J-Integrator: Uma Ferramenta para Integração de Bancos de Dados Heterogêneos

Abstract. Many applications nowadays involve data bases that differ both on their technology and architecture, and there is a need to integrate them. Although a number of tools exist to support the construction of a single schema, not many support the process of integrating them. In this paper we present J-Integrator, a tool that helps the user create an integrated schema by graphically defining mappings to local ones. It also translates queries defined on the integrated schema to queries to be shipped to local sources. Mappings between schemas are expressed in XML, which makes it easy to extend the tool by incorporating new integration operations.

Resumo. Em diversas aplicações no mundo de hoje é necessário integrar bancos de dados que foram projetados em momentos diferentes, usando arquiteturas e tecnologias heterogêneas. Embora existam diversas ferramentas que auxiliem a modelagem de um único esquema, há carência de ferramentas de apoio ao processo de geração de esquemas integrados. Neste artigo apresentamos o J-Integrator, uma ferramenta que permite a construção de esquemas integrados através de um editor gráfico, bem como a submissão de consultas a base integrada. Busca-se assim dar aos usuários o benefício de uma visão unificada dos dados sem que sejam necessárias alterações nos bancos de dados originais, preservando assim os investimentos feitos ao longo do tempo. Os mapeamentos entre esquemas são expressos em XML, o que facilita a extensão da ferramenta com novas operações de integração.

1. Introdução

Atualmente, é comum as organizações utilizarem 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 utilizam modelos **de dados distintos, representam os dados de uma maneira diferente, rodam em diferentes plataformas de hardware e normalmente são gerenciados por software 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 é a necessidade de 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 às informações localizadas em diferentes bancos de dados, preservando a autonomia dos mesmos. Ou seja, a construção de uma visão unificada e uma interface homogênea dos dados, sem a necessidade de realizar alterações nos bancos de dados existentes. Desta forma, a integridade dos bancos de dados são preservados, bem como os investimentos iniciais realizados na criação destes sistemas. Neste contexto, é importante desenvolver

mecanismos que permitam interoperabilidade entre os bancos de dados. Uma forma de prover tal interoperabilidade é definir um ou mais esquemas 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 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 pode resolver o problema de interoperabilidade entre as várias fontes de dados unificando os esquemas e criando um único 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 1995], possui algumas características interessantes que podem auxiliar no processo de integração de bancos de dados. Este modelo, que é uma extensão do modelo ER tradicional [CHEN 1976], 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 chamada 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.

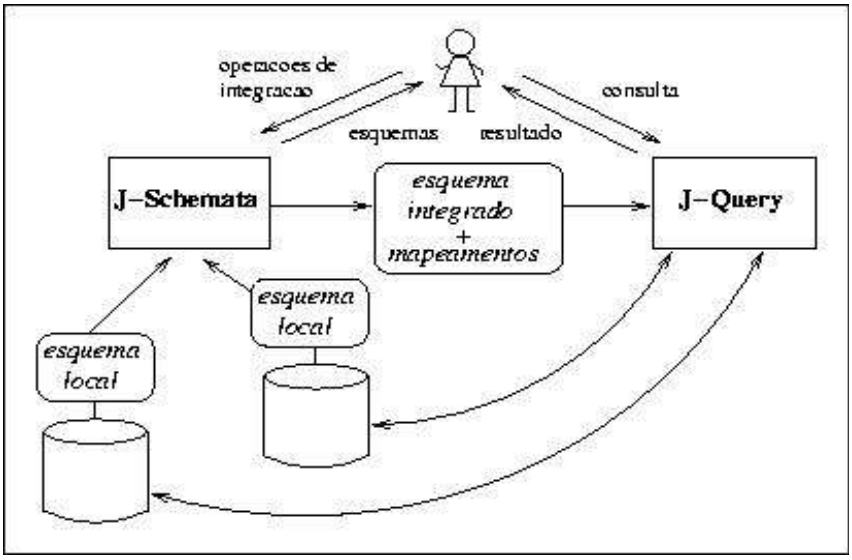

Figura 1: Arquitetura da ferramenta J-Integrator

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. 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.

Neste artigo descrevemos o desenvolvimento do *J-Integrator*, uma ferramenta de apoio a integração de esquemas de bancos de dados heterogêneos. A ferramenta é baseada no modelo ERC+ e possibilita o usuário visualizar os esquemas pré-existentes, bem como gerar o esquema integrado a partir dos conflitos existentes, utilizando um editor gráfico. A partir dos mapeamentos entre os esquemas, é também possível realizar consultas a base integrada. A ferramenta tem uma arquitetura flexível e todas as informações de correspondências entre esquemas são armazenadas em XML, de forma que a ferramenta possa ser facilmente estendida com novos módulos.

O restante do artigo está organizado da seguinte forma. A seção 1 descreve a arquitetura da ferramenta *J-Integrator* e o modelo ERC+. O módulo integrador de esquemas e o módulo de consultas são apresentados nas seções 3 e 4, respectivamente. Na seção 5 são discutidos trabalhos relacionados, e as considerações finais são apresentadas na seção 6.

2. Arquitetura da ferramenta *J-Integrator*

Integrar bases de dados é uma tarefa complexa e que não pode ser totalmente automatizada. Em função da extrema dificuldade de representar as estruturas semânticas das bases de dados, ferramentas automáticas ficam impossibilitadas de detectar e solucionar conflitos, e mesmo que pudessem, as formas de resolvê-las não são tarefas metódicas e exigem a interação humana.

O processo de integração tem como entrada um conjunto de esquemas e produz outro unificado que representa os esquemas de entrada, o esquema de saída, juntamente com informações de mapeamento, chamado esquema integrado. Na ferramenta *J-Integrator* este processo é realizado pelo módulo *J-Schemata,* como ilustrado na Figura 1. As informações geradas por este módulo podem então ser utilizadas para que o usuário possa realizar consultas ao esquema integrado através do módulo *J-Query*. Desta forma, as heterogeneidades das bases de dados fontes passam a ser transparentes ao usuário final.

O módulo J-Schemata pressupõe que os modelos a serem integrados foram previamente mapeados em um modelo de dados comum em uma fase de pré-integração, para tornarem-se homogêneos sintática e semanticamente. A utilização do modelo entidade relacionamento como base para a integração de bancos de dados é constante na literatura [BATINI 1984] [SPACCAPIETRA 1995]. A representação gráfica do modelo facilita a visualização de equivalências semânticas inter-esquemas. O modelo adotado por este trabalho, ERC+, deriva do modelo ER tradicional [CHEN 1976] e acrescenta aos conceitos básicos de entidade, relacionamento e atributo, as seguintes características:

- **Atributos complexos e multivalorados**
- **Generalização/especialização do tipo** *is_a* **e** *may_be_a* **(que serão descritos na seção 3)**
- **Operadores de reestruturação de esquema, que permitem a conversão de relacionamentos em atributos (*i-Join*)**
- **Existência de identificadores de objetos (*oids*)**
- **Representação gráfica de cardinalidades de relacionamentos e atributos**

Estas características levam-nos a acreditar que o modelo ERC+ é um modelo adequado para servir de fundamento teórico para a representação e reestruturação de esquemas oriundos de bases heterogêneas. A sua aplicação em uma ferramenta de integração que faz uso da visualização e reestruturação de vários esquemas com a subseqüente geração de um esquema global integrado é uma evolução normal e esperada. Na próxima seção, descreveremos o módulo *J-Schemata,* que destina-se a geração de esquemas integrados e informações de mapeamento dos modelos ERC+ criados no processo de pré-integração.

3. J-Schemata: O Módulo Integrador de Esquemas

O módulo J-Schemata tem como objetivo auxiliar o usuário nos processos de identificação de correspondências, e de geração tanto do esquema integrado como de informações de mapeamento. É utilizada a estratégia de integração binária em escada (binary ladder) como descrito em [BATINI 1984], por se tratar de uma abordagem que simplifica o processo de comparação entre esquemas e resolução de conflitos. Nesta estratégia, as integrações são realizadas aos pares. Uma das características importantes do J-Schemata é que ela utiliza o formato XML para descrever as informações de integração geradas. Isto facilita a extensão de novos módulos a ferramenta *J-Integrator*, como por exemplo um módulo para geração de ontologias. Além disso, a transformação dos dados para um outro formato pode ser facilmente realizada utilizando linguagens de consulta e transformação para XML como Xquery [BOAG 2005] e XSLT [CLARK 1999].

O módulo J-Schemata permite realizar as seguintes tarefas:

- **Visualizar os esquemas a serem integrados;**
- **Selecionar e categorizar os conflitos entre os esquemas;**
- **Visualizar e corrigir discordâncias do esquema integrado no processo;**
- **Gerar um arquivo XML com as informações de mapeamento dos esquemas iniciais e integrado.**

A interface da ferramenta, ilustrada na Figura 2, é composta por dois frames superiores, nos quais são carregados os esquemas iniciais, e um frame inferior, onde é formado o esquema integrado. Cada frame possui um menu que possibilita manipulações de esquemas, tais como carregar e salvar. Entre os frames superiores está uma paleta com botões que corresponde às regras de integração.

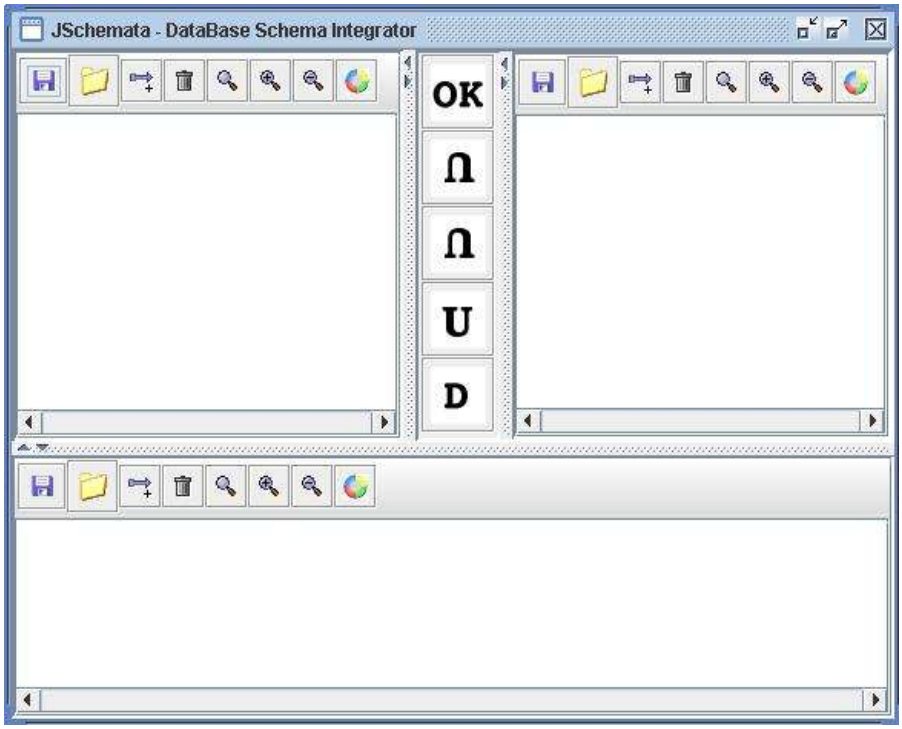

Figura 2: Interface com o usuário da ferramenta J-Schemata.

3.1. Regras de Integração

As regras de integração da ferramenta correspondem a criação de relacionamentos existentes no modelo ERC+. Cada uma delas, na ordem de cima para baixo na paleta central Figura 2, são descritas a seguir.

1. ***Migração direta de Entidades e Links sem Transformação*:** Os elementos presentes nos esquemas iniciais que não precisam ser transformados mas precisam participar do esquema integrado são copiados para o esquema integrado.

2. ***Disjunção Inter-Esquemas*: Esta transformação corresponde ao clássico relacionamento "*is-a*" e tem por objetivo identificar entidades similares, ou seja, que compartilham semelhanças, e generalizá-las em uma nova entidade. A nova entidade possuirá os atributos comuns das entidades específicas, e estas preservarão seus atributos específicos. As entidades Específicas se unem a entidade genérica por meio de uma**

ligação do tipo "*is_a*" direcionados à ela. Conforme descrito em [BATINI 1984]**, algumas incompatibilidades entre entidades e atributos de esquemas distintos precisam ser resolvidos a fim de permitir sua integração, visto que o mesmo conceito no mundo real pode ser percebido e representado de formas e por pessoas diferentes. Para isso são necessárias duas transformações: entidades em atributos e atributos em entidades.**

3. ***Disjunção Intra-Esquemas*: Esta regra de transformação é semelhante à anterior, porém integra entidades do mesmo esquema inicial.**

4. ***Intersecção Inter-Esquemas*: Esta transformação tem por objetivo integrar entidades com domínios semelhantes. As entidades iniciais serão copiadas fielmente para o esquema integrado com um ligação do tipo "*may_be_a*" unindo-as. O link "*may_be_a*" entre dois tipos de entidades E1 e E2 é usado para especificar que algumas (possivelmente todas) as instâncias de E1 descrevem os mesmos objetos do mundo real que as instâncias de E2. Ou seja, as populações de E1 e E2 podem compartilham identificadores de objetos e portanto sua interseção é não vazia.**

5. ***Conflito de Tipo de Dado*: Esta transformação permite modificar o nome e domínio dos atributos de uma entidade. Possibilita a compatibilização entre entidades que podem vir a ser integradas.**

Para ilustrar o uso da ferramenta, utilizaremos um exemplo de conflito do tipo disjunção envolvendo uma relação denominada `Aluno`, armazenando a população de alunos de uma Universidade, e outra relação denominada `DpFunc`, armazenando a população de funcionários. Tanto alunos como funcionários possuem nome e número de CPF como atributos em comum, embora suas instâncias registrem fatos do mundo real que não estão relacionados. A solução para este conflito consiste em criar uma entidade para generalizar os atributos em comum a estas duas relações e ligá-las através de um link "*is-a*", conforme ilustrado na Figura 3.

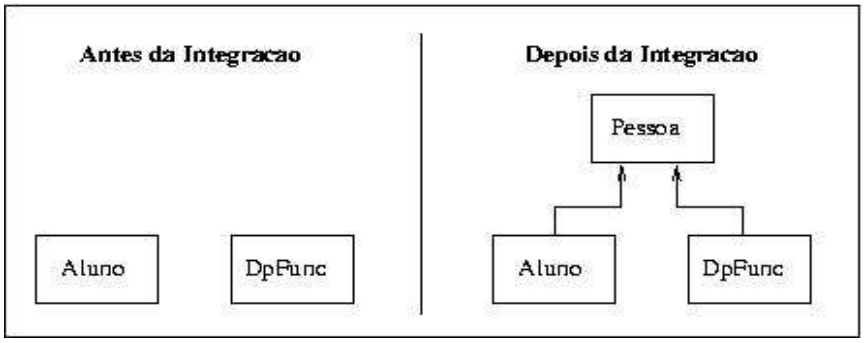

Figura 3: Exemplo de solução para o conflito do tipo disjunção

Na próxima seção descreveremos como as operações de integração bem como de mapeamentos entre esquemas são armazenados em um arquivo XML.

3.2. Armazenamento de Esquemas em Arquivos XML

Para armazenar as informações de soluções de conflito e mapeamentos entre esquemas no formato XML, foi criada uma Definição do Tipo de Documento (DTD). O documento XML é interpretado pela ferramenta no momento da leitura dos esquemas para representação gráfica no editor. A DTD é ilustrada na Figura 4.

O nó raiz do documento XML deve chamar-se `DatabaseSchema` **e deve conter três nós filhos:** `Tables, Links e MappingInformation.` **A entidade** `Tables` **representa as tabelas que fazem parte do esquema integrado e podem ser de dois tipos: TE (tipo entidade) e TA (tipo associação ou relacionamento). A entidade** `Links` **representa as ligações entre as entidades, que podem ser do tipo regular (um relacionamento),** *is-a*, **ou** *may-be-a*. **A entidade** `MappingInformation` **contém informações para acessar as relações originais, bem como mapeamentos entre os atributos das entidades do esquema integrado para atributos das relações originais.**

```
<?xml version="1.0" encoding="UTF-8" ?>

    <!ELEMENT DatabaseSchema (Tables, Links, MappingInformation)>
    <!ELEMENT Tables (Table*)>
    <!ELEMENT Links (Link*)>
    <!ELEMENT MappingInformation (Source*, Table*)>
    <!ELEMENT Source EMPTY>
    <!ELEMENT Table (Attribute*)>
    <!ELEMENT Link EMPTY>
    <!ELEMENT Attribute (Attribute*, AttrLink*)>
    <!ELEMENT DatabaseSchema name CDATA #REQUIRED>
    <!ATTLIST Source
           id ID #REQUIRED
           tableName  CDATA #REQUIRED
           URI    CDATA #REQUIRED >
    <!ATTLIST Table
            type CDATA #IMPLIED
            name CDATA #REQUIRED
            x CDATA #IMPLIED
            y CDATA #IMPLIED
            width CDATA #IMPLIED
            height CDATA #IMPLIED >
    <!ATTLIST Link
            type CDATA #REQUIRED
            source CDATA #REQUIRED
            target CDATA #REQUIRED >
    <!ATTLIST Attribute
            name CDATA #REQUIRED
            domain CDATA #IMPLIED
            obs CDATA #IMPLIED >
    <!ATTLIST AttrLink
            name CDATA #REQUIRED
            domain CDATA #REQUIRED
            source IDREF #REQUIRED
            obs CDATA #IMPLIED >
```

Figura 4: DTD utilizada pela ferramenta J-Schemata

A Figura 5 contém um exemplo de arquivo XML resultante da aplicação da regra de integração disjunção inter-esquemas ilustrado na Figura 3. As demais operações de integração também podem ser facilmente representadas no modelo: a interseção inter-esquemas é representada com a criação de uma entidade `link` **to tipo "*may_be_a*" com as informações de mapeamento correspondentes; conflitos de tipos de dados são registrados pelas entidades** `AttrLink` **que estabelecem a relação entre atributos de uma entidade do esquema integrado com atributos dos esquemas iniciais.**

Observe que as entidades `Tables` **e** `Links` **representam o modelo integrado gerado, que consiste das entidades** `Aluno`, `DpFunc` **e** `Pessoa` **com links "*is_a*" de** `Aluno` **para** `Pessoa` **e** `DpFunc` **para** `Pessoa`. **É este modelo que poderá ser utilizado para realizar processos de integração com outras bases de dados e consequentemente gerar novos esquemas integrados. Além disso, é sobre este modelo que o usuário poderá submeter consultas de forma transparente quanto a heterogeneidade das bases envolvidas. Para**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

 <DatabaseSchema name=”UNIVERSIDADE”>
  <Tables>
      <Table type="TE" name="ALUNO" y="209" x="579" width="50" height="20">
          <Attribute name="NUMALU-ALU" domain="INTEGER(10)" />
          <Attribute name="NOMEALU-ALU" domain="CHAR(60)"  />
          <Attribute name="ANO2GRAU-ALU" domain="INTEGER(10)" />
          <Attribute name="NUMCPF-ALU" domain="INTEGER(13)" />
      </Table>
      <Table type="TE" name="DPFUNC" y="22" x="264" width="50" height="20">
          <Attribute name="NOMEFUNC-DP" domain="CHAR(50)" />
          <Attribute name="NOME-DP" domain="CHAR(50)" />
          <Attribute name="NUMCPF-DP" domain="INTEGER(15)" />
      </Table>
      <Table type="TE" name="PESSOA" y="110" x="314" width="50" height="20">
          <Attribute name="NUMCPF-PESS" domain="INTEGER(13)" />
          <Attribute name="NOME-PESS" domain="CHAR(60)" />
      </Table>
  </Tables>
   <Links>
      <Link type="IS_A" source="ALUNO" target="PESSOA" />
      <Link type="IS_A" source="DPFUNC" target="PESSOA" />
   </Links>
   <MappingInformation>
       <Source id=”t1” tableName=”ALUNO” URI=”jdbc:postgresql:academico” />
      <Source id=”t2” tableName=”DPFUNC” URI=”jdbc:postgresql:pessoal” />
      <Table name="ALUNO">
          <Attribute name="NUMALU-ALU">
             <AttrLink name="NUMALU-ALU" domain="INTEGER(10)" source="t1" />
          </Attribute>
          <Attribute name="NOMEALU-ALU" >
             <AttrLink name="NOMEALU-ALU" domain="CHAR(60)" source="t1" />
          </Attribute>
          <Attribute name="ANO2GRAU-ALU" >
             <AttrLink name="ANO2GRAU-ALU" domain="INTEGER(10)" source="t1" />
          </Attribute>
          <Attribute name="NUMCPF-ALU" >
             <AttrLink name="NUMCPF-ALU" domain="INTEGER(13) source="t1" />
          </Attribute>
      </Table>
      <Table name="DPFUNC">
          <Attribute name="NOMEFUNC-DP" >
             <AttrLink name="NOMEFUNC-DP" domain="CHAR(50)" source="t2" />
          </Attribute>
          <Attribute name="NOME-DP" >
             <AttrLink name="NOME-DP" domain="CHAR(50)" source="t2" />
           </Attribute>
          <Attribute name="NUMCPF-DP" >
             <AttrLink name="NUMCPF-DP" domain="INTEGER(15)" source="t2" />
          </Attribute>
      </Table>
      <Table name="PESSOA">
          <Attribute name="NUMCPF-PESS" domain="INTEGER(13)" >
             <AttrLink name="NUMCPF-ALU" domain="INTEGER(13)" source="t1" />
             <AttrLink name="NUMCPF-DP" domain="INTEGER(15)" source="t2" />
          </Attribute>
           <Attribute name="NOME-PESS" domain="CHAR(60)" >
              <AttrLink name="NOMEALU-ALU" domain="CHAR(60)" source="t1" />
              <AttrLink name="NOMEFUNC-DP" domain="CHAR(50)" source="t2" />
           </Attribute>
      </Table>
   /MappingInformation>
 </DatabaseSchema>
```

Figura 5: Resultado da aplicação da regra Disjunção Inter-Esquemas.

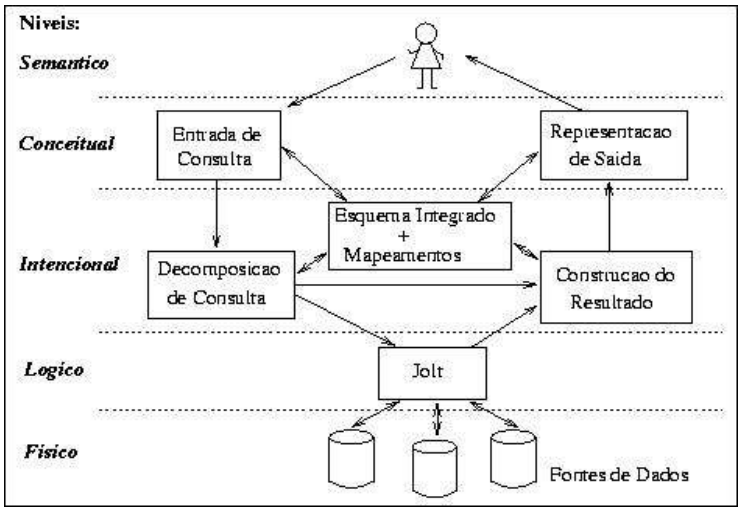

Figura 6: Arquitetura do módulo J-Query

executar tais consultas, as informações contidas na entidade `MappingInformation` são utilizadas para reformular a consulta do usuário em consultas adequadas a cada fonte de informação, como será descrito na próxima seção.

4. J-Query: O Módulo de Consultas à Base Integrada

O objetivo do módulo de consultas, *J-Query*, é permitir que o usuário execute consultas aos esquemas integrados de forma transparente sobre a localização e heterogeneidade dos dados. A arquitetura geral do módulo está ilustrada na Figura 6, e baseia-se no modelo proposto em [MACKINNON 1998]. Inicialmente, o usuário faz a entrada de uma consulta envolvendo as operações relacionais de seleção, projeção e junção (consultas SPJ) através de uma interface gráfica. O módulo J-Query consulta então as informações geradas pelo módulo J-Schemata para obter os mapeamentos entre o modelo integrado e as fontes de informação, bem como a localização das fontes de informação necessárias. A consulta é então desmembrada e cada subconsulta é enviada para o servidor adequado. A interação entre o módulo e as bases de dados é realizada por um servidor Jolt. O framework Jolt facilita o envio e obtenção de resultados de consultas a bancos de dados que estejam cadastrados no seu servidor, através de uma interface padrão. O módulo J-Query obtém as respostas do servidor Jolt e constrói o resultado, apresentando-o ao usuário final através de uma interface gráfica.

4.1. Componentes do módulo *J-QUERY*

Nesta seção serão detalhados os componentes do *J-Query* e o algoritmo de decomposição de consultas, mostrando como as informações geradas na integração de esquemas pelo módulo *J-Schemata* são utilizadas.

A entrada de uma consulta SPJ a um esquema integrado é realizada através de uma interface gráfica. Após digitar o nome do esquema integrado, é exibida uma janela com as abas de "Seleção", "Projeção" e "Junção" e, para cada operação, uma lista de

tabelas e atributos existentes no esquema é exibida de forma que o usuário possa expressar a consulta através de uma combinação de ações de apontar-clicar e digitação de dados. A lista de tabelas e atributos são obtidos do arquivo XML gerado pelo módulo *J-Schemata,* mais especificamente do elemento denominado `Tables` do documento. Em álgebra relacional, o tipo de consulta que a ferramenta suporta tem a seguinte forma:

$$\pi_{t1.a1,\ldots tn.an}(\sigma_{t'1.b1=v1\ \wedge\ \ldots\ \wedge\ t'm.bm<vm}(T1 \bowtie_{c1=c2} \ldots \bowtie_{d1=d2} Tp))$$

onde ti's, t'i's e Ti's são nomes de tabelas, ai's, bi's, ci's e di's são nomes de atributos, vi's são contantes, e a operação de seleção pode ser =, <>, >, <>, >= ou <=. Para descrever o algoritmo de decomposição de consultas, utilizamos uma representação tabular das operações como ilustrada abaixo.

proj [tab, atrib] = {(t1, a1), ..., (tn, an)}
sel [tab, atrib, op, const] = {(t'1, b1, =, c1), ..., (t'm, bm, <, cm)}
juncao [tab1, atrib1, tab2, atrib2] = {(T1, c1, T2, c2), ..., (Tp-1, d1, Tp, d2)}

Aqui, t[a,b,c] **representa uma tabela** t **com atributos** a, b **e** c**,** {a, b} **representa um conjunto com elementos** a **e** b**, e** (a, b) **representa um registro com atributos** a **e** b**.**

Para realizar a decomposição, é necessário mapear as relações e atributos do esquema integrado para as fontes de informação onde os dados estão efetivamente armazenados. Estas informações são obtidas do elemento `MappingInformation` do arquivo XML criado pelo módulo J-Schemata e que podem ser descritas de forma tabular na estrutura abaixo.

map: (fontes: {(tab, uri, atribs: {(atrib, dominio)})},
modeloGlobal:{(tab, atribs:{(nome, dominio, links: {(tab, atrib)})})})

O atributo fontes contém o nome das tabelas das bases locais (tab)**, bem como sua localização** (uri) **e seus atributos** (atribs)**. Estas informações são representadas no documento XML pelos elementos `source` e pelos elementos `AttrLink` que os referenciam. Todos os atributos de modeloGlobal são obtidos dos elementos `Table` em `MappingInformation`. Ou seja, para cada tabela do esquema integrado são mantidos o nome** (tab) **e atributos** (atribs)**, sendo que o mapeamento de cada atributo global para atributos em tabelas fonte é mantido em** links**. Por exemplo, o documento XML apresentado na Figura 5 tem a representação mostrada na Figura 7.**

O algoritmo apresentado na Figura 8 utiliza estas informações para construir consultas expressas nos esquemas das bases de dados locais que contribuem para a obtenção do resultado da consulta ao esquema integrado.

Detalharemos o algoritmo considerando que a seguinte consulta foi submetida ao esquema integrado na Figura 5:

$$\pi_{NUMCPF-PESS}(\sigma_{NOME-PESS='JOAO'}(PESSOA))$$

As linhas 1 e 2 identificam quais os atributos do esquema integrado são necessários para responder a consulta do usuário, considerando-se as operações de Seleção (sel)**, Projeção** (proj) **e Junção** (juncao)**. Ao final desse passo, a variável** tabAtrib **contém todos os nomes de atributos do modelo global que se deve acessar para atender a consulta. No exemplo, t**abAttrib={(PESSOA, NOME-PESS), (PESSOA, NUMCPF-PESS)}**.**

```
map: (fontes: {(ALUNO,  jdbc:postgresql:academico, atribs:
                    {(NUMALU-ALU, INTEGER(10)), (NOME-ALU, CHAR(60)),
                     (ANO2GRAU-ALU, INTEGER(10)), (NUMCPF-ALU, INTEGER(13))}),
                (DPFUNC, jdbc:postgresql:pessoal, atribs:
                    {(NOMEFUNC-DP, CHAR(50)), (NOME-DP, CHAR(50)), (NUMCPF-DP, INTEGER(15))})},
      modeloGlobal:{
            (ALUNO,  atribs:
              {(NUMALU-ALU, INTEGER(10), links: {(ALUNO, NUMALU-ALU)}),
               (NOME-ALU, CHAR(60), links: {(ALUNO, NOME-ALU)}),
               (ANO2GRAU-ALU, INTEGER(10), links: {(ALUNO, ANO2GRAU)}),
               (NUMCPF-ALU, INTEGER(13), links: {(ALUNO, NUMCPF-ALU)})}),
            (DPFUNC, atribs:
               {(NOMEFUNC-DP, CHAR(50), links:{(DPFUNC, NOMEFUNC-DP)}),
                (NOME-DP, CHAR(50), links:{(DPFUNC, NOME-DP)}),
                (NUMCPF-DP, INTEGER(15), links:{(DPFUNC, NUMCPF-DP)})}),
            (PESSOA, atribs:
               {(NUMCPF-PESS, INTEGER(13),
                    links: {(ALUNO,  NUMCPF-ALU), (DPFUNC, NUMCPF-DP)}),
                (NOME-PESS, CHAR(60),
                    links: {(ALUNO, NOMEALU-ALU), (DPFUNC, NOMEFUNC-DP)})})})
```

Figura 7: Representação tabular do mapeamento de esquemas

Algoritmo decomposicao

Entrada: *selecao*, *projecao*, *juncao*: uma consulta SPJ
map: mapeamento entre esquemas integrado e locais
Saida: relacao com atributos em *projecao*

// obtém tabelas e atributos utlizados na consulta
1. $tabAtrib[tab, atrib] := \pi_{tab,atrib}(selecao) \cup \pi_{tab,atrib}(projecao)$;
2. $tabAtrib := tabAtrib \cup \pi_{tab1,atrib1}(juncao) \cup \pi_{tab2,atrib2}(juncao)$;

// cria uma tabela local para cada tabela utilizada na consulta
3. para cada $globTab$ em $\pi_{tab}(tabAtrib)$ faca
4. $globAtribs := \pi_{atrib}(\sigma_{tab=globTab}(tabAtrib))$;
5. $tabLocal$ = CREATE TABLE $globTab$ com atributos $globAtribs$;
6. $mapsTab := \pi_{atribs}(\sigma_{tab=globTab}(map.modeloGlobal))$;

// obtém tabelas e atributos locais necessários para popular a tabela local
7. $fontes[tab, atrib] := \emptyset$;
8. para cada $globAtrib$ em $globAtribs$ faca
9. $fontes := fontes \cup \pi_{links}(\sigma_{atrib=globAtrib}(mapsTab))$;

// obtém de cada tabela local os atributos necessários
10. para cada $locTab$ em $\pi_{tab}(fontes)$ faca
11. $locAtribs := \pi_{atrib}(\sigma_{tab=locTab}(fontes))$;
12. $serv := \pi_{uri}(\sigma_{tab=locTab}(map.fontes))$;
13. res:= resultado da consulta "select :$locAtribs$ from :$locTab$" submetido ao servidor $serv$;
14. $tabLocal = tabLocal \cup res$;
15. $resFinal$ = execucao de $jucao$, $selecao$ e $projecao$ nas tabelas locais;
16. retorna $resFinal$;

Figura 8: Algoritmo de decomposição de consultas e construção do resultado

As linhas 3 a 5 criam em uma base de dados temporária as tabelas que têm em sua estrutura os atributos do modelo global necessários. Ou seja, para cada valor de globTab, é criada uma tabela com os nomes de atributos que foram colocados em globAtribs. **No exemplo, existe um único valor para a variável** globTab **=** PESSOA **e é criada uma tabela com atributos em** globAtribs = {NOME-PESSOA, NUMCPF-PESS}**). Chamaremos esta tabela de** PESSOA-T**.**

As linhas 6 a 9 identificam quais os atributos necessários nas fontes de informação. Ao final desse passo, a variável fontes **contém os nomes de todos os atributos, e respectivas tabelas, que devem ser recuperados nas fontes de informação. Dando sequência ao exemplo,** fontes={(ALUNO, NUMCPF-ALU), (DPFUNC, NUMCPF-DP), ALUNO, NOMEALU-ALU, DPFUNC, NOMEFUNC-DP}**. Uma consulta SQL é então construída para acessar cada uma das tabelas fontes projetadas sobre o conjunto de atributos em** locAtribs**. No exemplo, as consultas geradas são "select** NUMCPF-ALU, NOMEALU-ALU **from** ALUNO**" e "select** NUMCPF-DP, NOMEFUNC-DP **from** DPFUNC**".**
As linhas 10 a 14 copiam as tuplas trazidas das bases fonte para as tabelas temporárias criadas no banco de dados local.

As linhas 15 e 16 submetem ao sistema gerenciador de banco de dados local a consulta feita pelo usuário final e retornam para ele os dados solicitados. No exemplo, a consulta realizada é "select NUMCPF-PESS **from** PESSOA-T **where** NOME-PESS = 'JOAO'**". A relação resultante é então apresentada ao usuário em forma tabular.**

Os módulos *J-Schemata* e *J-Query* da ferramenta *J-Integrator* foram implementados com a linguagem de programação Java por ser um padrão aberto, independente de plataforma e oferecer várias soluções disponíveis que facilitam o seu desenvolvimento e manutenção. Para a implementação do módulo *J-Schemata*, a biblioteca *Jgraph*, um componente *Java Swing*, foi utilizada pois fornece componentes adequados ao desenho de grafos. O módulo *J-Query* utiliza o framework Jolt Reports em sua implementação. Este framework possibilita acessar qualquer banco de dados que possua um driver JDBC, e obter os dados em diversos formatos, dentre eles o XML, HTML, e PDF. Consultas SQL podem ser submetidas a um servidor Jolt por um documento XML com estrutura pré-estabelecida. A utilização do formato XML pela ferramenta *J-Integrator* serviu de padrão não somente para interação com esquemas de bases de dados, mas também para permitir estruturar a ferramenta de tal forma que a torne fácil de ser estendida. Um dos princípios no desenvolvimento foi justamente permitir que novos esforços para aprimorar a ferramenta se tornassem mais simples e modulares.

5. Trabalhos Relacionados

Embora o problema de integração de esquemas tenha sido extensivamente abordado na literatura, ele continua apresentando desafios, uma vez que trata-se de um problema de difícil automatização. Diversos sistemas gerenciadores de bancos de dados comerciais apresentam produtos que permite interoperabilidade, como o Transac-SQL e VQL do SQLServer, SQL*PLUS e SQL*STAR da Oracle, Ingres/Star e DataJoiner da IBM. Porém, estas ferramentas podem ser consideradas como sistemas de bancos de dados

homogêneos suportando conexões com bancos de dados externos através de "gateways" segundo [GARDARIN 1995]. Seu objetivo é facilitar a interoperabilidade entre bancos de dados e não possui facilidades para a geração de esquemas integrados resolvendo conflitos existentes, que é o objetivo do módulo *J-Schemata*.

A ferramenta *J-Integrator* tem objetivos e semelhanças com o sistema Nimble [DRAPER 2001], que utiliza o formato XML para representar resultados de consultas submetidas a diferentes fontes de dados, bem como com o sistema UNIBASE, que é dividido em dois subsistemas, *Unifier e Accessor*, que facilitam a integração de bases e consultas aos esquemas integrados, respectivamente. Outras abordagens para integração de esquemas são utilizados pelo CoopBDH [CAVINATO 2001], baseado no conceito de agentes cooperantes, e AutoMed [JASPER 2003] no qual a integração é realizada por sequências reversíveside transformações de esquemas.

Segundo [GARCIA-MOLINA 2001, p. 635-644], as três abordagens mais comuns para consultar bases distribuídas são: bancos de dados federados, na qual as fontes de informação são independentes, mas uma fonte pode solicitar dados de outras; datawarehouse, que mantém cópias de várias fontes; e mediação, na qual um componente de software (chamado *mediador*) acessa um *banco de dados virtual* com o qual o usuário interage. A ferramenta *J-Query* segue a abordagem de mediação. Dentro desta abordagem, um algoritmo para decomposição de consultas foi proposto em [LAMBRECHT 1997] usando o conceito de regras de reformulação relacionando visões do modelo global (esquema integrado) e as fontes de informação. No modelo proposto, a consulta do usuário vai sendo transformada usando estas regras até que se obtenha regras onde só se acesse fontes de informação diretamente. Outra abordagem é proposta em [LEVY 1996] na qual *registros de capacidade* são utilizados para manter informações sobre que tipo de consultas as fontes podem responder.

6. Conclusão

A integração de esquemas é, reconhecidamente, um processo de dificil automatização. A necessidade de se conhecer a fundo detalhes semânticos que muitas vezes escapam a capacidade descritiva dos modelos de dados nos levam a investir em ferramentas que apoiam a execução do processo manual dimuindo o custo final e contribuindo para um esquema global mais completo e adequado. A meta no desenvolvimento desta ferramenta foi apoiar o processo de integração de esquemas e consultas as mesmas. Por se tratar de uma tarefa bastante complexa, o foco foi mantido na representação e mapeamento de algumas regras de integração, através de uma interface gráfica amigável.

A ferramenta foi utilizada para fazer a integração de algumas bases de dados da Universidade Federal do Paraná e as regras de integração mostraram-se suficientes para a tarefa. Porém, a ferramenta pode ser estendida para dar suporte a novas regras de integração facilmente, pois ela foi desenvolvida de forma modular. Ou seja, a criação de novas regras e módulos funcionais não afeta significativamente o funcionamento de módulos já operantes. Além disso, a utilização do formato XML como meio de representação de esquemas e mapeamentos facilita a interface entre os módulos.

A ferramenta pode ser melhorada e estendida de diversas formas e dentre as idéias de trabalhos futuros citamos: implantação de princípios de interface homem-máquina para tornar a ferramenta mais intuitiva e amigável; criação de módulos tutoriais que relatem os princípios e técnicas utilizadas para realizar cada passo de integração; definição de cardinalidade dos relacionamentos; adição de módulos capazes de indicar possíveis pontos de integração; aplicação de algoritmos de teoria dos grafos para otimizar o diagrama ERC+ visualizado pela ferramenta *J-Schemata*; e estender o poder de expressão das consultas submetidas ao esquema integrado, incluindo consultas incompletas.

Referências

BATINI, C.; LENZERINI, M.; NAVATHE, S B. *A comparative analysis of methodologies for database schema integration*, ACM Computing Surveys (CSUR), Volume 18 Issue 4, December 1986.

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

BOAG, S et. al. *Xquery 1.0: An XML Query Language*, W3C Working Draft, http://www.w3.org/TR/2005/WD-xquery-20050404/

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.

CLARK, J. et. al. XSL Transformations (XSLT) Version 1.0, W3C Recommendation. http://www.w3.org/TR/xslt

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

GARCIA-MOLINA, H.; ULLMAN, J. D.; WIDOM, J. *Implementação de sistemas de bancos de dados*. **Rio de Janeiro: Campus, 2001.**

GARDARIN G.; GANNOUNI, S.; FINANCE, B. *IRO-DB: A Distributed System Federating Object and Relational Databases*. Object-Oriented Multibase Systems, Prentice Hall, 1995.

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

JAVA. http://java.sun.com.

JGRAPH. http://www.jgraph.org.

JOLT. http://www.jbanana.org/jbananaDownload/JBananaController?ServletState=10

LAMBRECHT, E.; KAMBHAMPATI, S. *Planning for information gathering: **a tutorial survey***. **ASU CSE Technical Report 96-017, May 1997.**

LEVY, A. Y.; RAJARAMAN, A.; ORDILLE, J. J. *Querying heterogeneous information sources using source descriptions*. **Proceedings of the 22nd International Conference onVery Large Databases. VLDB-96, Bombay, India, September 1996.**

MACKINNON, L.; MARWICK, D.; WILLIAMS, M. *A model for query decomposition and answer construction in heterogeneous distributed database systems*. **Journal of Intelligent Information Systems 11, 1998.**

NOGUEIRA, V. C. *J-Query: **uma ferramenta para consulta a bancos de dados heterogêneos distribuídos***. **Curitiba, 20045, 81 f. Dissertação (Mestrado em Informática) - Setor de Ciências Exatas, Universidade Federal do Paraná.**

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

SCOPIM, K. da S. *J-Schemas Integrator: **uma ferramenta para integração de banco de dados***. **Curitiba, 2003, 74 f. Dissertação (Mestrado em Informática) - Setor de Ciências Exatas, Universidade Federal do Paraná.**

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.**

UNIBASE. ***A System for Unifying Heterogeneous Databases***, **http://adrg.eller.arizona.edu/research/projects/heterogenous/heterogeneous.aspx**