UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2009.1

UMA FERRAMENTA DE TRANSFORMAÇÃO DE ESQUEMAS OBJETO-RELACIONAIS PARA ONTOLOGIAS

TRABALHO DE GRADUAÇÃO

Aluno: Felipe Franco (fanf@cin.ufpe.br)
Orientadora: Ana Carolina Salgado (acs@cin.ufpe.br)

Julho de 2009

Resumo

A popularização das ontologias, decorrente do surgimento da Web Semântica, trouxe consigo a necessidade de se ter as mais diferentes fontes de dados representadas em ontologias para que seja viável a integração entre elas.

O mapeamento de um esquema de banco de dados relacional em ontologia não é uma área de pesquisa e desenvolvimento nova. Vários métodos e ferramentas já foram desenvolvidos e implementados para atender essa necessidade. Porém, ainda existe uma carência de métodos e ferramentas com o objetivo de realizar o mapeamento de um esquema de bando de dados objeto-relacional em uma ontologia. Este trabalho visa o desenvolvimento de uma ferramenta que supra essa necessidade.

Palavras Chave: Ontologias, Mapeamento de Ontologias, Web Semântica, SGBD Objeto-Relacionais

Abstract

The popularizations of the ontologies, which happened because of the Semantic Web boom, brought with it the necessity of having the most different data sources represented by ontologies in order to make possible their integration..

The mapping between a relational data base and an ontology is not a new research and development area. Several methods and tools have been developed to fulfill this necessity. However, there is still a lack of methods and tools built to make the mapping of an object-relational database into an ontology. This work is intended to build a tool to fulfill this necessity.

Key words Ontology, Ontology Mapping, Semantic Web, DBMS Object-Relational

Sumário

Lista de Figuras

Lista de Quadros

1. Introdução

A popularidade das ontologias [Gruber, 1993; Guarino, 1998] vem aumentando cada vez mais desde o surgimento da Web Semântica [Berners-Lee et al., 2001]. Nos últimos anos, a quantidade de ontologias disponíveis na Web apresentou uma taxa de crescimento considerável. Entretanto, a maioria das fontes na Web continua disponibilizando seus dados através de esquemas representados em modelos de dados tradicionais, e.g. esquemas relacionais, documentos XML e páginas web [Ghawi and Cullot, 2007].

Uma forma para garantir a interoperabilidade entre fontes de dados heterogêneas na Web, é transformar os esquemas destas fontes (esquemas de origem) em ontologias (esquema de destino) [de Laborda and Conrad, 2005]. Durante este processo, são aplicadas regras para transformar os elementos (conceituais) do esquema de origem em elementos (conceituais) correspondentes do esquema de destino [Nyulas et al., 2007]. Por exemplo, se o esquema de origem for relacional, tabelas e colunas são transformadas em classes e propriedades da ontologia. Em geral, esta transformação produz um conjunto de correspondências entre os elementos contidos no esquema de origem e os elementos da ontologia [Ghawi and Cullot, 2007].

A transformação de esquemas tradicionais para ontologias é considerada uma pesquisa interdisciplinar nas comunidades de Banco de Dados e Web Semântica [Hu and Qu, 2007]. Atualmente, existem muitas abordagens e ferramentas que lidam com esta transformação [Cullot et al., 2007; Cerbah, 2008]. Basicamente, tais abordagens e ferramentas são classificadas em duas categorias principais [Ghawi and Cullot, 2007]: (i) abordagens para criação de uma ontologia a partir de um esquema de origem [Nyulas et al., 2007]; e (ii) abordagens para geração de correspondências entre um esquema de origem e uma ontologia já existente [Barrasa et al., 2004]. Este trabalho segue a primeira abordagem e possui como objetivo o desenvolvimento de uma ferramenta de transformação que permita a criação de uma ontologia a partir de um esquema de dados objeto-relacional. Desta forma, a ferramenta deve receber como entrada um esquema de origem e transformá-lo automaticamente em uma ontologia no formato OWL [OWL,

2009]. Para tal, deverão ser especificadas e codificadas regras de transformação entre os elementos do esquema de origem e da ontologia.

2. Conceitos Básicos

Nesta sessão, apresentaremos alguns conceitos básicos sobre os assuntos tratados no documento.

2.1. Conceitos Objeto-relacional

Com o advento e consolidação da programação orientada a objetos, uma grande quantidade de sistemas passou a ser desenvolvida em linguagens que seguem este paradigma. Paralelamente a essa consolidação, os sistemas de gerenciamento de banco de dados orientados a objetos surgiram [Junior,2000].

Um sistema de gerenciamento de banco de dados objeto-relacional pode ser tratado como uma forma de estender o sistemas de banco de dados relacionais de modo a que funcionalidades sejam incluídas para dar suporte a aplicações desenvolvidas em linguagens que seguem o paradigma de orientação a objetos.

As principais características de um sistema de gerenciamento de dados objeto-relacional são a possibilidade de extensão de tipos de dados básicos, o uso de objetos complexos, conceito de herança e regras.

SGBD, Sistema de Gerenciamento de Banco de Dados, objeto-relacionais permitem a criação de novos tipos e operadores para esses tipos. Tipos básicos e operadores em SGBDs relacionais, algumas vezes, não atendem por completo às necessidades do usuário. Nos SGBD objeto-relacionais, esse problema é tratado de forma que o usuário pode definir seus próprios tipos básicos e seus respectivos operadores.

Além disso, eles dão suporte a objetos complexos, objetos compostos por vários tipos básicos ou outros objetos complexos definidos pelo usuário. Também, listas, pilhas, filas, árvores e outros construtores têm suporte em SGBD objeto-relacionais, o que dá um grande poder ao desenvolvedor.

O conceito de herança também é um diferencial. Ele permite que tipos possam ser agrupados herdando funções e características dos tipos pais. Aumenta a modularidade e reutilização dos componentes já definidos.

2.2. Ontologias

Nas últimas décadas, ocorreu um aumento exponencial na quantidade de informação produzida e documentada. A grande maioria dessa informação está armazenada em computadores, principalmente em bancos de dados.

Grande parte dessa informação está na chamada “Web profunda” (do inglês *Deep Web*) [Bergman, 200]. Ela corresponde a páginas web estáticas e banco de dados que dão suporte a páginas Web.

Devido a essa grande quantidade de informação, a recuperação dessa informação e o seu tratamento passou a ser uma questão de alta relevância, visto que, muitas vezes, essa tarefa era realizada de maneira inapropriada, resultando em uma recuperação custosa ou mal feita.

A pesquisa sobre ontologias tem aumentado e se expandido bastante. O termo ontologia, para a filosofia, se refere à teoria sobre a natureza da existência, ou seja, sobre que tipos de coisas existem [Berners-Lee, 2001]. Já nos dias atuais, o termo tem se tornado muito comum nas áreas de Inteligência Artificial, Lingüística Computacional e Teoria dos Bancos de Dados. Seu uso tem sido notório em campos de pesquisa como: engenharia de conhecimento, representação de conhecimento, modelagem qualitativa, engenharia de linguagens, design de bancos de dados, modelagem informacional, extração e recuperação de informação, gerenciamento de conhecimento e no design de sistemas baseados em agentes.

Tendo como base a utilização de ontologias nos sistemas baseados em conhecimento, uma ontologia é um artefato constituído de um vocabulário específico usado para descrever uma certa realidade e um conjunto de inferências realizadas sobre esse vocabulário. Essas inferências são expressas na forma de uma lógica de primeira ordem onde cada palavra do vocabulário aparece como um termo unário (nomes) ou binário (relações) [Almeida, 2003].

Em outras palavras, uma ontologia é uma especificação de uma contextualização [Gruber,1993]. Uma ontologia tem como objetivo representar tudo que existe. O conjunto de objetos que podem ser definidos em um

domínio pré-definido e as relações entre eles são descritos através de um vocabulário definido representando o conhecimento. Nomes de classes, relações, funções e objetos podem ser associados de forma a que se possa expressar, de uma forma entendível ao homem, o que esses nomes descrevem e axiomas formais que definem o relacionamento entre esses termos.

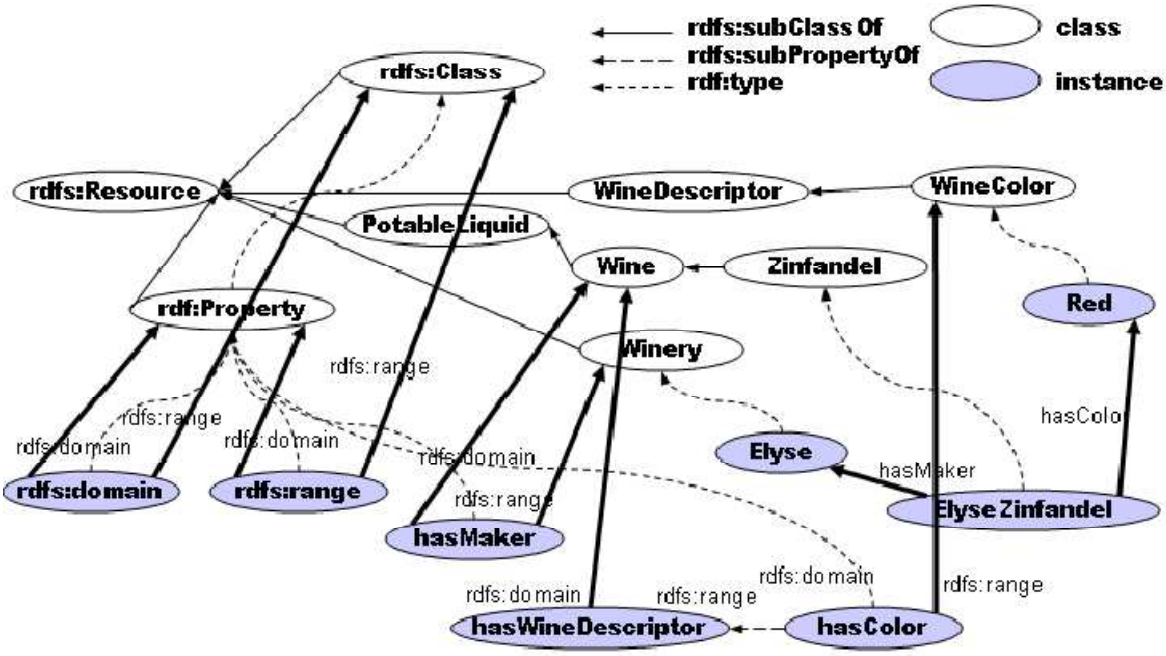

Figura 1 – Exemplo de Ontologia

Uma ontologia pode ser associada a um esquema conceitual de um banco de dados. Ela fornece uma descrição lógica dos dados compartilhados de forma que os programas e bancos interoperem sem ter que compartilhar estruturas de dados. Porém, enquanto um esquema conceitual define relações de dados, uma ontologia define termos pelos quais se podem representar conhecimento. A Figura 1, extraída de [Koide, 2004] apresenta a representação de uma ontologia exemplo.

Uma ontologia define classes de objetos, as relações entre eles, indivíduos e atributos. Classes são tipos de objetos ou conjuntos de objetos. Indivíduos são objetos básicos que pertencem a uma classe. Atributos são propriedades dos objetos e relacionamentos são o modo como objetos podem se relacionar entre si.

2.3 OWL

A OWL Web Ontology Language é a linguagem mais usada para se definir e instanciar ontologias na web. Uma ontologia OWL pode incluir descrições de classes, propriedades e suas instâncias. Dada uma ontologia OWL, pode-se raciocinar acerca de suas classes e seus indivíduos da ontologia através da semântica formal da linguagem OWL [McGuiness, 2004].

OWL foi criada com objetivo de ser a principal linguagem de especificação de ontologias usada na Web Semântica, onde um significado é explicitamente adicionado à informação de modo que processamento e integração da informação pelas máquinas se dê de forma fácil e automática.

Diferentemente de XML, que não impõe restrições semânticas no significado dos documentos, OWL possui um vocabulário mais rico para descrever propriedades de classes, como o relacionamento entre classes e características de propriedades.

OWL se decompõe em três: OWL Lite, OWL DL e OWL Full. OWL Lite dá suporte à classificação hierárquica e pequenas restrições. Por exemplo, pode-se fazer uma restrição de cardinalidade sobre um valor para que ele só possa ser zero ou um. OWL DL dá um grande poder de expressividade e garante a completude computacional (todas as conclusões podem ser computáveis) e a decidibilidade (serão concluídas em tempo finito). OWL Full garante máxima expressividade, porém sem garantias computacionais.

As estruturas básicas de OWL são *class, DatatypeProperty* e *ObjectProperty.* Uma *class*, ou classe, é o equivalente a um tipo em um esquema objeto-relacional. O relacionamento de uma *class* com seus atributos que não são outras classes se dá por meio de uma *DatatypeProperty.* O atributo nome de pessoa é mapeado em OWL no Quadro 1.

```
<owl:DatatypeProperty rdf:ID=" pessoa _nome">
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <rdfs:domain rdf:resource="# pessoa "/>
</owl:DatatypeProperty>
```

Quadro 1 - DataTypeProperty

O relacionamento de uma classe com outra classe é feita através de uma *ObjectProperty.* A Quadro 2 mostra o exemplo onde uma classe Telefone é atributo da classe Pessoa.

```
<owl:ObjectProperty rdf:ID=" pessoa _telefone">
        <rdfs:range rdf:resource="#telefone"/>
        <rdfs:domain rdf:resource="# pessoa "/>
</owl:DatatypeProperty>
```

Quadro 2 - ObjectProperty

3. Trabalhos Relacionados

Os trabalhos com foco no mapeamento entre banco de dados e ontologias existentes na literatura podem ser classificados em duas categorias: (1) os que buscam fazer o mapeamento de um banco de dados para uma ontologia já existente e (2) os que buscam fazer um mapeamento para uma ontologia gerada pela própria ferramenta, levando em consideração propriedades do banco de dados. Neste trabalho, focamos nos trabalhos envolvidos no segundo caso.

3.1. DataMaster

A ferramenta DataMaster [Nyulas 2007] é um plug-in para Protégé que permite importar a estrutura e os dados de banco de dados relacional de forma configurável de maneira que o esquema pode ser mapeado para ontologias como Protégé-OWL ou Relational.OWL [de Laborda 2005].

Usando esta ferramenta, pode-se mapear facilmente os dados de um banco para uma ontologia. Porém, o resultado pode não atender plenamente às expectativas dos usuários atraídos pelo poder e riqueza da Web Semântica, dado que o resultado da transformação se assemelha muito com o seu banco de dados relacional.

3.2. DB2OWL

DB2OWL[Cullot 2007] é outra ferramenta desenvolvida com o intuito de mapear bancos de dados relacionais em ontologias. As ontologias criadas são expressas em OWL-DL, que permite uma maior expressividade, diferentemente do DataMaster, que as expressa em OWL-Full [McGuiness 2004]. Durante o processo de mapeamento, um documento R2O [Barsa 2004] é automaticamente gerado com informações sobre a relação entre a ontologia gerada e o banco de dados original. Este documento possui informações como: uma descrição completa do esquema do banco; um conjunto de mapeamento entre os conceitos gerados e suas respectivas colunas; e um conjunto de mapeamentos entre as relações e os atributos. Esse documento é gerado no

intuito de que se possa fazer uma tradução de consultas ontológicas para consultas SQL. A Figura 2 mostra a arquitetura básica de DB2OWL.

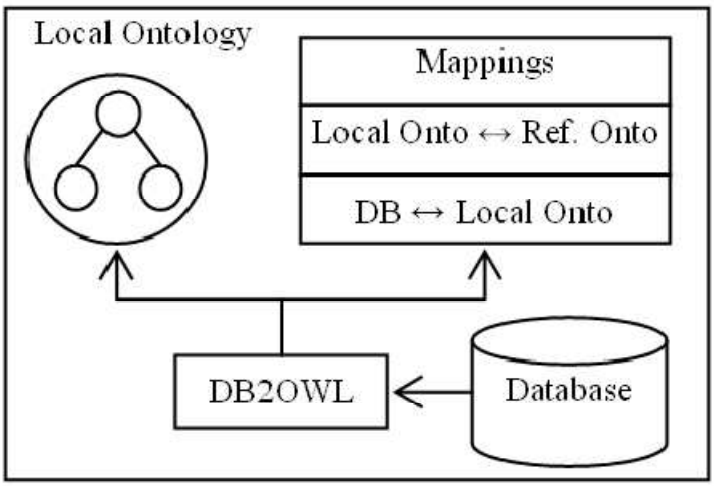

Figura 2 – Esquema DB2OWL

Em [Ghawi 2007], o autor defende uma arquitetura para consultas em bancos de dados heterogêneos que usam DB2OWL para criar ontologias locais para cada fonte de dados. A arquitetura consta de vários módulos. O primeiro módulo é o da base de conhecimento, onde fica a ontologia de referência global. Essa ontologia é criada de forma que cada conceito e atributo das ontologias locais tenham um conceito e atributo equivalentes nela. O módulo Provedor de Dados encapsula a fonte de informação e a ontologia local gerada pelo DB2OWL, assim como o documento de mapeamento entre a base e a ontologia local e outro de mapeamento entre a ontologia local e a de referência. O módulo de mapeamento tem como objetivo gerar os mapeamentos entre a ontologia local e a de referência. O módulo de consulta é por onde a consulta é feita pelo usuário, expressa na linguagem SPARQL [Prud'Hommeaux 2006]. A consulta é desmembrada em partes que seguem cada uma a o seu respectivo provedor de dados. O resultado é, então, agrupado e retornado ao usuário por meio do módulo de visualização que o apresenta ao usuário de maneira adequada. A Figura 3 mostra a arquitetura proposta em [Ghawi 2007].

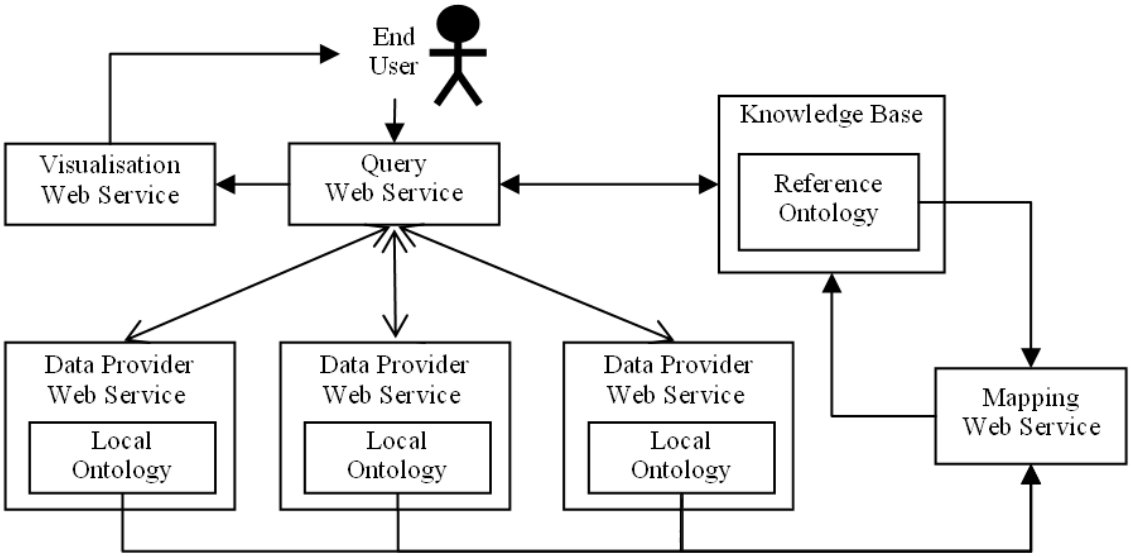

Figura 3 – Arquitetura proposta por Ghawi

3.3. RDBToOnto

Uma das ferramentas mais interessantes encontradas foi a RDBToOnto [Cerbah 2008]. A RDBToOnto é uma ferramenta de mapeamento entre um banco de dados relacional em uma ontologia que é gerada pela própria ferramenta. A diferença da RDBToOnto para as demais previamente citadas é que ela não gera apenas um espelho do esquema do banco de dados fonte. O objetivo dela é, no final do processo, encontrar ontologias que melhor representem a estrutura conceitual do banco. Para isso, deve-se identificar padrões estruturais que podem ser explorados para gerar uma ontologia mais precisa. Essa identificação pode ser feita automaticamente pela ferramenta, mas também pode ser configurada pelo usuário. Identificando esses padrões, a ferramenta possui um mecanismo orientado a dados que faz a "categorização" dos dados automaticamente. A Figura 4, extraída de [Cerbah 2008], mostra um exemplo onde as classes *Seafood, Beverage, Condiment* foram criadas com base na categorização. Nesse caso, a coluna categoria foi explorada para agregar valor contextual à ontologia.

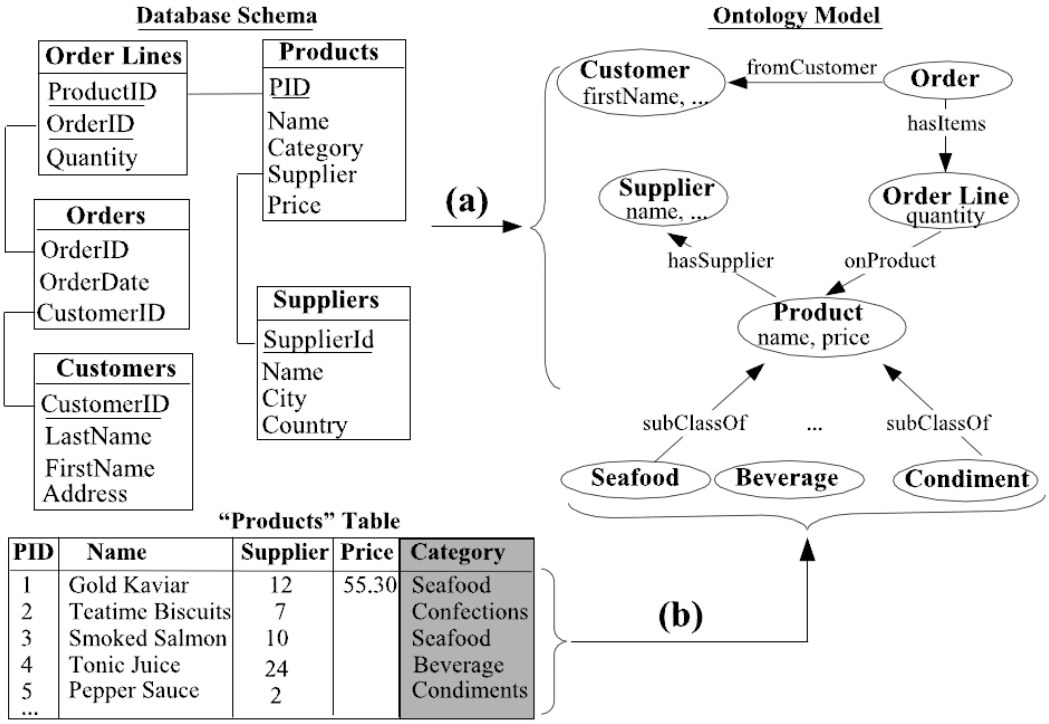

Figura 4 – RDBToOnto

3.4. RelOnto

A RelOnto [Lopes 2007] é muito similar a RDBToOnto. Porém possui uma pequena diferença. Enquanto a RDBToOnto apenas considera padrões em que o conteúdo do atributo vira subclasse na ontologia, a RelOnto vai um pouco mais além. Ela é capaz, através do mecanismo de categorização criado pela ferramenta, de identificar e extrair novas classes e relacionamentos que não seriam identificadas apenas com a análise do esquema. Com isso, é possível identificar o relacionamento entre atributos assim como possíveis especializações entre eles. Como no caso de uma tabela Departamento que possui uma coluna referente ao centro em que está.

Dentre as abordagens incluídas na categoria (1), indicada no início desta seção, merecem destaque a visAvis [Konstantinou 2006] e R2O [Barrasa 2004],

3.5. Análise Comparativa

A ferramenta DB2OWL se mostra bastante eficaz e a proposta de arquitetura feita em [Ghawi 2007] para as consultas a ontologias é muito

eficiente. Porém a ferramenta não faz o tratamento das classes candidatas como RDBToOnto e a RelOnto.

A ontologia gerada pela DataMaster é bastante limitada e o fato de ela só rodar no Protégé faz com que ela não seja muito relevante para esse trabalho.

De todas as ferramentas analisadas, a RelOnto é a mais completa. Ela estende a categorização, também apresentada por RDBToOnto não analisando, somente, o esquema e enriquecendo a ontologia gerada.

3.6. Considerações

Conforme analisado, nenhuma das ferramentas dá suporte ao mapeamento de conceitos objeto-relacional. Todas só tratam de bancos de dados relacionais.

A ferramenta criada nesse trabalho vem para suprir essa lacuna deixada pelas ferramentas existentes atualmente e dar suporte à criação de ontologias OWL a partir de bancos de dados objeto-relacional, mais precisamente, de esquemas gerados no SGBD Oracle.

Para isso, foi estendida a ferramenta RelOnto, que já faz o mapeamento de bancos de dados relacionais para ontologias OWL, para que ela também possa dar suporte a bancos de dados objeto-relacionais.

4. Processo de geração de ontologias

O processo de geração de ontologias da ferramenta RelOnto e sua extensão para incluir características objeto-relacionais é descrito nessa sessão do documento. Esse processo pode ser dividido em diversas etapas. Entre elas a etapa de extração do esquema e de metadados; classificação das relações, obtenção de cardinalidade e classificação dos relacionamentos; análise dos atributos e classes candidatas e, por fim, a aplicação das regras para a geração da ontologia.

A Figura 5 mostra o esquema do processo de geração de ontologias. Cada etapa desse processo será explicada detalhadamente a seguir.

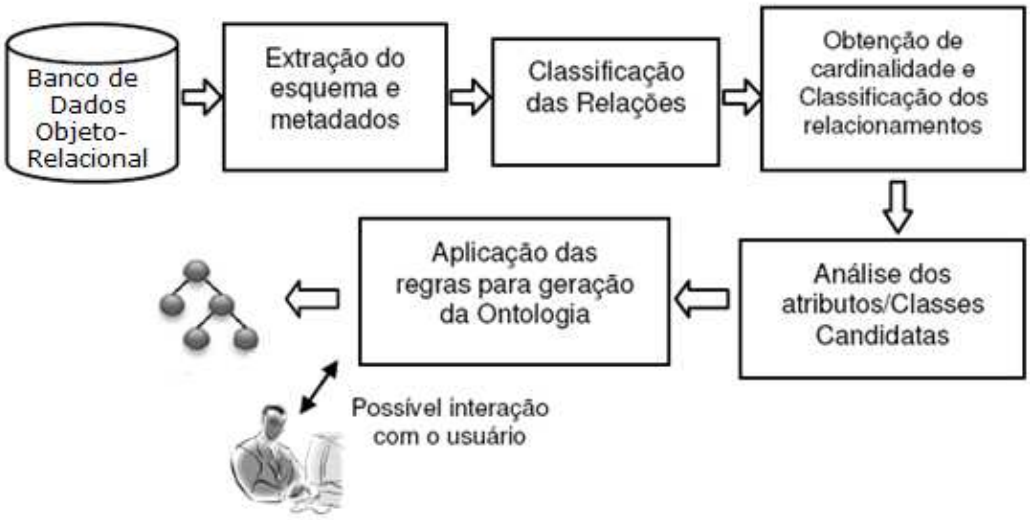

Figura 5 – Processo de geração de ontologias

4.1. Extração do esquema e metadados

É na etapa de extração de esquema e metadados que as informações relativas ao banco de dados do qual se pretende gerar uma ontologia OWL é extraída. Informações importantes como relações, atributos, chaves, restrições, herança, tipos, objetos complexos, entre outros são obtidas e armazenadas para que possa ser usada posteriormente na geração da ontologia.

Conceitos objeto-relacionais não eram tratados pela ferramenta. Assim, foi necessária a adequação para que referências, Nested Tables, Varrays, entre outros, fossem detectados e tratados corretamente.

4.2. Classificação das Relações

Com base na informação extraída na etapa anterior, é possível classificar as relações em sete tipos: relação de entidade forte, relação de entidade fraca, relação de entidade especializada, relação de entidade fragmentada, relação de associação, relação de atributo multivalorado e relação irregular. Essa classificação é feita com base nos trabalhos realizados por [Sonia,2008, Alhajj, 2003, Chiang,1995].

Relação de entidade forte é aquela na qual nenhuma chave primária é uma chave estrangeira. Uma relação de entidade fraca é aquela na qual sua chave primária ou parte dela é chave estrangeira. Relação de especialização define uma especialização de outra entidade e tem uma chave estrangeira como chave primária. Também pode ser uma relação fragmentada, uma relação onde a chave primária é uma chave estrangeira. Essa decisão é tomada com base na cardinalidade dos relacionamentos ou por meio da interferência do usuário. Um relacionamento N:N resulta uma relação de associação. Uma relação de atributo multivalorado é aquela onde a chave primária de uma relação é a combinação da chave primária de exatamente outra relação mais um atributo dela. Qualquer outro relacionamento é classificado como uma relação irregular.

Essa classificação só era feita para tabelas e foi necessário adicionar mais um componente para que os demais conceitos objeto-relacionais fossem tratados de forma apropriada. Esse componente tem como objetivo identificar os componentes objeto-relacionais extraídos do banco, como *ObjectTables* e tipos e fazer a classificação para esses componentes também. Esses componentes são, entre outros, *ObjectTable, Type, Nested Table* e *Varray.*

4.3 Obtenção de cardinalidade e classificação dos relacionamentos

Esta etapa é importante para que, durante a fase de geração da ontologia, seja possível criar as propriedades de objeto necessárias, bem como determinar as cardinalidades vinculadas a elas.

A cardinalidade dos relacionamentos é obtida. Valores mínimos e máximos são aferidos de acordo com o algoritmo apresentado em [Alhajj, 2003]. Além disso, as chaves são analisadas de acordo com a categoria do relacionamento.

De forma semelhante à etapa anterior, foi necessário a criação de outro módulo para que os conceitos objeto-relacionais também sejam classificados e sua cardinalidade obtida.

4.4. Análise dos atributos/ Classes candidatas

Essa é a última fase antes da ontologia ser gerada. Nessa fase, uma parte muito importante do processo de geração da ontologia é realizada, os atributos são analisados com o objetivo de se achar possíveis classes adicionais para enriquecer a ontologia. A Figura 6 mostra um exemplo de banco de dados no qual classes adicionais foram geradas para a ontologia de forma a enriquecê-la.

Departamento(idDepartamento, nome, centro)

Curso(codCurso, nome, idDepartamento)
idDepartamento referencia *Departamento*

Pessoa(idPessoa, CPF, nome, dataNascimento, sexo)

Estudante(idEstudante, tipo, matricula, codCurso, idOrientador)
idEstudante referencia *Pessoa* - *idOrientador* referencia *Professor*

Professor(idProfessor, idDept)
idProfessor referencia *Pessoa* - *idDept* referencia *Departamento*

Ensina (idProfessor, idCurso)
idProfessor referencia *Professor* - *idCurso* referencia *Curso*

Relação "Departamento"

idDepartamento	nome	centro
1	Computacao	Ciencias
2	Engenharia Eletrica	Tecnologia
3	Geografia	Ciencias

Relação "Estudante"

idEstudante	tipo	matricula	codCurso	idOrientador
1	Graduacao	12345	10	7
2	PosGraduacao	58745	12	18
3	Graduacao	36587	14	9

Figura 6 - Exemplo

Percebe-se que certos valores se repetem em algumas relações, centro na relação Departamento e tipo na Estudante, por exemplo. No caso de departamento poder-se-ia criar uma nova classe Centro que se relacionaria com Departamento. No caso de Estudante, classes poderiam ser criadas como especialização para identificar o tipo do estudante.

Depois de identificar os atributos que podem virar classes, chamadas de classes candidatas, eles devem ser classificados em duas categorias: os atributos que não representam especialização, o caso de centro, e os atributos que podem gerar especialização, o caso de tipo em Estudante. No caso dos atributos que podem virar especialização, só são classes candidatas aquelas nas quais as subclasses são disjuntas, como é o caso de estudante, e quando uma *flag* determina se a tupla pertence ou não à subclasse.

Esse processo de determinar se uma classe adicional será ou não criada conta com a interação com o usuário.

A identificação das classes candidatas se dá a partir da análise do conteúdo dos atributos. Detalhes adicionais de como essa análise é feita podem ser encontrados em [Lopes,2007].

Chaves primárias, atributos com a restrição UNIQUE e chaves estrangeiras não são analisados por motivos óbvios.

4.5. Geração da Ontologia

É nessa fase que, finalmente, a ontologia é gerada, com base em toda a informação obtida nas etapas que a antecederam. Esse processo de geração da ontologia pode ser dividido em 3 etapas: a conversão de relações, conversão de atributos e conversão de casos especiais.

As regras de criação da ontologia são baseadas em [Lopes,2007] e alterações foram feitas para que fosse possível dar suporte a esquemas objeto-relacionais.

4.5.1 Conversão de Relações

Relações de entidade forte, fraca e relações irregulares são transformadas em classes na ontologia. Relações de entidade especializada são transformadas em subclasses. Relações de associação são mapeadas como *Object Property*. Relações de atributo multivalorado e de entidade fragmentada não são mapeadas como classes. Atributos são adicionados às classes referentes para armazenar a informação.

4.5.2 Conversão de Atributos

Atributos que não são chaves estrangeiras ou REF são mapeadas como *DataTypeProperty* em OWL. Já as chaves estrangeiras e referências para objetos são mapeados como *ObjectProperty.* Chaves primárias e atributo com restrições NOT NULL e UNIQUE viram *functional property* em OWL.

4.5.3 Conversão dos Casos Especiais

A conversão de casos especiais leva em consideração a intervenção do usuário. Caso o usuário opte por não criar novas classes, os atributos são mapeados como descrito em 4.5.2.

No caso de serem transformadas em classes, as regras são aplicadas de acordo com a categoria dentre as determinadas na seção anterior. Classes candidatas de atributos que não representam especialização resultam em classes e em uma propriedade de tipo de dado, com domínio na classe criada. Uma *ObjectProperty* também é usada para relacionar a classe criada com a classe origem do atributo.

No caso de uma classe candidata em que as subclasses são disjuntas, os atributos dão origem a classes disjuntas, uma para cada instância. Utiliza-se *owl:disjointWith* para isso.

No caso de uma flag determinar se a tupla pertence ou não à subclasse, uma subclasse é criada e relacionada, através de uma *ObjectProperty ,*à classe de origem.

4.5.4 Conversão de conceitos objeto relacionais

Nesta sessão, apresentaremos como se dá o mapeamento dos conceitos objeto-relacionais dos esquemas representados no Oracle para ontologias OWL. Para isso, utilizamos o esquema objeto-relacional do Quadro 3 como exemplo. O esquema consiste em um *Type* Pessoa, que possui como atributos um identificador, CPF, data de nascimento, um VARRAY com três telefones que ele possa ter e um endereço, que possui como atributos o logradouro, número, bairro, cidade, estado e CEP; um *Type* Funcionário que herda de Pessoa e tem como atributos seu salário, função e uma NESTED TABLE com seus pedidos atendidos.O *Type* Pedido tem, por sua vez, a data e o código do pedido assim como uma referência para o Cliente, que também herda de pessoa, que fez o pedido.

```
CREATE OR REPLACE TYPE TP_PESSOA AS OBJECT(
  ID            NUMBER,
  CPF           VARCHAR2(11),
  NOME          VARCHAR2(50),
  DATANASCIMENTO DATE,
  TELEFONES     ARRAY_TELEFONE,
  ENDERECO      TP_ENDERECO
)NOT FINAL;
/
CREATE OR REPLACE TYPE TP_CLIENTE UNDER TP_PESSOA(
);

CREATE OR REPLACE TYPE ARRAY_TELEFONE AS VARRAY(3) OF TP_TELEFONE;

CREATE OR REPLACE TYPE TP_ENDERECO AS OBJECT(
  LOGRADOURO     VARCHAR2(50),
  NUMERO         NUMBER(5),
  BAIRRO         VARCHAR2(20),
  CIDADE         VARCHAR2(20),
  ESTADO         CHAR(2),
  CEP            VARCHAR2(9)
);
/

CREATE OR REPLACE TYPE TP_TELEFONE AS OBJECT(
  DDD    NUMBER(2),
  NUMERO  NUMBER(8),
);

CREATE OR REPLACE TYPE TP_PEDIDO AS OBJECT(
  DATA              DATE,
  CODIGO            NUMBER,
  REF_CLIENTE       REF TP_CLIENTE
);
/
CREATE OR REPLACE TYPE NESTED_REF_PEDIDO AS TABLE OF REF TP_PEDIDO;
/
CREATE OR REPLACE TYPE TP_FUNCIONARIO UNDER TP_PESSOA (
  SALARIO          DECIMAL(6,2),
  FUNCAO           VARCHAR2(15),
  PEDIDOSATENDIDOS  NESTED_REF_PEDIDO
```

Quadro 3 – Esquema exemplo

O tipo endereço resulta na *Class* de OWL descrita no Quadro 4. Nota-se que cada atributo do tipo endereço foi mapeado em um *DatatypeProperty* em OWL onde *rdfId* equivale ao nome do atributo, *rdfs:domain* o domínio do atributo, que, no caso, é a classe endereço e *rdfs:range* corresponde a *datatypes* de OWL correspondente: string no caso de endereço_cidade.

```
<owl:Class rdf:ID="endereco"/>
<owl:DatatypeProperty rdf:ID=" endereco _logradouro">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" endereco _numero">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" endereco _bairro">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" endereco _cidade">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" endereco _estado">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#char"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" endereco _cep">
        <rdfs:domain rdf:resource="# endereco "/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

Quadro 4 – Endereço

Descrita a *class* endereço, podemos descrever como ficaria a classe Pessoa, correspondente ao tipo pessoa, no Quadro 5.

```
<owl:Class rdf:ID="pessoa"/>

<owl:DatatypeProperty rdf:ID=" pessoa _ID">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" pessoa _CPF">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl:DatatypeProperty>
owl:DatatypeProperty rdf:ID=" pessoa _nome">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" pessoa _datadenascimento">
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID=" pessoa _endereco">
        <rdfs:range rdf:resource="#endereco"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl: ObjectProperty >
<owl:ObjectProperty rdf:ID=" pessoa _array_telefone">
```

```
        <rdfs:range rdf:resource="#array_telefone"/>
        <rdfs:domain rdf:resource="# pessoa"/>
</owl: ObjectProperty >
```

Quadro 5 - Pessoa

Podemos notar que a referência a endereço que existia no esquema objeto-relacional foi mapeado com uma *ObjectProperty* em OWL assim como o array_telefone.

No Quadro 6, são declaradas as *classes* que herdam de pessoa: funcionário e cliente. Cliente não possui nenhum atributo adicional. Porém, como a classe Funcionário possui outros atributos além dos de Pessoa, foi necessário que eles fossem adicionados à definição da classe.

```
<owl:Class rdf:ID="cliente">
        <rdfs:subClassOf>
        <owl:Class rdf:ID="pessoa"/>
        </rdfs:subClassOf>
</owl:class>

<owl:Class rdf:ID="funcionario">
        <rdfs:subClassOf>
        <owl:Class rdf:ID="pessoa"/>
        </rdfs:subClassOf>
</owl:class>

<owl:DatatypeProperty rdf:ID=" funcionario _salario">
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
        <rdfs:domain rdf:resource="# funcionario "/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=" funcionario _funcao">
              <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain rdf:resource="# funcionario "/>
</owl:DatatypeProperty>
```

Quadro 6 - Herança

.

O Quadro 7 mostra o mapeamento do tipo Telefone e do VARRAY de telefones com cardinalidade três em OWL.

Como OWL não possui, em sua definição, algum modo de se mapear um VARRAY ou uma NESTED TABLE, foi usado um artifício para se fazer esse mapeamento.

No caso de um VARRAY com cardinalidade x, uma *class* é criada para o VARRAY, que é associada à classe referente ao tipo que a tinha como atributo,

e x *ObjectPropeties* são definidas para associar cada elemento do VARRAY. O Quadro 7 mostra como esse mapeamento é feito para o VARRAY de telefones com 3 ocorrências que é atributo do tipo Pessoa e a *class* Telefone.

```
<owl:Class rdf:ID="telefone"/>

<owl:DatatypeProperty rdf:ID=" telefone _ddd">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
         <rdfs:domain rdf:resource="# telefone "/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID=" telefone _numero">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
         <rdfs:domain rdf:resource="# telefone "/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="array_telefone"/>

<owl: ObjectProperty rdf:ID=" array_telefone1">
        <rdfs:range rdf:resource="#telefone"/>
         <rdfs:domain rdf:resource="#array_telefone "/>
</owl: ObjectProperty >
<owl: ObjectProperty rdf:ID=" array_telefone2">
        <rdfs:range rdf:resource="#telefone"/>
         <rdfs:domain rdf:resource="#array_telefone "/>
</owl: ObjectProperty >
<owl: ObjectProperty rdf:ID=" array_telefone3">
        <rdfs:range rdf:resource="#telefone"/>
         <rdfs:domain rdf:resource="#array_telefone "/>
</owl: ObjectProperty >
```

Quadro 7 - Varray

O Quadro 8 retrata o mapeamento do tipo Pedido e da NESTED TABLE nested_ref_pedido

Pode-se observar que a referência de um Pedido a um Cliente foi mapeado em um *ObjectProperty* com domínio em Cliente e alcanço em Pedido. Isso ocorre por que esse é um relacionamento de 1:N entre Cliente e Pedido.

De maneira semelhante, uma NESTED TABLE representa um relacionamento 1:N entre o tipo que a contém e o tipo representado por ela, Funcionário e Pedido nesse caso. Logo, o mapeamento de uma NESTED TABLE em OWL resulta em *ObjectProperty* referenciando o domínio Cliente a Pedido.

```
<owl:Class rdf:ID="pedido"/>

<owl:DatatypeProperty rdf:ID=" pedido_data">
          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
           <rdfs:domain rdf:resource="# pedido "/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID=" pedido _codigo">
          <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
           <rdfs:domain rdf:resource="# pedido "/>
</owl:DatatypeProperty>

<owl: ObjectProperty rdf:ID=" cliente_pedido">
          <rdfs:range rdf:resource="#pedido"/>
           <rdfs:domain rdf:resource="#cliente "/>
</owl: ObjectProperty >

<owl: ObjectProperty rdf:ID=" funcionario_pedido">
          <rdfs:range rdf:resource="#pedido"/>
           <rdfs:domain rdf:resource="#funcionario "/>
</owl: ObjectProperty >
```

Quadro 8 – Nested Table

5. Arquitetura

A Figura 7 mostra a arquitetura da ferramenta proposta. Ela foi dividida em três camadas que englobam todos os passos da geração da ontologia descritos na sessão anterior. Essas três camadas são a Camada de Interação com o Banco de Dados, a Camada de Gerenciamento de Ontologia e a Camada de Interação com o Usuário. Essas três camadas serão detalhadas a seguir.

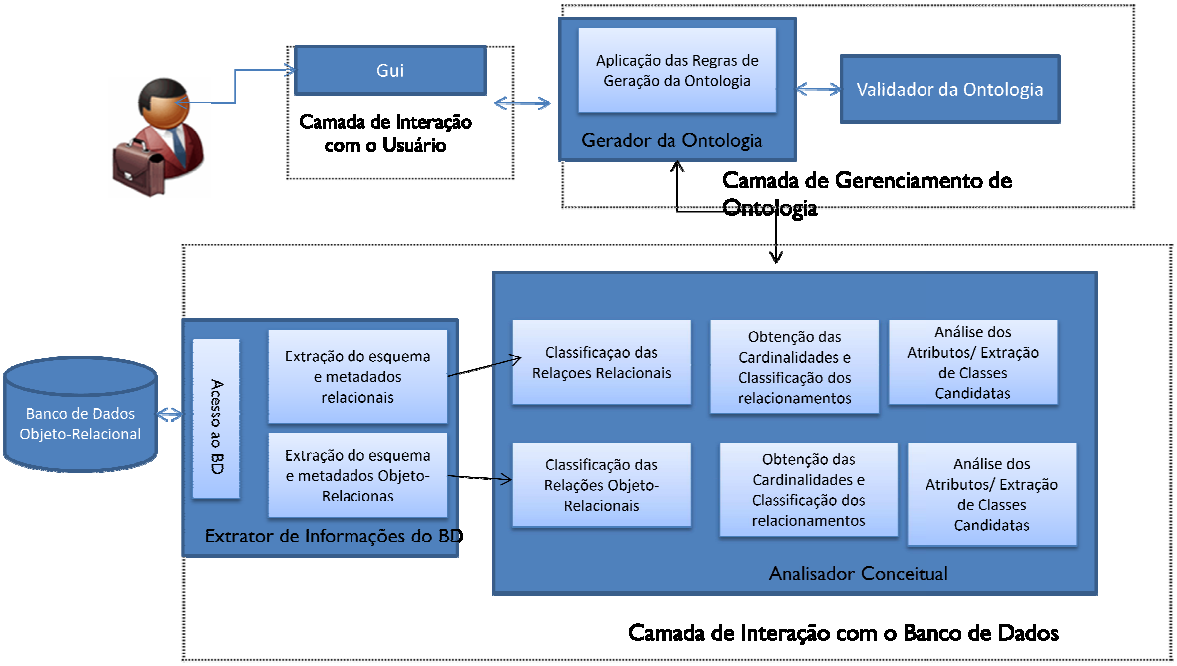

Figura 7 – Arquitetura da Ferramenta

6.1 Camada de Interação com o Banco de Dados

A camada de interação com o banco de dados é a camada que faz a interface entre a ferramenta e o banco de dados objeto-relacional a ser mapeado em uma ontologia. Ela é dividida em dois módulos: o Extrator de informação do DB e um Analisador Conceitual.

O módulo Extrator de Informação do banco de dados é a parte responsável por fazer o acesso direto ao banco de dados. Todo o acesso direto ao banco, seja para realizar consultas ou para extração de metadados, é feito nesse módulo. Esse módulo é dividido em duas partes: uma parte para

extração de informação de um BD relacional e outra para a extração de informação de um BD objeto-relacional. Essa criação foi necessária devido à diferença que existe entre eles tanto nos metadados quanto nas estruturas utilizadas.

No Analisador Conceitual, são realizadas as etapas de classificação das relações, obtenção das cardinalidades e classificação dos relacionamentos e a análise dos atributos e extração das classes candidatas.

Também foi necessária a dissociação entre a parte que faz o tratamento dos conceitos de um BD relacional e conceitos objeto-relacional.

Ao sair dessa camada, todos os dados relevantes sobre o banco de dados já foram extraídos e identificados de forma que a próxima camada, a de gerenciamento de uma ontologia, vai utilizar essa informação para gerar, de fato, a ontologia.

6.2 Camada de Gerenciamento de uma Ontologia

Assim como a camada anterior, a camada de gerenciamento de uma ontologia também é dividida em módulos. Esses módulos são o módulo gerador da ontologia e o validador da ontologia.

O módulo gerador da ontologia corresponde à etapa de aplicação das regras de geração da ontologia explicada na sessão anterior. Essa tarefa é realizada com base na informação extraída e tratada na camada anterior. Para essa tarefa, foram usados o framework JENA e a API Protege-OWL.

A informação sobre o mapeamento dos conceitos objeto-relacional foi adicionada para que o sistema fosse capaz de tratar de maneira correta tanto conceitos relacionais quanto objeto-relacionais. Desse modo, a ontologia gerada pela ferramenta é a mais correta possível.

O módulo validador da ontologia é responsável por verificar se a ontologia gerada está correta. Para isso,o *reasoner Pellet* é utilizado.

6.3 Camada de Interação com o Usuário

É a camada de interface do sistema com o usuário. É por meio dela que se dá toda a interação do usuário com a ferramenta. Através dela,

configurações de conexão com banco de dados são configuradas e também as configurações de geração da ontologia, que definem o grau de automação do processo, análise de instâncias e outras informações a respeito da ontologia a ser gerada.

6.4 Implementação

A ferramenta foi totalmente implementada usando a linguagem Java[1] na IDE Eclipse[2].

O acesso aos dados no SGBD Oracle 10g foi realizada utilizando o Oracle JDBC [3]e o driver ojdbc14,jar.

A manipulação e construção das ontologias foram feitas através da api Protege-OWL API[4] e do framework JENA[5]. A verificação da consistência da ontologia gerada para a sua validação foi feita utilizando o *reasoner* Pellet[6].

A adaptação da ferramenta RelOnto necessitou de várias alterações, desde as classes básicas de Java, que não davam suporte aos conceitos objeto-relacionais que foram adicionados; extração de metadados, que também não dava suporte à extração dos conceitos objeto-relacionais; algoritmos de classificação, que só classificavam conceitos relacionais e geração da ontologia que não dava suporte ao mapeamento dos conceitos que foram adicionados.

[1] http://java.sun.com/
[2] www.eclipse.org
[3] http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc_10201.html
[4] http://protege.stanford.edu/plugins/owl/api/
[5] http://jena.sourceforge.net/
[6] http://clarkparsia.com/pellet/

6. Conclusão

Nesse trabalho, foi proposta e desenvolvida uma ferramenta para mapeamento esquemas de bancos de dados objeto-relacionais em ontologias OWL. Tal ferramenta vem suprir uma lacuna deixada por outros trabalhos que se restringiam a outras fontes de dados como os bancos de dados relacionais.

7. Referências Bibliográficas

Barrasa, J., Corcho, O., and Gómez-Pérez, A.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: 2nd Workshop on Semantic Web and Databases (SWDB'04), pages 1069-1070, Toronto, Canada (2004)

Cerbah, F.: Learning Highly Structured Semantic Repositories from Relational Databases - The RDBtoOnto Tool. In: 5th European Semantic Web Conference (ESWC'08), pages 777-781, Tenerife, Spain (2008)

Cullot, N., Ghawi, R., and Yétongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping". In: 15th Italian Symposium on Advanced Database Systems (SEBD'07), pages 491-494, Torre Canne di Fasano (BR), Italy (2007)

de Laborda, C. P. and Conrad, S. Relational.OWL A Data and Schema Representation Format Based on OWL. In: 2nd Asia-Pacific Conference on Conceptual Modelling (APCCM'05), Volume 43 of CRPIT, pages 89-96, Newcastle, Australia (2005)

Ghawi, R., and Cullot, N.: Database-to-Ontology Mapping Generation for Semantic Interoperability. In: 3rd International Workshop on Database Interoperability (InterDB'07), held in conjunction with VLDB 2007, Vienna, Austria (2007)

Guarino, N.: Formal Ontology and Information Systems. In: 1st International Conference on Formal Ontologies in Information Systems, pages 3-15, Trento, Italy (1998)

Hu, W., Qu, Y.: Discovering Simple Mappings Between Relational Database Schemas and Ontologies. In: 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC'07), 4825:225-238, Busan, South Korea, (2007)

Nyulas, C., O'Connor, M., and Tu, S.: DataMaster - a Plug-in for Importing Schemas and Data from Relational Databases into Protégé. In: 10th International Protégé Conference, Budapest, Hungary (2007)

Prud'Hommeaux, E. and Seaborne, A,. SPARQL Query Language for RDF. World Wide Web Consortium, Working Draft WD-rdf-sparql-query-2006, (2006)

McGuiness D.L. and Harmelen F.V., eds., "OWL WebOntology Language Overview", W3C Recommendation, 2004. Disponível em: *http://www.w3.org/TR/owl-features/*.

Lopes, F. L. R, . de Oliveira, W. G. D., and Lóscio, B. F. Geração de Ontologias em *OWL* a partir de Bancos de Dados Relacionais. *Departamento de Computação - Universidade Federal do Ceará 2007*

Konstantinou N., Spanos, N., Chalas M., Solidakis E., and Mitrou N.,"VisAVis: An approach to an intermediate layer between ontologies and relational database contents," in *Proc. Int. Workshop on Web Information Systems Modeling*, Luxembourg, Grand Duchy of Luxembourg, 2006.

Berners-Lee, T., Hendler, J., and Lassila, O.: The Semantic Web. Scientific American (2001)

Gruber, T. R.: A translation approach to portable ontology specifications. Knowledge *& Application(IDEAS)*, Portugal, 2008.

Alhajj R. “Extracting the extended entity relationship model from a legacy relational database”, Information systems, volume 28, number 6, 2003

Chiang R. H. L, “A knowledge-based system for performing reverse engineering of relational databases”, Decision Support Systems, v.13 n.3-4, 1995, pp.295-312.

Lopes F.L.R., “*RelOnto:* Uma ferramenta para geração de ontologias a partir de bancos de dados relacionais”, Trabalho de Conclusão de Curso, Graduação em Ciência da Computação, Universidade Estadual do Ceará.

Sonia K.. and Khan S., “R2O transformation system: relation to ontology transformation for scalable data integration”, *in Proceedings of the 12th International Symposium on Database Engineering*

Koide, S, Kawamura,M. “*SWCLOS: A Semantic Web Processor on Common Lisp Object System”.* Galaxy Express Corporation, 2004

Bergman, M. K. . The deep web: Surfacing hidden value. Technical report, BrightPlanet LLC, Dezembro. 2000.

Junior, H. J. V, Cardoso, L. F., Lopes, M. A. M. “Extensão de Métodos de Acesso em SGBDs Relacionais-Objeto:Uma Avaliação da Interface para Índices Virtuais no Informix” Agosto, 2000

Almeida, M. B., BAX, M. P.. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. 2003.