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

Uma Avaliação de Abordagens Alternativas para Armazenar RDF em Banco de Dados Relacional

por

NELSON HENRIQUE ZANETE

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

Prof. José Valdeni de Lima
Orientador

Porto Alegre, dezembro de 2001

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Zanete, Nelson Henrique
Uma Avaliação de Abordagens Alternativas para Armazenar RDF em Banco de Dados Relacional / por Nelson Henrique Zanete. – Porto Alegre: PPGC da UFRGS, 2001.
92 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2001. Orientador: Lima, José Valdeni de.

1. Banco de dados. 2. Metadata. 3. RDF. I. Lima, José Valdeni de. II. Título.

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

“Das vossas mãos voltadas para o mundo, as graças chovem sobre nós”.

À Na. Sra. das Graças.

Agradecimentos

Sou parte de tudo o que conheci, fui influenciado por mais pessoas do que posso contar, mas quero agradecer em especial a algumas delas por tornarem este projeto possível.

Ao orientador e amigo Prof. Dr. José Valdeni de Lima, pela confiança, compreensão, dedicação e exemplo profissional.

A todos os professores do PPGC pelos ensinamentos e palavras de estímulo.

Ao coordenador Prof. Dr. Carlos A. Heuser pela dedicação e atenção durante o curso.

A Universidade Federal do Rio Grande do Sul e ao Programa de Pós-Graduação em Computação pela oportunidade concedida.

A Faculdade Paranaense pela confiança e apoio ilimitado.

Aos funcionários da secretaria e biblioteca que sempre nos atenderam com gentileza, informando e esclarecendo nossas dúvidas.

Aos amigos da Área de Informática do Instituto Agronômico do Paraná por entenderem minhas prioridades e colaborar nos momentos certos.

Aos amigos de mestrado com quem dividi as alegrias e dificuldades, em especial ao Tazima e ao Orlando.

A minha noiva Daniela pela compreensão, amor e incentivo em todos os momentos.

A minha mãe e minha irmã, por estarem sempre do meu lado, conversando e ouvindo meus anseios.

Sumário

Lista de Abreviaturas 7
Lista de Figuras 8
Lista de Tabelas 10
Resumo 12
Abstract 13
1 Introdução 14
2 Comunidades de descrição de recursos 16
2.1 Dublin Core Metadata Initiative 16
2.2 Australian Government Locator Service 17
2.3 EdNA – The Educational Network of Australia 17
2.4 Descrevendo e recuperando fotos usando HTTP e RDF 18
3 Resource Description Framework 19
3.1 Modelo RDF 19
3.1.1 Containers 21
3.1.2 Sintaxe RDF 23
3.1.3 Declarações sobre declarações 24
3.1.4 Modelo formal para RDF 25
3.2 Esquema RDF 27
3.2.1 O núcleo classe 28
3.2.2 O núcleo propriedades 29
3.2.3 O núcleo restrições 30
3.2.4 Documentação 30
4 Estratégias de elaboração e publicação de metadata RDF 31
4.1 Elaboração de esquemas 31
4.2 Geração de metadata 34
4.3 Inclusão de modelos RDF em imagens 35
4.4 Publicação das imagens 36
4.5 Parser RDF 36
5 Análise dos resultados do SiRPAC 38
5.1 O modelo básico RDF 40
5.2 Resultados sobre containers 41
5.3 Declarações sobre membros de um container 43
5.4 Containers definidos por um padrão de URI 44

5.5 Resultados do parser de declarações sobre declarações 45
5.6 Valores compartilhados 46
5.7 Relações não binárias 48
6 Distribuição e amostragem dos dados para avaliação 49
7 Abordagens existentes de armazenamento de RDF em banco de dados relacional 51
7.1 Abordagem de Brian McBride 51
7.2 Abordagem de Jonas Liljegren 54
7.3 Abordagem de Sergey Melnik 58
7.4 Abordagem de Eric Miller 61
8 Avaliação das abordagens 62
8.1 Proliferação de tabelas e informações mantidas 62
8.1.1 Metodologia utilizada 62
8.1.2 Desenvolvimento 62
8.2 Tempo de carga das declarações nos modelos relacionais 63
8.2.1 Metodologia utilizada 63
8.2.2 Desenvolvimento 65
8.3 Espaço consumido pelas declarações nos modelos 66
8.3.1 Metodologia utilizada 66
8.3.2 Desenvolvimento 66
8.4 Custo de consultas SQL 66
8.4.1 Metodologia utilizada 66
8.4.2 Desenvolvimento 67
9 Abordagem proposta 70
9.1 Apresentação do protótipo 73
10 Conclusões e Perspectivas 76
Anexo 1 Instruções SQL 78
Anexo 2 Algoritmo de geração do modelo RDF "aluno" e "professor" 86
Bibliografia 90

Lista de Abreviaturas

AGLS	Australian Government Locator Service
CRC	Cyclic Redundancy Code
DC	Dublin Core
DCQ	Dublin Core Qualifiers
DMP	Dedicação Mensal de Pesquisador
EdNA	Educational Network of Australia
E/S	Entrada e Saída
EOR	Extensible Open RDF
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ISO 8601	International Standard for the representation of dates and times
JPEG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extensions
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SiRPAC	Simple RDF Parser & Compiler
SQL	Structure Query Language
TPD	Tecnologia em Processamento de Dados
URI	Uniform Resource Identifiers
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Lista de Figuras

FIGURA 3.1 - Representação de modelo de dados RDF 20
FIGURA 3.2 - Valor da propriedade como recurso 20
FIGURA 3.3 - *Container* do tipo *Bag* 22
FIGURA 3.4 - *Container* do tipo *Sequence* 22
FIGURA 3.5 - *Container* do tipo *Alternative* 23
FIGURA 3.6 - Sintaxe RDF/XML para declaração *reified* 25
FIGURA 3.7 - Modelo grafo rotulado de declaração simples 25
FIGURA 3.8 - Simples declaração gráfica 26
FIGURA 3.9 - Hierarquia de Classes para o RDFS 28
FIGURA 3.10 - Exemplo de propriedades com restrições *domain* e *range* 30
FIGURA 4.1 - Modelo gráfico RDF do "EsquemaPessoas" 32
FIGURA 4.2 - Interface do Protégé-2000 com o "EsquemaPessoas" 32
FIGURA 4.3 - "EsquemaPessoas" gerado pelo Protégé-2000 33
FIGURA 4.4 - Modelo "professor" na sintaxe RDF/XML 34
FIGURA 4.5 - Modelo "aluno" na sintaxe RDF/XML 35
FIGURA 4.6 - Interface do RDFPic com a imagem de um professor 35
FIGURA 4.7- Interface do JigAdmin 36
FIGURA 5.1 - *Description* referindo-se a um identificador de recurso novo 38
FIGURA 5.2 - Recurso "FulanoDeTal.htm" 39
FIGURA 5.3 - Arquivo "Metadata.rdf" 39
FIGURA 5.4 - Entidade estruturada como objeto anônimo 40
FIGURA 5.5 - Entidade estruturada como objeto identificado 40
FIGURA 5.6 - Identificador gerado pelo SiRPAC para entidade anônima 41
FIGURA 5.7 - Identificador gerado pelo SiRPAC para entidade identificada 41
FIGURA 5.8 - *Container* "Estudantes" 42
FIGURA 5.9 - Resultado do SiRPAC para *container* do tipo *Bag* 42
FIGURA 5.10 - Declarações sobre o *container* "#Pages" 43
FIGURA 5.11 - Resultado do SiRPAC para declarações do *container* "#Pages" 43
FIGURA 5.12 - Declarações sobre os membros de um *container* 44
FIGURA 5.13 - Resultados do SiRPAC para membros de um *container* 44
FIGURA 5.14 - Emprego do atributo *aboutEachPrefix* 45
FIGURA 5.15 - Resultado do SiRPAC para o atributo *aboutEachPrefix* 45
FIGURA 5.16 - Modelo de declaração RDF 46

FIGURA 5.17 - Resultado do SiRPAC para modelo de declaração 46
FIGURA 5.18 - Um recurso como objeto de duas declarações..... 47
FIGURA 5.19 - Resultado com propriedades diferentes para o mesmo recurso. 47
FIGURA 5.20 - Modelo RDF para relações não binárias 48
FIGURA 5.21 - Resultado do SiRPAC para relações não binárias..... 48
FIGURA 7.1 - Modelo relacional de Brian McBride..... 51
FIGURA 7.2 - Conjunto de declarações *asserted*. 52
FIGURA 7.3 - Conjunto de declarações *reified* 53
FIGURA 7.4 - Modelo relacional de Jonas Liljegren 55
FIGURA 7.5 - Modelo RDF com o atributo *aboutEachPrefix* 56
FIGURA 7.6 - Resultado do *parser* para o modelo *aboutEachPrefix*..... 57
FIGURA 7.7 - Esquema relacional proposto por Sergey Melnik 59
FIGURA 7.8 - Conjunto de declarações RDF..... 60
FIGURA 7.9 – Proposta de Eric Miller..... 61
FIGURA 8.1 - Resultado do *parser* para recurso do modelo “aluno” 68
FIGURA 8.2 - Resultado do *parser* para recurso do modelo “professor” 68
FIGURA 9.1 – Custo após processamento do algoritmo de equivalência 72
FIGURA 9.2 - Esquema de funcionamento do protótipo..... 73
FIGURA 9.3 - Interface do protótipo de equivalência..... 74

Lista de Tabelas

TABELA 2.1 – Conjunto de Elementos *metadata* do AGLS 17
TABELA 3.1 – Sentenças sintaticamente diferentes com o mesmo significado 20
TABELA 3.2 – Partes de uma declaração RDF 20
TABELA 3.3 – Dois tipos de sintaxe RDF 23
TABELA 4.1 – Sintaxe de uso do SiRPAC 37
TABELA 6.1 – Distribuição das declarações nos modelos..... 49
TABELA 6.2 – Distribuição das declarações nos esquemas..... 49
TABELA 6.3 – Amostragem detalhada do conjunto de dados 50
TABELA 7.1 – Dados dispostos na tabela "rdfresource" 53
TABELA 7.2 – Dados dispostos na tabela "rdfnamespace" 53
TABELA 7.3 – Dados dispostos na tabela "rdfstatement" 54
TABELA 7.4 – Dados dispostos na tabela "rdfliteral" 54
TABELA 7.5 – Dados dispostos na tabela "rdfmodel"..... 54
TABELA 7.6 – Dados dispostos na tabela "rdfkeys" 54
TABELA 7.7 – Dados dispostos na tabela "prefix"..... 56
TABELA 7.8 – Dados dispostos na tabela "resources" 58
TABELA 7.9 – Dados dispostos na tabela "statements" 58
TABELA 7.10 – Dados dispostos na tabela "sequence"..... 58
TABELA 7.11 – Dados dispostos na tabela "resources" 60
TABELA 7.12 – Dados dispostos na tabela "namespaces" 60
TABELA 7.13 – Dados dispostos na tabela "literal" 61
TABELA 7.14 – Dados dispostos na tabela "triples" 61
TABELA 7.15 – Dados dispostos na tabela "models"..... 61
TABELA 7.16 – Dados dispostos na tabela "triple"..... 61
TABELA 8.1 – Informações abstraídas versus tabelas 63
TABELA 8.2 – Padrões de atributos usados nas abordagens 63
TABELA 8.3 – Índices utilizados na abordagem de Melnik 64
TABELA 8.4 – Índices utilizados na abordagem de Eric 64
TABELA 8.5 - Índices utilizados na abordagem de Brian..... 64
TABELA 8.6 – Índices utilizados na abordagem de Jonas 64
TABELA 8.7 – Tempo para carga de declarações em cada abordagem 65
TABELA 8.8 – Quantidade em bytes consumido em cada abordagem 66
TABELA 8.9 – Tipos de condições selecionadas para medição de custo..... 67

TABELA 8.10 – Índice de Custo computado para cada consulta 67
TABELA 9.1 – Tabela “triples” do modelo proposto 70
TABELA 9.2 – Tabela “recurso” do modelo proposto 70
TABELA 9.3 – Comparativo de instruções SQL 71
TABELA 9.4 – Índices de custo após processamento do algoritmo de equivalência 72
TABELA 9.5 – Tabela “Alunos” gerada pelo protótipo 74

Resumo

O *Resource Description Framework* (RDF) é uma infra-estrutura, que possibilita a codificação, troca e reuso de *metadata* estruturado. *Metadata* é dados sobre dados. O termo refere a qualquer dado usado para ajudar a identificação, descrição e localização de recursos eletrônicos na rede. O RDF permite adicionar declarações, sinônimos e palavras que não estão presentes nos recursos, mas que são pertinentes a eles. Uma declaração RDF pode ser desenhada usando diagramas de arcos e nodos, onde os nodos representam os recursos e os arcos representam as propriedades nomeadas. O modelo básico consiste em recursos, propriedades e objetos. Todas as coisas sendo descritas pelas declarações RDF são chamadas de recursos. Um recurso pode ser uma página da *Web* inteira ou um elemento específico HTML ou XML dentro de um documento fonte. Uma propriedade é um aspecto específico, característica, atributo, ou relação usada para descrever um recurso. O objeto pode ser um outro recurso ou um literal. Estas três partes, juntas, formam uma declaração RDF. O resultado do *parser* para recursos com *metadata* RDF, é um conjunto de declarações referentes aquele recurso. A declaração destas propriedades e a semântica correspondente delas estão definidas no contexto do RDF como um *RDF schema*. Um esquema não só define as propriedades do recurso (por exemplo, título, autor, assunto, tamanho, cor, etc.), mas também pode definir os tipos de recursos sendo descritos (livros, páginas *Web*, pessoas, companhias, etc.). O *RDF schema*, provê um sistema básico de tipos necessários para descrever tais elementos e definir as classes de recursos. Assim, os recursos constituindo este sistema de tipos se tornam parte do modelo RDF de qualquer descrição que os usa. A geração de modelos RDF pode ser conseguida através de algoritmos implementados com linguagens de programação tradicionais e podem ser embutidos em páginas HTML, documentos XML e até mesmo em imagens. Com relação a modelos em imagens, servidores *Web* específicos são usados para simular dois recursos sobre o mesmo URI, servindo ora a imagem ora a descrição RDF. Uma alternativa para armazenar e manipular grande quantidade de declarações RDF é usar a tecnologia de banco de dados relacional. Abordagens para armazenar declarações RDF em banco de dados relacional foram propostas, mas todas elas mantêm modelos diversos de diferentes fontes. Critérios de avaliação como tempo de carga, proliferação de tabelas, espaço, dados mantidos e custo de instruções SQL foram definidos. Duas abordagens apresentaram resultados satisfatórios. Com uma nova abordagem proposta por este trabalho se obteve melhores resultados principalmente no aspecto de consultas. A nova proposta provê mecanismos para que o usuário faça seu próprio modelo relacional e crie suas consultas. O conhecimento necessário pelo usuário se limita em parte aos modelos mantidos e ao esquema RDF.

Palavras-chaves: Banco de Dados, *Metadata* e RDF.

TITLE: "An Evaluation of Alternative Approaches for Storing RDF in Relational Database"

Abstract

The Resource Description Framework (RDF) is an infrastructure that enables the encoding, exchange and reuse of structured metadata. Metadata is data about data. The term refers to any data used to aid the identification, description and location of networked electronic resources. RDF allows adding declarations, synonyms and words that are not present in the resources, but that are pertinent to them. A declaration RDF can be drawn using nodes and arcs diagrams, where the nodes represent resources and the arcs represent named properties. The basic model consists of resources, properties and objects. All things being described by RDF expressions are called resources. A resource may be an entire Web page or a specific element HTML or XML within the source document. A property is a specific aspect, characteristic, attribute or relation used to describe a resource. The object can be another resource or it can be a literal. These three parts together form a RDF statement. The parser result for resources with RDF metadata is a set of statement referring to that resource. The declaration of these properties and their corresponding semantics are defined in the context of RDF as an RDF schema. A schema defines not only the properties of the resource (e.g., title, author, subject, size, color, etc.) but may also define the kinds of resources being described (books, Web pages, people, companies, etc.). The RDF schema provides a basic system of types needed to describe such elements and to define the classes of resources. Thus, the resources constituting this typing system become part of the RDF model of any description that uses them. The generation of models RDF can be gotten through algorithms implemented with traditional programming languages and they can be embedded in pages HTML, documents XML and even in images. With relationship to models in images, specific Web servers are used to simulate two resources on same URI, serving some times the image other times the description RDF. An alternative to store and manipulate large amounts of RDF statement is to use relational database technology. Approaches to store RDF statement in relational database were proposed, but all of them maintain several models from different sources. Evaluation criteria as load time, proliferation of tables, space, maintained data and cost of SQL instructions were defined. Two approaches presented satisfactory results. Better results were obtained with the new approach proposed by this work, mainly in the query aspect. The new proposal provides mechanisms for the user to do his own relational model and create his queries. The necessary knowledge by the user is limited partly to the maintained models and RDF schema.

Keywords: Database, Metadata and RDF.

1 Introdução

Considerando o fato de que os *Web Sites* mais importantes já utilizam a tecnologia de banco de dados, quase todos os comerciantes de banco de dados têm agora uma oferta para exportar dados em XML. Mas, quando a escala ou formato dos recursos existentes de um provedor de informação significa que eles possivelmente não possam ser codificados em XML, eles podem ser descritos por *metadata* codificada em XML. Esse é um método de baixo custo para fazer com que a informação relevante seja encontrada. **[LIG 99]**. O *Resource Description Framework* (RDF) é uma infra-estrutura, que possibilita a codificação, troca e reuso de *metadata* estruturado. **[MIL 98].** Atualmente mecanismos para lidar com RDF têm recebido considerável atenção e interesse do W3C - *World Wide Web Consortium*. Entre eles, linguagens de consultas baseadas em SQL - *Structure Query Language* e mecanismos de abstração de informações por inferência aos modelos têm sido propostos. Evidentemente estes trabalhos têm deixado de lado soluções triviais de banco de dados e de linguagens de consultas consagradas como o SQL. Abordagens relacionais capazes de manter *metadata* RDF em banco de dados têm sido propostas em **[MEL 2000]**, mas algumas questões ainda permanecem em aberto. Uma lista de fatores como tempo de carga, proliferação de tabelas, espaço consumido, dados mantidos e custo de instruções SQL têm sido questionados sobre os modelos relacionais criados para manter as declarações.

Visando promover o uso do RDF e responder questões desta natureza este trabalho apresenta estratégias de uso e publicação de recursos com *metadata* RDF, avalia e identifica dentre as abordagens existentes aquelas que se destacam por resultados positivos e ainda propõe um mecanismo para que o usuário crie e use seu próprio modelo relacional.

Baseando-se em princípios bem definidos pelas especificações oficiais do W3C uma nova abordagem é proposta por este trabalho. Ao contrário das abordagens existentes que mantêm modelos diversos e de diferentes fontes, a abordagem proposta preocupa-se em manter modelos relacionais específicos para cada modelo RDF. Soluções baseadas em abordagens existentes, em parte, também são utilizadas no sentido de garantir a manutenção de todos os modelos.

A estrutura do trabalho é descrita a seguir.

O segundo capítulo compreende os seguimentos de uso do RDF por comunidades de governo, redes educacionais de outros países, bibliotecas digitais e outras.

No terceiro capítulo, o RDF é apresentado em detalhes, tanto o modelo quanto o esquema. Comentários adicionais sobre as comunidades que influenciaram seu desenvolvimento e seguimentos de uso também estão presentes.

O capítulo 4 dedica-se às estratégias de uso e publicação de recursos com *metadata*. Estratégias de publicação de imagens com RDF são explicadas desde a inclusão dos modelos nos recursos até a publicação com servidores *Web* capacitados para esse propósito.

O quinto capítulo apresenta cada estrutura definida no *framework RDF* codificado na sintaxe RDF/XML seguido do correspondente resultado do *parser.* Particularidades da codificação e detalhes do conjunto de declarações de cada modelo são explicadas.

A distribuição e amostragem dos dados usados para avaliação estão detalhadas no capítulo 6.

No capítulo 7 é exibida além do modelo relacional de cada abordagem, uma amostra dos dados dispostos em cada tabela. As relações entre tabelas são comentadas.

O capítulo 8 aplica métodos comparativos e quantitativos a fim de avaliar os modelos relacionais. Todas as técnicas e metodologias utilizadas são explicadas. Resultados sobre cada critério de avaliação são apresentados e comentados.

O capítulo 9 dedica-se a uma nova abordagem. A solução, baseada em um protótipo possível de ser implementado sobre duas das abordagens existentes, garante que, no pior caso, os resultados alcançados sejam equivalentes às abordagens de origem.

Por fim o capítulo 10 apresenta as conclusões finais com sugestões adicionais.

2 Comunidades de descrição de recursos

Comunidades de descrição de recursos requerem habilidades para descrever certos tipos de recursos. Para recursos bibliográficos, por exemplo, atributos descritivos incluindo "autor", "título" e "assunto" são comuns.

Muitas comunidades estão adotando padrões de *metadata* como alternativas para a descrição de recursos e troca de informações. Alguns destes padrões já estão sendo aceitos por outras comunidades de peritos.

2.1 Dublin Core Metadata Initiative

O *Dublin Core* **[DUB 2000],** um simples modelo de descrição de conteúdo para recursos eletrônicos é um conjunto de elementos de *metadata* que pretende facilitar descoberta de recursos eletrônicos. Originalmente concebido para descrição de autoria de recursos da *Web*, chamou a atenção de comunidades de descrição de recurso formais como museus, bibliotecas, agências de governo e organizações comerciais.

A principal característica do conjunto de elementos do *Dublin Core* é o consenso e a experiência coletiva de uma variedade de especialistas interdisciplinares e internacionais da área de descrição de recursos. As características que o distingue como um candidato proeminente para descrição de recursos eletrônicos, entram em várias categorias:

Simplicidade: é pretendido que o *Dublin Core* seja utilizável por leigos como também por especialistas de descrição de recursos. A maioria dos elementos tem uma semântica comumente compreendida.

Semântica e interoperabilidade: Promovendo um conjunto de descritores de entendimento comum que ajudam a unificar outros conteúdos de dados padrões, incrementa a possibilidade de interoperabilidade através de disciplinas.

Extensibilidade: adicionalmente, o conjunto de elementos de *Dublin Core*, inclui flexibilidade suficiente e extensibilidade para codificar estrutura e semântica mais elaborada, inerente em padrões de descrição mais ricos.

Modularidade de *metadata* na rede: a diversidade de necessidades de *metadata* na rede requer uma infra-estrutura que apóia a coexistência de pacotes independentemente mantidos. O Consórcio W3C começou a implementar uma arquitetura para *metadata*. O RDF é projetado para suportar as muitas diferentes necessidades de *metadata* de vendedores e provedores de informação. Representantes dos objetivos de *Dublin Core* estão ativamente envolvidos no desenvolvimento desta arquitetura, trazendo a perspectiva da comunidade de biblioteca digital para afetar este componente importante da infra-estrutura da rede.

O *Dublin Core* conjunto de elementos de *metadata* é um conjunto de definições semânticas de 15 elementos descritivos. Muitos métodos diferentes podem ser usados para registrar ou transferir *metadata Dublin Core*. Métodos comuns incluem HTML, XML, RDF e bancos de dados relacionais.

O conjunto completo dos 15 elementos é: *Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage e Rights* e são usados para tentar descrever o significado dos elementos de um recurso.

Tipicamente, os elementos estão associados a um esquema de codificação, definindo um formato padrão. O elemento "*Date*" do conjunto acima, poderia estar

associado a um esquema de codificação como o ISO 8601 **[WOL 97]** que segue o formato de yyyy-mm-dd.

Verifica-se uma forte tendência ao uso de conjunto de elementos de *metadata* em comunidades que trocam informações. Muitas das comunidades tem-se baseado no conjunto de elementos *Dublin Core* para descrever recursos. O padrão *Dublin Core* tem colaborado e influenciado no desenvolvimento do RDF trazendo boas perspectivas para esta proposta.

2.2 Australian Government Locator Service (AGLS)

O *National Archives of Australia* é uma agência de manutenção do padrão de *metadata* AGLS, (*Australian Government Locator Service*). O padrão AGLS de *metadata* é um conjunto de 19 elementos descritivos que departamentos do governo e agências podem usar para melhorar a acessibilidade dos seus serviços e informações na Internet. **[NAT 2000].**

Os Governos Australianos endossaram o uso de *metadata* para facilitar descoberta de informações e serviços por todos os níveis: local, estadual e federal. Para ser efetivo, toda agência deveria usar os mesmos elementos. Elas são estimuladas a usar o padrão AGLS para interagir prontamente com outras agências.

AGLS é simples, contudo, poderoso o bastante para descrever coleções de recursos produzidos pelo governo. AGLS está baseado no padrão *Dublin Core* (DC) de *metadata* que é comumente usado em muitas indústrias e setores estatais.

O conjunto de *metadata* AGLS consiste em dezenove elementos. Os elementos estão listados abaixo em três categorias, relacionado ao propósito descritivo de cada um. (Tabela 2.1).

TABELA 2.1 – Conjunto de Elementos *metadata* do AGLS

Propósito	**Elementos**
Propriedade, Criadores da Propriedade do Recurso e Criadores do Recurso	Creator, Publisher, Contributor e Rights.
Conteúdo intelectual sobre o Recurso	Title, Subject, Description, Source, Language, Relation, Coverage, Function, Audience e Mandate.
Manifestação eletrônica ou Física do Recurso	Date, Type, Format, Identifier e Availability

2.3 EdNA - The Educational Network of Australia

EdNA - Rede de Educação da Austrália é uma rede da comunidade de educação australiana (sistema de educação governamental, não governamental, vocação educacional, treinamento, educação da comunidade de adultos e ensino superior).

É organizada com base no currículo australiano e suas ferramentas são grátis a pedagogos. É fundada pelo corpo responsável pela provisão de educação na Austrália.

Elementos de *metadata* são recomendados para uso na EdNA. Alguns setores têm identificado seus próprios conjuntos de elementos obrigatórios, em particular o setor das escolas.

A implementação de *metadata* é como um sistema simples, aplicável por toda a rede de educação, embora alguns componentes são orientados mais a um setor que

outro. (por exemplo, o elemento de *EDNA.Userlevel* é mais pertinente a escolas). Em uma fase posterior, provavelmente serão desenvolvidos elementos adicionais para tipos específicos de recursos. Assim sendo, informações de curso poderiam ter elementos adicionais que indicam o nível de qualificação, o modo de entrega e a duração do curso. Também há uma discussão relativa à descrição de *metadata* sobre pedagogia.

Estados, territórios, setores ou sistemas também podem desenvolver elementos de *metadata* para os seus propósitos. Estes poderiam estar no mesmo formato de elementos de *metadata Dublin Core /* EdNA.

O "*EdNA Reference Committee*" tem concordado que o padrão de *metadata* EdNA está baseado no padrão do conjunto de elementos *Dublin Core*.

O conjunto de *metadata* EdNA é composto por nove elementos, além dos já suportados 15 elementos *Dublin Core*. São eles: *EDNA.Entered, EDNA.Approver, EDNA.Reassessment, EDNA.Userlevel, EDNA.Categories, EDNA.Conditions, EDNA.Indexing, EDNA.Review, EDNA.Version.*

Estes elementos são criados especificamente para a rede de educação da Australia. O formato é semelhante aos elementos *Dublin Core*, mas estes elementos, não terão nenhum significado fora do ambiente da EdNA. **[COM 2001]**.

2.4 Descrevendo e recuperando fotos usando RDF e HTTP

Descrevendo e recuperando fotos usando RDF e HTTP é uma nota proposta por **[LAF 2000]** que apresenta um projeto para descrever e recuperar fotos digitalizadas com *metadata* RDF. As fotos são contempladas com modelos baseados em três esquemas:

- O *Dublin Core schema:* é um esquema geral para identificar trabalhos originais, tipicamente livros e artigos, mas também filmes, pinturas ou fotografias.
- O *Technical schema:* informa dados técnicos sobre a fotografia e a máquina fotográfica, como o tipo de máquina fotográfica, o tipo de filme, a data que o filme foi revelado o scanner e software usados para digitalizar.
- O *Content schema:* Este esquema é usado para categorizar o assunto da fotografia por meio de um vocabulário controlado. Este esquema permite recuperar fotografias baseando-se em tais características como retrato, retrato de grupo, paisagem, arquitetura, esporte, animais, etc.

Além disso, essa nota descreve um programa para entrar *metadata* rapidamente para um grande número de fotografias, um modo para servir as fotografias com *metadata* sobre o HTTP e algumas sugestões para métodos de procura e recuperação baseado nas descrições delas.

Como os esquemas são criados por comunidades diferentes, esquemas sobre a mesma categoria de coisas podem ser elaborados independentemente. Isso é verdade e espera-se que os esquemas mais populares sejam adotados pelas comunidades de descrição de recursos como um processo natural de avaliação e aceitação do padrão.

3 Resource Description Framework

Como resultado de muitas comunidades trabalhando em conjunto e concordando com princípios básicos de representação de *metadata* e transporte RDF tem sofrido influências de várias diferentes fontes. A principal influência tem vindo da comunidade de padronização da *Web* (*Web Standardization Community*), da comunidade de biblioteca (*Library Community*), da comunidade de documentos estruturados (*Structured Document Community*) e também da comunidade de representação de conhecimento (*Knowledge Representation Community*). Existem também outras áreas da tecnologia que contribuem para a criação do RDF entre elas as áreas de programação orientada a objetos e modelagem de linguagens, como também a área de banco de dados.

Características do RDF podem gradativamente proporcionar interoperabilidade entre aplicações que trocam informações compreensíveis por máquina na rede, em descoberta de recursos para prover melhores mecanismos de procura, em catalogação para descrever as relações de conteúdo e conteúdos disponíveis em um *Web Site* em particular, *Home-Page* ou biblioteca digitais por agentes de software inteligentes para facilitar a troca e compartilhamento de conhecimento, em avaliação de conteúdo, descrevendo coleções de páginas que representam um único documento lógico, para descrever direitos de propriedades intelectuais de *Home-Pages*, e para expressar preferências de privacidade para um usuário bem como políticas de privacidade de um *Web Site*. RDF com assinaturas digitais será a chave para a construção da *Web* de confiança para comércio eletrônico, colaboração e outras aplicações. **[W3C 99]**

3.1 Modelo RDF

O fundamento básico do RDF é um modelo para representar propriedades nomeadas e valores de propriedade. Propriedades de RDF podem ser pensadas como atributos de recursos que neste contexto corresponde aos tradicionais pares atributo-valor. Na terminologia de orientação a objetos, recursos correspondem a objetos e propriedades correspondem a variáveis de instância.

O modelo básico consiste em três tipos de objetos:

Recursos: Todas as coisas sendo descritas pelas expressões RDF são chamadas de recursos. Um recurso pode ser uma página da *Web* inteira, tal como um documento HTML, um recurso pode ser parte de uma página *Web*, por exemplo, um específico elemento HTML ou XML dentro de um documento origem. Um recurso pode ser também uma coleção inteira de páginas, por exemplo, um *Web Site* inteiro. Um recurso também pode ser um objeto que não está diretamente acessível via *Web*, como um livro impresso. Recursos são sempre referenciados por um URI, mais o opcional identificador de âncora. Qualquer coisa pode ter um URI, a extensibilidade de URI permite a introdução de identificadores para qualquer entidade imaginável.

Propriedades: Uma propriedade é um aspecto específico, característica, atributo, ou relação usada para descrever um recurso. Cada propriedade tem um significado específico, definem seus valores permitidos, os tipos de recursos que podem descrever e sua relação com outras propriedades.

Declarações: Um recurso específico junto com uma propriedade nomeada mais o valor daquela propriedade para aquele recurso é uma declaração RDF. Estas três partes individuais de uma declaração são chamadas respectivamente, assunto, predicado e objeto. O objeto de uma declaração (por exemplo, o valor de propriedade) pode ser outro recurso ou pode ser um literal como um recurso especificado por um URI ou uma simples *string* ou outro tipo de dados primitivo definido por XML. Em RDF, uma literal pode ter conteúdo que é marcação XML, mas não é avaliado pelos processadores.

Uma declaração RDF pode ser desenhada usando diagramas de arcos e nodos, onde os nodos (desenho oval) representam os recursos, os arcos (setas) representam as propriedades nomeadas. Nodos que representam literais serão desenhadas como retângulo.

O modelo de dados RDF é um modelo de sintaxe para representar expressões de RDF. A representação do modelo de dados é usada para avaliar equivalência de significado. Duas expressões RDF são equivalentes se e somente se suas representações de modelo de dados forem a mesma. Esta definição de equivalência permite algumas variações sintáticas nas expressões sem alterar o significado.

Considere as duas sentenças abaixo (tabela 3.1):

TABELA 3.1 – Sentenças sintaticamente diferentes com o mesmo significado

1.	O autor do Documento 1 é João
2.	João é autor do Documento 1

Para humanos, estas declarações levam o mesmo significado (quer dizer, o *João* é o autor de um documento particular). Para uma máquina, porém, estas são sentenças completamente diferentes. Considerando que os humanos são extremamente adeptos a extrair significado de diferentes construções sintáticas, máquinas permanecem grotescamente ineptas **[MIL 98].** Usando um modelo *triple*, recursos, propriedades e objetos (correspondentes valores), RDF tenta prover um método sem ambigüidades de expressar semântica em uma codificação legível por máquina.

As duas sentenças, acima descritas, no modelo proposto pelo RDF têm as seguintes partes (tabela 3.2):

TABELA 3.2 – Partes de uma declaração RDF

Recurso	Documento 1
Propriedade	Autor
Objeto	João

Nesse contexto, o esquema do grafo rotulado seria o mesmo para as duas sentenças, tornando a codificação "legível por máquina" semanticamente equivalente.

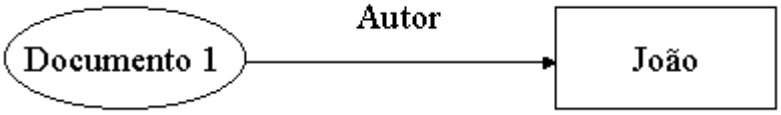

FIGURA 3.1 - Representação de modelo de dados RDF

A sentença abaixo mostra algumas características a mais para o criador do recurso “http://www.faccar.com.br/fulano.htm”, sendo elas “nome” e “e-mail”.

“O empregado referenciado pelo id85740 é chamado Fulano de Tal e tem o endereço de e-mail fulano@faccar.com.br. O recurso http://www.faccar.com.br/fulano.htm foi criado por este indivíduo”.

A idéia desta sentença é fazer o valor da propriedade “*Criador*” uma entidade estruturada. Em RDF tal entidade é representada como um outro recurso (Figura 3.2):

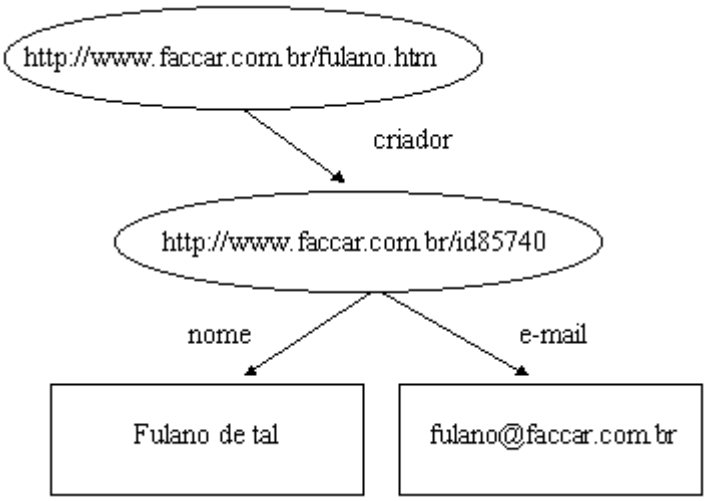

FIGURA 3.2 – Valor da propriedade como recurso

3.1.1 Containers

Freqüentemente é necessário se referir a uma coleção de recursos, por exemplo, para dizer que um trabalho foi criado por mais de uma pessoa ou listar os estudantes em um curso ou os módulos de software em um pacote. Em RDF são usados *containers* para manter tal lista de recursos ou literais.

O RDF define três tipos de objetos *container*:

Bag: Uma lista desordenada de recursos ou literal. São usadas *Bag* para declarar que uma propriedade tem valores múltiplos e que não existe nenhum significado para a ordem no qual os valores são dados. *Bag* poderia ser usada para dar uma lista de números de partes onde a ordem de processar as partes não importa. Valores duplicados são permitidos.

Por exemplo, para sentença “*The students in course 6.001 are Amy, Tim, John, Mary, and Sue*” o correspondente modelo RDF seria (figura. 3.3):

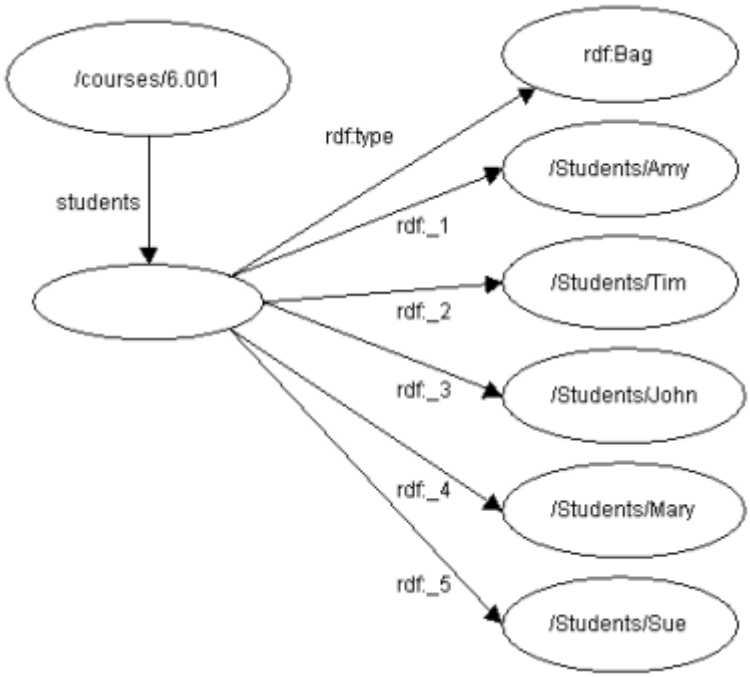

FIGURA 3.3 – *Container* do tipo *Bag*
Fonte: *W3C – Model and Syntax Specification*

Sequence: Uma lista ordenada de recursos ou literal. *Sequence* é usada para declarar que uma propriedade tem múltiplos valores e que a ordem dos valores é significante. Por exemplo, *Sequence* poderia ser usada para preservar a ordem alfabética de valores. Valores duplicados são permitidos. A (figura 3.4) apresenta um exemplo do modelo RDF para um *container Sequence* com os estudantes do curso 6.001 em ordem alfabética:

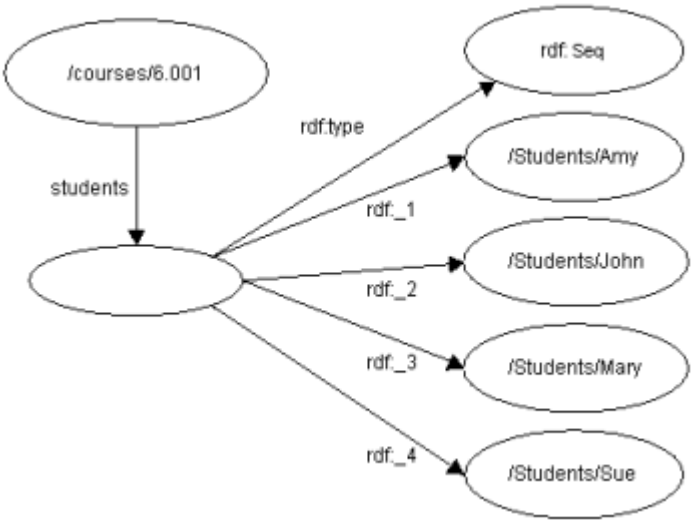

FIGURA 3.4 – *Container* do tipo *Sequence*

Alternative: Uma lista de recursos ou literais que representam alternativas para o único valor da propriedade. *Alternative* poderia ser usada para listar os locais da Internet aos quais um recurso poderia ser encontrado. Uma aplicação que usa uma propriedade cujo valor é um *container Alternative* está atenta que pode escolher, como apropriado, qualquer membro da lista.

Para sentença "*The source code for X11 may be found at ftp.x.org*, *ftp.cs.purdue.edu* or *ftp.eu.net*." o correspondente modelo RDF *Alternative* seria como mostra a (figura 3.5).

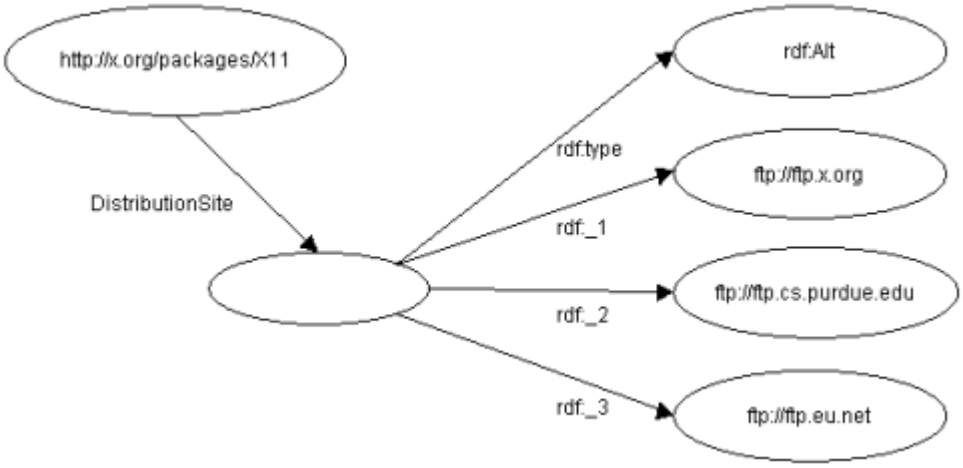

FIGURA 3.5 – *Container* do tipo *Alternative*
Fonte: *W3C – Model and Syntax Specification*

3.1.2 Sintaxe RDF

A principal meta do RDF é definir um mecanismo para representar *metadata* RDF tão bem como uma sintaxe para codificar e transportar este *metadata*. A sintaxe do RDF é baseada na XML. **[W3C 2000].** Uma das metas de RDF é tornar possível especificar semântica para dados baseados em XML de maneira padronizada. RDF e XML são complementares: RDF é um modelo de *metadata* que herda por referência muito dos aspectos de codificação que o transporte e o armazenamento de arquivo requerem, como internacionalização, conjunto de caracteres, etc.

Para os propósitos de criação e intercâmbio de *metadata* duas sintaxes XML para codificar um modelo de dados RDF são definidas: A "*serialization syntax*" expressa as capacidades completas dos modelos de dados e a "*abbreviated syntax*" que provê uma forma mais compacta para representar um subconjunto do modelo de dados. A (tabela 3.3) apresenta uma descrição RDF equivalente nas duas formas:

TABELA 3.3 – Dois tipos de sintaxe RDF

Serialization Syntax

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org">
    <s:Publisher>World Wide Web Consortium</s:Publisher>
    <s:Title>W3C Home Page</s:Title>
    <s:Date>1998-10-03T02:27</s:Date>
  </rdf:Description>
</rdf:RDF>
```

Abbreviated Syntax

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org"
       s:Publisher="World Wide Web Consortium"
       s:Title="W3C Home Page"
       s:Date="1998-10-03T02:27"/>
</rdf:RDF>
```

Também é importante entender que esta sintaxe de XML é somente uma possível sintaxe para RDF e que meios alternativos para o mesmo modelo de dados podem emergir. **[W3C 99]**

3.1.3 Declarações sobre declarações

Além de fazer declarações sobre recursos da *Web*, RDF pode ser usado para fazer declarações sobre declarações. Para fazer uma declaração sobre outra declaração, tem-se na verdade que construir um modelo da declaração original, este modelo é um novo recurso para o qual pode-se anexar propriedades adicionais.

Considere a seguinte sentença: "Fulano de Tal é o criador do recurso *http://www.faccar.com.br/fulano.htm*".

RDF consideraria esta sentença como um fato. Se, ao invés, fosse considerada a sentença "Beltrano de Tal disse que Fulano de Tal é o criador do recurso *http://www.faccar.com.br/fulano.htm*" não foi dito nada sobre o recurso ao invés, foi expresso um fato sobre uma declaração que Beltrano fez. Para expressar este fato em RDF, faz-se necessário modelar a declaração original como um recurso com quatro propriedades. Este processo é formalmente chamado de *reification* na comunidade de representação de conhecimento. O modelo de uma declaração é chamado como *reified*.

Para modelar declaração, RDF define as seguintes propriedades:

Subject: A propriedade *subject* identifica o recurso que é descrito pela declaração modelada; quer dizer, o valor da propriedade *subject* é o recurso sobre o qual a declaração original foi feita (em nosso exemplo, *http://www.faccar.com.br/fulano.htm*).

Predicate: A propriedade *predicate* identifica a propriedade original na declaração modelada. O valor da propriedade *predicate* é um recurso que representa a propriedade específica na declaração original (em nosso exemplo, *criador*).

Object: A propriedade *object* identifica o valor de propriedade na declaração modelada. O valor da propriedade *object* é o objeto na declaração original (em nosso exemplo, *Fulano de Tal*).

Type: O valor da propriedade *type* descreve o tipo do novo recurso. Todas as declarações *reified* são instâncias de *RDF:Statement,* quer dizer, elas têm uma propriedade de tipo cujo objeto é *RDF:Statement*.

Um recurso novo com as quatro propriedades acima representa a declaração original e pode ser usado como o objeto de outras declarações e tem declarações adicionais feitas sobre ele. A declaração modelada com as quatro propriedades não é uma substituição para a declaração original, ela é um modelo da declaração. Uma declaração e sua correspondente declaração *reified* existem independentemente em um gráfico RDF e qualquer um pode estar presente sem o outro. É dito que o gráfico de RDF contém o fato dado à declaração se e somente se a declaração está presente no gráfico, independente se a declaração *reified* correspondente está presente.

Para modelar a sentença acima, pode-se anexar outra propriedade à declaração *reified* (digo, "atribuidoPara") com um apropriado valor (neste caso "Beltrano de Tal"). Usando a sintaxe RDF/XML esta poderia ser escrita como (figura 3.6):

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:a="http://www.faccar.com.br/schema/">
  <rdf:Description>
    <rdf:subject resource="http://www.faccar.com.br/fulano.htm"/>
    <rdf:predicate resource="http:// www.faccar.com.br/schema/Criador"/>
    <rdf:object>Fulano de Tal</rdf:object>
    <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement" />
    <a:atribuidoPara>Ralph Swick</a:atribuidoPara>
  </rdf:Description>
</rdf:RDF>
```

FIGURA 3.6 – Sintaxe RDF/XML para declaração *reified*

3.1.4 Modelo formal para RDF

Esta seção apresenta três representações para o modelo de dados; como *triple*, como grafo rotulado e em RDF/XML. Estas representações têm significados equivalentes.

O modelo de dados RDF é definido formalmente como segue:

- Há um conjunto chamado *Resources;*
- Há um conjunto chamado *Literals*;
- Há um subconjunto de *Resources* chamado *Properties*;
- Há um conjunto chamado *Statements,* cada elemento no qual é uma *triple* na forma {*pred, sub, obj*}

Onde *pred* é uma propriedade (membro de *Properties*), *sub* é um recurso (membro de *Resources*), e *obj* ou é um recurso ou um literal (membro de *Literals*).

Pode-se ver um conjunto de declarações (membros de *Statements*) como um gráfico rotulado: cada recurso e literal é um vértice; uma triple {*p, s, o*} é um arco de *s* para *o*, rotulado por *p*. (Figura 3.7).

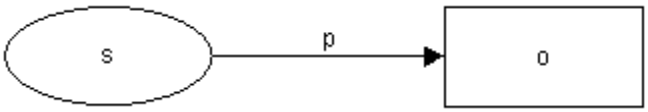

FIGURA 3.7 – Modelo grafo rotulado de declaração simples
Fonte: *W3C – Model and Syntax Specification*

Isso pode ser lido como:

o é o valor de *p* para *s*

ou da esquerda para a direita

s tem a propriedade *p* com um valor *o*

ou até mesmo

O *p* de *s* é *o*

Por exemplo, a sentença:
“Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila”

Seria representado graficamente como segue:

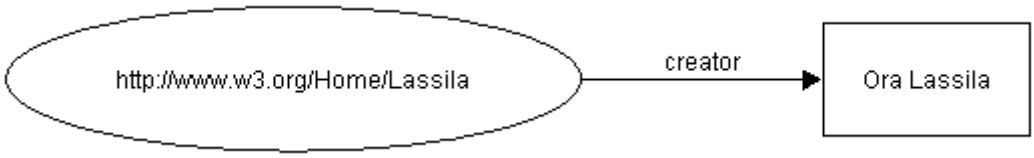

FIGURA 3.8 – Simples declaração gráfica
Fonte: *W3C – Model and Syntax Specification*

E a correspondente *triple* (membro de *Statements*) seria:

{creator, [http://www.w3.org/Home/Lassila], "Ora Lassila"}

A notação [*I*] denota o recurso identificado pelo URI *I* e as aspas denotam um literal. Usando *triple*, pode-se expressar como as declarações são *reified*. Dado a declaração:

{creator, [http://www.w3.org/Home/Lassila], "Ora Lassila"}

Pode-se expressar a *reification* como um novo recurso X como segue:

{type, [X], [RDF:Statement]}
{predicate, [X], [creator]}
{subject, [X], [http://www.w3.org/Home/Lassila]}
{object, [X], "Ora Lassila"}

Do ponto de vista de um processador de RDF, fatos (quer dizer, declarações) são *triples* que são membros de *Statements.* Tem-se meramente adicionados mais quatro *triples*.

A propriedade nomeada *type* é definida para prover tipo primitivo. A definição formal para *type* é:

- Existe um elemento de *Properties* conhecido como *RDF:type*;
- Membros de *Statements* da forma {*RDF:type, sub, obj*} tem que satisfazer o seguinte: *sub* e *obj* são membros de *Resources;*

Além disso, a especificação formal da *reification* é:

- Há um elemento de *Resources*, não contido em *Properties*, conhecido como *RDF:Statement*;
- Há três elementos em *Properties* conhecidos como *RDF:predicate*, *RDF:subject* e *RDF:object;*
- *Reification* da triple {*pred, sub, obj*} de *Statements* é um elemento *r* de *Resources* representando a triple *reified* e os elementos s_1, s_2, s_3, e s_4 de *Statements* tal que:

s_1: *{RDF:predicate, r, pred}*
s_2: *{RDF:subject, r, subj}*
s_3: *{RDF:object, r, obj}*
s_4: *{RDF:type, r, [RDF:Statement]}*

O recurso *r* na definição acima é chamado de declaração *reified*. Quando um recurso representa uma declaração *reified*, quer dizer, tem uma propriedade de *RDF:type* com um valor *RDF:Statement*, então aquele recurso tem que ter exatamente uma propriedade de *RDF:subject* uma propriedade de *RDF:object* e uma propriedade de *RDF:predicate*.

Freqüentemente é necessário representar uma coleção de recursos ou literais, por exemplo, para declarar que uma propriedade tem uma sucessão ordenada de valores. RDF define três tipos de coleções: lista ordenada, chamada *Sequence*, uma lista não ordenada chamada *Bag*, e lista que representa alternativas para um simples valor de uma propriedade, chamada *Alternative*.

Formalmente, estes três tipos de coleção estão definidos por:

- Existem três elementos de *Resources*, não contidos em *Properties*, conhecidos como *RDF:Seq*, *RDF:Bag*, e *RDF:Alt;*
- Existe um subconjunto de *Properties* correspondendo aos ordinais (*1, 2, 3, ...*) chamados *Ord*. Refere-se para elementos de *Ord* como *RDF:_1, RDF:_2, RDF:_3, ...*

Para representar uma coleção *c*, crie uma triple {RDF:type, c, t} onde *t* é que um das três tipos de coleções *RDF:Seq*, *RDF:Bag*, ou *RDF:Alt*. As *triple* restantes *{RDF:_1, c, r1},..., {RDF:_n, c, rn},...* apontam para cada membro *rn* da coleção. Para um único recurso de coleção pode haver no máximo uma *triple* cujo predicado é qualquer determinado elemento de *Ord* e os elementos de *Ord* devem ser usados em seqüência começando com *RDF:_1*. Para recursos que são instâncias do tipo de coleção *RDF:Alt*, deve haver exatamente uma *triple* cujo predicado é *RDF:_1* e isso é o valor padrão para recurso *Alternative* (quer dizer, sempre deve haver pelo menos uma alternativa).

3.2 Esquema RDF

O RDFS - *Resource Description Framework Schema Specification* **[W3C 2000a]** provê um sistema básico de tipos necessário para definir as classes de recursos e propriedades que podem ser usadas. Assim, o sistema de tipo definido se torna parte do modelo do RDF de qualquer descrição que os usa.

Um esquema define as condições que serão usadas em declarações de RDF e serão dados significados específicos a eles. **[W3C 99].** O esquema é um lugar onde são documentadas definições e restrições de uso para propriedades. Para evitar confusões entre independentes e possivelmente conflitantes definição do mesmo termo, RDF usa a facilidade de *namespace* XML. **[W3C 98]**. *Namespaces* simplesmente são um modo para amarrar um uso específico de uma palavra dentro de um contexto para o esquema onde a definição pretendida será encontrada. Em RDF, cada predicado usado em uma declaração deve ser identificado com exatamente um *namespace*, ou *schema*.

RDF representa uma evolução do modelo Warwick Framework em que o Warwick Framework permitiu representar cada vocabulário de *metadata* em uma sintaxe diferente. Em RDF, todos os vocabulários são expressos dentro de um único modelo bem definido. Isto permite gerenciar os vocabulários de maneira descentralizada pelas comunidades independentes de peritos. **[W3C 2000a]**.

Segundo **[BER 98]** os sistemas de cálculos de predicados tentados antes pela *KR Knowledge Representation Community* não tiveram um impacto profundo, porque todos

eles eram conceitualmente ou fisicamente centralizados em sua abordagem de organizar o conhecimento.

Como especificado em **[W3C 99]**, recursos podem ser instâncias de uma ou mais classes. Isto é indicado com a propriedade *rdf:type*. As classes são freqüentemente organizadas de uma maneira hierárquica, por exemplo, uma classe "Cachorro" poderia ser considerada uma subclasse da classe "Mamífero" que é uma subclasse da classe "Animal", enquanto significando que qualquer recurso do qual o *rdf:type* é de "Cachorro" é também considerado ser do tipo "Animal". O RDFS contém a propriedade *rdfs:subClassOf* para denotar tais relações entre classes.

O RDFS é semelhante aos sistemas de tipo de linguagens de programação orientado a objetos como *Java*. Porém, ele difere de tais sistemas que, em vez de definir uma classe em termos das propriedades que suas instâncias podem ter, um esquema RDFS define propriedades em termos de classes de recursos para o qual elas se aplicam. Por exemplo, pode-se definir a propriedade "Autor" para se ter um domínio de "Livro" e *range* literal, considerando que um sistema de orientação a objeto clássico poderia definir tipicamente a classe "Livro" com um atributo chamado "Autor" de tipo "literal".

Se uma classe for um subconjunto de outra, então há um arco de *rdfs:subClassOf* do nodo que representa a primeira classe ao nodo que representa a segunda. Semelhantemente, se um recurso for uma instância de uma classe, então há um arco de *rdf:type* do recurso para o nodo que representa a classe. (Figura 3.9):

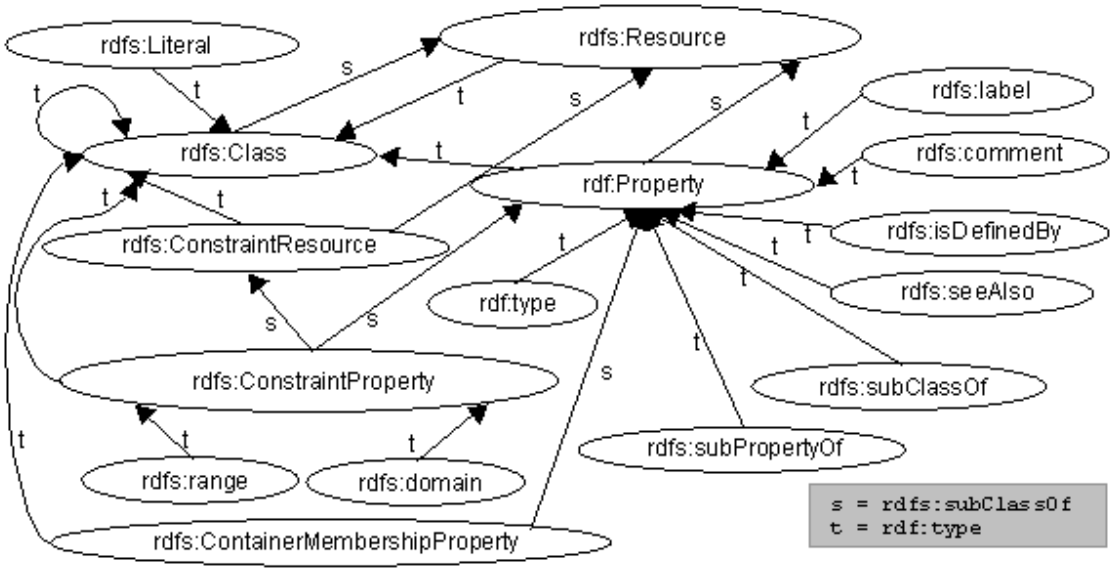

FIGURA 3.9 – Hierarquia de Classes para o RDFS
Fonte: *W3C – Schema Specification 1.0*

3.2.1 O núcleo classe

Os recursos seguintes são o núcleo de classes que estão definidos como parte do vocabulário do RDFS. Todo modelo de RDF que utiliza o esquema RDFS implicitamente inclui estes:

rdfs:Resource: todas as coisas descritas através de expressões RDF são chamadas *resources* e são consideradas instâncias da classe *rdfs:Resource*.

rdf:Property: representa o subconjunto de recursos de RDF que são propriedades.

rdfs:Class: Isto corresponde ao conceito genérico de tipo ou categoria, semelhante a noção de classe em linguagens de programação orientadas a objetos como *Java*. Quando um esquema definir uma classe nova, o recurso que representa aquela classe tem que ter uma propriedade de *rdf:type* cujo valor é o recurso *rdfs:Class*. Classes de RDF podem ser definidas para representar quase qualquer coisa, tal como páginas *Web*, pessoas, tipos de documento, bancos de dados ou conceitos abstratos.

3.2.2 O núcleo propriedades

Todo modelo RDF que também usa o mecanismo de esquema, implicitamente inclui as propriedades da classe *rdf:Property* e provêe um mecanismo para expressar relações entre classes e suas instâncias ou superclasses.

rdf:type: isto indica que um recurso é membro de uma classe, e assim tem todas as características que serão esperadas de um membro daquela classe. Quando um recurso tiver uma propriedade *rdf:type* cujo valor é alguma classe específica, pode-se dizer que o recurso é uma *instance of* dessa classe. O valor de uma propriedade de *rdf:type* para algum recurso é outro recurso que deve ser uma instância de *rdfs:Class*.

rdfs:subClassOf: Esta propriedade especifica uma relação de subconjunto/superconjunto entre classes. A propriedade *rdfs:subClassOf* é transitiva. Se a classe *A* é uma subclasse de alguma classe mais ampla *B*, e *B* é uma subclasse da classe *C*, então *A* também é implicitamente uma subclasse de *C*. Por conseguinte, recursos que também são instâncias da classe *A* serão instâncias de *C*, desde que *A* seja um subconjunto de *B* e *C*. Só instância de *rdfs:Class* podem ter a propriedade de *rdfs:subClassOf* e o valor da propriedade *rdf:type* sempre é *rdfs:Class*. Uma classe pode ser uma subclasse de mais de uma classe.

rdfs:subPropertyOf: a propriedade *rdfs:subPropertyOf* é uma instância de *rdf:Property* que é usado para especificar que uma propriedade é uma especialização de outra. Uma propriedade pode ser uma especialização de zero, uma ou mais propriedades. Se alguma propriedade *P2* é uma *subPropertyOf* de outra propriedade mais geral *P1*, e se um recurso *A* tem uma propriedade *P2* com um valor *B*, isto implica que o recurso *A* também tem uma propriedade de *P1* com o valor *B*.

*rdfs:seeAlso***:** a propriedade *rdfs:seeAlso* especifica um recurso que poderia prover informação adicional sobre o assunto recurso.

rdfs:isDefinedBy: a propriedade *rdfs:isDefinedBy* é uma subpropriedade de *rdfs:seeAlso*, e indica o recurso definindo o recurso assunto. Como *rdf:seeAlso*, esta propriedade pode ser aplicada a qualquer instância de *rdfs:Resource* e pode ter como seu valor qualquer *rdfs:Resource*.

3.2.3 O núcleo restrições

Um RDFS poderia descrever limitações nos tipos de valores válidos para alguma propriedade, ou nas classes para as quais faz sentido designar tais propriedades.

Esquemas RDF podem expressar restrições que relacionam itens de vocabulário de múltiplos esquemas desenvolvidos independentemente. Desde que são usadas referências URI para identificar classes e propriedades, é possível criar propriedades novas do qual as restrições *domain* ou *range* são definidas em outro *namespace*.

Alguns exemplos de restrições incluem:

- Que o valor de uma propriedade deveria ser um recurso de uma classe designada. Isto é conhecido como uma restrição *range*. Por exemplo, uma restrição *range* aplicado à propriedade "Autor" poderia expressar que o valor da propriedade "Autor" deve ser um recurso da classe "Pessoa";
- Que uma propriedade pode ser usada no recurso de uma certa classe. Isto é conhecido como uma restrição *domain*.

rdfs:ConstraintResource: este recurso define uma subclasse *rdfs:Resource* cujas instâncias são construções de RDF esquema, envolvido na expressão de restrição.

rdfs:ConstraintProperty: este recurso define uma subclasse de *rdf:Property*, cujas instâncias são usadas para especificar restrições. Esta classe é uma subclasse de *rdfs:ConstraintResource* e corresponde ao subconjunto daquela classe que representa propriedades. *Rdfs:domain* e *rdfs:range* são instâncias de *rdfs:ConstraintProperty*.

rdfs:range: é uma instância de *ConstraintProperty* usada para indicar classes que os valores de uma propriedade devem ser membros. O valor de uma propriedade *range* sempre é uma classe. Só são aplicadas restrições *range* a propriedades.

rdfs:domain: uma instância de *ConstraintProperty* que é usado para associar uma classe com propriedades que suas instâncias podem ter.

Como um exemplo simples considere a classe "VeiculosMotor". A propriedade "registradoPara" poderia ser aplicada para qualquer veículo a motor e seu valor é uma pessoa. (Figura 3.10)

```
<rdf:Description ID="registradoPara">
  <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#VeiculosMotor"/>
  <rdfs:range rdf:resource="#Pessoa"/>
</rdf:Description>
```

FIGURA 3.10 – Exemplo de propriedades com restrições *domain* e *range*

3.2.4 Documentação

As seguintes propriedades são providas para suportar documentação simples e anotações dentro do esquema RDF.

rdfs:comment: isto é usado para prover uma descrição legível de um recurso.

rdfs:label: isto é usado para prover uma versão legível de um nome de recurso.

4 Estratégias de elaboração e publicação de metadata RDF

Para facilitar a identificação, descrição e localização de recursos eletrônicos na rede, mecanismos de *metadata* têm sido utilizados. A idéia consiste em acrescentar propriedades e valores que pertencem aos recursos, mas que não estão presente neles. Isso é conseguido adicionando-se *metadata* RDF em páginas HTML, documentos XML e até mesmo em imagens. A declaração das propriedades (atributos) e a semântica correspondente estão definidas no contexto do RDF como um RDFS. Antes de elaborar o modelo de *metadata* deve-se realizar a escolha do esquema no qual ele será baseado. Um único modelo pode conter propriedades de mais de um esquema.

Apresenta-se a seguir estratégias que abrangem desde a criação de esquema até a publicação de recursos com *metadata* RDF.

Dos três tipos de recursos comentados neste tópico, optou-se em apresentar a título de motivação o uso do RDF para descrição e publicação de imagens, já que se trata de uma solução não convencional. O conjunto utilizado, relacionado a uma instituição de ensino, é composto basicamente com imagens de alunos e professores.

4.1 Elaboração de esquemas

Para facilitar a codificação do esquema, utilizou-se uma ferramenta usada por desenvolvedores de sistemas e *experts* de domínio para desenvolver sistemas baseados em conhecimento. O Protégé-2000 **[STA 2000]** foi desenvolvido pelo grupo de modelagem de conhecimento dentro do projeto de pesquisa de informática médica da Universidade de Stanford. Esta ferramenta permite a criação de sistemas de classes e propriedades na sintaxe RDFS. Segundo **[W3C 2000a]** a linguagem de especificação de esquema é uma linguagem influenciada por idéias da comunidade de representação de conhecimento como também da comunidade de linguagens de especificação de esquema de banco de dados. Isso torna evidente a relação de um software médico com o modelo RDF.

O esquema criado, conhecido por “EsquemaPessoas”, possui três classes que irão categorizar as imagens. São elas: “Pessoa”, “Aluno” e “Professor”. Cada classe possui um conjunto de propriedades correspondente e cada propriedade pertence a um domínio específico. Da mesma forma que nas linguagens orientada a objetos, “Aluno” e “Professor” são subclasses de “Pessoa”, portanto as propriedades nome e sexo definido em Pessoa são herdados por *Aluno* e *Professor*.

O Protégé-2000, além de oferecer suporte a codificação de esquemas, permite a geração automática de toda a documentação da hierarquia de classes.

A seguir, apresenta-se o modelo gráfico do “EsquemaPessoas” e a correspondente interface do Protégé-2000 (figura 4.1 e 4.2) respectivamente:

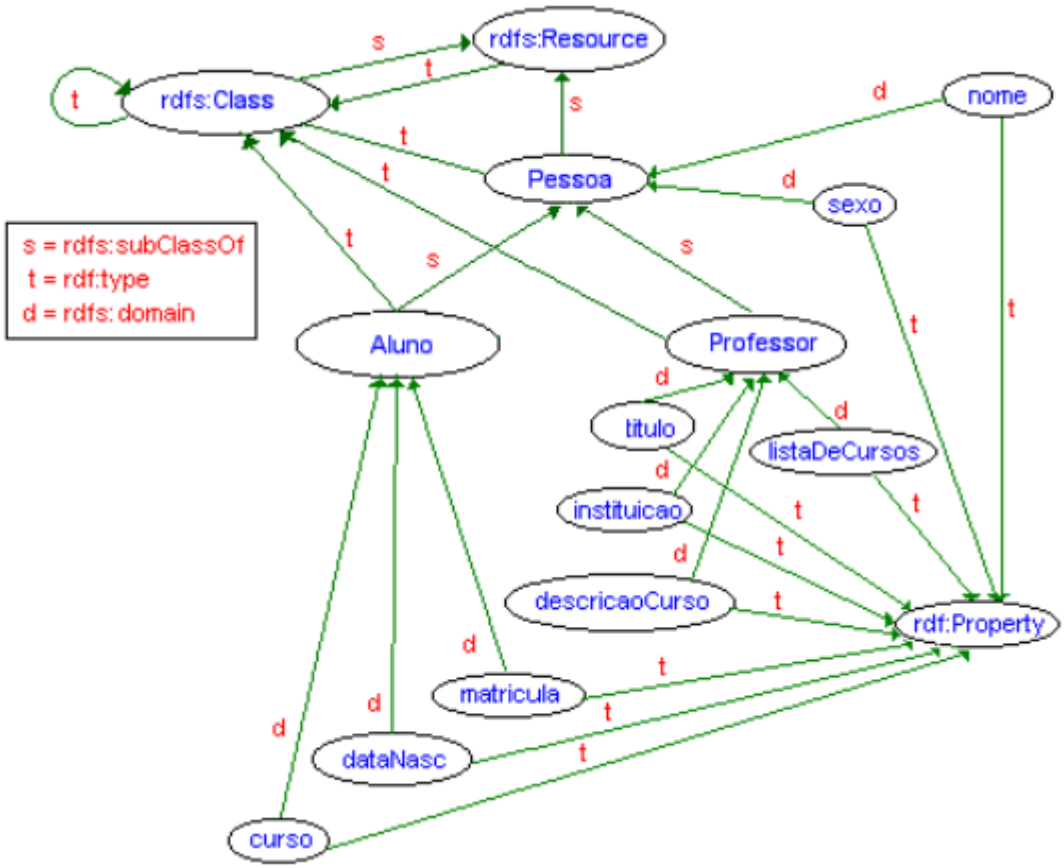

FIGURA 4.1 – Modelo gráfico RDF do "EsquemaPessoas"

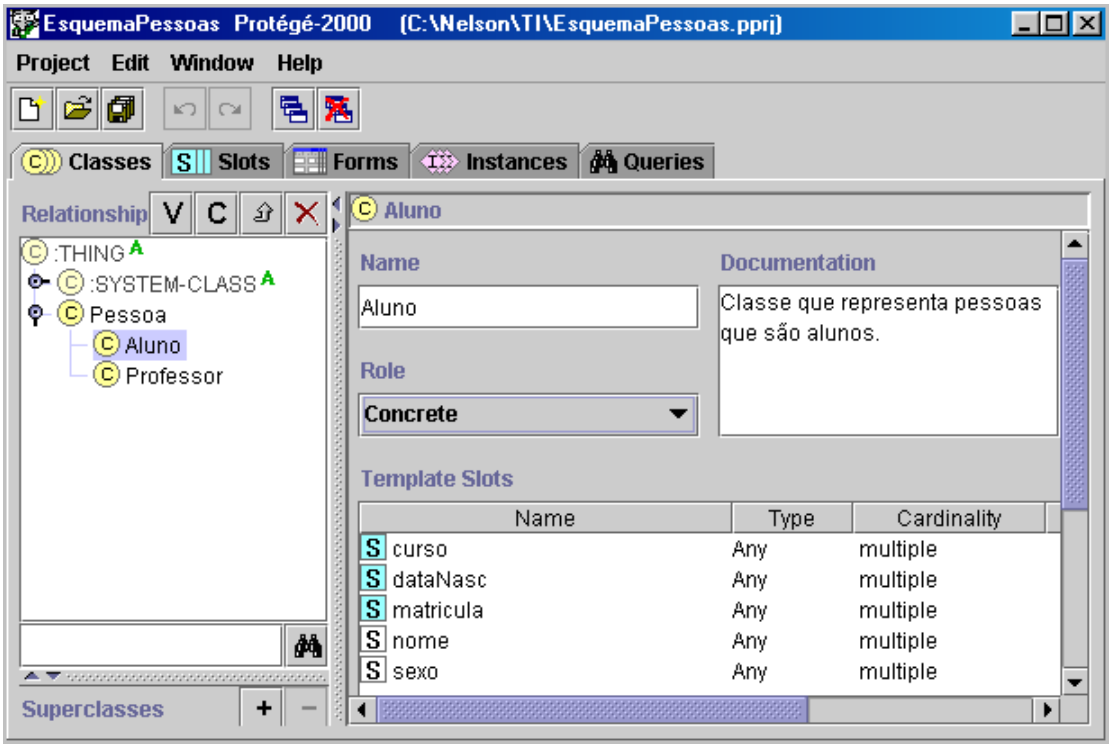

FIGURA 4.2 – Interface do Protégé-2000 com o "EsquemaPessoas"

A sintaxe RDF/XML do “EsquemaPessoas” gerada pelo Protégé-2000 pode ser visualizada por qualquer navegador compatível com a XML. (figura 4.3).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
        xmlns:EsquemaPessoas="http://ain/imagens/EsquemaPessoas#"
        xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
<rdfs:Class rdf:about="http://ain/imagens/EsquemaPessoas#Aluno"
        rdfs:comment="Classe que representa pessoas que são alunos.">
        <rdfs:subClassOf rdf:resource="http://ain/imagens/EsquemaPessoas#Pessoa">
</rdfs:Class>
<rdfs:Class rdf:about="http://ain/imagens/EsquemaPessoas#Pessoa"
        rdfs:comment="Classe que representa imagens de pessoas.">
        <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-
        19990303#Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://ain/imagens/EsquemaPessoas#Professor"
        rdfs:comment="Classe que representa pessoas que são professores.">
        <rdfs:subClassOf rdf:resource="http://ain/imagens/EsquemaPessoas#Pessoa"/>
</rdfs:Class>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#curso"
        rdfs:comment="Referente ao curso que o aluno está matriculado.">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Aluno" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#dataNasc"
        rdfs:comment="data de nascimento do aluno.">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Aluno" />
</rdf:Property>
        <rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#descricaoCurso"
        rdfs:comment="Referente a descrição do curso realizado pelo professor.">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Professor" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#instituicao"
        rdfs:comment="Representa a instituição que o professor adquiriu a titulação. ">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Professor" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#listaDeCursos"
        rdfs:comment="Representa uma bag de cursos que o professor possui.">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Professor" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#matricula"
        rdfs:comment="Referente ao número de matrícula do aluno na faculdade.">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Aluno" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#nome"
        rdfs:comment="nome da pessoa">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Pessoa" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#sexo"
        rdfs:comment="define o sexo da pessoa, M-masculino, F-feminino">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Pessoa" />
</rdf:Property>
<rdf:Property rdf:about="http://ain/imagens/EsquemaPessoas#titulo"
        rdfs:comment="representa o titulo obtido pelo professor. uma das tres opções:
        Doutor, Mestre ou Especialista. ">
        <rdfs:domain rdf:resource="http://ain/imagens/EsquemaPessoas#Professor" />
</rdf:Property>
</rdf:RDF>
```

FIGURA 4.3 – “EsquemaPessoas” gerado pelo Protégé-2000

4.2 Geração de metadata

Implementou-se dois algoritmos em Visual Basic com o propósito de gerar, na sintaxe RDF/XML modelos de *metadata*.

Cada algoritmo percorre a tabela de dados de professores e alunos, concatenado os valores de atributos com as propriedades definidas no "EsquemaPessoas". Propriedades do esquema RDF também são adicionadas.

Os resultados são modelos RDF para alunos e professores e que são adicionados nas imagens correspondentes.

Os modelos professor e aluno são apresentados como segue (figura 4.4 e 4.5 respectivamente):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://ain/schemas/EsquemaPessoas#">
<rdf:Description about="http://ain/schemas/PROF01.xml">
      <rdf:type rdf:resource="http://ain/schemas/EsquemaPessoas#Professor"/>
      <s:nome>Azenil Staviski</s:nome>
      <s:sexo>M</s:sexo>
          <s:listaDeCursos>
                <rdf:Bag>
                <rdf:li resource="#rec1" />
                <rdf:li resource="#rec2" />
                <rdf:li resource="#rec3" />
                <rdf:li resource="#rec4" />
                </rdf:Bag>
          </s:listaDeCursos>
</rdf:Description>
<rdf:Description ID="rec1">
      <s:titulo>Especialista</s:titulo>
      <s:instituicao>Universidade Estadual de Londrina</s:instituicao>
      <s:descricaoCurso>Ensino Superior</s:descricaoCurso>
</rdf:Description>
<rdf:Description ID="rec2">
      <s:titulo>Especialista</s:titulo>
      <s:instituicao>Faculdade de Ciências Contábeis de Arapongas</s:instituicao>
      <s:descricaoCurso>Administração Estatística Financeira</s:descricaoCurso>
</rdf:Description>
<rdf:Description ID="rec3">
      <s:titulo>Especialista</s:titulo>
      <s:instituicao>Universidade Estadual de Londrina</s:instituicao>
      <s:descricaoCurso>Elaboração, Gerência e Avaliação de Projetos de
      Governo</s:descricaoCurso>
</rdf:Description>
<rdf:Description ID="rec4">
      <s:titulo>Mestre</s:titulo>
      <s:instituicao>Universidade Estadual de Maringá</s:instituicao>
      <s:descricaoCurso>Mestrado em Economia</s:descricaoCurso>
</rdf:Description>
</rdf:RDF>
```

FIGURA 4.4 – Modelo "professor" na sintaxe RDF/XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://ain/imagens/EsquemaPessoas#">
<rdf:Description about="http://ain/exemplos/6078.xml">
      <rdf:type rdf:resource="http://ain/imagens/EsquemaPessoas#Aluno"/>
      <s:matricula>6078</s:matricula>
      <s:nome>SIMONE TITO FREITAS</s:nome>
      <s:dataNasc>01/05/75</s:dataNasc>
      <s:sexo>F</s:sexo>
      <s:curso>TPD</s:curso>
</rdf:Description>
</rdf:RDF>
```

FIGURA 4.5 – Modelo “aluno” na sintaxe RDF/XML

Os algoritmos de geração dos modelos “aluno” e “professor” são apresentados na íntegra no Anexo 2.

4.3 Inclusão de modelos RDF em imagens

O RDFPic **[NER 2000]** é uma ferramenta para embutir um modelo RDF de uma figura, na própria figura. Essa ferramenta permite embutir cada modelo na imagem correspondente.

A interface do RDFPic com a opção necessária para inserir os modelos RDF é apresentada como segue (figura 4.6):

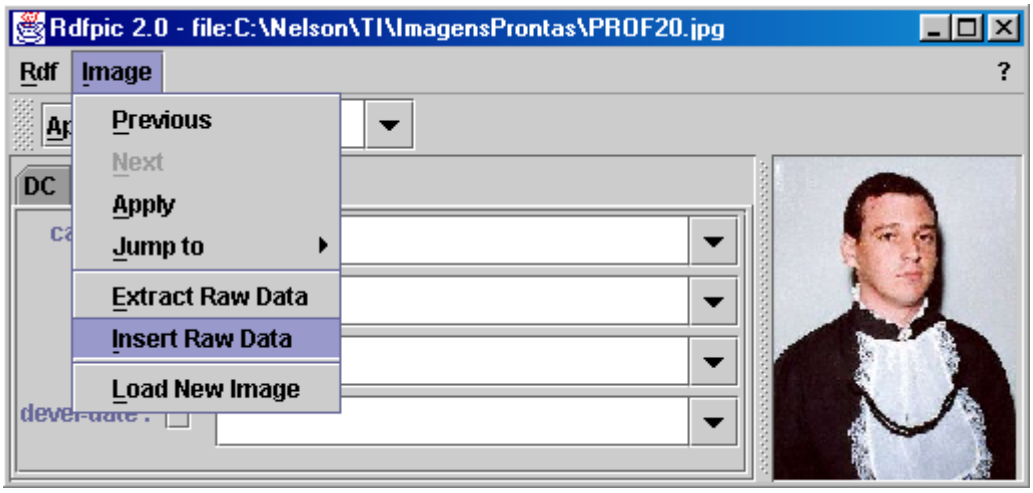

FIGURA 4.6 – Interface do RDFPic com a imagem de um professor

A ferramenta RDFPic só adiciona modelos em imagens JPEG. Primeiro é feito a carga da imagem e segundo a inclusão do modelo correspondente.

4.4 Publicação das imagens

O servidor Jigsaw **[JIG 2001]** é usado para simular dois recursos sobre o mesmo URI, primeiro a própria imagem, segundo o *metadata* embutido na mesma.

Utiliza-se o correspondente tipo MIME **[HOO 98]** no URI da imagem para que o servidor *Web* disponibilize somente a descrição RDF. (ex.: http://ain/imagem/6078.jpg;*text/xml*).

Essa capacidade estendida do Jigsaw se deve a um *Frame* criado para esse propósito. Um *Frame* é um objeto *Java* completo, contendo todas as informações necessárias para servir os recursos de imagens com descrições RDF usando protocolo HTTP.

Abaixo, apresenta-se a interface do JigAdmin, a ferramenta de administração do servidor Jigsaw. O quadro da esquerda apresenta a pasta onde estão relacionados os arquivos de imagem e o quadro da direita apresenta o conjunto de *Frames* disponível no servidor. O *Frame* usado para servir os modelos RDF embutido nas imagens está em destaque (figura 4.7):

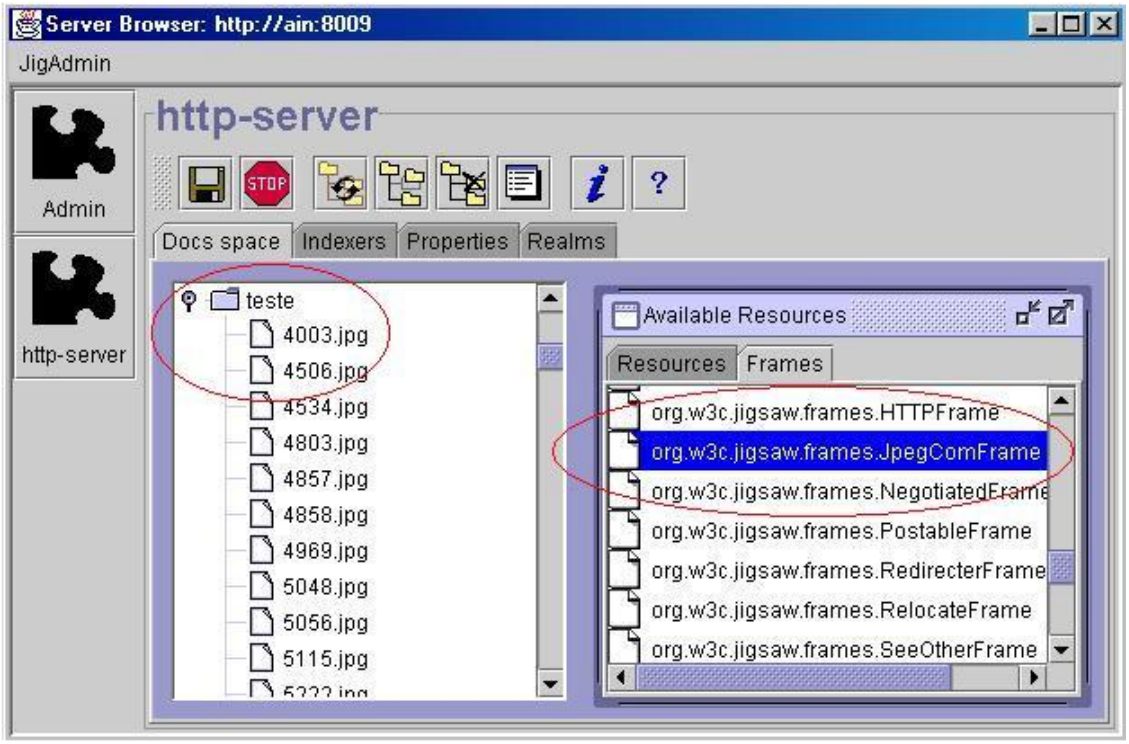

FIGURA 4.7 - Interface do JigAdmin

4.5 Parser RDF

Um parser RDF o SiRPAC **[SAA 99]** é um conjunto de classes de *Java* que pode analisar documentos RDF/XML em *triples* de um correspondente modelo de dados. Sendo o SiRPAC capaz de buscar uma descrição RDF de um determinado URI, o tipo MIME é só o que é necessário para trazer diretamente o conjunto de declarações referente ao *metatada* embutido. Apresenta-se na (tabela 4.1) a sintaxe para buscar a descrição RDF sobre uma imagem publicada na *Web*.

TABELA 4.1 – Sintaxe de uso do SiRPAC

java org.w3c.rdf.SiRPAC http://200.203.228.19:8001/rdf/prof01.jpg;text/xml

.

O capítulo cinco apresenta e comenta, todos os detalhes sobre o resultado do SiRPAC para modelos RDF/XML.

5 Análise dos resultados do SiRPAC

A análise dos resultados do *parser* se dá sobre vários modelos RDF como: O modelo básico RDF, *containers,* declarações sobre membros de um *container*, *containers* definidos por um padrão de URI, declarações sobre declarações, valores compartilhados e relações não binárias.

Segundo a gramática formal para RDF em **[W3C 99]** quando um atributo *about* é especificado com *Description* a declaração na *Description* refere-se ao recurso nomeado no *about*. Um elemento *Description* sem um atributo *about* representa um recurso novo, ou seja, um recurso que ainda não possui um URI.

O recurso novo tem um identificador de recurso formado pelo URI base do documento contendo a declaração RDF mais um identificador de âncora igual ao atributo *ID* do elemento *Description,* se presente.

Quando uma outra *Description* ou valor de propriedade se referir ao recurso novo ela usará o valor do atributo *ID* no seu atributo *about* (figura 5.1):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Bag ID="pages">
        <rdf:li resource="http://foo.org/foo.html" />
        <rdf:li resource="http://bar.org/bar.html" />
</rdf:Bag>
<rdf:Description about="#pages">
        <s:Creator>Ora Lassila</s:Creator>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.1 *Description* referindo-se a um identificador de recurso novo

Outros fatores devem ser considerados sobre o uso do atributo *about*. Considere uma descrição RDF embutida em uma página HTML, se for especificado um atributo *about* isso significa determinar um URI para esse recurso e todas as declarações na *description* referem-se ao recurso determinado no *about*.

Entretanto, se o recurso fosse tratado como uma entidade anônima, ou seja, se o atributo *about* fosse ignorado, um identificador para o recurso seria automaticamente gerado e desse modo, todas as declarações na *description* referir-se-iam a esse recurso.

Isso chama atenção para a seguinte situação: Considere uma descrição RDF embutida no recurso "*FulanoDeTal.htm*" codificado como um recurso anônimo, e publicado em "*http://ain/exemplos/*". (Figura 5.2).

```
<html>
<read><title>Exemplo de RDF embutido em HTML</title>
<!--
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:s="http://ain/exemplos/EsquemaExemplos#">
 <rdf:Description>
    <s:Criador>
       <rdf:Description>
          <s:nome>Fulano de Tal</s:nome>
          <s:email>Fulano@ain.com.br</s:email>
       </rdf:Description>
    </s:Criador>
 </rdf:Description>
</rdf:RDF>
-->
</read>
<body bgcolor="orange">
<h2>Fulano de Tal Home-page</h2>
</body>
</html>
```

FIGURA 5.2 – Recurso "FulanoDeTal.htm"

Neste caso o SiRPAC atribuirá um identificador para o recurso concatenando o endereço local onde o recurso está publicado, neste caso ("*http://ain/exemplos/*") + o recurso ("*FulanoDeTal.htm*") + identificador gerado para o primeiro recurso anônimo ("*#genid1*") resultando no URI "*http://ain/exemplos/FulanoDeTal.htm#genid1*".

A atribuição automática de identificadores de recursos pelo *parser* pode causar problemas na identidade do recurso. Observe um arquivo texto nomeado como "*Metadata.rdf*" contendo uma descrição RDF sobre o recurso "*FulanoDeTal.htm*", neste caso por se tratar de uma descrição não embutida em "*FulanoDeTal.htm*" o atributo *about* não poderia ser omitido. (Figura 5.3).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description about="http://ain/exemplos/FulanoDeTal.html">
      <s:Criador>
          <rdf:Description>
              <s:nome>Fulano de Tal</s:nome>
              <s:email>Fulano@ain.com.br</s:email>
          </rdf:Description>
      </s:Criador>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.3 – Arquivo "*Metadata.rdf*"

Nesse caso o identificador do recurso resultante do SiRPAC seria exatamente como descrito no atributo *about,* ou seja, "http://ain/exemplos/FulanoDeTal.htm" diferentemente do URI anterior "http://ain/exemplos/FulanoDeTal.htm#genid1", mas ambos referindo-se ao mesmo recurso. Essa diferença significa tratamento diferente no modo de manter e normalizar os dados nas tabelas dependendo da abordagem relacional.

Esse trabalho assume a posição de que a omissão do atributo *about* deve ser usada com cautela, principalmente quando esses modelos estiverem disponíveis para os robôs da rede e conseqüentemente, não se sabe como serão tratados futuramente por outros aplicativos.

5.1 O modelo básico RDF

O modelo básico apresenta a forma mais simples de codificação RDF/XML na forma de recursos, propriedades e objetos onde variações na codificação do modelo podem ocorrer. Verifica-se a seguir o resultado do SiRPAC para propriedades cujo valor são entidades estruturadas. Em RDF tal entidade (objeto) é representada como outro recurso. Duas formas de codificação para o mesmo modelo são apresentadas: a primeira apresenta a entidade estruturada como um objeto anônimo. A segunda apresenta o recurso como um objeto identificado (figuras 5.4 e 5.5 respectivamente). Segundo **[W3C 99]** a escolha do identificador pode ser feita com princípios de modelagem de dados.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description about="http://ain/testes/FulanoDeTal.html">
        <s:Criador>
            <rdf:Description>
                <s:nome>Fulano de Tal</s:nome>
                <s:email>Fulano@ain.com.br</s:email>
            </rdf:Description>
        </s:Criador>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.4 – Entidade estruturada como objeto anônimo

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description about="http://ain/testes/FulanoDeTal.html">
        <s:Criador rdf:resource="#ch291" />
</rdf:Description>
        <rdf:Description ID="ch291">
            <s:nome>Fulano de Tal</s:nome>
            <s:email>Fulano@ain.com.br</s:email>
        </rdf:Description>
</rdf:RDF>
```

FIGURA 5.5 – Entidade estruturada como objeto identificado

Nota-se no segundo modelo, que a entidade estruturada “Criador” refere-se ao recurso nomeado como “#ch291”. Isso difere do primeiro modelo onde a entidade “Criador” não possui uma identificação, ou seja, ela é anônima, referenciada apenas pelo aninhamento de marcações.

Após a submissão do primeiro modelo ao SiRPAC um identificador exclusivo para a entidade anônima foi automaticamente gerado. A garantia da exclusividade deste identificador se dá com a união de um “*contador*”, na forma de *(“genid2”, “genid3”,..,“genidN”*) mais o URI do arquivo que contém a própria descrição RDF que é representado pelo arquivo nomeado como “*basico1.xml*”.

Aqui, o contador é iniciado pelo número dois porque que a primeira *description* anônima é a segunda do modelo. Abaixo, apresenta-se o resultado do SiRPAC para o primeiro e segundo caso, respectivamente. (Figuras 5.6 e 5.7).

```
1) triple("http://ain/testes/FulanoDeTal.html",
     "http://ain/exemplos/EsquemaExemplos#Criador",
     "http://ain/exemplos/basico1.xml#genid2")

2) triple("http://ain/exemplos/basico1.xml#genid2",
     "http://ain/exemplos/EsquemaExemplos#nome",
     literal("Fulano de Tal"))

3) triple("http://ain/exemplos/basico1.xml#genid2",
     "http://ain/exemplos/EsquemaExemplos#email",
     literal("Fulano@ain.com.br"))
```

FIGURA 5.6 – Identificador gerado pelo SiRPAC para entidade anônima.

O segundo modelo codificado utiliza um identificador gerado com conceitos de modelagem de dados (chaves). O SiRPAC utiliza este como identificador interno entre as declarações resultantes.

```
1) triple("http://ain/testes/FulanoDeTal.html",
     "http://ain/exemplos/EsquemaExemplos# Criador",
     "http://ain/exemplos/basico2.xml#ch291")

2) triple("http://ain/exemplos/basico2.xml#ch291",
     "http://ain/exemplos/EsquemaExemplos#nome",
      literal("Fulano de Tal"))

3) triple("http://ain/exemplos/basico2.xml#ch291",
     "http://ain/exemplos/EsquemaExemplos#email",
     literal("Fulano@ain.com.br"))
```

FIGURA 5.7 – Identificador gerado pelo SiRPAC para entidade identificada

5.2 Resultados sobre containers

Containers são utilizados para se referir a uma coleção de recursos, para dizer que um trabalho foi criado por mais de uma pessoa ou para listar os estudantes de um curso. Por se tratar de modelos semelhantes nos três tipos de *containers*, apresenta-se o resultado do *parser* apenas para o tipo *Bag*.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description about="http://mycollege.edu/courses/6.001">
        <s:estudantes>
                <rdf:Bag>
                <rdf:li resource="http://mycollege.edu/students/Amy" />
                <rdf:li resource="http://mycollege.edu/students/Tim" />
                <rdf:li resource="http://mycollege.edu/students/John" />
                <rdf:li resource="http://mycollege.edu/students/Mary" />
                <rdf:li resource="http://mycollege.edu/students/Sue" />
                </rdf:Bag>
        </s:estudantes>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.8 – *Container* "Estudantes"

Este exemplo apresenta uma coleção de recursos que são estudantes de um curso. Para *containers* o SiRPAC atribui um identificador exclusivo para o recurso que representa a coleção "estudantes" e os recursos que são membros da coleção são expressos com um conjunto de propriedades especialmente definidas para este propósito. Como especificado em **[W3C 99]** o *parser* RDF nomeia as propriedades membros como [*"_1", "_2",.., "_N"*] e assim por diante. (Figura 5.9).

```
1) triple("http://mycollege.edu/courses/6.001",
        "http://ain/exemplos/EsquemaExemplos#estudantes",
        "http://ain/exemplos/container1.xml#genid2")

2) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag")

3) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_1",
        "http://mycollege.edu/students/Amy")

4) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_2",
        "http://mycollege.edu/students/Tim")

5) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_3",
        "http://mycollege.edu/students/John")

6) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_4",
        "http://mycollege.edu/students/Mary")

7) triple("http://ain/exemplos/container1.xml#genid2",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_5",
        "http://mycollege.edu/students/Sue")
```

FIGURA 5.9 – Resultado do SiRPAC para *container* do tipo *Bag*

5.3 Declarações sobre membros de um container

Entre os detalhes da elaboração do modelo das declarações, sobre os membros de um *container,* dois fatos devem ser considerados: O primeiro é quanto às declarações feitas sobre o próprio *container* e o segundo é sobre as declarações feitas sobre os membros do *container*.

Para esclarecer esta diferença, elaborou-se um modelo para cada situação:

O primeiro modelo mostra que "Ora Lassila" é o criador do *container* "pages" que é uma *Bag* de dois outros recursos, entretanto, não diz coisa alguma sobre os membros da *Bag.* (Figura 5.10 e 5.11 respectivamente).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Bag ID="pages">
        <rdf:li resource="http://foo.org/foo.html" />
        <rdf:li resource="http://bar.org/bar.html" />
</rdf:Bag>
<rdf:Description about="#pages">
        <s:Criador>Ora Lassila</s:Criador>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.10 – Declarações sobre o *container* "#Pages"

```
1) triple("http://ain/exemplos/container2.xml#pages",
         "http://ain/exemplos/EsquemaExemplos#Criador",
         literal("Ora Lassila"))

2) triple("http://ain/exemplos/container2.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag")

3) triple("http://ain/exemplos/container2.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#_1",
         "http://foo.org/foo.html")

4) triple("http://ain/exemplos/container2.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#_2",
         "http://bar.org/bar.html")
```

FIGURA 5.11 – Resultado do SiRPAC para declarações do *container* "#Pages"

Se "Ora Lassila" é o criador de cada um dos recursos existentes na *Bag*, um modelo ligeiramente diferente deve ser utilizado. O segundo modelo utiliza o atributo *aboutEach* para expressar esse fato.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Bag ID="pages">
        <rdf:li resource="http://foo.org/foo.html" />
        <rdf:li resource="http://bar.org/bar.html" />
</rdf:Bag>
<rdf:Description aboutEach="#pages">
        <s:Criador>Ora Lassila</s:Criador>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.12 – Declarações sobre os membros de um *container*

Essa pequena variação na sintaxe, com relação ao exemplo anterior, causa diferenças significativas no resultado do *parser*, pois a literal "Ora Lassila" é atribuída a cada membro do container como segue (figura 5.13):

```
1) triple("http://ain/exemplos/container3.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag")

2) triple("http://ain/exemplos/container3.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#_1",
         "http://foo.org/foo.html")

3) triple("http://ain/exemplos/container3.xml#pages",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#_2",
         "http://bar.org/bar.html")

4) triple("http://foo.org/foo.html",
         "http://ain/exemplos/EsquemaExemplos#Criador",
         literal("Ora Lassila"))

5) triple("http://bar.org/bar.html",
         "http://ain/exemplos/EsquemaExemplos#Criador",
         literal("Ora Lassila"))
```

FIGURA 5.13 – Resultados do SiRPAC para membros de um *container*

5.4 Containers definidos por um padrão de URI

Se for necessária a criação de declarações sobre todas as páginas de um *Web site*, pode-se empregar o atributo *aboutEachPrefix* para atribuir uma propriedade e seu correspondente valor a todas as páginas cujo URI apresenta o mesmo padrão. (Figura 5.14).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description aboutEachPrefix="http://ain/exemplos">
        <s:Criador>Ora Lassila</s:Criador>
</rdf:Description>
<rdf:Description about="http://ain/exemplos/page1">
        <s:titulo>Avaliacao da Erosao do Solo</s:titulo>
        <s:data>20/04/2001</s:data>
</rdf:Description>
<rdf:Description about="http://ain/exemplos/page2">
        <s:titulo>Manejo do Solo e Agua</s:titulo>
        <s:data>18/03/2001</s:data>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.14 – Emprego do atributo *aboutEachPrefix*

O modelo acima apresenta dois recursos “page1” e “page2” localizados no URI “http://ain/exemplos”. Isso faz com que o SiRPAC atribua o criador “Ora Lassila” aos dois recursos, gerando um conjunto de declarações. (Figura 5.15).

```
1) triple("http://ain/exemplos/page1",
        "http://ain/exemplos/EsquemaExemplos#Criador",
        literal("Ora Lassila"))

2) triple("http://ain/exemplos/page1",
        "http://ain/exemplos/EsquemaExemplos#titulo",
        literal("Avaliacao da Erosao do Solo"))

3) triple("http://ain/exemplos/page1",
        "http://ain/exemplos/EsquemaExemplos#data",
        literal("20/04/2001"))

4) triple("http://ain/exemplos/page2",
        "http://ain/exemplos/EsquemaExemplos#Criador",
        literal("Ora Lassila"))

5) triple("http://ain/exemplos/page2",
        "http://ain/exemplos/EsquemaExemplos#titulo",
        literal("Manejo do Solo e Agua"))

6) triple("http://ain/exemplos/page2",
        "http://ain/exemplos/EsquemaExemplos#data",
        literal("18/03/2001"))
```

FIGURA 5.15 – Resultado do SiRPAC para o atributo *aboutEachPrefix*

5.5 Resultados do parser de declarações sobre declarações

Segundo modelo formal do RDF em **[W3C 99]** para uma declaração ser utilizada como objeto de outra, ela deverá ser modelada com algumas propriedades adicionais. O modelo da figura 4.5.1 apresenta o fato de que “Fulano de Tal” disse que “Beltrano de Tal” é o criador do recurso *http://ain/exemplos/page3.htm*.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description>
       <rdf:subject resource="http://ain/exemplos/page3.htm" />
       <rdf:predicate resource="http://ain/exemplos/EsquemaExemplos#Criador"/>
       <rdf:object>Beltrano de Tal</rdf:object>
       <rdf:type resource="http://www.w3.org/1999/02/22-rdf-syntax-
       ns#Statement"/>
       <s:atribuidoPara>Fulano de Tal</s:atribuidoPara>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.16 – Modelo de declaração RDF

O SiRPAC atribui um identificador único para recurso que representa o fato e usa esse identificador em uma afirmação referenciada pela quinta declaração. (Figura 5.17). A quinta declaração do conjunto resultante, é dita afirmada porque as demais são consideradas declarações *reified* no modelo, nome usado pela comunidade de representação de conhecimento para expressar o processo de criar declarações sobre declarações.

```
1) triple("http://ain/exemplos/container5.xml#genid1",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#subject",
         "http://ain/exemplos/page3.htm")

2) triple("http://ain/exemplos/container5.xml#genid1",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate",
         "http://ain/exemplos/EsquemaExemplos#/Criador")

3) triple("http://ain/exemplos/container5.xml#genid1",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#object",
         literal("Beltrano de Tal"))

4) triple("http://ain/exemplos/container5.xml#genid1",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
         "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement")

5) triple("http://ain/exemplos/container5.xml#genid1",
         "http://ain/exemplos/EsquemaExemplos#atribuidoPara",
         literal("Fulano de Tal"))
```

FIGURA 5.17 – Resultado do SiRPAC para modelo de declaração

Como especificado em **[W3C 99]**, a declaração original foi modelada com quatro propriedades, o *subject*, *predicate*, *object* e *type* podendo esta declaração ser utilizada como objeto de outra. Isso é notado na última declaração resultante em que o recurso identificando a declaração original modelada *"genid1"*, recebe a propriedade "*atribuidoPara*" com o valor "Fulano de Tal".

5.6 Valores compartilhados

Um simples recurso pode ser objeto de mais do que uma declaração. Dois *containers* do tipo *Seq* foram utilizados para representar uma seqüência de trabalhos por data e assunto respectivamente. (Figura 5.18).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<Seq ID="trabalhosPorData">
        <li resource="http://www.dogworld.com/Aug96.doc" />
        <li resource="http://www.webnuts.net/Jan97.html" />
        <li resource="http://www.carchat.com/Sept97.html" />
</Seq>
<Seq ID="trabalhosPorAssunto">
        <li resource="http://www.carchat.com/Sept97.html" />
        <li resource="http://www.dogworld.com/Aug96.doc" />
        <li resource="http://www.webnuts.net/Jan97.html" />
</Seq>
</RDF>
```

FIGURA 5.18 – Um recurso como objeto de duas declarações.

O SiRPAC apresenta o conjunto resultante dos recursos com a correspondente propriedade nomeada [“_1”, “_2”, .. ,“_N”]. Nota-se que um mesmo recurso pode ter propriedades diferentes dependendo de sua seqüência de ordenação.

```
1) triple("http://ain/exemplos/container6.xml#trabalhosPorData",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq")

2) triple("http://ain/exemplos/container6.xml#trabalhosPorData",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_1",
        "http://www.dogworld.com/Aug96.doc")

3) triple("http://ain/exemplos/container6.xml#trabalhosPorData",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_2",
        "http://www.webnuts.net/Jan97.html")

4) triple("http://ain/exemplos/container6.xml#trabalhosPorData",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_3",
        "http://www.carchat.com/Sept97.html")

5) triple("http://ain/exemplos/container6.xml#trabalhosPorAssunto",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq")

6) triple("http://ain/exemplos/container6.xml#trabalhosPorAssunto",
        http://www.w3.org/1999/02/22-rdf-syntax-ns#_1",
        "http://www.carchat.com/Sept97.html")

7) triple("http://ain/exemplos/container6.xml#trabalhosPorAssunto",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_2",
        "http://www.dogworld.com/Aug96.doc")

8) triple("http://ain/exemplos/container6.xml#trabalhosPorAssunto",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#_3",
        "http://www.webnuts.net/Jan97.html")
```

FIGURA 5.19 – Resultado com propriedades diferentes para o mesmo recurso

5.7 Relações não binárias

O modelo de dados RDF somente suporta relações binárias. Uma técnica recomendada para representar relações não binárias é usar um recurso intermediário com propriedades adicionais indicando as relações restantes.

O modelo abaixo apresenta um uso comum desta capacidade, onde uma pessoa tem o valor de sua propriedade "peso" não somente como um número, mas acompanhado de uma unidade de medida. (Figura 5.20).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description ID="Fulano de Tal">
<s:peso rdf:parseType="Resource">
         <rdf:value>80</rdf:value>
         <s:unidade rdf:resource="http://ain/exemplos/EsquemaExemplos#Kg"/>
</s:peso>
</rdf:Description>
</rdf:RDF>
```

FIGURA 5.20 – Modelo RDF para relações não binárias

A propriedade "peso" na sintaxe RDF/XML foi modelada como um recurso tipo *rdf:parseType* e isso permite a utilização de propriedades/valores adicionais para a propriedade "peso". O valor principal dessa propriedade está indicado por *rdf:value*.

O resultado do SiRPAC para esse tipo de relação é apresentado como segue (figura 5.21):

```
1) triple("http://ain/exemplos/container7.xml#genid2",
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#value",
          literal("80"))

2) triple("http://ain/exemplos/container7.xml#genid2",
          "http://ain/exemplos/EsquemaExemplos#unidade",
          "http://ain/exemplos/EsquemaExemplos#Kg")

3) triple("http://ain/exemplos/container7.xml#Fulano de Tal",
          "http://ain/exemplos/EsquemaExemplos#peso",
          "http://ain/exemplos/container7.xml#genid2")
```

FIGURA 5.21 – Resultado do SiRPAC para relações não binárias

6 Distribuição e amostragem dos dados para avaliação

Utilizou-se para avaliação modelos e esquemas RDF de diferentes fontes, entre eles, os modelos analisados no capítulo quatro, modelos de exemplos disponíveis nas especificações oficiais do W3C, modelos gerados por algoritmos sobre uma base de dados, esquemas RDFS gerados com ferramentas como o Protégé-2000 **[STA 2000]** e modelos de esquema desenvolvidos por comunidades independentes citadas no primeiro capítulo. Os modelos gerados por algoritmos seguiram os mesmos princípios daqueles apresentados no capítulo três.

Os dados para testes são declarações resultantes do processo de *parser* sobre esse conjunto de modelos que, reunidos, somam 3.000 declarações.

As declarações estão distribuídas em 451 modelos RDF e 10 modelos RDFS. (Tabela 6.1 e 6.2 respectivamente).

TABELA 6.1 – Distribuição das declarações nos modelos

Nº de modelos	**Descrição**	**Nº de declarações**
4	Baseados no esquema DMP	66
210	Baseados no esquema programas/projetos	628
13	Baseados em exemplos oficiais	88
90	Baseados no esquema funcionários	540
64	Baseados no esquema alunos/pessoas	384
70	Baseados no esquema professores/pessoas	733
Total: 451		Total: 2.439

TABELA 6.2 – Distribuição das declarações nos esquemas

Nº de modelos	**Descrição**	**Nº de declarações**
1	DMP - dedicação mensal de pesquisador	77
1	Programas de Pesquisa	24
1	Dublin Core Qualifiers	80
1	Funcionários	24
1	Pessoas	48
1	Technical	36
1	EOR - extensible open rdf	40
1	Dublin Core	81
1	RDFS	106
1	RDF	45
Total: 10		Total...: 561

Como especificado no modelo formal em **[W3C 99]**, uma declaração possui três partes: assunto, predicado e objeto. Assunto e predicado são considerados recursos e objeto pode ser um recurso ou literal. Desse modo, as 3.000 declarações somam 9.000 partes, das quais 1.997 são literais e 7.003 são recursos.

Do total de literais aproximadamente 70% são exclusivas sendo os restantes literais duplicadas. Já do total de recursos apenas 12% são valores exclusivos e 88% são recursos duplicados.

A alta taxa de recursos duplicados se deve aos identificadores internos gerados pelo *parser* para manter o relacionamento entre as declarações de um mesmo modelo,

conforme demonstrado no capítulo 5. A tabela 6.3 apresenta a amostragem detalhada do conjunto de dados.

TABELA 6.3 – Amostragem detalhada do conjunto de dados

9.000 partes	**Valores exclusivos**	**Valores duplicados**	**Total**
Literais	1.396	601	1.997
Recursos	857	6.146	7.003
Declarações	2.986	14	3.000
Modelos	461	0	461
Namespaces	102	261	363

7 Abordagens existentes de armazenamento de RDF em banco de dados relacionais

Este capítulo tem como objetivo apresentar abordagens existentes para armazenamento de RDF em banco de dados relacionais. As abordagens apresentadas são às de: Brian McBride, Sergey Melnik, Eric Miller e Jonas Liljegren.

7.1 Abordagem de Brian McBride

A abordagem relacional de Brian McBride em **[MEL 2000]** possui seis tabelas: rdfresource, rdfnamespace, rdfliteral, rdfstatement, rdfmodel e rdfkeys. (Figura 7.1).

```
sql = "CREATE TABLE RDFRESOURCE"
            + "("
            + "Id INTEGER not null primary key,"
            + "NS INTEGER not null,"
            + "RoName varchar(255)" + ")";

sql = "CREATE TABLE RDFNameSpace"
            + "("
            + "Id INTEGER not null primary key,"
            + "NsName varchar(255)" + ")";

sql = "CREATE TABLE RDFLiteral"
            + "("
            + "Id INTEGER not null primary key,"
            + "VAL varchar (4000)" + ")";

sql = "CREATE TABLE RDFStatement"
            + "("
            + "Id INTEGER not null primary key,"
            + "Subject INTEGER not null,"
            + "Predicate INTEGER not null,"
            + "ObjResource INTEGER not null,"
            + "ObjLiteral INTEGER not null,"
            + "Res CHAR(1) not null" + ")";

sql = "CREATE TABLE RDFModel"
            + "("
            + "ModelId INTEGER not null,"
            + "Statement INTEGER not null,"
            + "Asserted CHAR(1) not null,"
            + "Reified CHAR(1) not null,"
            + "primary key(ModelId, Statement)" + ")";

sql = "CREATE TABLE RDFKEYS"
            + "("
            + "TableName char(10) not null primary key,"
            + "Key INTEGER not null" + ")";
```

FIGURA 7.1 - Modelo relacional de Brian McBride
Fonte: **[MEL 2000]** - *Storing RDF in a relational database*

A tabela de recursos “rdfresource” mantém todos os recursos. *id* é a chave primária da tabela. *ns* é a chave estrangeira da tabela “namespace*”*, referenciando o *namespace* para aquele recurso. *RoName* é o componente “*name*” de *Qname*. Segundo a sintaxe básica (*Basic Serialization Syntax*) em **[W3C 99]**, *Qname* é composto do nome de propriedade “*name*” mais o opcional prefixo de *namespace*. “[Nsprefix ‘ : ‘]”.

A tabela de *namespace* “rdfnamespace” tem um atributo *id* que é a chave primária para esta tabela e *NsName* é o *namespace* propriamente dito.

Literais, são colocados na tabela “rdfliteral” a qual tem seu próprio identificador *id* e o correspondente valor *val* do literal.

As declarações de cada recurso são colocadas na tabela “rdfstatement”. Um atributo *id* é a chave primária para essa tabela. *Subject, Predicate* e *ObjResource* são ponteiros para a tabela “rdfresource”. *ObjLiteral* é um ponteiro para a tabela “rdfliteral”. *Res* é um atributo que indica se o objeto é um recurso ou um literal.

Essa abordagem pode manipular múltiplos modelos. A tabela de modelos, “rdfmodel” mantém uma lista das declarações em cada modelo. Segundo Brian em **[MEL 2000]** cada modelo é um recurso e tem uma entrada na tabela de recursos. O atributo *ModelId* é o identificador daquele recurso. Assim declarações podem ser feitas sobre o modelo.

O atributo *asserted* indica se a declaração é fato no modelo e o atributo *reified* indica se a declaração é *reified* no modelo. A tabela “rdfkeys” é utilizada para controlar a seqüência de identificadores das outras tabelas.

Abaixo, seguem dois conjuntos de declarações utilizados para demonstração, o primeiro com declarações afirmadas *asserted* e o segundo com declarações *reified*. (Figura 7.2 e 7.3 respectivamente).

```
1) triple("http://ain/exemplos/basico1.xml#genid2",
     "http://ain/exemplos/EsquemaExemplos#nome",
     literal("Fulano de Tal"))

2) triple("http://ain/exemplos/basico1.xml#genid2",
     "http://ain/exemplos/EsquemaExemplos#email",
     literal("Fulano@ain.com.br"))

3) triple("http://ain/exemplos/FulanoDeTal.htm",
     http://ain/exemplos/EsquemaExemplos#Criador",
     http://ain/exemplos/basico1.xml#genid2")
```

FIGURA 7.2 – Conjunto de declarações *asserted*

Cada parte da declaração é inserida em uma tabela. Se a parte for o *subject* ou o *predicate* da declaração, elas são inseridas na tabela “rdfresource”, mas se a parte da declaração for o *object,* o atributo tipo é que determinará a tabela que irá receber a parte, se “rdfresource” ou “rdfliteral”.

Quando inserido na tabela, os recursos são contemplados com um identificador único baseado no seqüenciador “rdfkeys” e uma chave estrangeira correspondente ao *namespace,* caso exista, do contrário, a chave estrangeira do *namespace* na tabela “*rdfresource*” terá o valor zero. Então todos os recursos cuja chave estrangeira *ns* for zero, se referem ao recurso sendo descrito.

```
1) triple("http://ain/exemplos/reified.htm#rec01",
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#subject",
     "http://ain/exemplos/page3.htm")

2) triple("http://ain/exemplos/reified.htm#rec01",
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate",
     "http://ain/exemplos/EsquemaExemplos#Criador")

3) triple("http://ain/exemplos/reified.htm#rec01",
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#object",
     literal("Fulano de Tal"))

4) triple("http://ain/exemplos/reified.htm#rec01",
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement")

5) triple("http://ain/exemplos/reified.htm#rec01",
     "http://ain/exemplos/EsquemaExemplos#ditoPor",
     literal("Beltrano de Tal"))
```

FIGURA 7.3 – Conjunto de declarações *reified*

O segundo conjunto contém cinco declarações para o mesmo recurso *http://ain/exemplos/reified.htm#rec01*, sendo 4 delas declarações *reified*.

Os dados ficam assim dispostos nas tabelas, segundo a abordagem de Brian. (Tabelas de 7.1 a 7.6).

TABELA 7.1 – Dados dispostos na tabela "rdfresource"

ID	NS	RONAME
1	1	genid2
2	0	http://ain/exemplos/FulanoDeTal.htm
3	2	Nome
4	2	Email
5	2	Criador
6	3	rec01
7	4	subject
8	0	http://ain/exemplos/page3.htm
9	4	predicate
10	4	Object
11	4	type
12	4	Statement
13	2	ditoPor

TABELA 7.2 – Dados dispostos na tabela "rdfnamespace"

ID	NSNAME
1	http://ain/exemplos/basico1.xml#
2	http://ain/exemplos/EsquemaExemplos#
3	http://ain/exemplos/reified.htm#
4	http://www.w3.org/1999/02/22-rdf-syntax-ns#

TABELA 7.3 – Dados dispostos na tabela “rdfstatement”

ID	SUBJECT	PREDICATE	OBJRESOURCE	OBJLITERAL	RES
1	1	3	0	1	L
2	1	4	0	2	L
3	2	5	1	0	R
4	6	7	8	0	R
5	6	9	5	0	R
6	6	10	0	1	L
7	6	11	12	0	R
8	6	13	0	3	L

TABELA 7.4 – Dados dispostos na tabela “rdfliteral”

ID	VAL
1	Fulano de Tal
2	Fulano@ain.com.br
3	Beltrano de Tal

TABELA 7.5 – Dados dispostos na tabela “rdfmodel”

MODELID	STATEMENT	ASSERTED	REIFIED
1	1	S	N
1	2	S	N
1	3	S	N
2	4	N	S
2	5	N	S
2	6	N	S
2	7	N	S
2	8	S	N

TABELA 7.6 – Dados dispostos na tabela “rdfkeys”

TABLENAME	KEY
literal	3
namespace	4
resource	13
statement	8

7.2 Abordagem de Jonas Liljegren

A abordagem proposta por Jonas Liljegren em **[MEL 2000]** possui quatro tabelas. A tabela statement, resources, prefix e sequence (Figura 7.4).

```
CREATE TABLE statements
(
   id              int,
   pred            int,
   subj            int,
   obj             int,
   fact            int    -- 0 if only a reified statement
);

CREATE TABLE resources
(
   id              int,
   uri             text,
   value           text,
   isprefix        int,
   lang            text
);

CREATE TABLE prefix
(
   from            int,
   to              int
);

CREATE SEQUENCE internal_id;
```

FIGURA 7.4 – Modelo relacional de Jonas Liljegren
Fonte: **[MEL 2000]** - *Storing RDF in a relational database*

A tabela de declarações "statement" mantém todas os identificadores dos recursos que formam a declaração "pred", "subj" e "obj". O atributo "fact" difere as declarações *asserted* das declarações *reified* e o atributo "id" é o identificador da declaração. A tabela de recursos "resources" foi elaborada para manter recursos e literais, respectivamente nos atributos "uri" e "value". O atributo "lang" mantém informações sobre o idioma de marcação. Recursos *aboutEachPrefix* são indicados na tabela de recursos pelo atributo "isprefix". A tabela "prefix" é usada para manter uma lista de todos os recursos mapeados para o recurso *aboutEachPrefix*. **[LIL 2000]**.

A tabela "sequence" é usada para manter a seqüência de identificadores. Não existe declaração com o mesmo identificador que um registro na tabela de recursos, eles usam a mesma seqüência. Como a abordagem de Jonas destaca a preocupação com o atributo *aboutEachPrefix*, um conjunto de declarações que faz uso deste artifício será utilizado como exemplo (Figura 7.5).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF                    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://ain/exemplos/EsquemaExemplos#">
<rdf:Description aboutEachPrefix="http://ain/exemplos/">
        <s:Criador>Ora Lassila</s:Criador>
</rdf:Description>
<rdf:Description about="http://ain/exemplos/Projeto1">
        <s:titulo>Avaliacao da Erosao do Solo</s:titulo>
        <s:data>20/04/2001</s:data>
</rdf:Description>
<rdf:Description about="http://ain/exemplos/Projeto2">
        <s:titulo>Manejo do Solo e Agua</s:titulo>
        <s:data>18/03/2001</s:data>
</rdf:Description>
<rdf:Description about="http://ain/exemplos/Projeto3">
        <s:titulo>Densidade do Solo</s:titulo>
        <s:data>10/01/2001</s:data>
</rdf:Description>
</rdf:RDF>
```

FIGURA 7.5 – Modelo RDF com o atributo *aboutEachPrefix*

Pode-se observar no modelo acima três recursos sendo descritos: Projeto1, Projeto2 e Projeto3 e um quarto recurso *http://ain/exemplos* referente ao recurso *aboutEachPrefix* que será marcado na tabela "resource" no atributo isprefix.

A tabela "prefix" mantém um mapeamento através dos identificadores dos três recursos sendo descritos para o identificador do recurso *aboutEachPrefix.* Considere que, se os recursos Projeto1, Projeto2 e Projeto3 possuem respectivamente os *id's* "1", "2" e "3" e o recurso *aboutEachPrefix* o *id* "4" a tabela "prefix" ficaria como segue (Tabela 7.7):

TABELA 7.7 – Dados dispostos na tabela "prefix"

from	to
1	4
2	4
3	4

A (figura 7.6) apresenta o resultado do *parser* para o modelo *aboutEachPrefix* da (figura 7.5).

```
1) triple("http://ain/exemplos/Projeto1",
          http://ain/exemplos/EsquemaExemplos#Criador",
          literal("Ora Lassila"))

2) triple("http://ain/exemplos/Projeto2",
          "http://ain/exemplos/EsquemaExemplos#Criador",
          literal("Ora Lassila"))

3) triple("http://ain/exemplos/Projeto3",
          "http://ain/exemplos/EsquemaExemplos#Criador",
          literal("Ora Lassila"))

4) triple("http://ain/exemplos/Projeto1",
          "http://ain/exemplos/EsquemaExemplos#titulo",
          literal("Avaliacao da Erosao do Solo"))

5) triple("http://ain/exemplos/Projeto1",
          "http://ain/exemplos/EsquemaExemplos#data",
          literal("20/04/2001"))

6) triple("http://ain/exemplos/Projeto2",
          "http://ain/exemplos/EsquemaExemplos#titulo",
          literal("Manejo do Solo e Agua"))

7) triple("http://ain/exemplos/Projeto2",
          http://ain/exemplos/EsquemaExemplos#data",
          literal("18/03/2001"))

8) triple("http://ain/exemplos/Projeto3",
          "http://ain/exemplos/EsquemaExemplos#titulo",
          literal("Densidade do Solo"))

9) triple("http://ain/exemplos/Projeto3",
          "http://ain/exemplos/EsquemaExemplos#data",
          literal("10/01/2001"))
```

FIGURA 7.6 – Resultado do *parser* para o modelo *aboutEachPrefix*

As declarações um, dois e três mostram como o *parser* resolveu o atributo *aboutEachPrefix*. A declaração "**Criador = Ora Lassila**" foi atribuída aos três recursos sendo descritos (figura 7.6). Nota-se no conjunto resultante que não há evidências do uso do atributo *aboutEachPrefix.*

As outras tabelas do modelo proposto por Jonas são apresentadas a seguir.

TABELA 7.8 – Dados dispostos na tabela “resources”

id	uri	value	isprefix	lang
2	http://ain/exemplos/Projeto1		0	
3	http://ain/exemplos/EsquemaExemplos#Criador		0	
4		Ora Lassila	0	
6	http://ain/exemplos/Projeto2		0	
7		Ora Lassila	0	
9	http://ain/exemplos/Projeto3		0	
10		Ora Lassila	0	
12	http://ain/exemplos/EsquemaExemplos#titulo		0	
13		Avaliacao da Erosao do Solo	0	
15	http://ain/exemplos/EsquemaExemplos#data		0	
16		20/04/2001	0	
18		Manejo do Solo e Agua	0	
20		18/03/2001	0	
22		Densidade do Solo	0	
24		10/01/2001	0	

TABELA 7.9 – Dados dispostos na tabela “statements”

id	pred	subj	obj	fact
14	15	2	16	1
11	12	2	13	1
1	3	2	4	1
19	15	6	20	1
17	12	6	18	1
5	3	6	7	1
23	15	9	24	1
21	12	9	22	1
8	3	9	10	1

TABELA 7.10 – Dados dispostos na tabela “sequence”

internal_id
24

7.3 Abordagem de Sergey Melnik

A abordagem proposta por Sergey Melnik em **[MEL 2000]** é bastante semelhante à abordagem de Brian McBride. A abordagem contém cinco tabelas sendo elas triples, resources, namespaces, literals e models. (Figura 7.7).

```
CREATE TABLE triples(
    model bigint(20) DEFAULT '0' NOT NULL,
    subject bigint(20) DEFAULT '0' NOT NULL,
    predicate bigint(20) DEFAULT '0' NOT NULL,
    object bigint(20) DEFAULT '0' NOT NULL,
    objtype tinyint(3) unsigned DEFAULT '0' NOT NULL,
    KEYidx_subject_predicate (subject,predicate),
    KEY idx_model (model),
    KEY idx_object_predicate (object,predicate)
);

CREATE TABLE resources (
    hash bigint(20) DEFAULT '0' NOT NULL,
    ns bigint(20) DEFAULT '0' NOT NULL,
    name varchar(254) DEFAULT '' NOT NULL,
    PRIMARY KEY (hash)
);

CREATE TABLE namespaces (
    hash bigint(20) DEFAULT '0' NOT NULL,
    value varchar(254) DEFAULT '' NOT NULL,
    PRIMARY KEY (hash)
);

CREATE TABLE literals (
    hash bigint(20) DEFAULT '0' NOT NULL,
    value longtext NOT NULL,
    PRIMARY KEY (hash)
);

CREATE TABLE models (
    id bigint(20) DEFAULT '0' NOT NULL,
    uri varchar(254) DEFAULT '' NOT NULL,
    size int(11),
    PRIMARY KEY (id)
);
```

FIGURA 7.7 – Esquema relacional proposto por Sergey Melnik
Fonte: **[MEL 2000]** - *Storing RDF in a relational database*

Nesta abordagem a tabela principal “triples” armazena somente valores *hash* CRC64 (Cyclic Redundancy Code) de recursos e literais. Recursos URI são armazenados na tabela de recursos e literais são armazenados na tabela “literals”.

Recursos apontam para o correspondente *namespace* identificado por valores *hash*. A tabela de modelos mantém o URI de cada modelo e a chave primária mantém a relação com o atributo “model” da tabela triples. Observe o conjunto de declarações seguinte. (Figura 7.8).

```
1) triple("http://ain/exemplos/6011.xml",
        "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
        "http://ain/imagens/EsquemaPessoas#Aluno")

2) triple("http://ain/exemplos/6011.xml",
        "http://ain/imagens/EsquemaPessoas#matricula",
        literal("6011"))

3) triple("http://ain/exemplos/6011.xml",
        "http://ain/imagens/EsquemaPessoas#nome",
        literal("JEAN PAULO REGINATTO SOARES"))

4) triple("http://ain/exemplos/6011.xml",
        "http://ain/imagens/EsquemaPessoas#dataNasc",
        literal("10/05/81"))

5) triple("http://ain/exemplos/6011.xml",
        "http://ain/imagens/EsquemaPessoas#sexo",
        literal("M"))

6) triple("http://ain/exemplos/6011.xml",
        "http://ain/imagens/EsquemaPessoas#curso",
        literal("CC"))
```

FIGURA 7.8 – Conjunto de declarações RDF

Ao inserir as declarações nas tabelas uma função de *hash* é aplicada nos valores a serem armazenados e o resultado dessa função é utilizado na tabela como a chave primária para aquele registro. As tabelas abaixo apresentam os dados do conjunto de declarações processados e dispostos nas tabelas segundo proposta de Melnik. Convém notar que, no exemplo, utilizou-se uma função de *hash* CRC32. **[POW 2000]**.

TABELA 7.11 – Dados dispostos na tabela "resources"

hash	**ns**	**Name**
-1257966291	-1701842601	Aluno
-937936033		http://ain/schemas/6011.xml
-650646587	-1701842601	Curso
344431486	-1701842601	Matricula
1004821554	-1701842601	Nome
1130387884	-1701842601	sexo
1179401978	-1701842601	dataNasc
2118579704	1874517541	type

TABELA 7.12 – Dados dispostos na tabela "namespaces"

Hash	**value**
-1701842601	http://ain/schemas/EsquemaPessoas#
1874517541	http://www.w3.org/1999/02/22-rdf-syntax-ns#

TABELA 7.13 – Dados dispostos na tabela “literal”

Hash	value
75123803	10/05/81
141376813	M
880872684	CC
1722303845	6011
1880797605	JEAN PAULO REGINATTO SOARES

TABELA 7.14 – Dados dispostos na tabela “triples”

model	Subject	predicate	object	objtype
-937936033	-937936033	-650646587	880872684	1
-937936033	-937936033	344431486	1722303845	1
-937936033	-937936033	1004821554	1880797605	1
-937936033	-937936033	1130387884	141376813	1
-937936033	-937936033	1179401978	75123803	1
-937936033	-937936033	2118579704	-1257966291	0

TABELA 7.15 – Dados dispostos na tabela “models”

Id	uri	size
-937936033	http://ain/schemas/6011.xml	0

7.4 Abordagem de Eric Miller

A abordagem proposta por Eric Miller em **[MEL 2000]** é conhecida por abordagem ingênua. Mesmo assim, Miller ressalta que ingênua não significa ruim, e acrescenta que muitas aplicações que necessitam de um armazenamento simples para RDF poderiam utilizar esta solução.

```
TABLE triple(
        property varchar(255),
        resource varchar(255),
        value blob,
        hint char(1));
```

FIGURA 7.9 – Proposta de Eric Miller
Fonte: **[MEL 2000]** - *Storing RDF in a relational database*

A solução de Eric Miller é composta de uma única tabela chamada “triple” (figura 7.9). Esta tabela armazena sem nenhum tratamento todas as declarações resultantes do processo de *parser*. Abaixo se apresenta uma declaração disposta na tabela segundo abordagem proposta por Eric. (Tabela 7.16).

TABELA 7.16 – Dados dispostos na tabela “triple”

Property	resource	value	hint
http://ain/exemplos/EsquemaExemplos#Creator	http://bar.org/bar.html	Ora Lassila	L

8 Avaliação das abordagens

O objetivo desse capítulo é avaliar e comentar as quatro abordagens apresentadas anteriormente. Uma quinta abordagem, proposta por este trabalho, será apresentada em um capítulo à parte, porque os itens aplicados na avaliação das abordagens existentes não são em sua totalidade aplicáveis à nova.

O interesse da avaliação de abordagens para armazenamento de RDF em banco de dados relacionais é analisar estratégias existentes com o objetivo de destacar, contribuir e identificar soluções que possam ser usadas para esse fim.

Considera-se para avaliação, características como: proliferação de relações, informações do modelo RDF que são mantidas, espaço utilizado, tempo de carga dos modelos e custo de consultas SQL.

8.1 Proliferação de tabelas e informações mantidas

8.1.1 Metodologia utilizada

Método comparativo entre os tipos de informações abstraídas do conjunto de declarações e tabelas necessárias para mantê-las.

8.1.2 Desenvolvimento

Um comentário adicional que merece destaque é com relação aos dados de origem da declaração. Por exemplo, conjuntos de declarações resultantes de esquemas apresentam o recurso do qual ele foi originado como valor da propriedade *rdfs:isDefinedBy*. No entanto, se essa propriedade for ignorada no momento de codificação do esquema, a única maneira segura de obter o URI do qual o recurso foi originado é no momento da busca ou aplicação do *parser*. Apenas as abordagens de Brian e Melnik consideram essa informação, porque ambas mantêm uma tabela que guarda todas as declarações de cada modelo.

Outro aspecto importante sobre a abordagem de Melnik, é o uso da função *hash* CRC64 que é uma função utilizada em vários segmentos para prover valores *hash* "exclusivos". No entanto, em testes realizados por **[DIL 98]** a taxa de colisão teórica para o CRC64 e 18.2 milhões de identificadores de mensagens é de uma colisão a cada dois trilhões de identificadores de mensagens. Enquanto a abordagem de Brian faz uso de uma tabela para controlar a seqüência e exclusividade de identificadores, a abordagem de Melnik confia os identificadores a uma função de *hash*.

Quanto à necessidade de tabelas, esse trabalho assume a posição de que a tabela "prefix" utilizada na abordagem de Jonas Liljegren é desnecessária. Segundo a especificação *Containers defined by a uri pattern* em **[W3C 99]** o atributo *aboutEachPrefix* é apenas um artifício de sintaxe que declara que existe uma *Bag* do qual os membros são todos os recursos cujo identificador começa com uma *string* específica. Assim sendo, a declaração na *Description* que tem o atributo *aboutEachPrefix* se aplica individualmente para cada membro desta *Bag* sendo isso resolvido no processo de *parser*. Após o processo de *parser* o recurso *aboutEachPrefix* não pode mais ser identificado no conjunto resultante de declarações, o que impede o mapeamento de identificadores na tabela "prefix". (Figura 7.6).

Apresenta-se a seguir o comparativo sobre os tipos de informações que são abstraídas do conjunto de declarações e as tabelas definidas em cada abordagem para mantê-las.

TABELA 8.1 – Informações abstraídas versus tabelas

Informações/tabelas	**Modelos**	**Declarações**	**Recursos**	**Namespaces**	**Literais**
Modelos	**B,M**				
Declarações		E, **B,M**,J			
Recursos			**B,M**,J		
Namespaces			J	**B,M**	
Literais			J		**B,M**
Reification	**B**	J			

Legenda: B - Brian, J – Jonas, E – Eric, M - Melnik

As colunas indicam as tabelas (entidades) criadas pelas abordagens existentes. As linhas apresentam, no todo, as informações abstraídas do conjunto de declarações. As referências cruzadas indicam as abordagens existentes conforme a legenda.

O número de tabelas existente em cada abordagem pouco varia. Por conta de manter algumas particularidades do modelo algumas abordagens sugerem mais tabelas do que outras. Também todas elas, com exceção da abordagem de Eric Miller, se preocupam em normalizar os dados advindos das declarações.

Como pode ser visto as abordagens de Brian e Melnik se destacam por manter de forma normalizada o maior número de informações pertinentes aos modelos. (Tabela 8.1)

8.2 Tempo de carga das declarações nos modelos relacionais

8.2.1 Metodologia utilizada

Método quantitativo. Implementou-se um algoritmo de carga de declarações e tomada de tempo para cada modelo relacional. Os mesmos softwares, técnica de programação, recursos computacionais e conjunto de dados foram utilizados. Definições de tamanho e tipo de atributos foram padronizados para não influenciar, por suas diferenças, as medições de tempo. (Tabela 8.2).

TABELA 8.2 – Padrões de atributos usados nas abordagens

Abordagens/atributos	**Id's**	**URI's**	**Objetos**
Melnik	Número 8 Bytes	Texto 255 Bytes	Texto 255 Bytes
Eric	-	Texto 255 Bytes	Texto 255 Bytes
Brian	Número 4 Bytes	Texto 255 Bytes	Texto 255 Bytes
Jonas	Número 4 Bytes	Texto 255 Bytes	Texto 255 Bytes

Índices, mesmo que não tenham sido definidos em **[MEL 2000]** se fizeram necessário para alguns casos já que, sem os mesmos, as medidas de uma abordagem em relação à outra seriam sensivelmente prejudicadas. Os índices utilizados pelos algoritmos de carga de declarações em cada abordagem estão apresentados como seguem (tabelas de 8.3 a 8.6):

TABELA 8.3 – Índices utilizados na abordagem de Melnik

Abordagem	**Tabela**	**Nome do Índice**	**Atributo(s)**
Melnik	TRIPLES	PrimaryKey	model
			subject
			predicate
			object
	RESOURCES	PrimaryKey	hash
	NAMESPACE	PrimaryKey	hash
	LITERALS	PrimaryKey	hash
	MODELS	PrimaryKey	id

TABELA 8.4 – Índices utilizados na abordagem de Eric

Abordagem	**Tabela**	**Nome do Índice**	**Atributo(s)**
Eric	TRIPLE	PrimaryKey	property
			resource
			value

TABELA 8.5 - Índices utilizados na abordagem de Brian

Abordagem	**Tabela**	**Nome do Índice**	**Atributo(s)**
Brian	RDFRESOURCE	PrimaryKey	id
		Idx_roname	roname
			ns
	RDFNAMESPACE	PrimaryKey	id
		nsname	nsname
	RDFLITERAL	PrimaryKey	id
		val	val
	RDFSTATEMENT	PrimaryKey	id
		statement	subject
			predicate
			objResource
			objLiteral
	RDFMODEL	PrimaryKey	ModelId
			Statement
	RDFKEYS	PrimaryKey	TableName

TABELA 8.6 – Índices utilizados na abordagem de Jonas

Abordagem	**Tabela**	**Nome do Índice**	**Atributo(s)**
Jonas	STATEMENT	PrimaryKey	id
		statement	pred
			subj
			obj
	RESOURCES	PrimaryKey	id
		uri	uri
		value	value (não exclusivo)
	SEQUENCE	PrimaryKey	internal_id

8.2.2 Desenvolvimento

Todos os algoritmos evitam a entrada de valores duplicados nas tabelas. A fonte dos dados, distribuição e amostragem foram explicadas no capítulo 5.

A (tabela 8.7) apresenta o tempo de carga das declarações para cada abordagem.

TABELA 8.7 – Tempo de carga de declarações em cada abordagem

Abordagem	**Tempo de carga de 3.000 declarações**
Brian	1.15 min.
Jonas	1.10 min.
Melnik	0.45 min.
Eric	**0.01** min.

Vários fatores influenciam o tempo de carga das declarações. A tabela acima destaca o tempo de carga da abordagem de Eric. A abordagem de Eric na verdade é uma exceção às outras porque ela dispensa qualquer tratamento nas declarações antes do armazenamento, por isso o tempo de processamento necessário para abstrair as informações apresentadas na tabela 8.1 não é computado.

Os algoritmos de carga das outras três armazenam as informações conforme a tabela 8.1. Diferentemente da abordagem de Jonas, as de Brian e Melnik apresentam semelhanças tanto no modelo relacional quanto na forma de tratamento necessária para a inclusão dos valores nas tabelas, mas, no entanto, o tempo de carga de Melnik é quase a metade do que o tempo de carga de Brian. Isso acontece devido a técnicas usadas pela abordagem de Melnik que podem minimizar desde o uso de índices de procura até o uso de *join* em instruções SQL.

Observe o *namespace* X="http://ain/schemas/EsquemaPessoas#", agora considere a propriedade Y="curso" do *namespace* X, a técnica usada pela abordagem de Melnik aplica uma função de *hash* sobre o namespace X e inclui o valor correspondente na tabela de *namespace* usando o valor *hash* como a chave primária para esse registro. A mesma coisa é feita com a propriedade Y, que será incluída na tabela *resources*. A tabela *resources* possui a chave estrangeira da tabela *namespace* que mantém o valor *hash* do *namespace* para aquele recurso. A chave primária da tabela de *resources* mantém a função *hash* do par XY concatenado. Sempre que for necessário pesquisar se existe uma propriedade para aquele *namespace* aplica-se a função de *hash* no par concatenado XY e então se realiza uma pesquisa sobre a chave primária da tabela de *resources*, dispensando a pesquisa na tabela *namespace*.

O mesmo não acontece na abordagem de Brian. Quando é necessário pesquisar se existe uma propriedade para determinado *namespace*, duas pesquisas devem ser realizadas: uma para localizar o *namespace* e outra para localizar a propriedade. Por isso a necessidade de mais índices na abordagem de Brian do que na de Melnik.

A ocorrência de pesquisas em maior número, necessárias na abordagem de Brian se torna responsável pela diferença de tempo computada em relação à de Melnik.

8.3 Espaço consumido pelas declarações nos modelos

8.3.1 Metodologia utilizada

Método quantitativo. Obteve-se a quantidade de *bytes* ocupados pelas tabelas e respectivos índices em cada modelo relacional. Atributos que mantêm valores texto conforme (tabela 8.2), mesmo que tenham sido definidos com tamanho de 255 *bytes*, são considerados de tamanho variável. Para atributos numéricos considera-se o número de *bytes* indicados na tabela.

8.3.2 Desenvolvimento

Implementou-se um algoritmo para contagem do espaço consumido (em bytes) para cada tabela de cada abordagem. O algoritmo percorre todas as instâncias de cada tabela e computa para cada atributo do tipo texto apenas a quantidade de bytes ocupados pelo valor. Para valores numéricos considera-se o tamanho em bytes definidos na tabela 8.2 e valores nulos não são computados. Para índices o algoritmo segue o mesmo processo.

A (tabela 8.8) apresenta a comparação do espaço consumido em bytes para cada abordagem.

TABELA 8.8 – Quantidade em bytes consumido em cada abordagem

Abordagem	**Espaço consumido em bytes**
Brian	294.030 bytes
Jonas	**116.498** bytes
Melnik	311.054 bytes
Eric	593.188 bytes

Neste item o destaque ficou por conta da abordagem de Jonas. Uma boa média também ficou evidente na abordagem de Brian, considerando que esta guarda mais informações do modelo do que a de Jonas.

8.4 Custo de consultas SQL

Uma das vantagens de se utilizar banco de dados relacionais é aproveitar as poderosas operações de manipulação de dados.

Em particular, este tópico apresenta informações adicionais aos tópicos anteriores deste capítulo, porque os fatores discutidos até o momento podem influenciar no custo das instruções SQL.

8.4.1 Metodologia utilizada

Método comparativo. Para computar o custo de consultas SQL, optou-se pelo uso e instruções de diagnóstico do banco de dados Oracle 8i.

Antes de verificar o custo de cada instrução SQL todas as tabelas e dados em cada abordagem passaram por um processo padrão do Oracle que computa estatística. Essa estatística é usada pelo otimizador do banco de dados para tomar decisões inteligentes e executar as instruções SQL com o menor custo possível.

Nenhuma interferência quanto a modificação de regras de consultas ou uso de índices foi realizada. Esse processo de avaliação foi igualmente, em cada abordagem, por conta da ferramenta utilizada.

O referido "índice de custo" retornado pela ferramenta foi o valor numérico usado para avaliação.

Como especificado em **[BOB 95]** o valor numérico de uma instrução SQL é uma medida arbitrária, não corresponde a um valor real (por exemplo, milissegundos ou E/S lógica), portanto os custos analisados neste tópico seguem um método comparativo e não quantitativo.

8.4.2 Desenvolvimento

Foram criados arbitrariamente, com o objetivo de selecionar as principais propriedades e recursos dos referidos modelos, quatro tipos de condições para consultas. (Tabela 8.9).

TABELA 8.9 – Tipos de condições selecionadas para medição de custo

1ª	A primeira consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.
2ª	A segunda consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'.
3ª	A terceira consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'.
4ª	A quarta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'.

Para cada tipo de condição, quatro instruções SQL foram elaboradas, sendo uma para cada abordagem. Se um mesmo tipo de consulta for aplicado para abordagens diferentes, elas retornam a mesma informação. Demais detalhes sobre as instruções SQL e comentários são apresentados no ANEXO I deste trabalho.

A (tabela 8.10) apresenta o custo das consultas computados em cada abordagem.

TABELA 8.10 – Índice de Custo computado para cada consulta

Abordagem X Custo	**1º tipo**	**2º tipo**	**3º tipo**	**4º tipo**
Brian	23	49	18	147
Jonas	39	48	26	220
Melnik	**15**	**20**	**20**	**51**
Eric	93	8	3	199

Antes de opinar sobre os resultados do custo das instruções SQL deve-se considerar alguns aspectos como:

1- Forma de codificação dos modelos;
2- Numero necessários de *Join* nas instruções SQL;
3- Tipo de informação mantida pelos modelos relacionais.

Para explicar como estes aspectos podem influenciar sobre o custo das instruções SQL apresentam-se dois casos, o primeiro sobre o modelo aluno e o segundo sobre o modelo professor do capítulo 3. A idéia básica é mostrar que, dependendo da forma de codificação do modelo, uma instrução SQL pode tornar-se mais ou menos complexa quando se pretende obter o mesmo valor de retorno de uma consulta.

Considere que se deseja obter o recurso dado um valor de propriedade em diferentes modelos de codificação. Observe primeiro o modelo aluno. (Figura 8.1).

Assunto	Predicado	Objeto
http://ain/schemas/6057.xml	http://ain/schemas/EsquemaPessoas#matricula	6057
http://ain/schemas/6057.xml	http://ain/schemas/EsquemaPessoas#nome	AIRTON CARLOS THOMAZINI
http://ain/schemas/6057.xml	http://ain/schemas/EsquemaPessoas#dataNasc	26/03/69
http://ain/schemas/6057.xml	http://ain/schemas/EsquemaPessoas#sexo	M
http://ain/schemas/6057.xml	http://ain/schemas/EsquemaPessoas#curso	TPD
http://ain/schemas/6057.xml	http://www.w3.org/1999/02/22-rdf-syntax-ns #type	http://ain/schemas/Esquema Pessoas#Aluno

FIGURA 8.1 – Resultado do *parser* para recurso do modelo aluno

Para recuperar o recurso cujo valor da propriedade curso seja igual a “TPD”, basta que a instrução SQL selecione o assunto onde o predicado seja igual a “curso” e objeto igual a “TPD” (isso no modelo aluno). Agora considere o modelo professor. (figura 8.2).

Assunto	Predicado	Objeto
http://ain/prof20.xml#genid2	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/rdf-syntax-ns#Bag
http://ain/prof20.x ml#genid2 **(5)**	http://www.w3.org/1999/02/22-rdf-syntax-ns#_1	http://ain/prof20.xml#rec1 **(4)**
http://ain/prof20. xml #rec1	http://ain/EsquemaPessoas#instituicao	Universidade Federal de São Carlos
http://ain/prof20. xml #rec1 **(3)**	http://ain/EsquemaPessoas#descricaoCurso **(2)**	Computação **(1)**
http://ain/prof20. xml #rec1	http://ain/EsquemaPessoas#titulo	Especialista
http://ain/PROF20.xml	http://ain/EsquemaPessoas#sexo	M
http://ain/PROF20.xml	http://ain/EsquemaPessoas#nome	Nelson Henrique Zanete
http://ain/PROF20.xml **(7)**	http://ain/EsquemaPessoas#listaDeCursos	http://ain/prof20.xml#genid2 **(6)**
http://ain/PROF20.xml	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ain/EsquemaPessoas# Professor

FIGURA 8.2 – Resultado do *parser* para recurso do modelo professor

Para recuperar o recurso cujo valor da propriedade “descricaoCurso” seja igual a “Computação” é necessário que a instrução SQL selecione o assunto onde o predicado seja igual a “descricaoCurso” e objeto igual a “Computação” e ainda com o resultado recuperado aplicar recursivamente outra instrução que recupera o assunto para este objeto até que se obtenha o recurso raiz do modelo. Observe em negrito a seqüência de informações pesquisadas para encontrar o recurso desejado. A seqüência pode ser entendida da seguinte maneira: Com os valores conhecidos (1) e (2) recupera-se o assunto (3). O assunto (3) é então procurado em objeto, encontrando o (4). De posse do assunto da declaração do objeto (4) que é o assunto (5) procura-se em objeto encontrando (6).

O assunto da declaração do objeto (6) é o recurso desejado (raiz) identificado na seqüência por (7).

Como demonstrado, é necessária uma instrução SQL mais complexa para recuperar um recurso no modelo professor do que no modelo aluno, dado um simples valor de propriedade.

A segunda e terceira consulta (tabela 8.9) desejam obter informações em modelos semelhantes, em nível de codificação, ao modelo aluno. Isso faz com que as

abordagens de Jonas e Eric possam sofrer aumento no custo de instruções SQL se os modelos consultados fossem semelhantes ao modelo professor. Já as abordagens de Brian e Melnik, que usam informações da tabela modelo para recuperar o mesmo tipo de informação, tanto para modelos alunos como para modelos professores o custo das instruções para estas consultas permaneceriam estáveis.

Para destacar a importância da informação de origem das declarações, mantida pela tabela modelo nas abordagens de Brian e Melnik, suponha que o recurso (7) da (figura 8.2) fosse objeto de declaração de outro modelo. A única forma de estabelecer uma condição de término de uma instrução recursiva seria determinada pelos limites do modelo.

Considerando o conjunto de consultas apresentadas, observa-se que a variabilidade do custo de instruções na abordagem de Melnik se mantém menor do que as outras. No quarto tipo de consulta, em que na maioria das abordagens ocorreu alto custo das instruções, a abordagem de Melnik também se destaca. Isso se deve a eliminação de *join* nas instruções SQL por causa do uso da função *hash* já discutido anteriormente.

9 Abordagem proposta

Pode-se definir pelo escopo das abordagens avaliadas anteriormente, que todas elas foram idealizadas para armazenar modelos diversos e de diferentes fontes. Nota-se também que para realizar consultas sobre tais abordagens, faz-se necessário que o usuário memorize o modelo relacional e também os modelos RDF mantidos em cada um deles. Características comuns em todas as abordagens avaliadas dificultam a construção de instruções SQL. Isso se deve ao fato das propriedades RDF estar armazenadas como valores de atributos de tabelas no modelo relacional. Então, sempre que é necessário verificar se uma propriedade possui determinado valor, duas condições devem ser avaliadas pela instrução SQL, uma para a propriedade e outra para o valor.

O objetivo principal da abordagem proposta é prover um mecanismo que processe os modelos RDF mantidos no repositório de forma que as propriedades do modelo RDF se tornen atributos de tabelas do banco de dados e os valores de objetos RDF se tornen valores de atributos das tabelas do banco de dados.

Estes objetivos foram conseguidos através de um algoritmo aplicado sobre um repositório (modelo relacional) definido exclusivamente para esta proposta. Convém notar que o modelo relacional sobre o qual o algoritmo atua, não pretende em nenhum momento servir de comparação com os outros modelos. Nada impede ao nível de lógica de programação, que este algoritmo possa ser implementado usando as abordagens de Brian ou de Melnik.

O modelo relacional (repositório) usado por esta proposta consiste em duas tabelas: "recurso" e "triples". A tabela "recurso" guarda todos os recursos sendo descritos pelos modelos RDF e a tabela "triples" mantém todas as declarações de cada recurso. O conjunto de declarações resultante de cada modelo é contemplado com o identificador exclusivo do recurso relacionado. Nesta abordagem a tabela de recurso faz papel equivalente à tabela de modelo das abordagens de Brian e Melnik.

A seguir apresentam-se os dados dispostos na tabela "triples" e "recurso" respectivamente. (Tabelas 9.1 e 9.2).

TABELA 9.1 – Tabela "triples" do modelo proposto

Orig.	Assunto	Predicado	Objeto
142	http://ain/schemas/6097.xml	http://www.w3.org/1999/02/22-rdf-syntax-ns #type	http://ain/schemas/Esquema Pessoas#Aluno
142	http://ain/schemas/6097.xml	http://ain/schemas/EsquemaPessoas#matricula	6097
142	http://ain/schemas/6097.xml	http://ain/schemas/EsquemaPessoas#nome	MELISA ALVES FORISTIERI
142	http://ain/schemas/6097.xml	http://ain/schemas/EsquemaPessoas#dataNasc	24/04/80
142	http://ain/schemas/6097.xml	http://ain/schemas/EsquemaPessoas#sexo	F
142	http://ain/schemas/6097.xml	http://ain/schemas/EsquemaPessoas#curso	TPD

TABELA 9.2 – Tabela "recurso" do modelo proposto

id	Uri
142	http://ain/schemas/6097.xml

Segundo **[W3C 99]** a representação do modelo de dados é usada para avaliar equivalência de significado. Duas expressões RDF são equivalentes se e somente se suas representações de modelo de dados forem à mesma. Como em RDF, cada predicado usado em uma declaração deve ser identificado com exatamente um *namespace* optou-se pela criação de um mecanismo (protótipo) onde o usuário possa criar suas próprias estruturas de tabelas relacionando cada atributo ao par *namespace#propriedade* correspondente.

O protótipo, após criar as tabelas definidas pelo usuário, percorre as declarações RDF do repositório populando as tabelas com os dados dos modelos RDF. Se o par *namespace#propriedade* definido para cada atributo da tabela do modelo relacional for igual ao predicado da declaração do modelo RDF, o valor do atributo da tabela será o valor do objeto da declaração.

Como a estrutura definida pelo usuário será o alvo de suas consultas, pode-se pensar no modelo relacional usado pelo repositório como uma caixa preta o que dispensa o usuário de memoriza-lo no momento de construir uma consulta. A tabela 9.3 apresenta três instruções SQL com a seguinte condição: Selecione 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.

TABELA 9.3 Comparativo de instruções SQL

Abordagem	Instrução SQL	Custo
Brian	select r.roname, concat(ns1.nsname, r1.roname) as pred, lit.val from rdfresource r, rdfnamespace ns1, rdfmodel m, rdfresource r1, rdfstatement st, rdfliteral lit where r.id=m.modelid and m.statement=st.id and st.predicate=r1.id and r1.ns=ns1.id and st.objliteral=lit.id and lit.val='Doutor' and ns1.nsname= 'http://ain/schemas/EsquemaPessoas#' and r1.roname= 'titulo';	23
Melnik	select m.uri, concat(ns1.value, r1.name) as pred, lit.value from resources r1, namespaces ns1, models m, literals lit, triples t where m.id=t.model and t.predicate=r1.hash and r1.ns=ns1.hash and t.object=lit.hash and t.predicate= -2016262896 and t.object= -589537348;	15
Zanete	select recurso, titulo from titulo where titulo='Doutor';	6

Observa-se que a terceira instrução SQL ficou mais simplificada porque a propriedade ‘titulo’ que era o valor do predicado da declaração RDF agora é um atributo da tabela Titulo.

Um aspecto importante que deve ser considerado é quanto ao índice de custo computado na última instrução. Este valor é real, mas foi computado após a aplicação do algoritmo de equivalência. Portando a diminuição no índice de custo é uma conseqüência da simplicidade da instrução SQL e não considera o esforço computacional da execução do algoritmo. Essa proposta não mediu o custo de criação de tabelas, muito menos a complexidade e tempo para popular às tabelas de dados definidas pelo usuário.

Assim sendo, essa proposta assume que qualquer esforço computacional que possa ser empregado para simplificar o modelo relacional deve ser usado no sentido de favorecer para o usuário a criação de consultas mais simples. Além disso, esses processos poderiam ser executados em segundo plano com o auxílio de ferramentas elaboradas para esse propósito.

A (tabela 9.4) mostra o índice de custo de instruções SQL das abordagens existentes em comparação com a proposta, após o processamento do algoritmo de equivalência.

TABELA 9.4 – Índice de custo após processamento do algoritmo de equivalência

Abordagem X Custo	**1º tipo**	**2º tipo**	**3º tipo**	**4º tipo**
Brian	23	49	18	147
Jonas	39	48	26	220
Melnik	15	20	20	51
Eric	93	8	3	199
Zanete	6	33 *	4	10

Observa-se que a abordagem proposta apresenta melhores índices de custo de instrução SQL em relação às outras, porém com algumas limitações.

Como a abordagem proposta é baseada em equivalência de modelos, faz se necessário à elaboração prévia da estrutura do usuário para que o processo possa verificá-la em relação aos modelos RDF que estão armazenados. Por exemplo: o segundo tipo de consulta deseja recuperar todas as propriedades de um recurso. O fato é que a abordagem proposta, parte do pressuposto que se conheça ao menos uma propriedade do recurso que se deseja recuperar, por isso o segundo tipo de consulta usada para avaliação das abordagens existentes não se aplica a esta.

Este trabalho considera aceitável essa limitação, porque nada impede que as consultas possam ser realizadas sobre o modelo relacional do repositório da abordagem proposta e, além disso, se um recurso já é conhecido, suas propriedades podem ser obtidas diretamente da origem. O segundo tipo de consulta aplicado sobre o modelo relacional do repositório da abordagem proposta, resulta em custo 33 conforme destacado. (Tabela 9.4).

A (figura 9.1) apresenta um gráfico completo comparando as abordagens existentes com a proposta, após o processamento do algoritmo de equivalência.

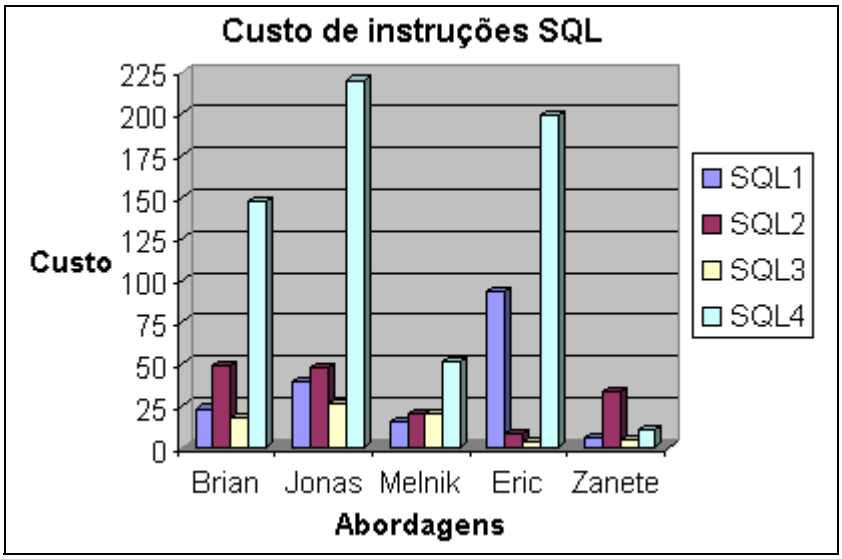

FIGURA 9.1 – Custo após processamento do algoritmo de equivalência

9.1 Apresentação do protótipo

Para aplicar os conceitos da abordagem proposta e demonstrar didaticamente os resultados obtidos, desenvolveu-se um protótipo sobre o banco de dados Microsoft Access 2000. A (figura 9.2) apresenta o esquema de funcionamento do protótipo.

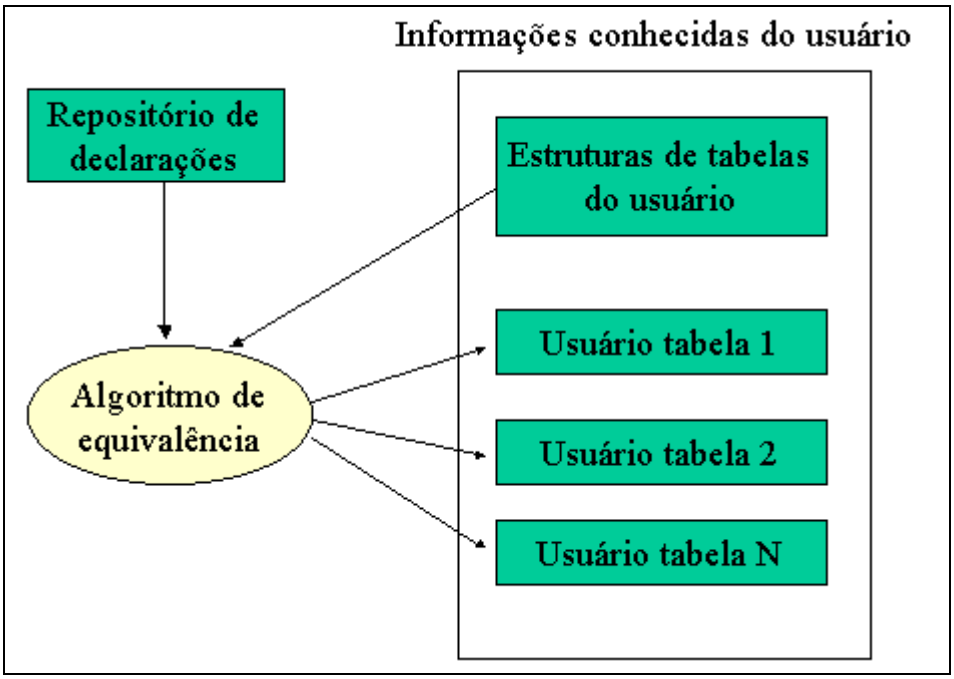

FIGURA 9.2 – Esquema de funcionamento do protótipo

O protótipo possui uma interface onde o usuário monta suas próprias estruturas de tabelas como num banco relacional. A cada atributo adicionado o usuário deve informar o par *namespace#propriedade* relacionado ao esquema conhecido. Mais de uma tabela pode ser incluída. È necessário também que se defina três atributos obrigatórios para cada tabela, são eles: Chave primária (P), atributo recurso (R) e atributo origem (E) (figura 9.3). O atributo chave primária recebe um identificador exclusivo para cada registro adicionado na tabela, o atributo recurso recebe o recurso correspondente àquele registro e o atributo origem serve como chave estrangeira mantendo a relação entre tabelas.

FIGURA 9.3 – Interface do protótipo de equivalência

Aplicando-se a verificação de equivalência na tabela criada pelo usuário o algoritmo cria no banco de dados a tabela "Alunos" e adiciona as instâncias extraídas dos modelos equivalentes.

A (tabela 9.5) apresenta parte da tabela "Alunos" do banco de dados.

TABELA 9.5 – Tabela "Alunos" gerada pelo protótipo

ID	ORIGEM	RECURSO	NOME	MATRICULA	SEXO
1083	5	http://ain/schemas/6015.xml	VALÉRIA GALLÃO	6015	F
1084	6	http://ain/schemas/6016.xml	LILIAN NUNES SILVA	6016	F
1085	7	http://ain/schemas/6018.xml	JOÃO BAVARESCO	6018	M
1086	10	http://ain/schemas/6021.xml	SANDRA DA SILVA	6021	F
1087	11	http://ain/schemas/6053.xml	CARLOS SORPRESO	6053	M

Observando-se a (tabela 9.5) verifica-se que as propriedades dos modelos RDF, indicadas por *namespaces* no protótipo, são agora atributos da tabela aluno, tornando o modelo relacional mais legível.

Enfim, alguns detalhes ainda podem ser conseguidos com o uso do protótipo. Por exemplo, supondo que determinado modelo RDF faça uso de uma *Bag* cuja quantidade de membros é variável, pode-se instruir o protótipo para ignorar a equivalência na tabela de membros. Isso é conseguido com o uso da palavra reservada "ignorar_equivalencia" na coluna *namespace* do atributo que constitui a chave primária.

Nesse caso a tabela definida pelo usuário deve ter um atributo para cada membro da *Bag*, limitado é claro pela quantidade de atributos suportada por cada tabela do banco de dados. Se isso não for o bastante, a única alternativa seria uma tabela para cada membro da *Bag*.

Basicamente, o algoritmo usado pelo protótipo está descrito como segue:

1) São criadas no banco de dados uma tabela para cada estrutura definida pelo usuário;

2) Seleciona da tabela “triples” o conjunto de declarações de uma mesma origem;

3) Seleciona deste, quantos subconjuntos existirem do mesmo assunto;

4) Verifica se os predicados de cada subconjunto, em sua totalidade, equivalem a alguma das estruturas definida pelo usuário.

5) Se sim, os valores correspondentes são inseridos como instâncias desta tabela.

6) Se for usada a palavra chave “ignorar_equivalencia”, serão inseridas instâncias se, e somente se, um ou mais predicado for equivalente a um ou mais “namespace” da tabela definida pelo usuário.

10 Conclusões e perspectivas

Atualmente, comunidades de descrição de recursos têm-se preocupado com a localização, acesso e troca de informações em um contexto específico. Para esse grupo de pessoas, a linguagem XML tem prometido bons resultados. A proposta de *metadata* RDF não pretende substituir as tecnologias existentes de troca de informações, no entanto, sugere a adição de mais um recurso a elas.

Módulos de software necessitam ser capazes de reconhecer as propriedades que eles são projetados para processar, até mesmo em face às "colisões" que acontecem quando a marcação usa o mesmo tipo de elemento ou nome de atributo. Formas de codificação contempladas pelo RDF e RDFS garantem que as propriedades possuem nomes universais cujo escopo se estende além do conteúdo do próprio *metadata*. Um modelo padrão e bem definido garante a gerência e manutenção da semântica por comunidades independentes de peritos.

Mesmo permitindo manter o significado correspondente a cada marcação do modelo, o RDF e RDFS ainda necessitam de ferramentas e mecanismos de validações. Um esquema pode definir que a propriedade “autor” é usada para indicar recursos que são membros da classe pessoa, mas não diz como uma aplicação deveria agir no processamento dessa classe de informação. Espera-se que diferentes aplicações possam usar estas restrições de diferentes modos. Mecanismos de validação poderiam evitar erros no uso das marcações; editores interativos poderiam sugerir valores válidos para propriedades; aplicações de inferência no modelo de classes poderiam avisar por inconsistências etc...

Com o significado e tipo das propriedades (esquema) colocados juntos com o par atributo-valor de cada recurso, consultas, validações, verificações de integridade e decisões internas de processos de softwares se tornam mais fáceis.

Avaliando-se abordagens sobre este contexto, verificou-se que duas delas, Brian e Melnik se destacaram das demais, por apresentar um comportamento uniforme principalmente no aspecto mais desejável que são as consultas.

Verificou-se que todas as abordagens existentes se preocupam em manter qualquer tipo de modelo ou esquema RDF independente da forma de codificação. Essa preocupação tem levado todas as abordagens a considerar as propriedades de um modelo RDF como valor de um atributo do modelo relacional e isso obriga que a instrução SQL avalie duas condições para recuperar o mesmo dado. Se a propriedade do modelo fosse um atributo da tabela do banco de dados, apenas uma comparação seria necessária. Recursos são recuperados através de valores de propriedades, exatamente da mesma maneira que uma instância é recuperada em uma tabela de banco de dados. Então, esse mecanismo natural do banco de dados foi o artifício utilizado pela nova proposta para possibilitar as instruções SQL mais simples.

A legibilidade e memorização do modelo relacional que mantém as declarações RDF se tornam mais fácil porque o modelo foi elaborado pelo próprio usuário com o auxílio do protótipo.

Este estudo chamou atenção para outras aplicações com esta solução. A criação de ferramentas de consultas e manutenção para recursos com *metadata* RDF com as idéias propostas, poderiam ser facilmente implementadas. Um modelo relacional genérico (repositório) seria o responsável por manter todos os documentos, sem nenhum problema quanto a mudanças nas suas estruturas, enquanto que uma ferramenta, semelhante ao protótipo, seria a responsável por manter todas as estruturas de tabelas equivalentes à versão e modelo do *metadata* armazenado. Uma recuperação de documentos por versão poderia ser implementada, de forma a otimizar o algoritmo de

equivalência. Estudos mais detalhados poderiam medir se o esforço computacional despendido para criar e popular as tabelas dos modelos relacionais do usuário em prol da simplificação das instruções SQL seja viável.

Visando reduzir a sobrecarga de informações nos mecanismos de procura, filtros especializados para recursos com *metadata* RDF poderiam ser criados e integrados às ferramentas populares de indexação de textos, considerando é claro o mecanismo de propriedades e *namespace*. Ferramentas de recuperação para um esquema particular poderiam ser criadas por um e utilizadas por muitos.

As tabelas definidas pelos usuários poderiam ser incrementadas com tipos de atributos específicos para cada valor. (ex.: atributos data com o tipo *date*).

Na internet discrepantes modelos de descrição interferem na habilidade de procurar através dos limites de uma disciplina. A busca de novos mecanismos com uso do RDF constituem os desafios à continuidade dessa pesquisa.

Anexo 1 Instruções SQL

A tabelas seguintes apresentam respectivamente: nome da abordagem, descrição da consulta, instrução SQL e custo correspondente.

TABELA A.1.1 - Primeiro tipo de consulta elaborada sobre a abordagem de Brian

Abordagem	**Brian McBride**
Descrição	Esta consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.
SQL_1	select r.roname, concat(ns1.nsname, r1.roname) as pred, lit.val from rdfresource r, rdfnamespace ns1, rdfmodel m, rdfresource r1, rdfstatement st, rdfliteral lit where r.id=m.modelid and m.statement=st.id and st.predicate=r1.id and r1.ns=ns1.id and st.objliteral=lit.id and lit.val= 'Doutor' and ns1.nsname='http://ain/schemas/EsquemaPessoas#' and r1.roname='titulo';
Custo	23

TABELA A.1.2 – Segundo tipo de consulta elaborada sobre a abordagem de Brian

Abordagem	**Brian McBride**
Descrição	Esta consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'.
SQL_2	select r.roname, concat(ns1.nsname, r1.roname) as pred, lit.val from rdfresource r, rdfnamespace ns1, rdfmodel m, rdfresource r1, rdfstatement st, rdfliteral lit where r.id=m.modelid and m.statement=st.id and st.predicate=r1.id and r1.ns=ns1.id and st.objliteral=lit.id and r.roname='http://ain/schemas/6016.xml';
Custo	49

TABELA A.1.3 – Terceiro tipo de consulta elaborada sobre a abordagem de Brian

Abordagem	**Brian McBride**
Descrição	Esta consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'.
SQL_3	select concat(ns1.nsname, r1.roname) as rec, lit.val from rdfresource r1, rdfnamespace ns1, rdfliteral lit, rdfnamespace ns2, rdfresource r2, rdfstatement st where ns1.id=r1.ns and r1.id=st.subject and ns2.id=r2.ns and r2.id=st.predicate and lit.id=st.objliteral and ns1.nsname= 'http://ain/schemas/EsquemaPessoas#' and r1.roname='matricula' and ns2.nsname='http://www.w3.org/TR/1999/PR-rdf-schema-19990303#' and r2.roname='comment';
Custo	18

TABELA A.1.4 – Quarto tipo de consulta elaborada sobre a abordagem de Brian

Abordagem	**Brian McBride**
Descrição	Esta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'.
SQL_4	select r.roname from rdfresource r where r.id in (select m.modelid from rdfmodel m, rdfstatement st, rdfnamespace ns, rdfresource r where ns.id=r.ns and r.id=st.predicate and ns.nsname= 'http://ain/schemas/EsquemaPessoas#' and r.roname='nome' and m.statement=st.id) and r.id in (select m.modelid from rdfmodel m, rdfstatement st, rdfnamespace ns, rdfresource r where ns.id=r.ns and r.id=st.predicate and ns.nsname='http://ain/schemas/EsquemaPessoas#' and r.roname='sexo' and m.statement=st.id) and r.id in (select m.modelid from rdfmodel m, rdfstatement st, rdfnamespace ns, rdfresource r where ns.id=r.ns and r.id=st.predicate and ns.nsname= 'http://ain/schemas/EsquemaPessoas#' and r.roname='matricula' and m.statement=st.id) and r.id in (select m.modelid from rdfmodel m, rdfstatement st, rdfnamespace ns, rdfresource r where ns.id=r.ns and r.id=st.predicate and ns.nsname='http://ain/schemas/EsquemaPessoas#' and r.roname='curso' and m.statement=st.id) and r.id in (select m.modelid from rdfmodel m, rdfstatement st, rdfnamespace ns, rdfresource r where ns.id=r.ns and r.id=st.predicate and ns.nsname= 'http://ain/schemas/EsquemaPessoas#' and r.roname='dataNasc' and m.statement=st.id);
Custo	147

TABELA A.1.5 – Primeiro tipo de consulta elaborada sobre a abordagem de Melnik

Abordagem	**Sergey Melnik**
Descrição	Esta consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'. Nota-se que na instrução SQL foi utilizado o valor hash CRC32 pré-processado do par propriedade/valor.
SQL_1	select m.uri, concat(ns1.value, r1.name) as pred, lit.value from resources r1, namespaces ns1, models m, literals lit, triples t where m.id=t.model and t.predicate=r1.hash and r1.ns=ns1.hash and t.object=lit.hash and t.predicate= -2016262896 and t.object= -589537348;
Custo	15

TABELA A.1.6 – Segundo tipo de consulta elaborada sobre a abordagem de Melnik

Abordagem	**Sergey Melnik**
Descrição	Esta consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'. Nota-se que na instrução SQL foi utilizado o valor hash CRC32 pré-processado do recurso.
SQL_2	select m.uri, concat(ns1.value, r1.name) as pred, lit.value from resources r1, namespaces ns1, models m, literals lit, triples t where m.id=t.model and t.predicate=r1.hash and r1.ns=ns1.hash and t.object=lit.hash and m.id=2050549583;
Custo	20

TABELA A.1.7 – Terceiro tipo de consulta elaborada sobre a abordagem de Melnik

Abordagem	**Sergey Melnik**
Descrição	Esta consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'. Nota-se que na instrução SQL foi utilizado o valor hash CRC32 pré-processado da propriedade 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#comment' como predicado e da propriedade 'http://ain/schemas/EsquemaPessoas#matricula' como recurso.
SQL_3	select concat(ns1.value, r1.name) as prop, lit.value from namespaces ns1, resources r1, literals lit, triples t where t.object=lit.hash and t.subject=r1.hash and r1.ns=ns1.hash and t.predicate=126210730 and t.subject=344431486;
Custo	20

TABELA A.1.8 – Quarto tipo de consulta elaborada sobre a abordagem de Melnik

Abordagem	**Sergey Melnik**
Descrição	Esta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'. Nota-se que na instrução SQL foi utilizado o valor hash CRC32 pré-processado das propriedades.
SQL_4	select m.uri from models m, triples t where t.model=m.id and t.model in (select t.model from triples t where t.predicate=1004821554) and t.model in (select t.model from triples t where t.predicate=344431486) and t.model in (select t.model from triples t where t.predicate=1130387884) and t.model in (select t.model from triples t where t.predicate= -650646587) and t.model in (select t.model from triples t where t.predicate=1179401978) group by m.uri;
Custo	51

TABELA A.1.9 – Primeiro tipo de consulta elaborada sobre a abordagem de Eric

Abordagem	**Eric Miller**
Descrição	Esta consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.
SQL_1	select t1.recurso, t3.propriedade, t3.valor from triple t1, triple t2, triple t3 where t1.valor=t2.recurso and t2.valor=t3.recurso and t3.propriedade= 'http://ain/schemas/EsquemaPessoas#titulo' and t3.valor='Doutor';
Custo	93

TABELA A.1.10 – Segundo tipo de consulta elaborada sobre a abordagem de Eric

Abordagem	**Eric Miller**
Descrição	Esta consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'.
SQL_2	select recurso, propriedade, valor from triple where recurso= 'http://ain/schemas/6016.xml' and hint='L';
Custo	8

TABELA A.1.11 – Terceiro tipo de consulta elaborada sobre a abordagem de Eric

Abordagem	**Eric Miller**
Descrição	Esta consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'.
SQL_3	select recurso, valor from triple where propriedade= 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#comment' and recurso='http://ain/schemas/EsquemaPessoas#matricula';
Custo	3

TABELA A.1.12 – Quarto tipo de consulta elaborada sobre a abordagem de Eric

Abordagem	**Eric Miller**
Descrição	Esta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'.
SQL_4	select recurso from triple where recurso in (select recurso from triple where propriedade='http://ain/schemas/EsquemaPessoas#nome') and recurso in (select recurso from triple where propriedade= 'http://ain/schemas/EsquemaPessoas#matricula') and recurso in (select recurso from triple where propriedade= 'http://ain/schemas/EsquemaPessoas#sexo') and recurso in (select recurso from triple where propriedade= 'http://ain/schemas/EsquemaPessoas#curso') and recurso in (select recurso from triple where propriedade= 'http://ain/schemas/EsquemaPessoas#dataNasc') group by recurso;
Custo	199

TABELA A.1.13 – Primeiro tipo de consulta elaborada sobre a abordagem de Jonas

Abordagem	**Jonas Liljegren**
Descrição	Esta consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.
SQL_1	select r1.uri, r2.uri, r3.value from statement st1, statement st2, statement st3, resources r1, resources r2, resources r3 where st1.obj=st2.subj and st2.obj=st3.subj and st1.subj=r1.id and st3.pred=r2.id and st3.obj=r3.id and r2.uri='http://ain/schemas/EsquemaPessoas#titulo' and r3.value='Doutor';
Custo	39

TABELA A.1.14 – Segundo tipo de consulta elaborada sobre a abordagem de Jonas

Abordagem	**Jonas Liljegren**
Descrição	Esta consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'.
SQL_2	select r1.uri, r2.uri, r3.value from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r2.uri='http://ain/schemas/6016.xml' and r3.value!=' ';
Custo	48

TABELA A1.15 – Terceiro tipo de consulta elaborada sobre a abordagem de Jonas

Abordagem	**Jonas Liljegren**
Descrição	Esta consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'.
SQL_3	select r2.uri, r3.value from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri='http://www.w3.org/TR/1999/PR-rdf-schema-19990303#comment' and r2.uri='http://ain/schemas/EsquemaPessoas#matricula';
Custo	26

TABELA A.1.16 – Quarto tipo de consulta elaborada sobre a abordagem de Jonas

Abordagem	**Jonas Liljegren**
Descrição	Esta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'.
SQL_4	select r.uri from resources r where r.uri in (select r2.uri from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri= 'http://ain/schemas/EsquemaPessoas#nome') and r.uri in (select r2.uri from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri= 'http://ain/schemas/EsquemaPessoas#matricula') and r.uri in (select r2.uri from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri='http://ain/schemas/EsquemaPessoas#sexo') and r.uri in (select r2.uri from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri='http://ain/schemas/EsquemaPessoas#curso') and r.uri in (select r2.uri from resources r1, resources r2, resources r3, statement st where r1.id=st.pred and r2.id=st.subj and r3.id=st.obj and r1.uri='http://ain/schemas/EsquemaPessoas#dataNasc');
Custo	220

TABELA A.1.17 – Primeiro tipo de consulta elaborada sobre a abordagem de Zanete

Abordagem	**Nelson Henrique Zanete**
Descrição	Esta consulta seleciona todos os recursos que possuem a propriedade 'http://ain/schemas/EsquemaPessoas#titulo' igual a 'Doutor'.
SQL_1	select recurso, titulo from titulo where titulo='Doutor';
Custo	6

TABELA A.1.18 – Segundo tipo de consulta elaborada sobre a abordagem de Zanete

Abordagem	**Nelson Henrique Zanete**
Descrição	Esta consulta seleciona todas as propriedades do recurso 'http://ain/schemas/6016.xml'.
SQL_2	select t.assunto, t.predicado, t.objeto from recurso r, triples t where r.id=t.origem and t.assunto='http://ain/schemas/6016.xml' and t.tipo='L';
Custo	33

TABELA A.1.19 – Terceiro tipo de consulta elaborada sobre a abordagem de Zanete

Abordagem	**Nelson Henrique Zanete**
Descrição	Esta consulta seleciona o significado da propriedade 'http://ain/schemas/EsquemaPessoas#matricula'.
SQL_3	select concat(esquema, propriedade), significado from esq where propriedade='matricula';
Custo	4

TABELA A.1.20 – Quarto tipo de consulta elaborada sobre a abordagem de Zanete

Abordagem	**Nelson Henrique Zanete**
Descrição	Esta consulta seleciona todos os recursos que tenham a propriedade nome, matricula, sexo, curso e datanasc do esquema 'http://ain/schemas/EsquemaPessoas#'.
SQL_4	select recurso from alunos;
Custo	10

Anexo 2 Algoritmo de geração do modelo RDF “Aluno” e “Professor”

Os algoritmos de geração dos modelos de *metadata* RDF foram escritos em Visual Basic e atuam sobre duas tabelas do Microsoft Access 2000. As figuras seguintes apresentam os algoritmos para alunos e professores respectivamente.

```
Private Sub Comando2_Click()

' ************************************************
' *** Algoritmo de geração de metadata RDF para alunos ***
' ************************************************

' Criação de variáveis auxiliares
Dim wmatricula As Long
Dim wnome As String
Dim wsexo As String
Dim wdatanasc As String
Dim wcurso As String
Dim wbanco As Database
Dim walu As Recordset
Dim wdir As String
Dim wresult As String

Set wbanco = CurrentDb
Set walu = wbanco.OpenRecordset("DadosAlunos")

' Complemento da pasta de destino
wdir = Me![Texto0]
wresult = Right(wdir, 1)
If wresult <> "\" Then
   wdir = wdir & "\"
End If

' Inicio da rotina para alunos
walu.MoveFirst
Do Until walu.EOF
   wmatricula = 0
   wnome = " "
   wsexo = " "
   wdatanasc = " "
   wcurso = " "
   wmatricula = walu![codigo]
   wnome = walu![nome]
   wsexo = walu![Sexo]
   wdatanasc = walu![datanasc]
   wcurso = walu![curso]

   Open wdir & wmatricula & ".xml" For Output As #1 ' Abre arquivo de saída

   ' Concatena variáveis com marcações RDF
   Print #1, "<?xml version=" & Chr(34) & "1.0" & Chr(34) & " encoding=" _
   & Chr(34) & "ISO-8859-1" & Chr(34) & " ?>"
   Print #1, "<rdf:RDF"

   Print #1, "xmlns:rdf=" & Chr(34) & _
   "http://www.w3.org/1999/02/22-rdf-syntax-ns#" & Chr(34)
   Print #1, "xmlns:s=" & Chr(34) & "http://ain/imagens/EsquemaPessoas#" _
   & Chr(34) & ">"
   Print #1, "<rdf:Description about=" & Chr(34) & "http://ain/exemplos/" _
```

```
   & wmatricula & ".xml" & Chr(34) & ">"
   Print #1, "<rdf:type rdf:resource=" & Chr(34) & _
   "http://ain/imagens/EsquemaPessoas#Aluno" & Chr(34) & "/>"
   Print #1, "<s:matricula>" & wmatricula & "</s:matricula>"
   Print #1, "<s:nome>" & wnome & "</s:nome>"
   Print #1, "<s:dataNasc>" & wdatanasc & "</s:dataNasc>"
   Print #1, "<s:sexo>" & wsexo & "</s:sexo>"
   Print #1, "<s:curso>" & wcurso & "</s:curso>"
   Print #1, "</rdf:Description>"
   Print #1, "</rdf:RDF>"
   Close #1 ' Fecha arquivo de saída
   walu.MoveNext
Loop
walu.Close ' Fecha conjunto de registros

MsgBox ("Dados gerados com sucesso")

End Sub
```

FIGURA A.2.1 – Algoritmo de geração de metadata RDF do modelo “Aluno”

```
Private Sub Comando0_Click()

' ****************************************************
' *** Algoritmo de geração de metadata RDF para professores ***
' ****************************************************

' Criação de variáveis auxiliares
Dim wcodigo As String
Dim wnome As String
Dim wsexo As String
Dim wbanco As Database
Dim wprof As Recordset
Dim i, j As Integer
Dim teste As ListaDeCursos
Dim wdir As String
Dim wresult As String

' Complementa pasta de destino
wdir = Me![Texto0]
wdir = Me![Texto0]
wresult = Right(wdir, 1)
If wresult <> "\" Then
   wdir = wdir & "\"
End If

Set wbanco = CurrentDb
Set wprof = wbanco.OpenRecordset("DadosProfessores")
wprof.MoveFirst
Do Until wprof.EOF
   wcodigo = " "
   wnome = " "
   wsexo = " "
   wcodigo = wprof![codigo]
   wnome = wprof![nome]
   wsexo = wprof![Sexo]


   Open wdir & wcodigo & ".xml" For Output As #1 ' Abre arquivo de saída
   ' Concatena variáveis com marcações RDF
   Print #1, "<?xml version=" & Chr(34) & "1.0" & Chr(34) & " encoding=" & Chr(34) & _
   "ISO-8859-1" & Chr(34) & " ?>"
   Print #1, "<rdf:RDF"
   Print #1, "xmlns:rdf=" & Chr(34) & "http://www.w3.org/1999/02/22-rdf-syntax-ns#" _
   & Chr(34)
   Print #1, "xmlns:s=" & Chr(34) & "http://ain/imagens/EsquemaPessoas#" _
   & Chr(34) & ">"
   Print #1, "<rdf:Description>"
   Print #1, "<rdf:type rdf:resource=" & Chr(34) & _
   "http://ain/imagens/EsquemaPessoas#Professor" & Chr(34) & "/>"
   Print #1, "<s:nome>" & wnome & "</s:nome>"
   Print #1, "<s:sexo>" & wsexo & "</s:sexo>"
   Print #1, "<s:listaDeCursos>"
   Print #1, "<rdf:Bag>"

 ' Zera o recurso tipo record
   For i = 1 To 10
        teste.ID(i) = " "
        teste.Titulo(i) = " "
        teste.Instituicao(i) = " "
        teste.Descricao(i) = " "
   Next i
```

```
  i = 0
  j = 0

  ' Carrega o recurso do tipo record
  Do Until wprof![codigo] <> wcodigo
       i = i + 1
       teste.ID(i) = wprof![codigo]
       teste.Titulo(i) = wprof![Titulo]
       teste.Instituicao(i) = wprof![Instituicao]
       teste.Descricao(i) = wprof![curso]
       Print #1, "<rdf:li resource=" & Chr(34) & "#rec" & i & Chr(34) & "/>"
       wprof.MoveNext
       If wprof.EOF Then
          Exit Do
       End If
  Loop
  Print #1, "</rdf:Bag>"
  Print #1, "</s:listaDeCursos>"
  Print #1, "</rdf:Description>"
  j = 0
  j = i
  i = 0
  ' Monta as descrições da Bag descarregando o tipo record
  For i = 1 To j
       Print #1, "<rdf:Description ID=" & Chr(34) & "rec" & i & Chr(34) & ">"
       Print #1, "<s:titulo>" & teste.Titulo(i) & "</s:titulo>"
       Print #1, "<s:instituicao>" & teste.Instituicao(i) & "</s:instituicao>"
       Print #1, "<s:descricaoCurso>" & teste.Descricao(i) & "</s:descricaoCurso>"
       Print #1, "</rdf:Description>"
  Next i
  Print #1, "</rdf:RDF>"
  Close #1 ' Fecha arquivo de saída
Loop
wprof.Close ' Fecha conjunto de registros
MsgBox ("Dados gerados com sucesso")

End Sub
```

FIGURA A.2.2 – Algoritmo de geração de metadata RDF do modelo “Professor”

Bibliografia

[BER 98] BERNERS-LEE, Tim. ***Web design issues. What a semantic can represent.*** W3C. [September 17, 1998]. Disponível em: <http://www.w3.org/DesignIssues/RDFnot.html>. Acesso em: 06 nov. 2000.

[BOB 95] BOBROWSKI, Steven M. ***Dominando o Oracle7***. São Paulo: Makron Books, 1995. p. 573 –577.

[COM 2001] COMMONWEALTH OF AUSTRALIA. ***EdNA Metadata Standard Homepage***. [January 10, 2001]. Disponível em: <http://standards.edna.edu.au/metadata/index.html>. Acesso em: 03 out. 2000.

[DIL 98] DILLON, Matt. ***18.2 million dataset CRC-64 test & Program source.*** [October 11, 1998]. Disponível em: <http://web.mit.edu/afs/sipb/user/kolya/diablo/contrib/crctest/>. Acesso em: 16 out. 2001.

[DUB 2000] DUBLIN CORE METADATA INITIATIVE. ***About the Dublin Core Metadata Initiative.*** [October 18, 2000]. Disponível em: <http://dublincore.org/about/>. Acesso em: 15 out. 2001.

[HOO 98] HOOD, EARL. ***MIME (Multipurpose Internet Mail Extensions).*** University of California, Irvine. [July 11, 1998]. Disponível em: <http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html>. Acesso em: 16 out. 2001.

[INT 99] INTERNATIONAL FEDERATION OF LIBRARY ASSOCIATIONS AND INSTITUTIONS. ***Digital Libraries:*** *Metadata Resources*. [February 15, 1999]. Disponível em: <http://www.ifla.org/II/metadata.htm>. Acesso em: 27 jun. 2000.

[JIG 2001] JIGSAW TEAM. ***Jigsaw Overview.*** [July 13, 2001]. Disponível em: <http://www.w3.org/Jigsaw/Overview.html>. Acesso em: 16 out. 2001.

[LAF 2000] LAFON, Yves; BOS, Bert. ***Describing and retrieving photos using RDF and HTTP.*** [September 28, 2000]. Disponível em: <http://www.w3.org/TR/2000/NOTE-photo-rdf-20000928/>. Acesso em: 15 out. 2001.

[LIG 99] LIGHT, Richard. ***Iniciando em XML.*** São Paulo: Makron Books, 1999. 404 p.

[LIL 2000] LILJEGREN, Jonas. ***Description of the database implementation.*** [May, 2000]. Disponível em: <http://www-db.stanford.edu/~melnik/rdf/db-jonas.html>. Acesso em: 16 out. 2001.

[MEL 2000] MELNIK, Sergey. ***Storing RDF in a relational database.*** [May 11, 2000]. Disponível em: <http://www-db.stanford.edu/~melnik/rdf/db.html>. Acesso em: 05 mar. 2001.

[MIL 98] MILLER, Eric. ***An Introduction to the Resource Description Framework***. [November 5, 1998]. Disponível em: <http://www.dlib.org/dlib/may98/miller/05miller.html>. Acesso em: 03 out. 2000.

[NAT 2000] NATIONAL ARCHIVES OF AUSTRALIA. ***Australian Government Locator Service (AGLS) introduction.*** [March 30, 2000]. Disponível em: <http://www.naa.gov.au/recordkeeping/gov_online/ agls/summary.html>. Acesso em: 25 set. 2000.

[NER 2000] NERBONNE, Eamon. ***RDFPic.*** [2000]. Disponível em: <http://jigsaw.w3.org/rdfpic/>. Acesso em: 23 maio 2001.

[POW 2000] POWELL, David. ***Delphi 4 CRC32 implementation.*** [June 23, 2000]. Freeware Utilities from ISS Ltd. Disponível em: <http://www.iss.u-net.com/crc32.htm>. Acesso em:16 out. 2001.

[SAA 99] SAARELA, Janne. ***SiRPAC - Simple RDF Parser & Compiler***, W3C, latest release V1.14. [May 21, 1999]. Disponível em: <http://www.w3.org/RDF/Implementations/SiRPAC/>. Acesso em: 13 maio 2000.

[STA 2000] STANFORD MEDICAL INFORMATICS. ***Protégé-2000:*** *Home Page.* [1998-2000]. Stanford University. Disponível em: <http://protege.stanford.edu>. Acesso em: 05 mar. 2001.

[W3C 2000] W3C (WORLD WIDE WEB CONSORTIUM). ***Extensible Markup Language (XML).*** [October 6, 2000]. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 27 out. 2000.

[W3C 98] W3C (WORLD WIDE WEB CONSORTIUM). ***Namespaces in XML.*** [January 19, 1998]. Disponível em: <http://www.w3.org/TR/1998/NOTE-xml-names-0119.html>. Acesso em: 04 dez. 2000.

[W3C 99] W3C (WORLD WIDE WEB CONSORTIUM). ***Resource Description Framework (RDF) Model and Syntax Specification.*** [February 22, 1999]. Disponível em: <http://www.w3.org/TR/REC-rdf-syntax/>. Acesso em: 23 maio 2000.

[W3C 2000a] W3C (WORLD WIDE WEB CONSORTIUM). ***Resource Description Framework (RDF) Schema Specification 1.0.*** [March 27, 2000]. Disponível em: <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>. Acesso em: 14 nov. 2000.

[WOL 97] WOLF, Misha WICKSTEED, Charles. ***Date and Time Formats.*** W3 Consortium. [October 12, 1997]. Disponível em: <http://www.w3.org/TR/NOTE-datetime>. Acesso em: 02 out. 2000.