado da informação. De qualquer forma, a pesquisa da integração de esquemas foi especializada em duas áreas segundo RAM (1999)

a) Integração de visões

b) Integração de base de dados.

A integração de visões é usada como uma ferramenta de auxílio no projeto da base de dados e produz uma descrição global e conceitual de uma base de dados proposta, unindo diferentes requisições de dados ou visões do usuário.

De outra forma, a integração da base de dados é usada para produzir um esquema global que representa uma coleção de bases de dados relacionadas em toda a organização. Este esquema global é uma visão virtual de todas as bases de dados juntas em um ambiente de base de dados distribuído.

4.1 Abordagens para integração de esquemas.

Há duas abordagens básicas de integração de esquemas: esquema global e sistemas de bancos de dados federados.

4.1.1 Integração via esquema global:

A integração via esquema global é a integração completa (ou não) dos vários bancos de dados com o objetivo de fornecer uma única visão do banco de dados heterogêneo. Pode-se disponibilizar um esquema global geral, ou vários esquemas globais parciais (visões) de acordo com as necessidades dos vários usuários.

Vantagens:

- O usuário tem uma visão única dos dados integrados.

Problemas:

- O processo de integração requer a intervenção do usuário (mapeamento dos esquemas para o esquema global - conflitos semânticos, estruturais e comportamentais), por isso só pode ser parcialmente automatizado;

- A autonomia dos bancos de dados participantes é sacrificada para resolução de conflitos semânticos durante a criação do esquema global;
- Diversos métodos de integração existem para mais de dois bancos de dados: combinação dois a dois, combinação de todos de uma só vez, etc., e cada método pode definir esquemas globais completamente diferentes;
- A integração dos esquemas é estática, a cada modificação nos esquemas locais o esquema global tem que ser revisado.

4.1.2 Integração de esquemas via bancos de dados federados:

A integração não tem que ser total, e depende das necessidades dos usuários. Semelhante à integração via esquema global com a utilização de diversas visões sobre o conjunto de dados integrados, mas cada banco de dados participante tem que conhecer todos os demais conforme SHELT (1990). Os bancos de dados federados podem ser fortemente ou fracamente acoplados, definindo diferentes compromissos entre integração e não-integração:

- *Fracamente acoplados*: a responsabilidade de criar e manter a federação fica a cargo do usuário, não existindo um controle central da federação. O objetivo deste tipo de federação é de tornar mais dinâmico e flexível o compartilhamento dos dados dos bancos de dados componentes. Adequado para sistemas que tem por objetivo a leitura dos dados.
- *Fortemente acoplados*: há um número finito de esquemas federados criados e administrados por apenas um indivíduo (administrador da federação). Adequado para sistemas que tem por objetivo a leitura e a atualização dos dados.

4.2 Primeira Fase: Pré-integração

Na fase de pré-integração, esquemas que correspondem às bases de dados individuais a serem integrados devem ser traduzidos para um esquema usando um modelo de dados comum.

O propósito desta fase é transformar os esquemas a serem integrados em um esquema mais homogêneo tanto sintaticamente quanto semanticamente, e desta maneira, facilitar a especificação final do esquema integrado. Esta tarefa deve ser realizada, pois os bancos de dados iniciais podem ter sido modelados utilizando diferentes modelos de dados (ER – Entidade Relacionamento, Orientado a Objetos, Hierárquico, Objeto-Relacional, etc).

Isto pode ser uma tarefa complexa, primeiramente, porque muitas representações de esquemas podem não capturar a semântica pretendida dos bancos de dados, segundo porque quem está integrando os esquemas pode não ter sido o responsável por modelar o esquema inicial.

Portanto, o processo de integração requer interação intensiva com os projetistas e administradora do sistema para um melhor entendimento da semântica dos bancos de dados e garantir que as semânticas dos esquemas integradas não violem a semântica dos bancos de dados iniciais.

A nossa proposta é de que todas as bases de dados iniciais sejam modeladas utilizando um modelo de dados comum: o ERC+. Este modelo reúne requisitos de aplicação de banco de dados combinando características da semântica tradicional do modelo de dados com capacidades de orientação a objetos tais como orientação estrutural dos objetos, herança e identidade dos objetos como demonstrado no capítulo 3 deste trabalho.

As principais dificuldades geralmente encontradas nesta fase são a complexidade dos esquemas iniciais, a falta de documentação adequada para tradução dos esquemas para o esquema de dados desejado. Por esta razão, esta fase demanda um grande tempo para adquirir o conhecimento para a compreensão da semântica adequada dos objetos em questão.

A fase de pré-integração é justificada, entre outras razões, pela relativa fraqueza semântica dos esquemas e pela heterogeneidade dos modelos de dados utilizados nesses esquemas. Esta fase pode representar uma oportunidade para mostrar os eventuais limites do sistema atual.

4.3 Segunda Fase: Identificação das correspondências entre os objetos dos esquemas.

O objetivo desta fase é identificar objetos entre os esquemas iniciais que possam estar relacionados, isto é, entidades, atributos e relações que estejam em conflitos e classificar os relacionamentos entre estes objetos. Isto é feito examinando a semântica dos objetos nos diferentes bancos de dados e identificando os relacionamentos baseados na sua semântica. A semântica de um objeto pode ser averiguada analisando propriedades esquemáticas das entidades, atributos e relacionamentos nos esquemas bem como pela interação com os analistas responsáveis pelos bancos de dados tirando proveito de seus conhecimentos sobre o domínio da aplicação. A comparação entre os esquemas é realizada em dois níveis:

a) Comparação entre os elementos dos esquemas (entidades, atributos e relações);
b) Comparação entre a população dos esquemas: A comparação entre as instâncias permite descobrir relações lógicas (inclusão, intersecção) entre os dados e os valores nos banco de dados, indicando tipos de relacionamento generalização/especialização entre os elementos envolvidos nos esquemas.

A principal meta deste passo é gerar um conjunto de relacionamentos confiáveis entre os objetos dos bancos de dados. É importante que esses relacionamentos sejam precisos, pois eles serão usados como entrada na fase de geração do esquema integrado.

A atividade fundamental nessa etapa consiste em verificar todos os conflitos na representação dos mesmos objetos em esquemas diferentes. Pode-se usar para classificar os relacionamentos entre os objetos, o conjunto de técnicas e processos descritos por BATINI e LENZERINE (1986), SPACCAPIETRA (1992) e LENZERINE (1984).

Existem vários tipos de conflitos que podem existir entre os objetos de esquemas a serem integrados. Iremos examinar agora alguns destes tipos de conflitos em detalhes:

4.3.1 Conflitos de Nomes

Os esquemas em modelos de dados incorporam nomes para os vários objetos representados. Pessoas de diferentes áreas de aplicação da mesma organização consultam aos mesmos dados usando as suas próprias terminologias e nomes. Isto resulta em uma proliferação de nomes, bem como uma possível inconsistência entre nomes nos componentes dos esquemas. Os relacionamentos problemáticos entre nomes são de dois tipos:

- **Homônimos:** Quando o mesmo nome for usado para dois conceitos diferentes causando inconsistências. Considere os dois esquemas mostrados na Figura 5. Ambos os esquemas incluem uma entidade nomeada *Equipment*. Entretanto, o *Equipment* na Figura 5a refere-se a computadores, copiadoras, etc., visto que na Figura 5b refere-se a equipamentos como condicionadores de ar. É óbvio que unir as duas entidades no esquema integrado resultaria em produzir uma entidade para dois objetos conceituais distintos.

- **Sinônimos**: Quando o mesmo conceito for descrito por dois ou mais nomes. A menos que diferentes nomes melhorem o entendimento de diferentes usuários, eles não são justificados. Um exemplo aparece na Figura 6, onde o *Client* e o *Customer* são sinônimos; as entidades com estes dois nomes nos dois esquemas referem-se ao mesmo conceito no mundo real. Neste caso, manter duas entidades distintas no esquema integrado resultaria na modificação de um único objeto por meio de duas entidades diferentes.

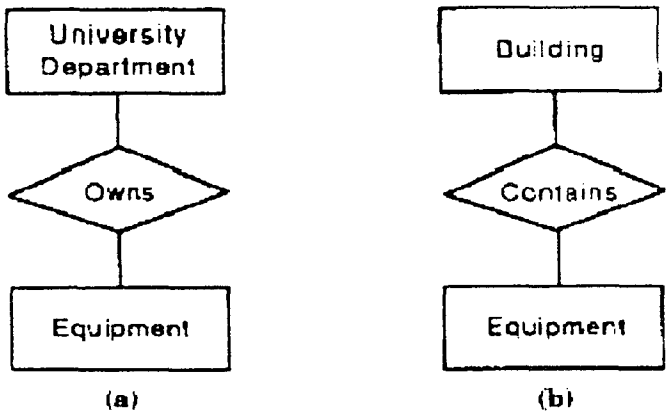

Figura 5 Exemplo de Homônimo, BATINI (86)

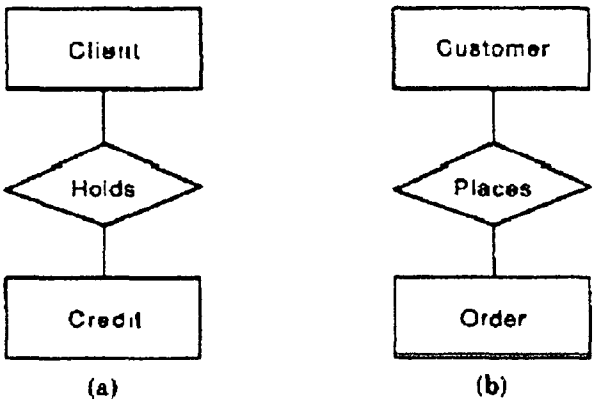

Figura 6 Exemplo de sinônimo, BATINI (86)

Note que homônimos podem ser detectados comparando conceitos com o mesmo nome em esquemas diferentes, ao passo que sinônimos podem somente ser detectado após uma especificação externa.

Dicionários de dados têm sido utilizados como ferramenta adjunta para o processo de integração de esquemas para um melhor gerenciamento de nomes.

4.3.2 Conflitos Estruturais

Utilizamos o termo conflitos estruturais para englobar conflitos que aparecem em conseqüência de uma escolha diferente de modelar construções ou restrições de integridade. A seguinte classificação distingue os seguintes tipos de conflitos:

a) **Conflito de Tipo**. Estes aparecem quando o mesmo conceito é representado por diferentes construções de modelagem em esquemas diferentes. Este é o caso quando, por exemplo, uma classe de objetos é representada como uma entidade em um esquema e como atributo em outro esquema.

b) **Conflitos de dependência**. Estes aparecem quando um grupo de conceitos é relacionado a si mesmos com diferentes dependências em esquemas diferentes. Por exemplo, o relacionamento Casamento entre homem e mulher é 1: 1 em um esquema, mas pode ser m: n em outro.

c) **Conflitos de chave**. As diferentes chaves são associadas ao mesmo conceito em esquemas diferentes. Por exemplo, nome pode ser a chave do empregado em um esquema e a identificação de Empregado pode ser a chave do empregado em outro esquemas.

d) **Conflitos de Comportamento**. Estes surgem quando as diferentes políticas de inserção/deleção são associadas com a mesma classe de objetos em esquemas distintos. Por exemplo, em um esquema, um departamento pode ter a permissão de existir sem empregados, visto que outro, quando o último empregado associado com um departamento é deletado, é feita a deleção automática do departamento. Note que estes conflitos podem surgir somente quando o modelo dos dados permitir a representação comportamental das propriedades dos objetos.

4.3.3 Tarefas a serem realizadas

Duas tarefas principais devem ser realizadas nesta fase.

a) Primeiro deve-se identificar todos os objetos que possuam algum tipo de relacionamento entre si nas bases de dados iniciais.

b) Uma vez que um conjunto potencial de objetos relacionados tenha sido identificado, a segunda tarefa é classificar os relacionamentos entre esses objetos em várias categorias.

4.4 Terceira Fase: Geração do Esquema Integrado.

A geração do esquema integrado é o processo de geração de um ou mais esquemas integrados a partir dos esquemas iniciais. Estes representam a semântica dos bancos de dados a serem integrados e são usados como entrada para o processo de integração. A saída do processo é um ou mais esquemas integrados representando a semântica dos esquemas iniciais. Esses esquemas integrados poderão ser usados para receber consultas que são remitidas para os bancos de dados iniciais.

Para conseguir este objetivo, devem-se resolver os conflitos, que por sua vez exigem que transformações desses esquemas sejam feitas. A fim de resolver um conflito, o desenvolvedor deve compreender os relacionamentos semânticos entre os conceitos envolvidos no conflito. Às vezes os conflitos não podem ser resolvidos porque surgiram em conseqüência de alguma inconsistência básica. Neste caso, os conflitos são relatados aos usuários, que devem guiar o desenvolvedor em sua definição.

4.4.1 Regras de integração

Os esquemas integrados são gerados de acordo com as correspondências entre os objetos dos esquemas inicias identificadas previamente e com suas respectivas regras de integração. As regras de integração são usadas para resolver os tipos de correspondências identificadas entre os esquemas. Para cada tipo de conflito encontrado na fase anterior deve existir uma determinada regra de integração que solucione o caso.

As figuras 7, 8 e 9 mostram alguns exemplos de regras de integração que solucionam determinados tipos de correspondências conforme descritas no processo de integração realizado por SCOPIM e KATSURAGAWA (2001)

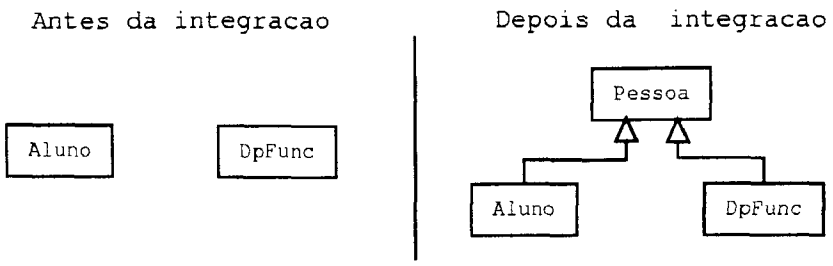

Figura 7 Regra 1 - Integrar tipo de correspondência disjunção

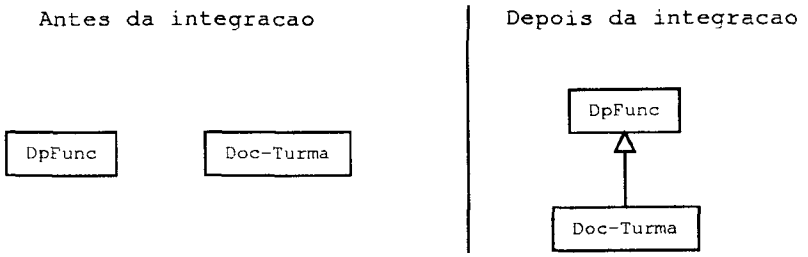

Figura 8 Regra 2 – Integrar tipo de correspondência inclusão de domínio

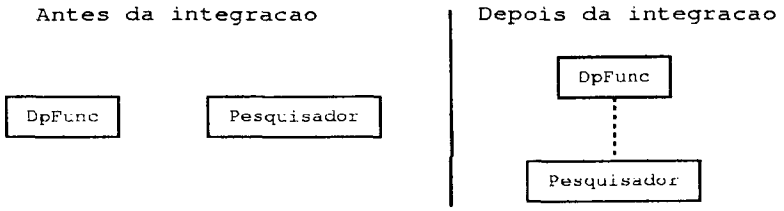

Figura 9 Regra 3 - Integrar tipo de correspondência intersecção de domínio

4.4.2 Geração dos esquemas intermediários

Após os esquemas dos bancos de dados iniciais serem modelados em um modelo de dados comum, os objetos conflitantes entre esses bancos de dados serem identificados e classificados e as regras de integração especificadas, a fase de integração está pronta para ser realizada.

Existem várias estratégias para o processamento da integração. A figura 12 mostra quatro variações possíveis denominadas estratégias de processamento da integração. Cada estratégia é mostrada na forma de uma árvore. Os nós da folha correspondem aos componentes do esquema e os nós não folha correspondem aos resultados intermediários da integração. O nó da raiz é o resultado final. A classificação preliminar das estratégias é binária contra n-ária.

As estratégias binárias permitem a integração de dois esquemas de cada vez. Estas são chamadas estratégias escada (*binary ladder*) quando um novo componente do esquema é integrado com um resultado intermediário existente em cada etapa. Uma estratégia binária é dita equilibrada quando os esquemas são divididos em pares no início e são integrados em uma forma simétrica.

As estratégias n-árias permitem a integração de n esquemas de uma só vez (n > 2). Uma estratégia n-ária é one-shot quando os n esquemas são integrados em uma única etapa; caso contrário ela é dita iterativa. A última classificação é o caso mais geral. A figura 10 mostra várias estratégias de integração.

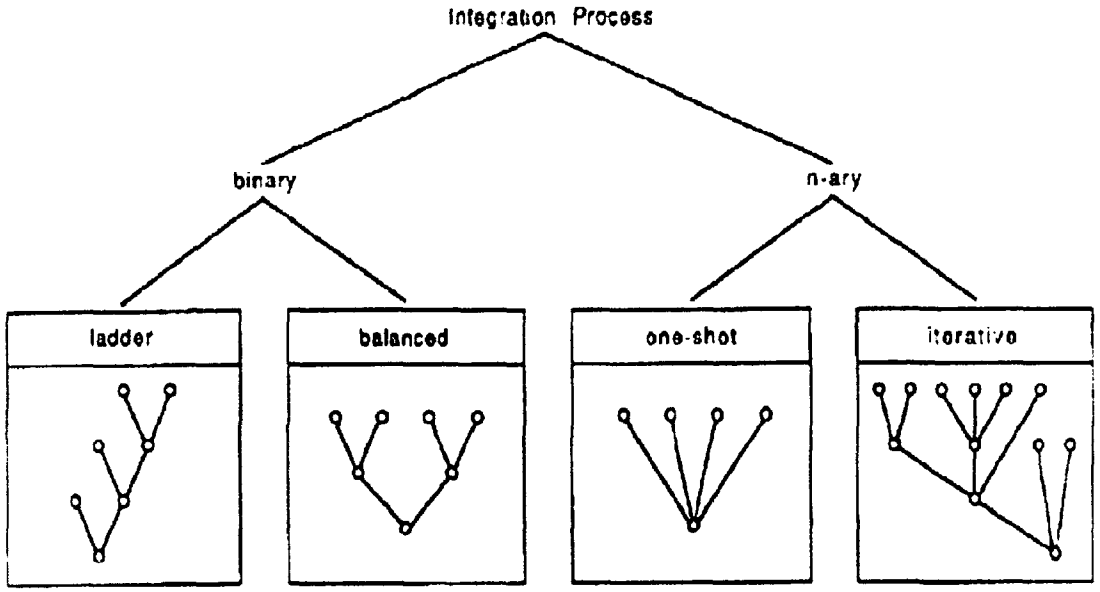

Figura 10 Tipos de Estratégias de Processamento da Integração, , BATINI (86)

A vantagem da estratégia binária é a simplificação das atividades de comparação e adequação para cada passo da integração. É evidente que a maioria das metodologias adotam estratégias binárias devido ao aumento da complexidade do processo de integração com respeito ao número de esquemas a serem integrados. Em geral, um algoritmo de união de n esquemas pode ter uma complexidade n^2. A desvantagem dessa alternativa é o aumento no número de operações de integração para se chegar a uma análise final.

A estratégia n-ária de integração realiza uma análise de n esquemas de uma vez. As óbvias vantagens disso são: um aumento considerável de análise semântica que pode ser

realizada antes da união de esquemas, evitando assim a necessidade de transformações futuras no esquema integrado; o número de passos para integração é minimizado.

Referência	Tipo de estratégia de Integraçã o	Equilíbrio
Al-Fedaghi and Scheuermann[1981]	One-shot n-ária	-
Batini e Lenzerini[1984]	Binária	Escada
Casanova and Vidal[1983]	Binária	Equilibrada
Dayal and Hwang[1984]	Binária	Sem declaração
ElMasri et al.[1987]	One-shot n-ária	-
Kahn[1979]	Binária	Sem declaração
Motro e Buneman[1981]	Binária	Sem declaração
Mannino e Effelsberg[1984]	Binária	Sem declaração
Navathe e Gadgil[1982]	n-ária interativa	-
Teorey and Fry[1982]	Binária	Equilibrada
Yao et al.[1982]	One-shot n-ária	-
Wiederhold and ElMasri[1979]	Binária	Escada

Figura 11 Estratégias de processamento de integração

Para desenvolvimento da ferramenta J-Schemas Integrator foi utilizado a técnica da estratégia binária do tipo escada (*binary ladder*).

4.4.3 Geração da informação de mapeamento entre os esquemas

O processo de integração não somente gera o esquema integrado como também possibilita a geração de informações de mapeamento entre os objetos pertencentes ao esquema integrado e os objetos pertencentes aos esquemas iniciais.

A geração da informação de mapeamento do esquema é executada concorrentemente com ambas as fases de pré-integração e geração do esquema integrado. Estabelecer mapeamentos durante a fase de pré-integração é necessário para garantir consultas corretas para os banco de dados iniciais. O mapeamento gerado durante o processo de geração do

esquema integrado envolve armazenar informações sobre mapeamento entre os objetos nos esquemas integrados (transformados) e objetos nos esquemas locais. Tais mapeamentos são importantes para construção das consultas que serão enviadas ao esquema integrado e remetidas aos esquemas iniciais.

5 A ferramenta J-Schemas Integrator

A integração de esquemas de bancos de dados é uma tarefa que não pode ser totalmente automatizada. Alguns conflitos semânticos existentes entre os esquemas não podem ser detectados por uma ferramenta de integração de esquemas e necessitam da interação humana.

Não é o objetivo desta ferramenta automatizar todo o processo de integração. A meta principal da ferramenta é servir de apoio ao processo de integração de esquemas. O usuário utilizará a ferramenta para:

a) Visualizar os esquemas a serem integrados

b) Selecionar e categorizar os conflitos existentes entre os esquemas

c) Visualizar a geração do esquema integrado corrigindo eventuais discordâncias no processo.

d) Visualizar a informação de mapeamento gerado pela ferramenta entre o esquema integrado e os esquemas iniciais.

5.1 Tecnologias utilizadas na implementação da ferramenta.

A Ferramenta J-Schemas Integrator foi implementada utilizando a linguagem de programação Java, a biblioteca JGraph e a linguagem XML .

XML é uma linguagem de marcação de dados que provê um formato para descrever dados estruturados. O Apêndice 1 desta dissertação explica com maiores detalhes a linguagem XML.

Não foi utilizada nenhuma biblioteca proprietária para a implementação desta ferramenta.

5.1.1 Mas por que Java?

Java oferece um grande conjunto de soluções para o desenvolvimento de software comercial e acadêmico reduzindo a complexidade, o tempo de desenvolvimento e facilitando a manutenção das aplicações sem comprometer a qualidade do software produzido.

- **Um padrão aberto. Plataforma independente de fornecedor.**

  A Sun Microsystems definiu um processo aberto através do qual a comunidade internacional de desenvolvedores pode participar ativamente dos rumos e decisões

acerca do desenvolvimento da tecnologia Java, o *Java Community Process* (JCP). O processo tem como objetivo permitir:

- A participação da comunidade Java na proposta, seleção e desenvolvimento das APIs Java;
- A integração da comunidade Java com outras entidades como consórcios empresariais, órgãos de padronização, grupos de pesquisa;
- A verificação através de auditorias de que o processo é rigidamente seguido pelos participantes;
- Assegurar que cada nova especificação será suportada por uma implementação de referência e um conjunto de testes de conformidade.

O JCP fomenta a evolução da tecnologia Java de maneira aberta e participativa, acompanhando o desenvolvimento tecnológico mundial e demanda por novas soluções tecnológicas.

- **Independência de plataforma.**

Java permite o desenvolvimento de soluções portáveis entre diferentes sistemas e plataformas de hardware, de desktop PC a Mainframes. Mais ainda, entre diferentes dispositivos com PDA (Personal Digital Assitant) e celulares.

"Write once, Run AnywhereTM"

Java viabiliza começar um projeto de forma mais modesta, adotando uma plataforma de hardware mais simples com software gratuito reduzindo assim o risco do investimento. Posteriormente, a solução pode ser estendida e escalonada, novos recursos podem ser agregados, ou o sistema pode ser migrado para outras plataformas de hardware e software mais robustas.

- **Um padrão de fato (mercado).**

Lançada em 1995, Java vem ao longo dos anos sendo adotada por várias empresas pelo mundo como solução tecnológica, além dos fabricantes de softwares citados acima. Em alguns segmentos, como o desenvolvimento de aplicações servidoras, Java apresenta um alto grau de maturidade e estabilidade comprovadas pelo grande número de casos de sucesso.

- **Simples e fácil de aprender, mas com qualidade.**

Java é uma linguagem de programação simples e bem projetada com uma baixa curva de aprendizado. Compromissada com as boas técnicas da Engenharia de Software, a linguagem favorece a obtenção de fatores de qualidade de software como modularidade, manutenção e extensão. Java utiliza o paradigma de orientação a objetos, um padrão que surgiu a mais de 30 anos e que na última década tem sido adotada pela maioria das empresas que buscaram evolução tecnológica.

- **Mais que uma linguagem, uma plataforma.**

  Java define não apenas uma linguagem de programação, mas também uma plataforma para o desenvolvimento de software. A plataforma Java difere das demais pelo fato de ser uma plataforma de software apenas, executando no topo de diferentes plataformas de hardware provendo um ambiente de desenvolvimento e execução de aplicações baseadas na tecnologia Java.

5.2 A Biblioteca JGraph

As bibliotecas Java disponíveis hoje para criação de interfaces para o usuário oferecem componentes para listas, tabelas e árvores, mas os componentes do tipo grafo são raramente implementadas. Por isto foi utilizado a biblioteca JGraph que é uma implementação de código aberto (*open-source*) e gratuita de um componente de grafo desenvolvimento inteiramente dentro dos padrões especificados pela JCP.

JGraph é um componente Java Swing que suporta opções de edição e display. Esta introdução esboça a estrutura geral do JGraph e a literatura que foi usada para implementá-lo. Para informações adicionais consulte o portal do JGraph em http://www.jgraph.org.

5.2.1 Visão Geral

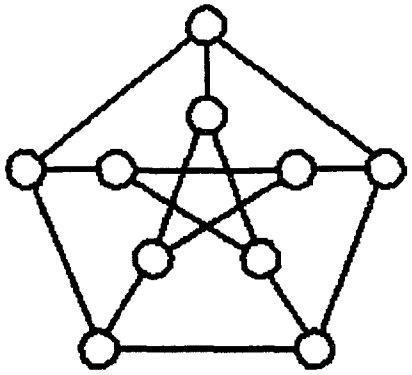

Figura 12 O grafo de Peterson

A intenção do projeto jgraph é fornecer uma implementação de um componente de grafo livre e de código aberto e inteiramente dentro dos padrões especificados pelo *Java Community Process* (JCP), comunidade responsável pela evolução das especificações e padronização do java, para construção de componentes visuais em Swing. Como um componente Swing para grafos, JGraph é baseado na teoria matemática das redes, chamada teoria de grafos, e a biblioteca Java de interface do usuário Swing, que define a arquitetura. Combinando estes dois, um componente para visualizar e editar grafos foi obtido.

Os seguintes princípios são seguidos pela implementação de JGraph.

- Compatibilidade 100% com a biblioteca Swing Java
- Solução clara e eficiente
- Tempo de download pequeno.
- 100 % Java e por isto independente de plataforma.

A idéia por trás destes princípios é baseada na experiência com outras bibliotecas de grafos, que tipicamente são frameworks grandes e complexos e que não aderem a nenhum padrão especificado.

Ao contrário, no JGraph, a arquitetura básica é a mesma utilizada como padrão nos componentes Swing, e os métodos e variáveis seguem a convenção para código Java determinado pela Sun Microsystems. Isto tem a vantagem de diminuir o tempo de aprendizado desta nova biblioteca, e o código fonte pode ser mais fácil reutilizado, resultando em um menor tempo de desenvolvimento.

5.2.2 Teoria de Grafos

O conceito de grafo é baseado em uma teoria matemática bem fundada, chamada de teoria de grafo, que é rigorosamente definida em BIGGS (1989), AIGNER(1993) e OTTMANN (1993). Um grafo consiste de vértices e arestas conectando pares de vértices. O exato padrão geométrico para ligação não é especificado.

Como uma generalização matemática das listas e das árvores, os grafos entram em cena onde as listas e as árvores não são suficientes para modelar as relações entre os objetos. Por exemplo, em uma árvore cada nó tem no máximo um pai, e zero ou mais filhos, enquanto que em um grafo, cada nó tem simplesmente zero ou mais vizinhos. (no exemplo da lista, cada nó tem no máximo dois vizinhos).

O termo simples vizinho enfatiza um conceito mais genérico que substitui o termo pai e filho utilizado no contexto de árvores. O termo célula será usado no lugar do termo nodos durante toda a explicação neste documento para distinguir entre elementos de grafos dos elementos de listas e árvores.

Formalmente, um grafo G consiste de um conjunto não vazio de elementos V(G) e um subconjunto E(G) de um conjunto de pares não ordenados de elementos distintos de V(G). Os elementos de V(G) chamados vértices de G, podem ser representados por pontos. Se (x,y)) Î E(G), então a aresta (x,y) pode ser representado por um arco juntando x e y. Então é dito que x e y são adjacentes, e a aresta (x,y) é incidente com x e y. Se (x,y) não é uma aresta, então os vértices x e y são ditos não adjacentes.

A Teoria de Grafos também oferece algoritmos padrões para resolver problemas comuns em grafos. Por exemplo, o algoritmo de Dijkstra pode ser usado para encontrar o menos caminho entre dois vértices de um grafo. O algoritmo Dijkstra e outros algoritmos padrões são explicados em SEDGEWICK (1991), NIEVERGELT(1993) e HAREL(1992). (Uma implementação do algoritmo Dijkstra já vem junto com a aplicação exemplo na distribuição do Jgraph).

Resumindo, grafos são usados como um paradigma dentro da biblioteca Jgraph para visualizar uma rede de objetos relacionados. Uma rodovia, uma rede de computadores, uma molécula, a arquitetura de um software ou **um esquema de banco de dados são** exemplos de grafos. Como grafos são generalizações matemáticas de listas e árvores, a biblioteca JGraph pode ser usado também para mostrar estrutura simples, como por exemplo a estrutura de árvore de um sistema de arquivos.

5.2.3 A Biblioteca Swing

Swing é uma biblioteca gráfica que já vem embutida dentro da plataforma Java e provê elementos que podem ser colocados na janela de uma aplicação. Tais elementos são chamados de componentes da interface do usuário ou simplesmente de componentes. Exemplos de componentes comuns são botões, listas, árvores e tabelas.

Swing é baseado na AWT que significa Abstract Windowing Toolkit ou caixa de ferramentas abstrata para criação de janelas. AWT é uma outra biblioteca gráfica que vem junto na plataforma Java e pode ser utilizado independente de Swing. Portanto você pode escolher se deseja trabalhar com a biblioteca AWT ou com a biblioteca Swing quando for desenvolver aplicações gráficas com a linguagem Java. Ao contrário de AWT, Swing é mais sofisticado e provê muito mais funcionalidades e componentes que AWT.

Uma boa introdução para Java e AWT é dada em BISHOP(1998), JACKSON(1999), FLANAGAN(1997) e STÄRK(2001) é indicado para programadores mais experientes. Ao contrário de AWT, Swing é somente explicado em GEARY(1999), e existe um tutorial web muito bom de Swing em *THE SWING TUTORIAL*[9].

As referências mencionadas acima discutem com grandes detalhes os componentes existentes na biblioteca Swing e AWT, mas não explica como criar novos componentes. A biblioteca Jgraph é justamente a implementação de uma família de componentes que são adicionados na arquitetura do Swing.

O código fonte de Swing deve ser usado para elaborar as informações requeridas:

[9] *The Swing Tutorial*, http://java.sun.com/docs/books/tutorial/index.html, Sun Microsystems

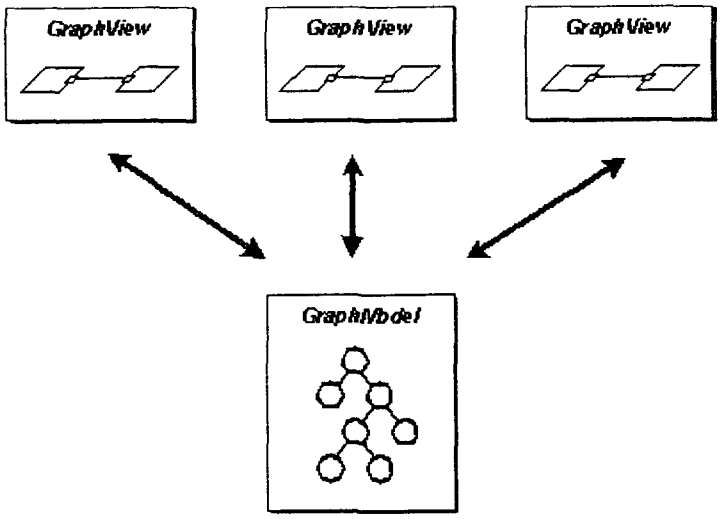

Figura 13 Model-View-Control (MVC)

O padrão de projeto MVC provê uma separação clara do código de uma aplicação em 3 camadas: Modelo (dados da aplicação), Controle (camada que faz todo o controle da aplicação) e Visualização (camada responsável por renderizar os dados na tela). O Swing MVC é usado para separar o modelo da visualização e estabelecer uma relação entre eles, tal qual mais de uma visualização pode ser relacionada com o mesmo modelo, e todas as visualizações são automaticamente atualizadas quando o modelo muda.

Jgraph é largamente baseado no componente de árvores do Swing, chamado JTree, o qual é explicado em *JTree API Specification*[10]. Algumas idéias foram adotadas dos componentes de texto do Swing, chamados de elementos da interface, que é discutido em Violett - *The Element Interface*[11] e as visualizações em *View Interface API Specification*[12]. A classe JGraph é uma extensão (sub-classe) da classe JComponent, que é a classe pai de todos os componentes Swing.

JGraph se enquadra em todos os padrões Swing, tais como look & feel plugáveis, transferência de dados entre os componentes, acessibilidade, internacionalização e serialização. Também, todos os conceitos utilizados no JGraph são conceitos bem conhecidos do Swing. Para características mais avançadas tais como fazer/desfazer, impressão e suporte a XML, padrões adicionais do Swing foram usados. Além de seguir os padrões do Swing, a

[10] *JTree API Specification*, http://java.sun.com/j2se/1.4/docs/api/javax/swing/JTree.html, Sun Microsystems.
[11] Violett, *The Element Interface*, http://java.sun.com/products/jfc/tsc/articles/text/element_interface/, Sun Microsystems

biblioteca JGraph também está em conformidade com a convenção de nomenclatura de nomes de métodos e variáveis para o código fonte e comentários no padrão javadoc.

5.2.4 Decompondo o JGraph

A decomposição do JGraph dentro de módulos divide a definição global de grafos e MVC em uma série de sub-definições tratáveis. Seguindo o padrão de projeto MVC, o framework é dividido dentro de pacotes que definem modelo, visão, controle e modelo de eventos.

5.2.5 Fontes e Literaturas relacionadas

O componente JGraph é largamente baseado em *JTree API Specification*[13]. Algumas idéias foram usadas de Violett - *The Element Interface*[14]. As descrições dos padrões de projetos usam os nomes descritos por GAMMA e outros (1995)

Um livro padrão sobre Swing Geary(1999) é certamente uma boa base para entender a arquitetura do Jgraph. Algumas implementações de algoritmos de grafos do Jgraph foram baseadas em SEDGEWICK(1991) e NIEVERGELT and HINRICHS(1993). Também, para entender a fundo a biblioteca JGraph, o desenvolvedor tem que estar familiarizado com os conceitos mais importantes da teoria de grafos, que podem ser obtidos em BIGGS (1989)

A especificação (API) para utilizar os componentes disponibilizados pela biblioteca JGraph podem ser encontradas no portal onde está hospedado o projeto JGraph. http://www.jgraph.org.

5.2.6 Conclusões

A biblioteca JGraph (utilizada pela ferramenta J-Schemas Integrator) suporta opções estendidas de edição e visualização de grafos, mas a semântica do grafo, isto é, o significado de vértices e arestas, é definido pela aplicação. Entretanto, a biblioteca JGraph é altamente

---

[12] ***View Interface API Specification***, http://java.sun.com/j2se/1.4/docs/api/javax/swing/text/View.html, Sun Microsystems
[13] ***JTree API Specification***, http://java.sun.com/j2se/1.4/docs/api/javax/swing/JTree.html, Sun Microsystems.
[14] Violett, ***The Element Interface***, http://java.sun.com/products/jfc/tsc/articles/text/element_interface/, Sun Microsystems

customizável, e a classe provê ganchos para subclasses. Em casos nos quais customização não é suficiente, o código fonte do componente é livre e disponível para que possamos mudar a implementação padrão.

5.3 Interface do Usuário

O editor gráfico da ferramenta possui três frames principais. Os dois frames localizados na parte superior do editor possibilitam ao usuário carregar dois esquemas a serem integrados. O frame localizado na parte inferior da janela principal é responsável por mostrar o esquema integrado na medida em que este for sendo integrado e gerado.

Em cima de cada um dos três frames, conforme mostra a figura 14, existe uma paleta de botões para que o usuário possa:

a) Carregar um esquema ERC+ a partir de um documento XML

b) Salvar um esquema ERC+ para um documento XML

c) Remover tabelas e associações entre tabelas do modelo

d) Mostrar todos os atributos (campos) de uma determinada tabela ou associação.

Entre os dois frames superiores existe uma paleta de botões correspondente as regras de integração implementadas e disponíveis na ferramenta. A qualquer momento novas regras podem ser construídas e adicionadas a arquitetura da ferramenta.

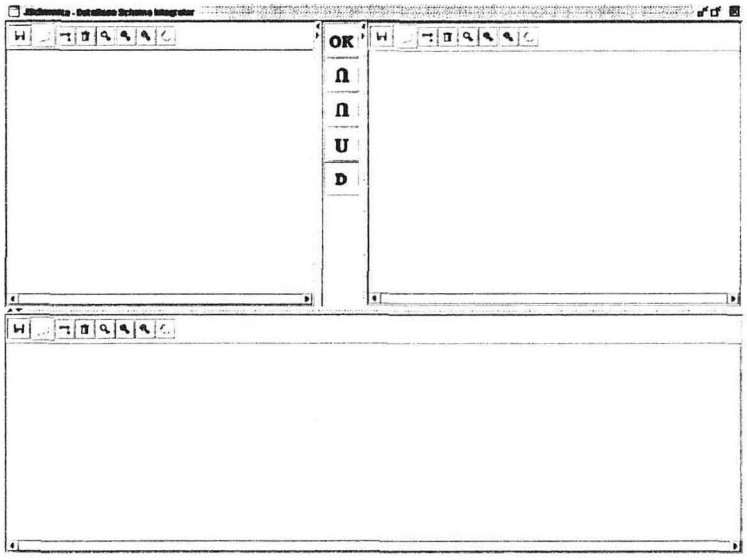

Figura 14 Interface visual da ferramenta J-Schemas Integrator

Todos os ícones na tela apresentam *tool tips*, isto é, quando o usuário passar o mouse sobre estes ícones, alguma ajuda (dica) lhe será fornecida indicando o propósito do ícone.

5.4 Representação XML de um Esquema ERC+

O editor gráfico da ferramenta possibilita o carregamento de esquemas no modelo ERC+ a partir de documentos XML. A ferramenta nesta primeira versão não possibilita que o esquema seja carregado diretamente de um SGBD[15]. Portanto, na primeira fase de integração, que é a fase de pré-integração, o integrador deverá representar o esquema a ser integrado através de um documento XML.

Em um documento XML as regras que definem um documento são ditadas por DTDs[16], as quais ajudam a validar os dados quando a aplicação que os recebe não possui internamente uma descrição do dado que está recebendo. Para maiores detalhes sobre XML e DTD consultar o apêndice 1 desta dissertação.

Foi definido a DTD, no qual é representado o modelo de esquema XML que é interpretado pela ferramenta no momento da leitura dos dados para representação gráfica no editor. O DTD que representa o esquema ERC+ suportado pela ferramenta pode ser visualizado na figura 15.

[15] Sistema Gerenciador de Banco de Dados
[16] Document Type Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DatabaseSchema (Tables, Links)>
<!ELEMENT Tables (Table+)>
<!ELEMENT Links (Link+)>
<!ELEMENT Table (Attribute+)>
<!ATTLIST Table
    type CDATA #REQUIRED
    name CDATA  #REQUIRED
>
<!ELEMENT Link EMPTY>
<!ATTLIST Link
    type CDATA #REQUIRED
    source CDATA #REQUIRED
    target CDATA #REQUIRED
    cardinality CDATA #REQUIRED
>
<!ELEMENT Attribute (Attribute*)>
<!ATTLIST Attribute
    name CDATA #REQUIRED
    domain CDATA #REQUIRED
    obs CDATA #REQUIRED
>
```

Figura 15 DTD do esquema ERC+.

No XML que representa o esquema foi definido suporte a todos os objetos suportados no modelo ERC+. O nó root do documento XML deve chamar-se *DatabaseSchema* e deve conter dois nós filhos: *Tables* e *Links*.

O nó *Tables* representa o conjunto de todas as tabelas contidas no modelo ERC+ a ser integrado. Estas tabelas podem ser do tipo TE (Tipo de Entidade) ou TA (Tipo de Associação). Abaixo da hierarquia do nó *Tables* podem existir um ou mais nós filhos chamados *Table*.

Cada nó *Table* representa uma tabela modelada no modelo ERC+. Cada tabela poderá ser categorizada em TE ou TA dependendo de sua finalidade e obrigatoriamente deverá ser dados um nome para cada tabela. Dependendo do tipo da tabela TE ou TA, estas serão visualizadas com cores diferentes pela ferramenta gráfica do editor de integração. Abaixo da hierarquia do nó *table* podem existir um ou mais nós filhos chamado *Attribute* que representarão os atributos mono ou multivalorados da tabela em questão.

Para cada nó *Attribute* deverá ser obrigatório associar um nome e o domínio do atributo. Como XML é um modelo hierárquico para definirmos o conceito de atributos multi-valorados existente no modelo ERC+, definimos uma hierarquia de nós *Attribute*. Assim

podemos ter nós *Attribute* dentro de nós *Attribute*. Podemos visualizar um exemplo de instância de um DTD com atributos multivalorados através da figura 16.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DatabaseSchema>
    <Tables>
        <Table type="TE" name="Aluno">
            <Attribute name="NUMALU-ALU" domain="NUMBER(07)" obs="REQUIRED NULL 0"/>
            <Attribute name="NOMEALU-ALU" domain="ALPHA(41)" obs="REQUIRED NUL"/>
            <Attribute name="ENDERECO" domain="MULTI" obs="SIZE VARYING">
                <Attribute name="RUA-ALU" domain="ALPHA(20)" obs="REQUIRED NULL 0"/>
                <Attribute name="BAIRRO-ALU" domain="ALPHA(20)" obs="REQUIRED NUL"/>
            </Attribute>
        </Table>
        <Table type="TE" name="Programa">
            <Attribute name="NUMALU-PROG" domain="NUMBER(07)" obs="REQUIRED NULL 0 matrícula do aluno"/>
            <Attribute name="CODCUR-PROG" domain="ALPHA(04)" obs="REQUIRED NULL código do curso"/>
        </Table>
        <Table type="TE" name="Disciplina">
            <Attribute name="SIGLA-DISCIP " domain="ALPHA(06)" obs="REQUIRED NULL"/>
            <Attribute name="NOME-DISCIP " domain="ALPHA(50)" obs="REQUIRED NULL"/>
            <Attribute name="DEPTO-DISCIP " domain="ALPHA(10)"/>
        </Table>
        <Table type="TA" name="Matricula">
            <Attribute name="NUMALU-MATRIC" domain="NUMBER(7)" obs="REQUIRED NULL 0"/>
            <Attribute name="CODCUR-MATRIC" domain="ALPHA(04)" obs="REQUIRED NULL"/>
            <Attribute name="TURMA-MATRIC" domain="ALPHA(02)" obs="REQUIRED NULL turma"/>
            <Attribute name="DISCIP-MATRIC" domain="ALPHA(6)" obs="REQUIRED NULL código da disciplina"/>
        </Table>
    </Tables>
    <Links>
        <Link type="NORMAL" source="Matricula" target="Aluno" cardinality="N-1"/>
        <Link type="NORMAL" source="Matricula" target="Disciplina" cardinality="N-1"/>
        <Link type="NORMAL" source="Matricula" target="Programa" cardinality="1-1"/>
    </Links>
</DatabaseSchema>
```

Figura 16 Exemplo de um documento XML contendo esquema ERC+ com atributos multivalorados

O nó *Links* agrupa todos os links existentes entre as tabelas do modelo ERC+. Cada nó filho do nó Link representa um determinado link entre duas tabelas do modelo ERC+. Estes links podem ser do tipo NORMAL e nestes casos somente expressam a cardinalidade da ligação, ou estes links podem ser do tipo *IS-A* ou *MAY-BE-A* que são dois tipos de links existentes no modelo ERC+. Um exemplo de link *IS-A* pode ser observado na Figura 17.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <DatabaseSchema>
  - <Tables>
    - <Table type="TE" name="PESSOA">
        <Attribute name="NUMFUNC-DP" domain="NUMBER(6)" obs="REQUIRED NULL LOW-VALUE" />
        <Attribute name="NOME-DP" domain="ALPHA(41)" obs="REQUIRED NULL" />
        <Attribute name="ESPECIALIDADE-DP" domain="ALPHA(8)" />
        <Attribute name="NUMRG-DP" domain="ALPHA(15)" />
        <Attribute name="TIPODOCUM-ALU" domain="ALPHA(06)" />
      </Table>
    - <Table type="TE" name="DPFunc">
        <Attribute name="SUBAGENCIA-DP" domain="ALPHA(1)" />
        <Attribute name="NUMCONTA-DP" domain="NUMBER(8)" />
        <Attribute name="DIGCONTA-DP" domain="ALPHA(1)" />
        <Attribute name="NACION-DP" domain="NUMBER(3) INITIALVALUE IS 1" />
        <Attribute name="CONVENIO-DP" domain="ALPHA(1)" />
      </Table>
    - <Table type="TE" name="Aluno">
        <Attribute name="NUMCPF-ALU" domain="NUMBER(12)" />
        <Attribute name="NUMMILI-ALU" domain="ALPHA(12)" />
        <Attribute name="SERMILI-ALU" domain="ALPHA(01)" />
        <Attribute name="ESCOLA2GRAU-ALU" domain="ALPHA(60)" obs="SIZE VARYING" />
        <Attribute name="CIDADE2GRAU-ALU" domain="ALPHA(30)" obs="SIZE VARYING" />
        <Attribute name="PAIS2GRAU-ALU" domain="ALPHA(15)" obs="SIZE VARYING" />
      </Table>
    </Tables>
  - <Links>
      <Link type="IS_A" source="DPFunc" target="PESSOA" />
      <Link type="IS_A" source="Aluno" target="PESSOA" />
    </Links>
</DatabaseSchema>
```

Figura 17 Outro exemplo de um esquema ERC+ mostrando links do tipo IS-A

5.5 Visualização dos Esquemas

A ferramenta possibilita a **visualização** em uma interface gráfica, de esquemas de bancos de dados representado no modelo ERC+. Um modelo ERC+ representando o esquema deve estar descrito em um documento XML conforme descrito na seção anterior. Esses documentos XML contém a representação dos esquemas a serem visualizados dentro do editor. O usuário poderá visualizar os dois esquemas que serão integrados nas duas janelas localizadas no frame superior do editor. No frame inferior, o usuário poderá visualizar o esquema gerado (esquema integrado) a partir dos esquemas iniciais.

O usuário pode carregar o esquema a partir de um documento XML ou poderá salvar a modelagem apresentada na tela para um documento no formato XML para posterior leitura. O formato do XML terá que obedecer ao DTD especificado pela seção anterior, como mostrado na figura 15.

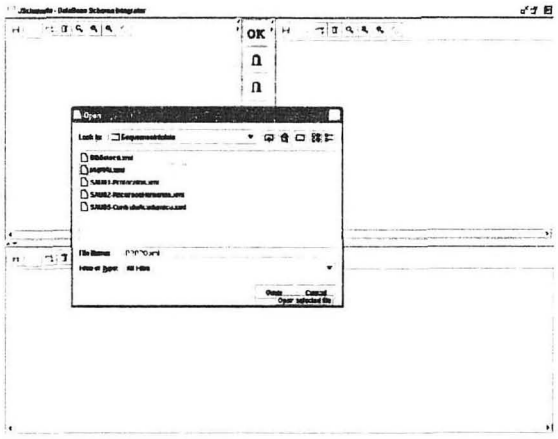

Figura 18 Ferramenta está tentando abrir um esquema ERC+

a partir de um documento XML

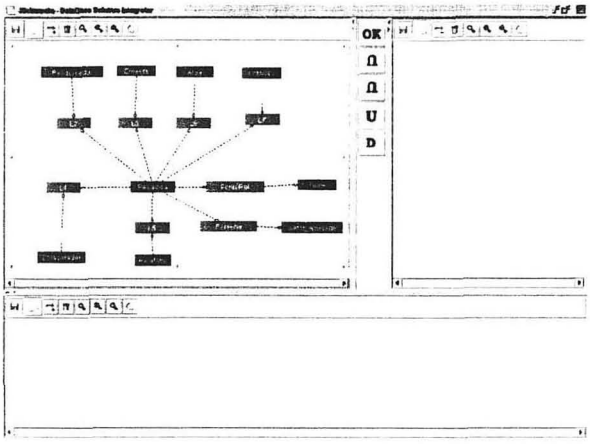

Figura 19 Esquema visual ERC+ é mostrado no frame após ser carregado

a partir do arquivo XML

Quando o modelo ERC+ é carregado, somente as tabelas e os links são mostrados na interface gráfica. O usuário poderá aumentar ou diminuir o zoom, e poderá reorganizar o modelo para que os objetos contidos nele fiquem na posição que mais agradar ao usuário. A biblioteca JGraph utilizada para suporte a utilização de grafos em um ambiente gráfica, disponibiliza vários algoritmos para ordenação e reagrupamento do objetos contidos na tela. Nesta primeira versão desta ferramenta não foi possível adicionar chamadas as estes algoritmos, mas pode ser uma melhoria futura a ser feita na ferramenta. No esquema

visualizado pelo usuário na tela, as tabelas do tipo TE (Tipo de Entidade) aparecem com destaque de cor diferente das tabelas do tipo TA (Tipo de Associação).

Também é possível visualizar os atributos de uma determinada tabela seja ela um TE ou um TA conforme visto na figura 21. Para visualizar os atributos de uma determinada tabela, é necessário selecionar o objeto na tela e clicar sobre o ícone correspondente na paleta de botões localizado acima de todos os frames internos.

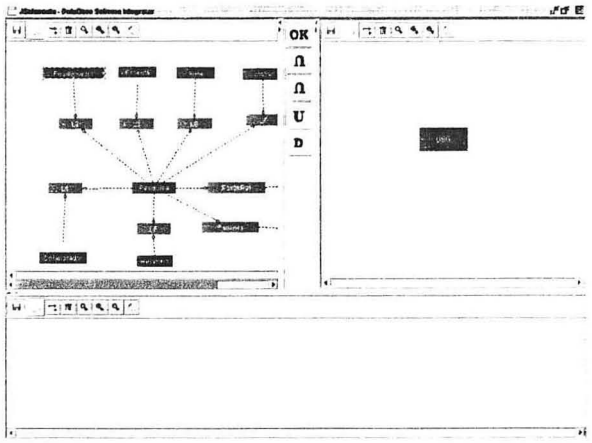

Figura 20 Dois esquemas ERC+ carregados pela ferramenta. Um em cada frame superior

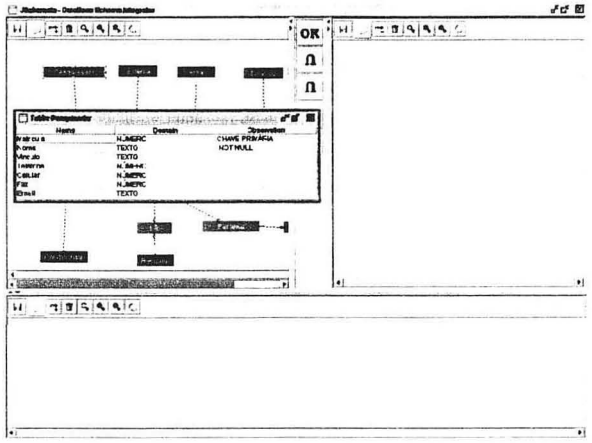

Figura 21 Atributos mostrados para uma determinada tabela selecionada no frame superior esquerdo

Para visualizar os atributos do tipo multivalorados suportados pelo modelo ERC+, basta selecionar a linha correspondente ao atributo multivalorado na tabela de atributos. Os atributos do tipo multivalorado devem ter o domínio igual a **MULTI** na definição de domínio

do documento XML. Um exemplo de visualização de atributo multivalorado pode ser observado na figura 22.

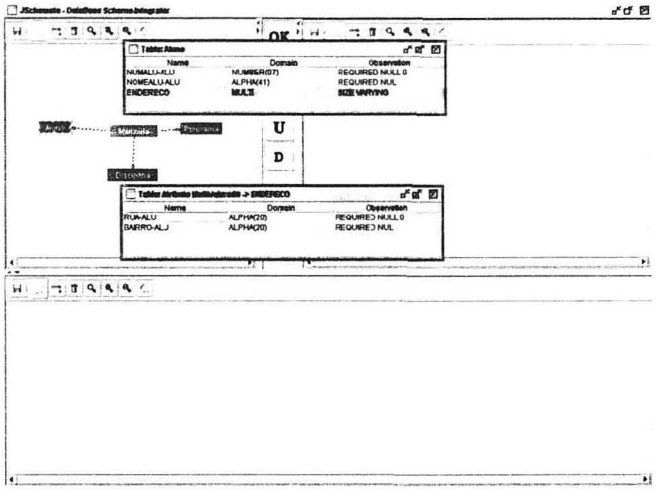

Figura 22 Exemplo de visualização de atributo multivalorado

5.6 Seleção e categorização dos objetos conflitantes entre os esquemas

Uma vez carregado os esquemas ERC+ a serem integrados na ferramenta, o integrador poderá utilizar as regras de integração existentes na ferramenta para identificar e resolver os conflitos existentes entre os objetos dos esquemas.

A ferramenta J-Schemas Integrator possui uma arquitetura flexível na qual podem ser inseridas novas regras de integração que solucionarão algum tipo de conflito existente entre os objetos dos esquemas. As regras de integração são identificadas pela paleta de botões existente entre os dois frames superiores.

Para que o usuário identifique e resolva determinado tipo de conflito existente entre dois objetos, ele deve executar o seguinte procedimento:

1. Selecionar na paleta de botões, qual regra de integração o integrador desejará utilizar.
2. Uma vez selecionada a regra de integração, um novo painel gráfico é aberto na tela, solicitando que o integrador de esquemas indique quais objetos nos esquemas iniciais estão em conflito.
3. Quando o integrador selecionar os objetos em conflitos, a regra automaticamente será encarregada de solucionar o conflito, gerando

informação para o esquema integrado e gerando informação de mapeamento do esquema integrado para os esquemas iniciais.

4. Estas informações geradas pela regra de integração são armazenadas em um documento XML e enviada para o módulo integrador de esquemas para que este gere parcialmente o esquema integrado.

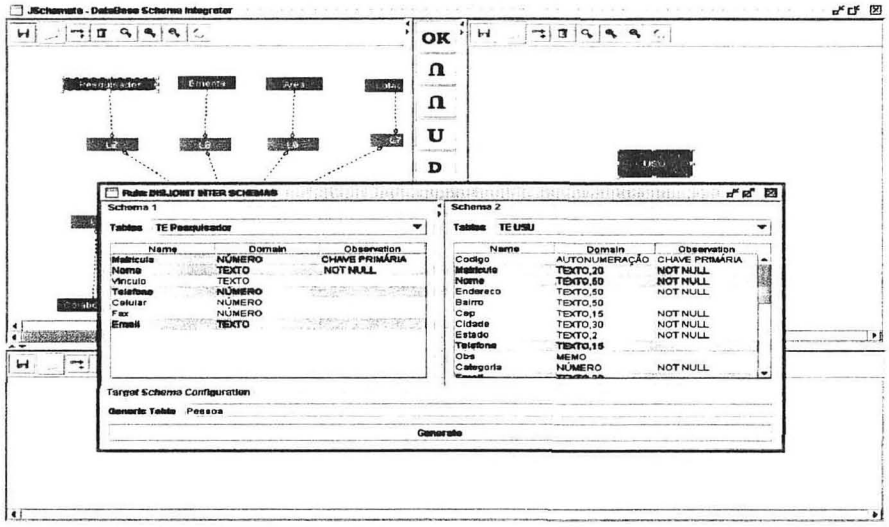

Figura 23 Regra de integração invocada pelo usuário pedindo as informações necessárias para solução do conflito.

Para esta versão da ferramenta foram implementadas cinco regras de integração para validar o funcionamento da ferramenta. Estas regras são as regras utilizadas para solucionar os conflitos encontrados no processo de integração de esquemas das bases de dados da UFPR executado por SCOPIM e KATSURAGAWA (2001). Porém novas regras poderão ser implementadas. As regras implementadas foram:

1. Intersecção de conjuntos entre duas tabelas de esquemas diferentes.
2. Disjunção de atributos entre tabelas de esquemas diferentes (figura 23)
3. Disjunção de atributos entre tabelas do mesmo esquema.
4. Tipo de dados conflitantes entre duas tabelas.
5. Migrar objetos do esquema inicial para o esquema integrado sem realizar transformações.

5.7 Geração da Informação de Mapeamento

A informação de mapeamento serve para coletar dados de transformação entre os objetos do esquema inicial para o esquema integrado. Estes dados poderão ser usados futuramente para fazer as transformações necessárias quando uma consulta for submetida ao esquema integrado e tiver que ser re-submetida aos esquemas iniciais.

Todas as regras de integração inseridas na ferramenta devem gerar como saída de seu processo, isto é, como saída do resultado do processo de resolução de conflito entre os objetos dos esquemas iniciais, uma especificação descrevendo o processo de integração e a informação de mapeamento gerada durante o processo de solução do conflito solucionado.

Toda regra de integração gera um documento XML, o qual contém o estado anterior dos objetos antes de serem integrados e o estado atual dos objetos no modelo integrado, isto é, o estado após o processo de integração.

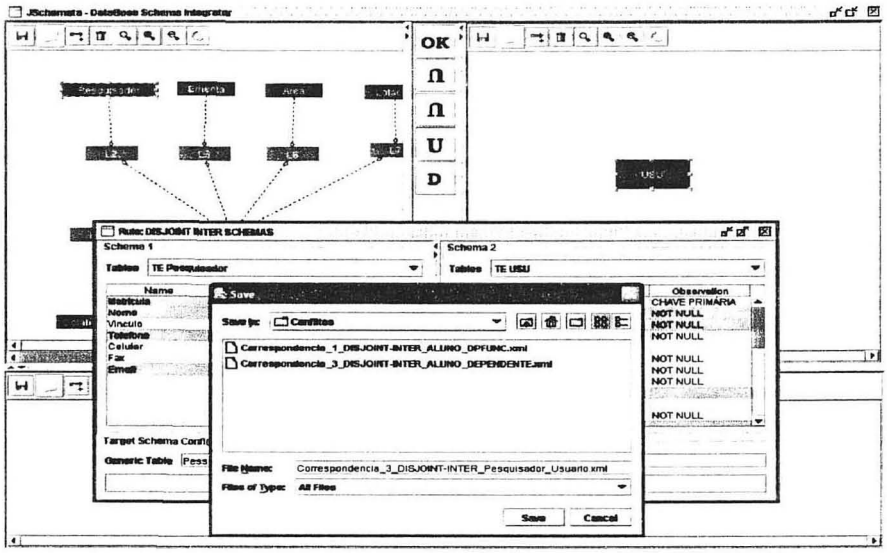

Figura 24 Sempre após a resolução de alguma correspondência entre os objetos conflitantes, as regras de integração salvam a informação de mapeamento em um documento XML.

Foi estabelecido como padrão a geração de um documento XML contendo a informação de mapeamento e a informação utilizada para o processo de integração pelo módulo integrador. Também foi definido um DTD com o padrão estrutural das *tags* XML para o documento descrevendo a informação de mapeamento. O DTD especificado para representação da informação de mapeamento pode ser observada na figura 25.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DatabaseSchema (MappingInformation, Tables, Links)>
<!ELEMENT MappingInformation (Table+, Link+)>
<!ELEMENT Tables (Table+)>
<!ELEMENT Attribute EMPTY>
<!ELEMENT Links (Link+)>
<!ELEMENT Table (Attribute+)>
<!ATTLIST Table
    type CDATA #REQUIRED
    name CDATA #REQUIRED
>
<!ELEMENT Link EMPTY>
<!ATTLIST Link
    type CDATA #REQUIRED
    source (Aluno | DPFunc) #REQUIRED
    target CDATA #REQUIRED
>
<!ATTLIST Attribute
    name CDATA #REQUIRED
    domain CDATA #REQUIRED
    obs CDATA #IMPLIED
>
```

Figura 25 DTD utilizado pelas regras de integração para geração da informação de mapeamento

Na figura 27 é mostrado um exemplo da informação de mapeamento gerada após aplicada a regra de integração do tipo DISJUNÇÃO para resolver um conflito existente entre dois objetos de esquemas a serem integrados.

O exemplo ilustrado mostra a resolução de conflito existente entre uma tabela denominada Aluno pertencente a um esquema e uma tabela denominada DpFunc pertencente a um outro esquema. A tabela Aluno armazena a população de alunos de uma Universidade enquanto que a tabela de DpFunc armazena a população de funcionários de uma Universidade. O conflito do tipo DISJUNÇÃO ocorre quanto tabela com populações diferentes possuem atributos em comum. Tanto aluno quanto funcionários possuem nome, endereço, telefone, e muitos outros atributos em comum. A solução para este tipo de conflito é criar uma entidade mais genérica generalizando os atributos em comum a estas duas tabelas

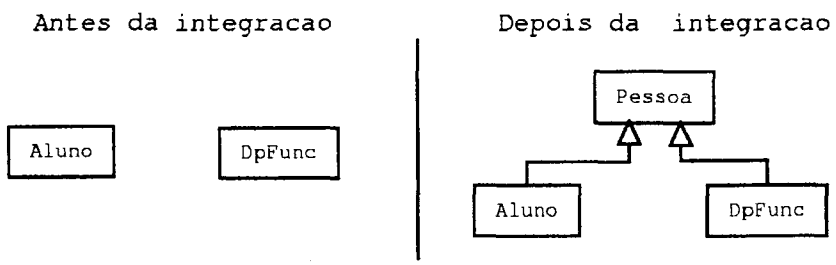

Figura 26 Exemplo de solução para o *conflito do tipo* DISJUNÇÃO

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <DatabaseSchema>
  - <MappingInformation>
    - <Table type="TE" name="DPFunc" y="22" x="264" width="90" height="20">
        <Attribute name="NUMFUNC-DP" domain="NUMBER(6)" obs="REQUIRED NULL LOW-VALUE" />
        <Attribute name="NOME-DP" domain="ALPHA(41)" obs="REQUIRED NULL" />
        <Attribute name="DATANASC-DP" domain="NUMBER(6)" obs="REQUIRED NULL LOW-VALUE" />
      </Table>
    - <Table type="TE" name="Aluno" y="209" x="579" width="50" height="20">
        <Attribute name="NUMALU-ALU" domain="NUMBER(07)" obs="REQUIRED NULL 0" />
        <Attribute name="NOMEALU-ALU" domain="ALPHA(41)" obs="REQUIRED NUL" />
        <Attribute name="ENDERECO-ALU" domain="ALPHA(36)" obs="SIZE VARYING" />
        <Attribute name="COMPLENDER-ALU" domain="ALPHA(15)" obs="SIZE VARYING Complemento do endereço do aluno" />
      </Table>
    </MappingInformation>
    <Tables>
    - <Table type="TE" name="PESSOA">
        <Attribute name="NUMFUNC-DP" domain="NUMBER(6)" obs="REQUIRED NULL LOW-VALUE" />
        <Attribute name="NOME-DP" domain="ALPHA(41)" obs="REQUIRED NULL" />
        <Attribute name="DATANASC-DP" domain="NUMBER(6)" obs="REQUIRED NULL LOW-VALUE" />
      </Table>
    - <Table type="TE" name="DPFunc">
        <Attribute name="SUBAGENCIA-DP" domain="ALPHA(1)" />
        <Attribute name="NUMCONTA-DP" domain="NUMBER(8)" />
        <Attribute name="DIGCONTA-DP" domain="ALPHA(1)" />
      </Table>
    - <Table type="TE" name="Aluno">
        <Attribute name="NUMCPF-ALU" domain="NUMBER(12)" />
        <Attribute name="NUMMILI-ALU" domain="ALPHA(12)" />
        <Attribute name="SERMILI-ALU" domain="ALPHA(01)" />
      </Table>
    </Tables>
  - <Links>
      <Link type="IS_A" source="DPFunc" target="PESSOA" />
      <Link type="IS_A" source="Aluno" target="PESSOA" />
    </Links>
  </DatabaseSchema>
```

Figura 27 Exemplo de informação de mapeamento gerado durante o processo de integração e resolução de conflitos.

Um template de integração é uma implementação de uma regra de integração. Este template deve solucionar determinado tipo de conflito entre objetos. Novos templates de integração podem ser inseridos na ferramenta através da configuração de um arquivo de configurações utilizado pela ferramenta.

Neste arquivo deve ser configurado um identificador para esta regra de integração, uma imagem (que é a imagem que aparece no caixa de botões localizado entre os dois frames superiores do editor) e o nome da classe Java que deverá ser instanciada para que a regra de integração possa executar. O arquivo de configuração de templates atual pode ser visualizado através da figura 28.

Assim a ferramenta irá permitir que à medida que novos templates forem criados, eles possam ser plugados na arquitetura da ferramenta e possam ser utilizados para solucionar algum tipo de conflito particular.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Rules>
  - <Mapping>
      <rule id="None Conflict" gif="images/ok.png" toolTip="NONE CONFLICT"
        guiJavaClass="org.ufpr.mestrado.kemmel.integrator.rules.noneconflict.FrameGUI" />
      <rule id="DISJOINT INTER-Schemas" gif="images/disjoint.png" toolTip="DISJOINT INTER-SCHEMAS"
        guiJavaClass="org.ufpr.mestrado.kemmel.integrator.rules.disjointInter.FrameGUI" />
      <rule id="DISJOINT INTRA-Schemas" gif="images/disjoint.png" toolTip="DISJOINT INTRA-SCHEMAS"
        guiJavaClass="org.ufpr.mestrado.kemmel.integrator.rules.disjointIntra.FrameGUI" />
      <rule id="INTERSECTION" gif="images/union.png" toolTip="INTERSECTION"
        guiJavaClass="org.ufpr.mestrado.kemmel.integrator.rules.intersection.FrameGUI" />
      <rule id="DataType" gif="images/data.png" toolTip="CONFLICTING DATA TYPE"
        guiJavaClass="org.ufpr.mestrado.kemmel.integrator.rules.dataType.FrameGUI" />
    </Mapping>
  </Rules>
```

Figura 28 Arquivo de configuração da ferramenta na qual são mapeadas as regras de integração. Novas regras de integração deverão ser incluídas neste arquivo.

Um template de integração receberá como entrada de seu processamento os objetos que estão em conflitos nos esquemas iniciais

Um template de integração gerará como saída de seu processamento as seguintes informações: objeto integrado que deverá ser gerado no esquema integrado e informações de mapeamento entre este objeto integrado e os objetos nos esquemas iniciais.

5.8 Geração do esquema integrado.

O esquema integrado estará sendo gerado, na medida em que os conflitos existentes entre os objetos forem sendo identificados e categorizados pelo integrador de esquemas. A cada conflito identificado pelo integrador, a ferramenta gerará a informação (representada através de um documento XML) correspondente aos objetos que estão sendo incluídos no esquema integrado. Também será gerado pela ferramenta neste momento uma outra informação (representada através de um outro documento XML) da informação de mapeamento entre estes objetos incluídos no esquema integrado e os pertencentes ao esquema inicial.

Na medida que o usuário for clicando sobre os objetos dos esquemas iniciais e for categorizando os conflitos existentes entre eles, o usuário poderá visualizar o esquema integrado sendo gerado e a informação de mapeamento existente entre os esquemas. Portanto, existirão três janelas de visualização na interface gráfica do editor.

O esquema integrado gerado é armazenado no mesmo padrão (DTD) dos esquemas de dados iniciais. Isto é necessário, pois este mesmo esquema integrado pode ser reutilizado para

a integração com um outro esquema qualquer, conforme descrito na estratégia de integração escada binária (*binary ladder*).

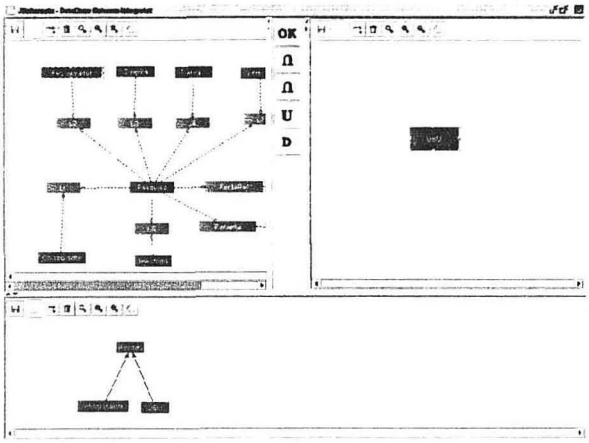

Figura 29 Exemplo de esquema integrado gerado (frame inferior) após solução de tipo de conflito DISJUNÇÃO

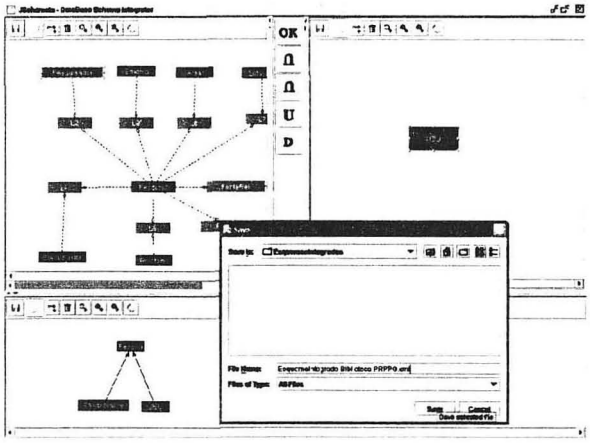

Figura 30 O esquema integrado gerado pode ser salvo para o formato XML para que possa ser utilizado em futuras integrações com outros esquemas.

5.9 Limitação da Ferramenta J-Schemas Integrator

Normalmente, a obtenção do diagrama da base de dados seguindo algum tipo de modelo físico ou lógico, é efetuada a partir de conexão direta com a base de dados. Para isto é necessário ter algum driver específico para conexão com o determinado tipo de banco. Porém a ferramenta J-Schemas Integrator limita-se a fazer a leitura do diagrama a partir de documentos XML localizados fisicamente no sistema de arquivos ou na rede. A ferramenta

efetua a leitura do esquema existente nestes documentos XML e, a partir das regras internas contidas nestes documentos, constrói um modelo gráfico representando os objetos encontrados e seus relacionamentos. A geração do esquema integrado também é armazenada em documentos XML ao invés de descarregar a estrutura resultante da integração diretamente em algum banco de dados.

Portanto esta implementação do J-Schemas Integrator não possui interação alguma com banco de dados. A limitação da possibilidade de acesso diretamente ao banco de dados pode ser resolvida posteriormente com adição de algum novo módulo à ferramenta que possibilite exportar ou importar o modelo do banco de dados para o formato XML interpretado pela ferramenta.

6 Conclusão

A integração de esquemas de bancos de dados é uma tarefa que não pode ser totalmente automatizada. Alguns conflitos semânticos existentes entre os esquemas não podem ser detectados por uma ferramenta de integração.

Não é o objetivo deste trabalho automatizar todo o processo de integração. A meta principal foi construir uma ferramenta que sirva como apoio ao processo de integração de esquemas. O usuário utilizará a ferramenta para visualizar os esquemas a serem integrados, selecionar e identificar os objetos conflitantes entre os esquemas e visualizar a geração do esquema integrado e das informações de mapeamento entre os esquemas originais e o esquema integrado.

O processo de geração do esquema integrado depende muito da compreensão dos objetos nos sistemas iniciais. Isto é feito através de uma minuciosa análise de cada esquema. Para fazer esta análise é necessário modelar todas as bases usando um mesmo modelo de dados, auxiliando no desenvolvimento do esquema integrado e representando os relacionamentos, entidades, objetos, leis e integridade dos sistemas. O modelo de dados ERC+ possui algumas características interessantes que podem auxiliar no processo de integração de bancos de dados e por esta razão ele foi escolhido como modelo de dados comum na modelagem dos esquemas.

Como benefício gerado no processo de integração pode-se citar: disseminação do conhecimento de aplicações distribuídas e heterogêneas; visão unificada dos dados partindo do esquema integrado; possibilidade de construção a suporte de sistemas de decisão e data warehouses partindo do esquema integrado, dentre outras.

6.1 Sugestões para trabalhos futuros

Considerando a ferramenta J-Schemas Integrator, esforços futuros podem ser conduzidos no sentido de tornar a ferramenta ainda mais completa. Isso deve ser alcançado a partir de implementação de outros componentes e módulos cujo desenvolvimento não estavam previsto no escopo deste trabalho. O J-Schemas Integrator abre várias possibilidades de trabalhos que poderiam acrescentar as funcionalidades do editor, dentre elas:

- Módulo para semi-automatizar e facilitar a detecção de possíveis conflitos estruturais, descritivos e semânticos entre os esquemas. Como vimos nos capítulos anteriores,

para detectar conflitos entre os objetos podemos comparar os elementos dos esquemas (entidades, atributos e relações) e também compara ar populações dos esquemas, pois este tipo comparação entre as instâncias permite descobrir relações lógicas (inclusão, intersecção) entre os elementos.

- Módulo para automatizar ou semi-automatizar a fase de pré-integração, extraindo um esquema diretamente de um banco de dados e traduzindo para a estrutura XML suportada por esta ferramenta.
- Módulo para recebimento e realizações de consultas nos bancos de dados envolvidos na integração. Este módulo teria acesso as informações contidas no repositório da ferramenta (toda informação pertencente a esquemas integrados, esquemas iniciais e informações de mapeamento entre os esquemas integrados e os esquemas iniciais são armazenados em documentos XML durante o processo de integração) e realizaria consultas nas bases de dados iniciais. Isso preservaria a autonomia dos bancos de dados que foram integrados, permitindo acesso transparente a todos os bancos de dados sem a necessidade de que os usuários conhecessem os banco de dados iniciais, suas localizações e seus modelos de dados.
- Implementação de novos templates de integração para novos casos de conflitos.
- Melhoria na parte visual da ferramenta utilizando princípios e métodos descritos pela disciplina de IHC – Interface Homem Máquina.

7 Referências

AIGNER, **Diskrete Mathematik**, Vieweg, Braunschweig/Wiesbaden, 1993.

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

BATINI, C., LENZERINI, M. **A Methodology for Data Schema Integration in the Entity Relationship Model**, IEEE Transactions on Software Engineering, 1984

BIGGS, **Discrete Mathematics (Revised Edition)**, Oxford University Press, 1989, New YorkNY

BISHOP, **Java Gently**, Addison-Wesley, 1998.

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

BOUGUETTAYA, A., BENATALLAH, B., ELMAGARMID **A.. Management of heterogeneous and autonomous database systems** - Chapters 1, 5 e 10. Morgan-Kaufmann, 1999.

CAVINATO, E; RIBEIRO, H.G. **Integração do Banco de Dados no Ambiente Multiagentes**. Anais do IX Encontro dos Jovens Pesquisadores da Universidade de Caxias do Sul, 12 e 13 de setembro de 2001, pág. 97

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

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

DRAPER, D., HALEVY, A.Y., WELD, D.S. **The Nimble XML Data Integration System**. Proceedings of the International Conference on Data Engineering 2001(ICDE 2001).

FLANAGAN, **Java in a nutshell (2nd Edition),** O'Reilly Associates, 1997.

GAMMA, HELM, JOHNSON, and VLISSIDES, **Design Patterns**, Addison-Wesley, 1995.

GARDARIN G.; GANNOUNI, S.; FINANCE, B. **IRO-DB: A Distributed System Federating Object and Relational Databases.** Object-Oriented Multibase Systems, Prentice Hall, 1995. Disponível em: http://citeseer.nj.nec.com/gardarin95irodb.html

GEARY, **Graphic Java 2, Volume II: Swing (3rd Edition)**, Sun Microsystems, 1999.

GEARY, **Graphic Java 2, Volume III: Advanced Swing (3rd Edition)**, Sun Microsystems, 1999.

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

HALEYV, A.Y. **Data Integration: A Status Report**.. Proceedings of BTW 2003 - Database Systems for Business, Technology and Web (Invited Talk).

HAREL, **Algorithmics**, Addison-Wesley, 1992.

JACKSON, MCCLELLAN, **Java 1.2 by example**, Sun Microsystems, 1999.

JASPER, E., TONG, A., MCBRIEN P.J., POULOVASSILIS, **A. View Generation and Optimisation in the AutoMed Data Integration Framework**, In Proceedings of CAiSE03 Forum, Editors: J. Eder and T. Welzer, Univ. of Maribor Press, Pages 29-32, 2003

PASQUAL J., **Uso de XML para Interoperabilidade entre bases heterogêneas,** Programa de pós-graduação em informática. Universidade Federal do Paraná. 2002. Dissertação de mestrado

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

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

MARCÍLIO, Dalton Luiz, **Análise e Metodologias de Integração de Esquemas – Parte 1 ,2,3** http://www.pr.gov.br/batebyte/edicoes/2002/bb123/analise.htm

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

NIEVERGELT, HINRICHS, **Algorithms and data structures**, Prentice-Hall, 1993. Englewood CliffsNJ

OTTMANN, WIDMAYER, **Algorithmen und Datenstrukturen (2. Auflage),** BI-Wiss.-Verl, 1993.

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

PARENT, C., SPACCAPIETRA, **S. Issues and approaches of database integration.** Communications of the ACM, 41(5):166-178,1998

RAM, S., RAMESH, S. **Schema Integration: Past, Present and Future. Management of Heterogeneous and Autonomous Databases Systems**, Morgan Kaufmann Publishers, 1999.

SCOPIM, K. S.; KATSURAGAWA, R. **An Implementation of Database Integration Strategies**. Trabalho de Graduação. Departamento de Informática, Universidade Federal do Paraná, 2001.

SEDGEWICK, **Algorithmen**, Addison-Wesley, 1991.

SHELT, A.P, LARSON J., **"Federated Database Systems for managing heterogeneous, distributed and autonomous Databases"**, ACM Computing Surveys, Vol 22, No 3, 1990.

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

SPACCAPIETRA, S., PARENT, C., DUPONT Y.. **Independent Assertions for Integration of Heterogeneous Schemas**. Very Large Database Journal, Vol 1 (1), 1992.

STÄRK, SCHMID, BÖRGER, **Java and the Java Virtual Machine**, Springer, 2001.Heidelbery

WORLD WIDE WEB CONSORTIUM (W3C). **A schema language for XML**. http://www.w3c.org/XML/schema.HTML. Acessado em 10 / agosto / 2003.

UNIBASE: **A System for UNIfying Heterogeneous DataBASEs**, http://vishnu.bpa.arizona.edu/projects/page9.html

APÊNDICE

1 XML

1.1 Introdução

Extensible Markup Language (XML) é linguagem de marcação de dados (meta-markup language) que provê um formato para descrever dados estruturados. Isso facilita declarações mais precisas do conteúdo e resultados mais significativos de busca através de múltiplas plataformas. O XML também vai permitir o surgimento de uma nova geração de aplicações de manipulação e visualização de dados via internet.

O XML permite a definição de um número infinito de tags. Enquanto no HTML, se as tags podem ser usadas para definir a formatação de caracteres e parágrafos, o XML provê um sistema para criar tags para dados estruturados.

Um elemento XML pode ter dados declarados como sendo preços de venda, taxas de preço, um título de livro, a quantidade de chuva, ou qualquer outro tipo de elemento de dado. Como as tags XML são adotadas por intranets de organizações, e também via Internet, haverá uma correspondente habilidade em manipular e procurar por dados independentemente das aplicações onde os quais são encontrados. Uma vez que o dado foi encontrado, ele pode ser distribuído pela rede e apresentado em um browser como o Internet Explorer 5 de várias formas possíveis, ou então esse dado pode ser transferido para outras aplicações para processamento futuro e visualização.

1.2 Noções de HTML

1.2.1 Introdução

Na internet atualmente quase todas as páginas se resumem em HTML (HyperText Markup Language). O termo hypertext é definido por textos que têm links para outros textos. Já o termo markup language define anotações para a estrutura de um texto. O design de documentos html tem duas características importantes:

- Documentos html são feitos para prover estrutura lógica da informação destinada à apresentação de páginas da rede mundial de computadores.
- A linguagem html contém um conjunto de tags com um número fixo para definir a estrutura do documento, e cada tag tem a sua semântica já definida. O CSS (Cascading Style Sheets) permite a separação da estrutura lógica da

aparência da página. Mas, embora o layout possa ser separadamente definido no CSS, o html é destinado especificamente para hipertexto, e não para informação em geral!

1.2.2 Evolução do HTML

Essa linguagem foi desenvolvida em 1992 por Tim Berners Lee e Robert Caillau no CERN, que é o Centro Europeu de Pesquisas de Física de Partículas. O html é um exemplo do SGML (Standard Generalized Markup Language). Originalmente o html definia estritamente a estrutura lógica de um documento, e não a sua aparência física. Mas, com a pressão dos usuários (principalmente da indústria), as versões posteriores do html foram forçadas a prover cada vez mais e mais controle da aparência do documento. Algumas datas importantes:

- *1992: html foi definido*
- 1993: algumas definições físicas da aparência, tabelas, formulários e equações matemáticas (HTML+)
- 1994: HTML 2.0 (padrão para as características principais) e 3.0 (uma extensão do HTML+, entendido como um rascunho de padrão).
- 1995 e 1996: Netscape e Internet Explorer definem seus próprios padrões e surge o HTML 3.2 baseado nas implementações correntes.
- 1997: O HTML 4.0 é desenvolvido separando a apresentação da estrutura com style sheets (folhas de estilo).
- 1999: Definição do HTML 4.01 (suaves modificações da versão anterior).
- 2000: O XHTML 1.0 é criado, o qual consiste de uma versão XML do HTML 4.01.

1.3 Comparações entre HTML e XML

HTML e XML são primos. Eles derivam da mesma inspiração, o SGML. Ambos identificam elementos em uma página e ambos utilizam sintaxes similares. Se você é familiar com HTML, também o será com o XML. A grande diferença entre HTML e XML é que o HTML descreve a aparência e a ações em uma página na rede enquanto o XML não descreve nem aparência e ações, mas sim o que cada trecho de dados é ou representa ! Em outras palavras, o XML descreve o conteúdo do documento !

Como o HTML, o XML também faz uso de tags (palavras encapsuladas por sinais '<' e '>') e atributos (definidos com name="value"), mas enquanto o HTML especifica cada sentido para as tags e atributos (e frequentemente a maneira pela qual o texto entre eles será exibido em um navegador), o XML usa as tags somente para delimitar trechos de dados, e deixa a interpretação do dado a ser realizada completamente para a aplicação que o está lendo. Resumindo, enquanto em um documento HTML uma tag <p> indica um parágrafo, no XML essa tag pode indicar um preço, um parâmetro, uma pessoa, ou qualquer outra coisa que se possa imaginar (inclusive algo que não tenha nada a ver com um p como por exemplo autores de livros).

Os arquivos XML são arquivos texto, mas não são tão destinados à leitura por um ser humano como o HTML é. Os documentos XML são arquivos texto porque facilitam que os programadores ou desenvolvedores "debuguem" mais facilmente as aplicações, de forma que um simples editor de textos pode ser usado para corrigir um erro em um arquivo XML. Mas as regras de formatação para documentos XML são muito mais rígidas do que para documentos HTML. Uma tag esquecida ou um atributo sem aspas torna o documento inutilizável, enquanto que no HTML isso é tolerado. As especificações oficiais do XML determinam que as aplicações não podem tentar adivinhar o que está errado em um arquivo (no HTML isso acontece), mas sim devem parar de interpretá-lo e reportar o erro.

1.4 Características da linguagem XML

1.4.1 Representação estruturada dos dados

O XML provê uma representação estruturada dos dados que mostrou ser amplamentemente implementável e fácil de ser desenvolvida.

Implementações industriais na linguagem SGML (Standard Generalized Markup Language) mostraram a qualidade intrínseca e a força industrial do formato estruturado em árvore dos documentos XML.

O XML é um subconjunto do SGML, o qual é otimizado para distribuição através da web, e é definido pelo Word Wide Web Consortium(W3C), assegurando que os dados estruturados serão uniformes e independentes de aplicações e fornecedores.

O XML provê um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações desde simples até as mais complexas, dentre elas:

- Um simples documento.
- Um registro estruturado tal como uma ordem de compra de produtos.
- Um objeto com métodos e dados como objetos Java ou controles ActiveX.
- Um registro de dados. Um exemplo seria o resultado de uma consulta a bancos de dados.
- Apresentação gráfica, como interface de aplicações de usuário.
- Entidades e tipos de esquema padrões.
- Todos os links entre informações e pessoas na web.

Uma característica importante é que uma vez tendo sido recebido o dado pelo cliente, tal dado pode ser manipulado, editado e visualizado sem a necessidade de reacionar o servidor. Dessa forma, os servidores tem menor sobrecarga, reduzindo a necessidade de computação e reduzindo também a requisição de banda passante para as comunicações entre cliente e servidor.

O XML é considerado de grande importância na Internet e em grandes intranets porque provê a capacidade de interoperação dos computadores por ter um padrão flexível e aberto e independente de dispositivo. As aplicações podem ser construídas e atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados.

1.4.2 Separação entre dados e apresentação

A mais importante característica do XML se resume em separar a interface com o usuário (apresentação) dos dados estruturados. O HTML especifica como o documento deve ser apresentado na tela por um navegador. Já o XML define o conteúdo do documento. Por exemplo, em HTML são utilizadas tags para definir tamanho e cor de fonte, assim como formatação de parágrafo. No XML você utiliza as tags para descrever os dados, como exemplo tags de assunto, título, autor, conteúdo, referências, datas, etc...

O XML ainda conta com recursos tais como folhas de estilo definidas com Extensible Style Language (XSL) e Cascading Style Sheets(CSS) para a apresentação de dados em um navegador. O XML separa os dados da apresentação e processo, o que permite visualizar e processar o dado como quiser, utilizando diferentes folhas de estilo e aplicações.

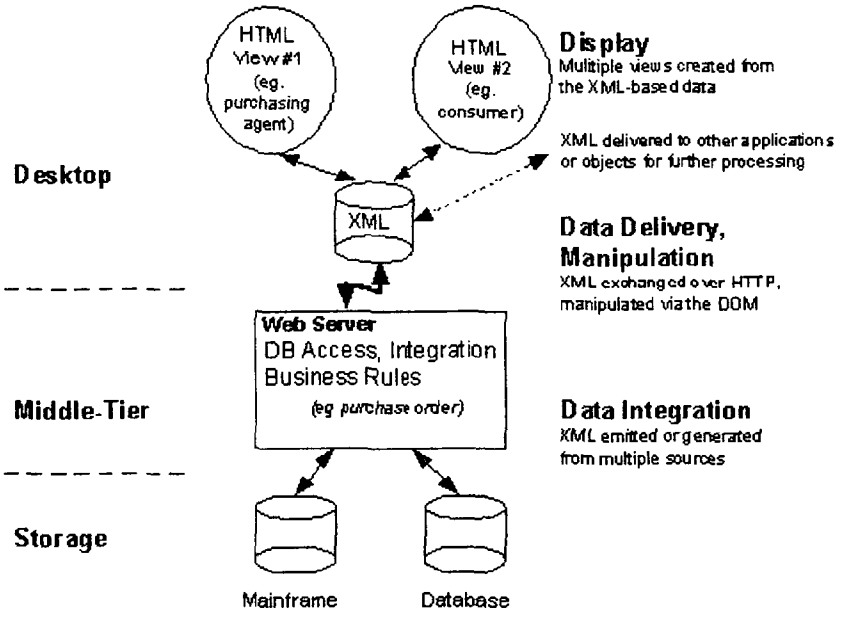

Figura 31 Exemplo de aplicação Web em três níveis, a qual é flexível e permite a troca de dados entre mainframes e os clientes (desktops).

Essa separação dos dados da apresentação permite a integração dos dados de diversas fontes. Informações de consumidores, compras, ordens de compra, resultados de busca, pagamentos, catálogos, etc... podem ser convertidas para XML no middl- tier (espécie de servidor), permitindo que os dados fossem trocados online tão facilmente como as páginas HTML mostram dados hoje em dia. Dessa forma, os dados em XML podem ser distribuídos através da rede para os clientes que desejarem.

1.5 Definição conceitual do XML

1.5.1 Estrutura do documento

Um documento XML é uma árvore rotulada onde um nó externo consiste de:

- Dados de caracteres (uma sequência de texto)
- Instruções de processamento (anotações para os processadores), tipicamente no cabeçalho do documento
- Um comentário (nunca com semântica acompanhando)
- Uma declaração de entidade (simples macros)
- Nós DTD (Document Type Declaration)

Um nó interno é um elemento, o qual é rotulado com:

- um nome ou
- um conjunto de atributos, cada qual consistindo de um nome e um valor

Normalmente, comentários, declarações de entidades e informações DTD não são explicitamente representadas na árvore.

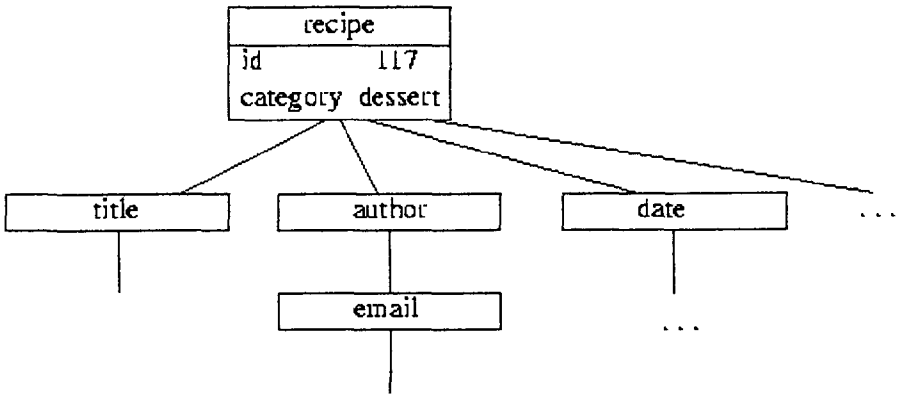

Figura 32 Estrutura de uma árvore XML

1.5.2 Uma visão prática das tags

Um documento XML é um texto (em formato Unicode) com tags de marcação (markup tags) e outras informações.

As markup tags denotam a seguinte estrutura:

```
...<bla attr="val" ...>...</bla>...
   |  |          | |
   |  |          | uma tag finalizadora de elemento
   |  |          o contexto do elemento
   |  um atributo com nome attr e valor val, com valores delimitados por ' ou "
      uma tag inicializadora de elemento com nome bla
Notação para elementos vazios: ...<bla attr="val".../>...
```

Os documentos XML são sensíveis à letras maiúsculas e minúsculas. Um documento XML é bem formatado quando segue algumas regras básicas. Tais regras são

mais simples do que para documentos HTML e permitem que os dados sejam lidos e expostos sem nenhuma descrição externa ou conhecimento do sentido dos dados XML.

Documentos bem estruturados:

- tem casamentos das tags de início e fim
- as tags de elemento tem que ser apropriadamente posicionadas

Os elementos não podem se sobrepor. Um exemplo de sobreposição é o seguinte:

```
<title>Descrição dos diveros modelos de carros<sub> da marca Ford
</title> Alexandre Manso</sub>
```

E, corrigindo o erro:

```
<title>Descrição dos diversos modelos de carros <sub> da marca Ford</sub>
 <author> Alexandre Manso</author> </title>
```

Caracteres especiais podem ser digitados usando referências de carcteres Unicode. Exemplo:

```
&#38; = &.
```

Seções CDATA são formas alternativas de se usar dados de caracteres, como:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

Informações adicionais:

```
<!-- comment -->
```

um comentário que será ignorado por todos os processadores.

```
<?target data...?>
```

uma instrução para um processador; target identifica o processador para o qual ela foi direcionada e data é a string contendo a instrução.

```
<!ENTITY name value>
```

declara uma entidade com um nome e um valor; expandida usando a referência entity: &name (entidades externas e referências de entidades de parâmetros são ignorados aqui).

```
<!ELEMENT ...>, <!ATTLIST ...>, ...
```

informações DTD (melhores alternativas são: DSD, XML Schema, que serão explicados posteriormente)

1.6 Documentos com DTDs

No XML as regras que definem um documento são ditadas por DTDs (Document Type Definitions), as quais ajudam a validar os dados quando a aplicação que os recebe não possui internamente uma descrição do dado que está recebendo. Mas os DTDs são opcionais e os dados enviados com um DTD são conhecidos como dados XML válidos. Um analisador de

documentos pode checar os dados que chegam analisando as regras contidas no DTD para ter certeza de que o dado foi estruturado corretamente. Os dados enviados sem DTD são conhecidos como dados bem formatados. Nesse caso, o documento pode ser usado para implicitamente se auto-descrever, como no caso da coleção de receitas em XML de um exemplo desse tutorial.

Com os dados XML válidos e com os bem-formatados, o documento XML se torna auto-descritivo porque as tags dão idéia de conteúdo e estão misturadas com os dados. Devido ao formato do documento ser aberto e flexível, ele pode ser usado em qualquer lugar onde a troca ou transferência de informação é necessária. Desta forma, podemos usar o XML para descrever informações sobre páginas HTML, ou descrever dados contidos em objetos ou regras de negócios, ou transações eletrônicas comerciais. O XML pode ser inserido dentro de documentos HTML, o que foi definido pelo W3C como "data-islands". Esse recurso permite que um documento HTML possa ter múltiplas formas de visualização quando se faz uso da informação de semântica contida no XML.

O que define formalmente quais elementos e quais combinações possíveis são permitidas dentro de um documento XML é o "schema", ou seja, esquema. Existem novos esquemas propostos ao W3C, dentre eles estando o DCD (Document Content Description), que provêm a mesma funcionalidade dos DTDs, e que, pelo fato de linguagens esquema serem extensíveis, os desenvolvedores podem aumentá-los com informações adicionais, tais como regras de apresentação, tornando essas novas linguagens esquema mais poderosas que os DTDs.

As DTDs são formas de se descrever classes de documentos XML (como gramáticas para outras linguagens).

Problemas com DTDs:

- Se muito simples não tem poder expressivo de descrição.
- Se for muito complexa terá uma sintaxe horrível.

Um exemplo de DTD:

```
<!DOCTYPE recipecollection [
...
<!ELEMENT recipe
(title,author?,date?,description,ingredients,preparation,related)>
<!ATTLIST recipe id ID
```

```
#REQUIRED
category (breakfast|lunch|dinner|dessert|unknown)
#IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author ANY>
...
]>
```

A solução para as DTDs: usar linguagens de esquemas (schema languages) tais como DSD,XML Schema, etc...

Os documentos, para serem validados, tem que ser bem formados e também estarem em conformidade com a DTD dada.

1.7 Padrões da estrutura do XML

1.7.1 Definição

O XML é baseado em padrões de tecnologia comprovadamente otimizados para a Web.

Os padrões que compõem o XML são definidos pelo W3C (World Wide Web Consortium) e são os seguintes:

- **Extensible Markup Language (XML)**[1] - é uma Recomendação, que é vista como o último estágio de aprovação do W3C. Isso significa que o padrão é estável e pode ser aplicado à Web e utilizado pelos desenvolvedores de ferramentas.

- **XML Namespaces**[2] - é também uma Recomendação, a qual descreve a sintaxe de namespace, ou espaço de nomes, e que serve para criar prefixos para os nomes de tags, evitando confusões que possam surgir com nomes iguais para tags que definem dados diferentes.

- **Document Object Model (DOM) Level 1**[3] - é uma Recomendação que provê formas de acesso aos dados estruturados utilizando scripts, permitindo aos desenvolvedores interagir e computar tais dados consistentemente.

- **Extensible Stylesheet Language (XSL)**[4]- é atualmente um rascunho. O XSL apresenta duas seções: a linguagem de transformação e a formatação de objetos. A linguagem de transformação pode ser usada para transformar documentos XML em algo agradável para ser visto, assim como transformar

para documentos HTML, e pode ser usada independentemente da segunda seção (formatação de objetos). O Cascade Style Sheet (CSS) pode ser usado para XML simplesmente estruturado mas não pode apresentar informações em uma ordem diferente de como ela foi recebida.

- **XML Linking Language (XLL)**[5] - e **XML Pointer Language (XPointer)**[6] - são também rascunhos. O XLL é uma linguagem de construção de links que é similar aos links HTML, sendo que é mais poderosa, porque os links podem ser multidirecionais, e podem existir a nível de objetos, e não somente a nível de página.

1.8 Uma noção sobre DOM

DOM[7] é uma API (Applications Programming Interface) independente de plataforma e linguagem que é utilizada para manipular as árvores do documento XML (e HTML também).

DOMs são ideais para linguagem script, como exemplo ECMAScript.

Essa API é definida em vários níveis:

- **Nível 0**: Funções existentes conhecidas das linguagens script dos browsers
- **Nível 1**: Funcionalidade para navegação em documentos e manipulações.
- **Nível 2**: Adiciona modelos de style sheets (folhas de estilo), filtros, modelos de eventos, e suporte a namespaces.
- **Nível 3**: Possibilita as opções de carregar e salvar, DTDs, schemas, visualização de documentos e status de formatação

SAX[8] - Simple API for XML

É baseada em eventos (eventos analisados são reportados para as aplicações através de chamadas callback).

1.9 Principais benefícios da linguagem XML

O XML tem por objetivo trazer flexibilidade e poder às aplicações Web. Dentre os benefícios para desenvolvedores e usuários temos:

- Buscas mais eficientes

- Desenvolvimento de aplicações Web mais flexíveis. Isso inclui integração de dados de fontes completamente diferentes, de múltiplas aplicações; computação e manipulação local dos dados; múlitiplas formas de visualização e atualização granulares do conteúdo.
- Distribuição dos dados via rede de forma mais comprimida e escalável.
- Padrões abertos

1.9.1 Buscas mais eficientes

Os dados em XML podem ser unicamente "etiquetados", o que permite que, por exemplo, uma busca por livros seja feita em função do nome do autor. Atualmente, uma busca com o nome do autor poderia levar a qualquer site que tivesse referência a tal nome, não importando se fosse o autor do livro ou simplesmente um livro sobre o autor. Sem o XML é necessário para a aplicação de procura saber como é esquematizado e construído cada banco de dados que armazena os dados de interesse, o que é impossível. O XML permitiria definir livros por autor, título, assunto, etc..., o que facilitaria enormemente a busca.

1.9.2 Desenvolvimento de aplicações flexíveis para a Web

O desenvolvimento de aplicações Web em três camadas, ou three-tier, é altamente factível com o XML. Os dados XML podem ser distribuidos para as aplicações, objetos ou servidores intermediários para processamento. Esses mesmos dados também podem ser distribuídos para o desktop (pc e similares) para ser visualizado em um navegador.

1.9.3 Integração de dados de fontes diferentes

Atualmente é praticamente impossível a procura em múltiplos bancos de dados e incompatíveis. O XML permite que tais dados possam ser facilmente combinados. Essa combinação seria feita via software em um servidor intermediário, estando os bancos de dados na extremidade da rede.

Os dados poderiam ser distribuídos para outros servidores ou clientes para que fizessem o processamento, a agregagem e a distribuição.

1.9.4 Computação e manipulação locais

Os dados XML recebidos por um cliente são analizados e podem ser editados e manipulados de acordo com o interesse do usuário. Ao contrário de somente visualizar os dados, os usuários podem manipulá-los de várias formas. Os recursos disponíveis do

Document Object Model (DOM) permitem que os dados sejam manipulados via scripts ou outra linguagem de programação.

A separação da interface visual dos dados propriamente ditos permite a criação de aplicações mais poderosas, simples e flexíveis.

1.9.5 Múltiplas formas de visualizar os dados

Os dados recebidos por um usuário podem ser visualizados de diferentes formas uma vez que o XML define somente os dados e não o visual. A interpretação visual poderia ser dada de várias maneiras diferentes, de acordo com as aplicações. Os recursos de CSS e XSL permitem essas formas particulares de visualização.

1.9.6 Atualizações granulares dos documentos

Os dados podem ser atualizados de forma granular, evitando que uma pequena modificação no conjunto de dados implique na busca do documento inteiro novamente. Dessa forma, somente os elementos modificados seriam enviados pelo servidor para o cliente. Atualmente, uma modificação em um item de dados acarreta na necessidade de atualização da página inteira.

O XML também permite que novos dados sejam adicionados aos já existentes, sem a necessidade de reconstrução da página.

1.9.7 Fácil distribuição na Web

Assim como o HTML, o XML, por ser um formato baseado em texto aberto, pode ser distribuído via HTTP sem necessidade de modificações nas redes existentes.

1.9.8 Escalabilidade

Devido ao fato dos documentos XML separarem completamente os dados da forma com a qual são visualizados, autores de aplicações de visualização de dados podem torná-las muito poderosas e interativas, permitindo ao usuário visualizar os dados da forma que lhe agrade. Dessa forma, a interatividade, em termos, não dependeria tanto da comunicação cliente servidor, mas sim seria feita "offline", reduzindo o tráfego do link com o servidor.

1.9.9 Compressão

A compressão de documentos XML é fácil devido à natureza repetitiva das tags usadas para definir a estrutura dos dados. A necessidade de compressão é dependente da aplicação e da quantidade de dados a serem movidos entre clientes e servidores. Os padrões de compressão do HTTP 1.1 podem ser usados para o XML.

1.10 Exemplo de estruturas HTML e XML

1.10.1 Exemplo de receita em html

```
<h1> Bolo de banana</h1>
<h2> Miguel Furtado furtado@predialnet.com.br</h2>
<h3>Sunday, 04 Jun 2000</h3>
O bolo de banana é feito com banana prata e possui um sabor maravilhoso.
Pode ser servido quente ou frio.
 <table>
 <tr><td> 3 1/2 copos <td> de leite desnatado longa vida
(leite em pó não é indicado).
 <tr><td> 2 colheres de sopa <td> de açúcar.
<tr><td> 4 <td> bananas prata picadas em pedaços pequenos.
<tr><td> 1 colher de chá <td> canela.
<tr><td> 4 pedaços <td> noz moscada.
</table>
Combine tudo no liquidificador e bata até misturar bem.
 Leve ao forno por 20 minutos. Pronto. É só servir!Receitas relacionadas:
<a href="#BoloChocolate">Bolo de Chocolate</a>
```

Esse é um exemplo de receita em HTML. Observe bem a estrutura porque ela será comparada com uma estrutura XML.

1.10.2 Exemplo de receita em XML

```
<recipe id="117" category="sobremesa">
<title> Bolo de banana </title>
<author>
<email> Miguel Furtado furtado@predialnet.com.br </email>
</author>
<date>Sunday,04 Jun 2000</date>
<description> O bolo de banana é feito com banana prata e
possui um sabor maravilhoso.
```

```
Pode ser servido quente ou frio. </description>
<ingredients> ... </ingredients>
<preparation>
 Combine tudo no liquidificador e bata até misturar bem.
 Leve ao forno por 20 minutos. Pronto. É só servir! </preparation>
<related url="#BoloChocolate">Bolo de Chocolate</related></recipe>
```

O exemplo mostra que:

- As marcações são utilizadas puramente na estrutura lógica
- Existe apenas uma única escolha do nível de detalhamento da marcação
- Há a necessidade de uma espécie de "gramática" específica para a coleção de receitas em XML
- Também é necessário style-sheet para definir a semântica da apresentação do documento.

1.11 Inserindo XML em documentos HTML

1.11.1 Apresentando dados XML via HTML

Um documento XML não indica por si só como a informação deve ser visualizada.

Os dados simplesmente contém os fatos (do tipo quem ordenou um livro e por qual preço). Já o HTML é uma linguagem ótima para a apresentação de dados para o usuário final. Um bom exemplo seria um empregado de uma livraria virtual acessando os registros de dados para encontrar as listas de entrada de pedidos. Tais dados poderiam estar sendo representados em XML e a visualização dos mesmos em HTML. Este tipo de livraria precisaria que o seu servidor convertesse os registros de XML para HTML. Isso também poderia ser feito pelo próprio navegador do usuário.

Os mecanismos de ligação de dados e folhas de estilos podem ser usados para organizar os dados XML em uma apresentação visual, e para adicionar interatividade. A ligação de dados (data binding) é um aspecto do HTML Dinâmico (DHTML), a qual move itens individuais de dados de uma fonte de informação (por exemplo XML) para uma apresentação HTML, permitindo que o HTML seja usado como um modelo de exibição de dados XML. Isso é parecido com uma ligação de mensagens em processamento de palavras.

O XSL (Extensible Stylesheet Language) pode adicionar poderes ainda maiores à esse processo. Uma folha de estilos XSL contém instruções de como retirar informação de um

documento XML e transformá-la para outro formato, como o HTML. Essa transformação é feita de forma declarativa, e não via scripts. E mais, o XSL usa o XML como sua sintaxe, o que evita que autores de documentos XML tenham que aprender outro tipo de linguagem de marcação de dados.

Os Cascade Style Sheets (CSSs) também podem ser usados para dados XML, só que de estrutura mais simplificada, e são usuais em tais situações. Entretanto, os CSSs não provém uma estrutura de apresentação que se difere da estrutura da fonte de dados. Com o XSL, é possível gerar estruturas de apresentação (em HTML, por exemplo) que são bem diferentes das estruturas de dados originais dos documentos XML, como mostrado a seguir.

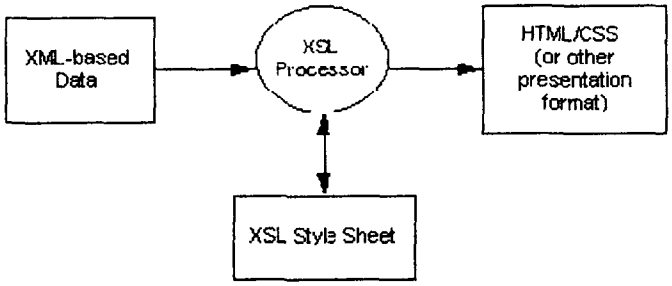

Figura 33 Transformando dados estruturados em XML para uma apresentação

O XSL é a ferramenta que realizará tal transformação. O XSL provê independência semântica e estrutural do conteúdo e da apresentação.

1.12 Ferramentas XML

1.12.1 Parsers (analisadores)

Programas de análise da formatação e gramática mais conhecidos (Parsers):

- Expat (www.jclark.com/xml/expat.html): Escrito em C (passado para outras linguagens), e utilizado por LIBWWW, Apache, Netscape, DSD, ...
- XML4J (www.alphaworks.ibm.com/tech/xml): Tem como desenvolvedora a AlphaWorks, em Java, é baseado em Apache Xerces, e suporta DOM e SAX.

Existem muitos outros parsers no mercado.

1.12.2 Editores

São utilitários que permitem a edição do documento XML.

- Xeena (www.alphaWorks.ibm.com/tech/xeena) Desenvolvido pela AlphaWorks, em Java, com a sintaxe de visualização de árvore direcionada para a edição.

Muitos outros editores estão disponíveis no mercado.

1.12.3 Browsers (navegadores)

Dentre os mais conhecidos estão o Netscape Navigator 5 e o Internet Explorer 5.

Realizam várias funções, dentre elas análise e validação de documentos XML, apresentação visual (rendering) com XSL e CSS, e acesso aos scripts via DOM.

Referências:

[1] **Extensible Markup Language (XML)**, http://www.w3.org/TR/REC-xml/t_top Acessado em 10/agosto/2003

[2] **XML Namespaces**, http://www.w3.org/TR/REC-xml-names/t_top Acessado em 10/agosto/2003

[3] **Document Object Model (DOM) Level 1**, http://www.w3.org/TR/REC-DOM-Level-1/t_top Acessado em 10/agosto/2003

[4] **Extensible Stylesheet Language (XSL)**, http://www.w3.org/TR/WD-xsl/t_top Acessado em 10/agosto/2003

[5] **XML Linking Language (XLL)**, http://www.w3.org/TR/WD-xlink Acessado em 10/agosto/2003

[6] **XML Pointer Language (XPointer)**, http://www.w3.org/TR/WD-xptr/t_top Acessado em 10/agosto/2003

[7] **DOM** , http;//www.w3.org/DOM. Acessado em 10/agosto/2003

[8] **SAX** , www.megginson.com/SAX Acessado em 10/agosto/2003