UNIVERSIDADE FEDERAL DE SANTA CATARINA

APLICAÇÃO DE ONTOLOGIAS PARA O SUPORTE AO PROCESSAMENTO ETL EM SOLUÇÕES DE BUSINESS INTELLIGENCE

Dhiogo Cardoso da Silva

Florianópolis – SC
2006/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Aplicação de ontologias para o suporte ao processamento ETL em soluções de Business Intelligence

Dhiogo Cardoso da Silva

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis - SC
2006/2

Dhiogo Cardoso da Silva

Aplicação de ontologias para o suporte ao processamento ETL em soluções de Business Intelligence

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação

Banca Examinadora:

_____
Prof. Dr. José Leomar Todesco
Universidade Federal de Santa Catarina
Orientador

_____
Dr. Denilson Sell
Co-orientador

_____
Prof. Dr. Ronaldo dos Santos Mello
Universidade Federal de Santa Catarina

RESUMO

Diversas empresas não obtêm conhecimentos compulsórios para o andamento de seus negócios embora haja muitos investimentos onerosos em soluções de BI e na construção de Data Warehouses. Percebe-se que boa parte das arquiteturas de BI implantadas não são flexíveis às mudanças de requisitos, das regras de negócio corporativas e muitas vezes não possibilitam novas análises para o gestor. A arquitetura SBI proposta por Sell (2006) visa tornar solúveis essas limitações com base na semântica e no conhecimento do negócio da organização, por meio de ontologias e inferências semânticas. A fim de melhorar o desempenho das ferramentas OLAP e também flexibilizar o processo de Data Warehousing com foco no processamento ETL, propõe-se uma alternativa à abordagem empregada por Sell. Através de uma ferramenta desenvolvida, todas as inferências semânticas feitas sobre a ontologia são armazenadas no DW, em uma estrutura apropriada pra esse propósito, denominada modelo tripla. Além da arquitetura SBI, a ferramenta baseia-se nos principais formalismos de representação de recursos e de ontologias da Web Semântica, tais como RDF e OWL. Um estudo de caso relacionado à Plataforma Lattes Institucional da UFSC foi realizado a fim de demonstrar a aplicabilidade do sistema. Com isso, novas análises puderam ser apresentadas pelo DW através da junção entre o seu modelo dimensional e o modelo tripla proposto.

Palavras-chave: Business Intelligence; Data Warehousing; Ontologias; Arquitetura SBI.

ABSTRACT

Various companies don't obtain compulsory knowledge for the progress of their businesses although there are costly investments in Business Intelligence solutions and in the construction of Data Warehouses. It has to be mentioned that most implemented BI architectures are not flexible to changing requirements, corporate business rules and do not allow managers to do new analysis in many cases.The SBI architecture proposed by Sell (2006) aims at to become soluble to these limitations based on semantics and on knowledge of the organization's business using ontologies and semantic inferences. In order to improvement the performance of OLAP tools and also the flexibilty of the Data Warehousing process, with focus on ETL processing, an alternative approach applied by Sell is proposed. Through the developed system, all semantic inferences done on an ontology are saved in the Data Warehouse within an appropriate structure for this proposal, named triple model. Beyond the SBI architecture, the tool is based on principle formalisms of resource and ontologies representation of Web Semantics like RDF and OWL. A case study on top of the Institutional Lattes Platform of the Federal University of Santa Catarina has been realized to demonstrate the applicability of the system. New analysis can be presented by Data Warehouses by joining the dimensional model and the proposed triple model.

Keywords: Business Intelligence; Data Warehousing; Ontology; SBI Arquitecture.

LISTAS DE FIGURAS

LISTAS DE TABELAS

LISTAGENS

LISTAS DE ABREVIATURAS

BI - Business Intelligence

CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico

CRM - Customer Relationship Management

DAML - Darpa Agent Markup Language

DL - Description Logics

DW - Data Warehouse

EIS - Enterprise Information Systems

ERP - Enterprise Resource Planning

MER - Modelo Entidade-Relacionamento

OCML - Operational Conceptual Modelling Language

OIL - Ontology Inference Layer

OLAP - On-Line Analytical Processing

OLTP - On-Line Transaction Processing

OWL - Web Ontology Language

PLI - Plataforma Lattes Institucional

RDF - Resource Description Framework

RuleML - Rule Markup Language

SBI - Semantic Business Intelligence

SGBD - Sistema Gerenciador de Banco de Dados

SQL - Structured Query Language

SWRL - Semantic Web Rule Language

W3C - World Wide Web Consourtium

XML - eXtensible Markup Language

SUMÁRIO

1 Introdução

A busca por informações estratégicas e essenciais para gestão empresarial impulsionou a evolução dos sistemas de informação. Antes, as organizações aplicavam os recursos computacionais apenas para a automatização de seus processos e ao suporte às operações de negócio. Esses recursos, chamados de sistema transacionais, não são projetados para oferecer suporte analítico e devido ao grande volume de dados da empresa, tornaram-se ainda mais inviáveis para análises. Com isso, muitas tecnologias informacionais úteis aos processos decisórios corporativos ganharam mercado e contribuíram para o crescimento da área de BI (*Business Intelligence*).

Não obstante oferecer um ambiente propício para análises, verifica-se que as soluções de BI ainda apresentam algumas limitações referecentes a novas necessidades analíticas. A arquitetura típica de BI concebida na maioria das organizações, mesmo assistida pelos seus componentes de âmbito informacional, como DW (*Data Warehouse*) e ferramentas OLAP (*On-Line Analytical Process*), oferece pouca ou quase nenhuma flexibilidade para alteração e incorporação de novas regras de negócio pela organização. Isto é, uma vez definidos os requisitos de projeto, bem como os indicadores utilizados nas análises, e desenvolvida uma plataforma de BI para suporte dessas informações, torna-se custoso para a empresa adicionar novas variáveis de interesse e regras de negócio ou extrair novos indicadores. Dessa forma, as análises exploratórias realizadas sobre um grande repositório de dados comumente limitam-se às definições feitas *a priori* e não são capazes de gerar novas informações estratégicas para a corporação (SELL, 2006).

Visto que o mercado competitivo faz com que as corporações busquem sempre novos conhecimentos relevantes para o seu serviço ou produto, é necessária uma forma flexível para auxiliar na tomada de decisão e estender as arquiteturas de BI tradicionais. Para isso, Sell propõe em sua tese de doutorado, cujo título é “Uma Arquitetura para Business Intelligence baseada em Tecnologias

Semânticas para Suporte a Aplicações Analíticas", uma mudança de paradigma a essas arquiteturas – a arquitetura Semantic Business Intelligence (SBI).

A arquitetura SBI utiliza a semântica do negócio da organização para realizar o processamento analítico, através de diversos componentes inéditos e ontologias. O uso de ontologias, ou seja, a utilização de um vocabulário explícito, através do qual se definem regras, relacionamentos entre classes, funções e objetos para compartilhamento de um universo de discurso, (GRUBER, 1993) referentes ao negócio da organização, foca em aspectos semânticos para poder proporcionar maleabilidade às soluções de BI convencionais na arquitetura SBI. Essas ontologias são usadas na arquitetura SBI para subsidiar o processamento de inferências semânticas sobre as fontes de dados da organização. Assim, novas perspectivas analíticas podem ser extraídas ou inferidas do DW e apresentadas para o cliente, com base na terminologia de seu negócio.

1.1 Problemática

A abordagem adotada por Sell em sua pesquisa oferece meios flexíveis de se realizar o processamento OLAP. Isto é, as análises feitas com base na semântica do negócio, suportadas pela arquitetura SBI, são sempre submetidas *in real time* ao DW sem haver nenhum tipo de armazenamento de inferências. Dessa forma, dada uma nova necessidade informacional, novas consultas devem ser reprocessadas sobre as fontes de dados. De acordo com o volume de dados, a complexidade da consulta, ou ainda, a quantidade de acessos ao DW da organização, esse processamento OLAP pode implicar em baixo desempenho. Além disso, o reprocessamento analítico pode ocasionar a replicação de instâncias da ontologia do domínio da organização.

Para dar facilidade e rapidez ao processamento OLAP, nesta pesquisa adota-se uma abordagem distinta da tese tratada. O DW é utilizado tanto como fonte de informações, para criar o conjunto de instâncias da ontologia de domínio proposto pelo cliente, como também para armazenar as deduções semânticas feitas sobre a ontologia, dado um conjunto de regras. Logo, este trabalho visa solucionar a seguinte problemática focada na arquitetura SBI: é possível utilizar as

ontologias e regras de negócios de modo flexível a fim de prover uma ferramenta, baseada na arquitetura SBI, capaz de auxiliar no processo de *Business Intelligence* e suportar o processamento e armazenamento de inferências semânticas sobre o Data Warehouse?

A resolução do problema baseia-se no princípio de que novas informações possam ser inferidas e adicionadas flexivelmente ao DW através de ontologias e regras de negócio. Todo o processamento e armazenamento de inferências ocorre em modo *batch*, tal como os processos ETL tradicionais. Conseqüentemente, diferentes visões de análise podem ser mostradas pelo DW, além das já existentes.

Para tal, uma ontologia deve ser definida e associada a um modelo de dados propício para a extração de informações, como o modelo proposto na construção de um DW – modelo dimensional ou modelo estrela. Deste modo, as terminologias específicas e os conceitos pertencentes ao contexto da organização devem ser modelados para produzir uma ontologia de domínio, que posteriormente será vinculada ao DW através de outra ontologia. A partir da ontologia de domínio e da aplicação de regras de negócio sobre essa ontologia, geram-se as novas informações através de inferências semânticas sobre o DW. Essas inferências são salvas no DW em uma estrutura adicional ao modelo estrela, denominada modelo tripla, na qual é uma das contribuições deste trabalho relacionada a arquitetura SBI.

O modelo tripla proposto é capaz de relacionar quaisquer duas dimensões aos conceitos inferidos sobre a ontologia. Assim, as novas análises já se encontram armazenadas e prontas para extração por parte das ferramentas OLAP, sem necessidade de processamento sobre o DW. O modelo tripla, analogamente ao modelo RDF/OWL é composto por Sujeitos, Predicados e Objetos. Os sujeitos e os Objetos referem-se às dimensões do DW e os Predicados estão relacionados aos conceitos inferidos sobre a ontologia.

1.2 Objetivos

1.2.1 Objetivo geral

Como objetivo geral deste trabalho pretende-se, através do projeto e da implementação de um aplicativo, empregar ontologias no processamento e armazenamento de inferências semânticas a fim de melhorar o desempenho das ferramentas OLAP na arquitetura SBI, dando flexibilidade ao processo de Data Warehousing[1], especificamente no que diz respeito a ETL. Essa flexibilidade é oferecida por meio de uma estrutura complementar ao modelo estrela, chamada de modelo tripla. O DW além de possuir as informações de interesse da organização, passa também a armazenar conclusões e resultados de regras de inferência. Com isso, ocorre um aumento nas capacidades de análises para apoio à tomada de decisão no processo de BI.

1.2.2 Objetivos Específicos

Os objetivos específicos são:

1 aplicar a semântica das informações e os conceitos de ontologia de domínio para a construção de um sistema de informação maleável às necessidades estratégicas dos clientes no processo de BI;

2 empregar linguagens formais de padrão aberto já existentes e expressivas para a representação de ontologias e de regras de negócio de uma organização;

3 adequar e usufruir de recursos ou motores de inferências disponíveis no mercado para a construção da ferramenta que dêem suporte a linguagem de representação de ontologias e de regras de negócio. Essa ferramenta

[1] Data Warehousing consiste em um processamento informacional para suportar o processo de tomada de decisão relacionado à construção do Data Warehouse, complementando o processamento transacional existente, que suporta a operação do negócio (KIMBALL, 2002 ; INMON , 2005).

obedece a arquitetura SBI é capaz de deduzir informações a partir do mapeamento entre ontologias e repositório de dados; e

4 desenvolver o módulo funcional chamado Gerenciador de Ontologias e os Repositórios de Ontologias da arquitetura semântica, propostos na tese de doutorado de Denilson Sell (2006), a fim de propor uma nova solução de BI. O Gerenciador de Ontologias é desenvolvido para que qualquer mecanismo de inferência, repositório de ontologias e fonte de dados externos possam ser integrados.

1.3 Delimitação do Escopo

Este trabalho está basicamente vinculado à arquitetura SBI de Sell (2006). Conforme esse autor, a maioria dos trabalhos relacionados à BI limita-se à aplicação de ontologias apenas para integração semântica e para apoiar as buscas sobre conteúdos, sem oferecer novas características analíticas.

Sell identifica somente três pesquisas de BI que se ligam a este estudo por propor novas extensões às arquiteturas de BI, são elas: SEWASIE (BERGAMASCHI; QUIX; JARKE, 2005), BIKM (CODY et al., 2002) e a solução de Priebe e Permul (PRIEBE; PERMUL, 2003).

SEWASIE é um acrônimo de *Semantic Webs and Agents in Integrated Economies* cujo objetivo é fornecer acesso inteligente a fontes de dados heterogêneas na Web através de semântica dos dados. Em sua arquitetura, diversos agentes realizam consultas sobre bases de dados específicas a partir de um mapeamento de uma ontologia de domínio. BIKM também oferece integração semântica de fontes de dados heterogêneas, todavia contém algoritmos de mineração de dados para extração de dimensões e tabelas de fatos a partir de documentos. Já a solução de Priebe e Permul possibilta que informações contidas em fontes de dados não estruturadas possam ser oferecidas a partir de consultas OLAP e conforme o contexto no qual são executadas. Para isso é utilizada uma ontologia que relaciona documentos às análises feitas pelo usuário.

As pesquisas de BI mencionadas anteriormente não utilizam as ontologias a fim de auxiliar de maneira flexível o processamento ETL e não oferecem novas

análises exploratórias com a terminologia usada pelos próprios clientes. A proposta deste trabalho oferece uma abordagem alternativa relacionada ao armazenamento de deduções semânticas sobre a arquitetura SBI, que é o principal tema relacionado.

Baseando-se na arquitetura SBI, apresentada na Figura 1 abaixo, foca-se nos seguintes elementos constituintes: *Ontologia BI, Ontologia do Domínio, DW; Reasoner e Gerenciador de Ontologias.* Esses módulos, que auxiliam no processamento e armazenamento de inferências semânticas na arquitetura, são descritos na seqüência do documento.

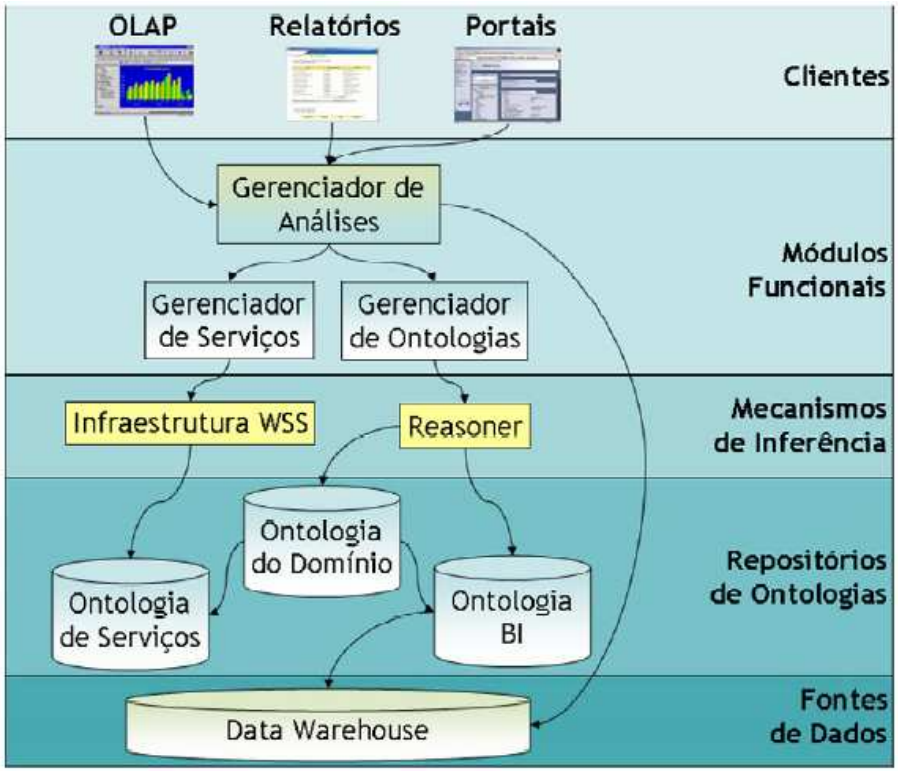


FIGURA 1 – Arquitetura Semântica de Business Intelligence (SBI)
(Fonte: SELL, 2006)

Todos esses elementos relacionados ao escopo deste trabalho pertencem à camada Back-End da arquitetura de BI tradicional e ao processo de Data Warehousing e, como é observado na Figura 1, distribuem-se em quatro camadas da arquitetura SBI. Sendo assim, as ferramentas de análise de informações, ou

OLAP, que não estão vinculadas ao objetivo do projeto poderão usufruir do conteúdo gerado por esses elementos.

Com exceção do DW e do Reasoner, os demais elementos foram construídos no projeto do aplicativo. Para compor o repositório de ontologias adotou-se OWL[2] (Web Ontology Language) e o Jena Web Semantic Framework (JENA, 2006) como mecanismo de inferência (ou Reasoner da Figura 1 acima), diferentemente do empregado na tese de Sell, em que é adotado OCML[3] (Operational Conceptual Modelling Language). A descrição e o motivo da adoção desses recursos são relatados adiante. O módulo funcional Gerenciador de Ontologias foi desenvolvido com as APIs Java.

A arquitetura SBI e as tecnologias empregadas são detalhadas no Capítulo 3.

1.4 Justificativa

As tecnologias de informação que tratam da introdução da semântica nos dados e da adaptação de novas regras de negócio para propiciar uma melhor gestão de negócios são contemporâneas e ainda possuem alguns obstáculos ou limitações a vencer. O Gartner Group (GARTNER, 2006) estima que até 2007 mais de 50% dos projetos de Data Warehouse irão falhar ou terão aceitação limitada, devido à falta de atenção à qualidade da informação ou por não estarem adequados aos assuntos do negócio. Outro dado importante é apresentado pela AT&T, que declara que as regras de negócio mudam em média 8% a.m. nas organizações. Haja vista esse cenário de insucesso e de mudanças de requisitos e de regras, é necessária uma solução de BI mais flexível e com foco na qualidade da informação. Logo, é uma área de pesquisa importante que necessita de mais apoio, contribuição e desenvolvimento, o que justifica o presente projeto.

A arquitetura SBI introduz novas soluções às arquiteturas de BI, tornando solúveis os pontos supracitados. Entretando, o modo aplicado por Sell traz algumas deficiências quanto ao desempenho das ferramentas OLAP, nas quais este trabalho dispõe-se a resolver.

---

[2] OWL <http://www.w3.org/TR/owl-features>.
[3] OCML <http://kmi.open.ac.uk/projects/ocml>.

O estado-da-arte no que se refere às linguagens de representação de ontologias e motores de inferência é utilizado para conceber uma ferramenta fundamentada na arquitetura SBI a fim de gerar informações com uso de ontologias e semântica dos dados.

1.5 Metodologia

Para idealizar o projeto, têm-se cinco etapas principais, são elas:

1) levantamento da literatura e tecnologia disponível;
2) adoção de uma linguagem formal para representação de ontologias e um mecanismo de inferência;
3) definição de uma ontologia e de regras de negócio junto a um DW que trata de assuntos acadêmicos, como estudo de caso;
4) construção propriamente dita do sistema de informação; e
5) aplicabilidade do estudo de caso.

Na etapa 1, buscaram-se informações literárias sobre os conceitos e as tecnologias associados aos processos de construção do DW e de Business Intelligence, bem como as ferramentas mais utilizadas. Com isso, puderam-se obter as características necessárias para a concepção do aplicativo.

A etapa 2 envolve a escolha da linguagem de representação de ontologias para compor a base de conhecimento do sistema. Para futuramente obter uma integração com outros sistemas, optou-se pelo uso de OWL, por ser expressiva, de padrão aberto e baseada na Web Semântica, de tal modo que o usuário possa usufruir de editores de ontologias e ontologias de terceiros. Jena Semantic Web Framework é utilizado nesta pesquisa como mecanismo de inferência, já que dá suporte à OWL e possui recursos nativos úteis nas deduções semânticas.

A elaboração das regras de inferência e da ontologia de domínio acadêmico é realizada na etapa 3. Nessa etapa modelaram-se classes e relacionamentos associados às atividades profissionais das pessoas dentro da instituição de ensino. Utilizou-se o editor de ontologias denominado Prótégé[4] para criação da ontologia e de seu respectivo mapeamento para o DW. Todas as informações amostrais do

[4] Prótégé <http://protege.stanford.edu>.

DW são oriundas da Plataforma Lattes Institucional[5] (PLI) e foram concedidas pela UFSC para fins de pesquisa.

A quarta etapa é a que compreende o desenvolvimento do sistema propriamente dito. Alguns componentes da arquitetura SBI são implementados com a linguagem Java, e para sua conclusão, além das ferramentas citadas, contou-se também com o apoio de outros recursos de software externos.

Por fim, a quinta e última etapa é focada na aplicação do estudo de caso e na viabilidade da ferramenta desenvolvida. Os testes foram realizados em dois ambientes computacionais diferentes, e uma ferramenta analítica ainda foi ajustada para exibir as novas análises oferecidas.

1.6 Organização do Trabalho

O capítulo atual dá o contexto geral no qual a pesquisa está situada e os seus principais objetivos. São apresentadas também as justificativas e as contribuições do trabalho, bem como a metodologia utilizada. Alguns trabalhos relacionados são brevemente listados.

No Capítulo 2, dá-se uma fundamentação teórica a respeito dos principais conceitos de BI, Data Warehousing e aplicação de semântica por meio de ontologias. Alguns comparativos entre linguagens de representação de ontologias são feitos.

O Capítulo 3 especifica a proposta deste trabalho, na qual os módulos da arquitetura SBI são detalhados com as contribuições e as tecnologias empregadas. Todo o desenvolvimento de um sistema de informação e sua aplicação sobre um estudo de caso também são explanados.

E, por fim, no último capítulo descrevem-se as considerações finais e os possíveis trabalhos futuros que podem ser desenvolvidos.

[5] Plataforma Lattes Institucional <http://lattes.ufsc.br>.

2 Fundamentação Teórica

Este capítulo oferece embasamento teórico para a compreensão dos assuntos relacionados à integração de semântica e aplicação de ontologias às soluções de Business Intelligence. Logo abaixo, faz-se uma síntese dos elementos e processos que compõem a arquitetura de BI tradicional, e das iniciativas que tangenciam a área de gestão do conhecimento.

2.1 Arquitetura de Business Intelligence

Embora sejam úteis na automação dos processos operacionais e habituais das empresas, os sistemas OLTP (On-Line Transaction Processing) não são adequados para as necessidades informacionais. Esses sistemas transacionais servem como alavanca para as principais tarefas cotidianas e, geralmente, não são construídos para as análises ou para a gestão estratégica empresarial. Entre esses principais recursos e tecnologias que circundam o ambiente operacional, pode-se citar: sistemas ERP (Enterprise Resource Planning), que coordenam os processos e as atividades dentro da empresa, e CRM (Customer Relationship Management) operacional, que administra as necessidades dos clientes.

Os analistas de negócio ou dirigentes da instituição precisam de uma visão diferenciada sobre os dados primitivos, oriundos dos sistemas operacionais. De fato, o custo de análises cometidas sobre esses dados é maior se comparado a dados já derivados e trabalhados. A Tabela 1 a seguir traz as características dos dados primitivos, manuseados pelos sistemas OLTP, e dados derivados, manipulados pelos sistemas analíticos.

TABELA 1 – Características dos dados primitivos e derivados

Características \ Tipos de dados		Dados primitivos	Dados derivados
Utilização		Baseados em aplicações	Baseados em assuntos ou negócios
Nível de detalhamento		Mais detalhados	Resumidos ou refinados
Volatilidade	**Alterações**	Podem ser atualizados	Não são atualizados
	Consultas	Exatos em relação ao momento do acesso	Representam valores de momentos já decorridos ou instantâneos
Finalidade	**Público-alvo**	Comunidade funcional	Comunidade gerencial
	Foco	Voltado para transações	Voltado para análises
Processamento	**Forma de processamento**	São processados repetitivamente	São processados de forma heurística
	Requisitos de processamento	São conhecidos com antecedência	Não são conhecidos anteriormente
	Volume de dados	Pequena quantidade de dados utilizada em um processo	Grande quantidade de dados utilizada em um processo
Acessibilidade	**Probabilidade**	Alta probabilidade de acesso	Baixa ou modesta probabilidade de acesso
	Volume de dados	Acessados por unidade	Acessado um conjunto por vez
Disponibilidade		Alta disponibilidade	Disponibilidade atenuada
Controle e gerência		Gerenciados em sua totalidade	Gerenciados por subconjunto
Redundância		Não contemplam redundância	A redundância não pode ser ignorada
Estrutura		Estrutura fixa, conteúdo variável	Estrutura flexível

(Fonte adaptada: INMON, 2005)

As necessidades por sistemas de suporte à decisão, que tornam as informações mais acessíveis ao gestor, contribuíram para a evolução da área de inteligência de negócio (ou em inglês *Business Intelligence*). Segundo Wayne Eckerson, diretor de pesquisa do Data Warehouse Institute, inteligência de negócio pode ser definida:

> Business Intelligence ou Inteligência de Negócio consiste em processos, ferramentas e tecnologias necessárias para transformar dados e informação e informação em conhecimento e planejar a direção efetiva das atividades de negócios. (ECKERSON, 2006).

Para Ralph Kimball (2002), BI pode descrever em âmbito geral os recursos informacionais internos e externos da organização necessários para propiciar melhores decisões sobre o negócio. A inteligência de negócio é a transformação metódica e consciente dos dados de qualquer fonte de dados em novas formas de proporcionar a informação dirigida aos negócios e orientada aos resultados (BIERE, 2003).

Muitos dos conceitos de *Business Intelligence* estão comumente envolvidos com o *Data Warehousing.* O DW desempenha um papel fundamental em qualquer arquitetura de BI. Conforme Inmon (2005), os processos de inteligência de negócio sem a presença dele ficariam apenas na teoria. A Figura 2 abaixo mostra resumidamente a disposição dos componentes da arquitetura de BI, também chamada de Corporate Information Factory (CIF).

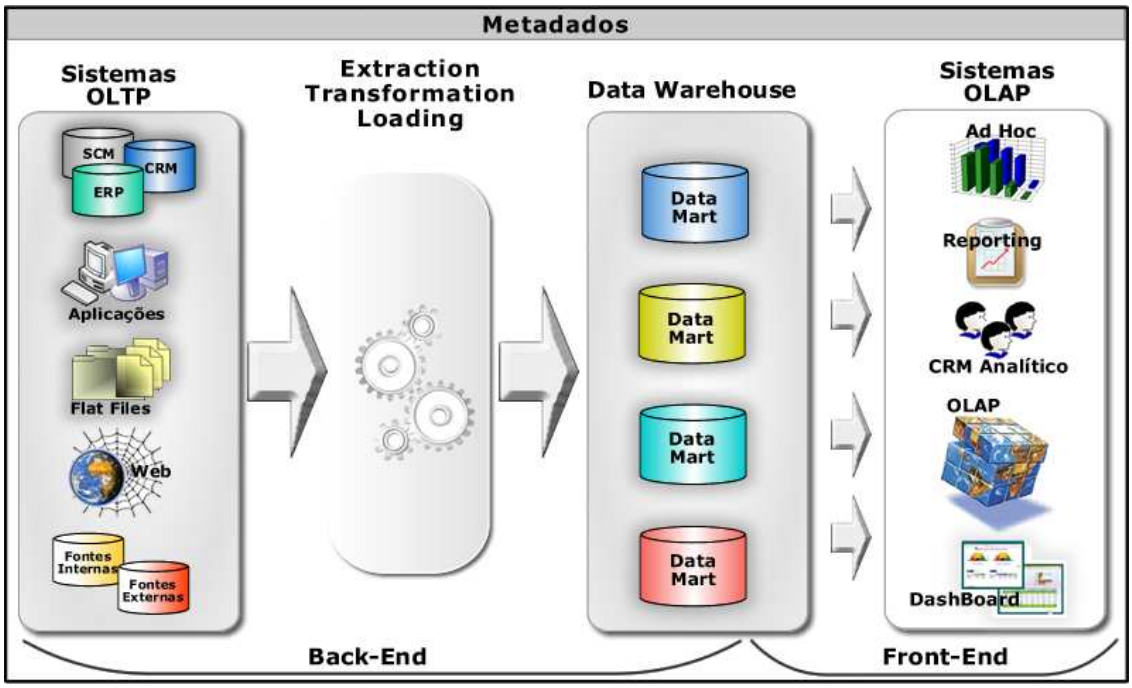


FIGURA 2 – Arquitetura de Business Intelligence Tradicional

A arquitetura anterior pode ser dividida em duas partes: Back-End (ou ainda Back-Room) e Front-End (ou também denominada de Front-Room). Kimball (1998) afirma que o Back-Room é o ponto de encontro dos administradores de banco de dados (DBAs - Data Base Administrator) e o Front-Room, dos administradores de negócio (MBAs - Master Business Administrator). Para Inmon (1997), o Back-End está atrelado aos produtores de informação, enquanto o Front-End aos consumidores de informação.

Ambas as divisões têm em comum um repositório de informações (DW), cuja construção deve-se aos componentes e processos de Back-End para o uso dos elementos de Front-End. As seções a seguir detalham essas duas divisórias.

2.1.1 Back-End

A parte Back-End da arquitetura compreende os elementos iniciais do processo de BI, isto é, a infra-estrutura e as técnicas necessárias para guiar a transformação e a integração dos dados num repositório útil às análises e à extração de conhecimento denominado Data Warehouse (DW).

2.1.1.1 Data Warehouse e Data Mart

Kimball define o DW como um grande processo em que há uma conglomeração de dados corporativos, nos quais os dados operacionais são estruturados especificamente para consultas, desempenho e facilidade de análises (2002).

Uma das definições mais citadas é a de autoria de Inmon (INMON, 2005), que afirma que DW é coleção de dados para suporte a decisão:

- **orientado a assuntos**: pois armazena informações organizadas em temas particulares de interesse;
- **integrado**: devido à capacidade de unificar as informações oriundas de fontes internas e externas da organização;

- **não volátil**: uma vez inseridas as informações no DW, não ocorrem mais alterações. As aplicações-cliente apenas realizam consultas às informações do repositório; e
- **variável em relação ao tempo**: por garantir histórico devido à não volatilidade, os dados têm diferentes valores associados ao tempo.

Um DW é também uma composição de Data Marts. Esses Data Marts são orientados a assuntos ou temas destinados a atender a uma área específica da organização. Geralmente são desenvolvidos para assistir a um departamento ou setor de uma empresa, e, por integrarem um menor conjunto de dados de um determinado contexto, apresentam menor custo de desenvolvimento inicial.

Existem duas abordagens a respeito da constituição de um DW:

- **top-down:** proposta por Inmon, sugere a construção do DW por completo e posteriormente a sua divisão em Data Marts; e
- **bottom-up:** proposta por Kimball, recomenda a construção de Data Marts que devem ser integrados para constituir o DW. Kimball (2002) introduz o conceito denominado *Arquitetura Bus*, em que as estruturas dimensionais devem estar em conformidade para que os Data Marts possam ser unidos.

Independentemente da abordagem adotada, são necessários três processos seqüenciais para a construção do Data Warehouse. Esses processos denominados Extração, Transformação e Carga - ETC (ou em inglês *Extraction*, *Transformation and Loading* - ETL) são apresentados na próxima seção.

2.1.1.2 Extraction, Transformation and Loading (ETL)

Os processos de Extração, Transformação e Carga, imprescindíveis na construção do DW, envolvem muitos componentes e técnicas da arquitetura de BI. Kimball (2004) comenta que os sistemas ETL consomem cerca de 70% dos recursos necessários para a implementação e a manutenção do DW. Dentre alguns dos elementos relacionados à ETL, listam-se: as fontes de dados operacionais nas quais os dados são capturados; a área de estagiamento (ou em inglês *Data Staging Area*), região onde os dados são transformados; os

metadados, que fornecem uma documentação dos dados e dos processos; a área de armazenamento de dados (em inglês *Operational Data Store*), que integra os dados operacionais, e outros.

Logo abaixo se resume separadamente cada um dos três processos.

1. **Extração** (Extraction): responsável por extrair os dados dos sistemas operacionais da empresa. Kimball (2004) declara que a extração denota a leitura e a compreensão das fontes de dados operacionais, e a cópia das partes que são necessárias para inserção na área de estagiamento. Essa área serve como um repositório temporário em que os dados são ajustados para a sua transição ao destino. Esses dados podem ser oriundos de inúmeras fontes: internas e externas, sistemas legados, arquivos, Web etc.

2. **Transformação** (Transformation): corresponde à rotina de limpeza, validação e preparação dos dados extraídos. Para garantir a qualidade das informações do DW, filtros e eliminações de inconsistências devem ser realizados na área de estagiamento. Além disso, deve-se optar por uma estrutura ou um formato comum para esses dados, visto que podem ser provenientes de fontes distintas. Dentre os exemplos de transformação de dados, pode-se citar a padronização do formato de datas, a troca de valores nulos para textos que representam o valor não informado e a adição de uma chave primária simples para cada tupla da base de dados (também chamada de chave artificial ou em inglês *surrogate key*).

3. **Carga** (Loading): ao final de todo o processo de transformação, os dados devem ser integrados ao núcleo do Data Warehouse. Basicamente há dois tipos de cargas, a inicial e a incremental. A inicial geralmente é mais prolongada, pois todos os dados devem ser inseridos pela primeira vez no DW. Já a incremental é realizada sazonalmente conforme as alterações dos dados no sistema operacional. Ambas preferencialmente devem ser executadas em períodos que não prejudiquem a execução dos processos

funcionais da empresa, visto que é uma transação com grande volume de dados.

Neste trabalho, os processos ETL utilizam como fonte de dados o próprio DW, embora a ferramenta desenvolvida não possua essa limitação.

2.1.1.3 Modelagem Dimensional

Um dos diferenciais evidentes entre os bancos de dados destinados aos sistemas operacionais e o Data Warehouse é a modelagem de dados. O modelo Entidade–Relacionamento (MER), projetado nos SGBDs relacionais, não deve ser utilizado para a construção do Data Warehouse. O desempenho e a facilidade de navegabilidade entre as tabelas para se apresentarem as informações são alguns dos motivos.

A alternativa para o MER é o modelo dimensional ou também chamado de modelo estrela. Conforme Kimball (2002), a modelagem dimensional é uma técnica de desenvolvimento que procura apresentar os dados em uma estrutura padrão que seja intuitiva e permita o acesso de alto desempenho. A idéia da modelagem dimensional está focada na denormalização da estrutura de dados, ou seja, visa-se à obtenção de um modelo simples para acesso às informações mesmo que haja redundância de dados. Quando a modelagem apresenta dimensões normalizadas, tem-se o chamado modelo *Snowflake*, que, embora seja legal, deve ser evitado. Um exemplo de modelo estrela que aborda um contexto de vendas é mostrado na Figura 3 a seguir.

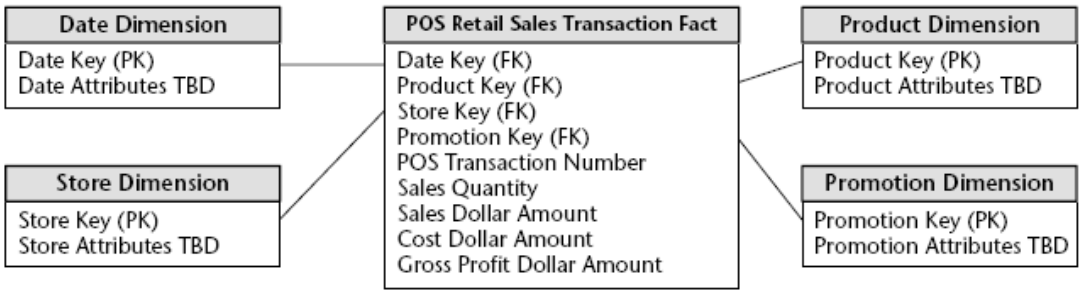


FIGURA 3 – Exemplo de modelo estrela

(Fonte: KIMBALL, 2002)

(O termo TBD (*to be determined*) usado na Figura 3 referece-se a outros atributos que poderiam ser de colocados no modelo)

O modelo estrela da Figura 3 possui quatro dimensões associadas a uma tabela de fato, que se encontra no centro da figura. As dimensões armazenam as descrições textuais do negócio, as quais servem como filtro na realização de consultas. As tabelas de fato contêm os valores numéricos ou medidas de negócio relacionadas às dimensões, isto é, fatos aditivos. Porém, é possível que a tabela de fato tenha informações não aditivas ou semi-aditivas sobre o assunto tratado, como, por exemplo, a tabela de fato da Figura 3 apresenta um atributo (POS Transaction Number) que representa o número do cupom fiscal da venda, que poderia ser utilizado para análises de cestas de compras (Market Basket Analysis[6]). Outra distinção é que as tabelas de fato são muito esparsas, exigindo muito espaço para armazenamento.

O DW agrega dados derivados dos sistemas operacionais e, por esse motivo, tem um nível de detalhamento de informações geralmente menor comparado a esses sistemas. Esse nível de detalhamento contido nas unidades de dados do DW, também chamado de granularidade, é um fator importante a ser considerado no projeto do modelo dimensional. Uma granularidade alta (*grão elevado*) oferece menor nível de detalhe, e conseqüentemente têm-se informações menos analíticas. O contrário, com uma granularidade baixa (grão pequeno) alcança-se um maior nível de detalhe e, por conseguinte, análises mais ricas.

2.1.2 Front-End

Um dos objetivos do Data Warehouse é tornar a informação o mais acessível possível para o usuário final (KIMBALL, 1998). A região que contém os componentes de interface com o Data Warehouse, dedicados às análises de negócio, é chamada de Front-End ou Front-Room.

[6] Market Basket Analysis – artifício utilizado para determinar a relação de compra casada entre produtos.

Inúmeras tecnologias são desenvolvidas em torno do Front-End a fim de atender à comunidade gerencial, entre as principais citam-se as ferramentas OLAP (On-Line Analytical Processing), Enterprise Information System[7] (EIS), Data Mining[8] e consultas Ad Hoc[9].

2.1.2.1 On-Line Analytical Processing (OLAP)

As ferramentas OLAP permitem que os usuários possam navegar de forma amigável pelo modelo multidimensional do Data Warehouse, ou seja, executar uma análise multidimensional. Esse acesso flexível às informações, combinado com o cruzamento das informações contidas nas dimensões do DW, forma o chamado Cubo OLAP ou ainda Hipercubo (KIMBALL, 1998; IMNON, 1997).

Existem diferentes variações de OLAP, conforme o tipo de base de dados que é usado para acesso (SELL apud HARRISON; 2006), são elas: ROLAP (Relational OLAP) – quando há uma acessibilidade através de banco de dados relacionais; MOLAP (Multidimensional OLAP) – quando aplicada em banco de dados multidimensionais; e HOLAP (Hybrid OLAP) – quando há uma junção das duas formas anteriores.

As operações comuns desempenhadas pelos sistemas OLAP são:

- **slice-dice:** acessa igualmente o Data Warehouse através de qualquer uma de suas dimensões. São processos de separação e de combinação das informações em diversas visões (KIMBALL, 1998);
- **drill-up e drill-down:** são operações inversas que permitem navegar até um nível de detalhe imediatamente superior (drill-up) ou inferior (drill-down). (INMON, 2005);
- **drill-across:** altera o nível de análise de uma dimensão, alternando um nível intermediário dentro de uma mesma dimensão, por exemplo, numa dimensão relacionada à Geografia, que é composta de país, região

[7] Enterprise Information System – Sistema de Informação Empresarial.
[8] Data Mining – Conjunto de metodologias e técnicas que visam à descoberta de conhecimento em grande volume de dados.
[9] Consulta *ad hoc* – Segundo Inmon (INMON, 2005), são os acessos casuais que manipulam dados conforme parâmetros ainda não utilizados, geralmente executados de forma iterativa e heurística.

geográfica, Estado e município, pode-se realizar uma análise no nível região geográfica e, posteriormente, buscar os municípios dessa região, saltando o nível Estado; e

- **drill-through:** ocorre quando o usuário faz análises de distintas visões proporcionadas por outras dimensões, por exemplo, o usuário realiza análises de indicadores pela dimensão geografia e posteriormente passa a analisar sobre a dimensão tempo.

Uma arquitetura de BI pode ter ainda uma camada de metadados, que é comum ao Back-End e ao Front-End. Os metadados são responsáveis pela documentação do processo de Data Warehousing e também pela catalogação dos dados, desde a sua origem até o seu destino e exibição pelo ferramental OLAP.

2.1.3 Contexto de BI nas Organizações

Percebe-se através da mídia que existe uma grande demanda por soluções de BI. Conforme a pesquisa realizada pela International Data Corporation (IDC), o mercado de ferramentas e aplicações de BI expandirá de 3,9 bilhões de dólares em 2003 para aproximadamente 5 bilhões de dólares em 2007. Isso denota uma taxa de crescimento de quase 5% ao ano, a qual é muito superior comparada aos índices de outros segmentos de mercado de software nos últimos anos. Levando-se em conta as vendas de hardwares, servidores e sistemas gerenciadores de banco de dados, o mercado do BI excede 100 bilhões de dólares anuais (ECKERSON, 2006).

O Data Warehousing Institute [10] (TDWI), referência mundial que faz premiações das melhores práticas em BI, apresenta muitos casos de sucesso:

- uma das maiores linhas áreas gerou 40 milhões de dólares em novos investimentos e poupou 31 milhões de dólares em custos no último ano devido a apenas quatro das 35 aplicações analíticas que atuam em seu ambiente de BI;

[10] TDWI <http://www.dw-institute.com>.

- uma empresa varejista de eletrônica atribui $1,3 milhão a uma solução de BI que garantiu a melhoria de sua variedade de produtos e controle de estoque. A mesma solução conservou também em um ano $ 2,3 milhões em seu inventário, um resultado de pedidos de entrega mais exatos ao seu fornecedor; e
- um departamento financeiro estadual conseguiu a aquiescência de seus impostos em 10 milhões de dólares em um ano, conciliando a satisfação dos pagantes graças a um novo recurso de BI.

O rumo dos investimentos em soluções de BI mencionadas acima não se encontra muito distante do cenário brasileiro, é o que indica a conferência “IDC Brasil Business Intelligence & Business Performance Management”, realizada em abril de 2006 e publicada pela ComputerWord (COMPUTERWORLD, 2006). Nessa edição é descrita uma estimativa de aumento nos investimentos no mercado de *Business Intelligence* de 25% a mais em relação a 2004. Outro ponto importante revelado é que 57% das 800 companhias de grande e de médio porte pesquisadas devem investir em soluções de BI em 2006.

2.2 Tecnologias Semânticas

Adicionar conhecimento ao seu negócio é um dos principais motivos que levam as organizações a investirem em recursos computacionais. Porém, tal investimento pode ser desvantajoso quando esses recursos não oferecem meios para se obterem benefícios ou informações relevantes sobre o mercado em que a empresa atua. Em muitos casos, isso acontece devido à forma como os dados estão estruturados ou definidos, e também como as ferramentas extraem e manipulam esses dados. Assim, muitas pesquisas são realizadas com o objetivo de favorecer um ambiente para que as máquinas, através da incorporação de semântica nos dados, possam interpretar e gerar conhecimento útil para um fim qualquer.

A transformação de dados em informações e por sua vez em conhecimentos está relacionada ao significado que essas entidades representam e ao contexto no qual se situam, como é mostrado por Watson (2003) na Figura 4.

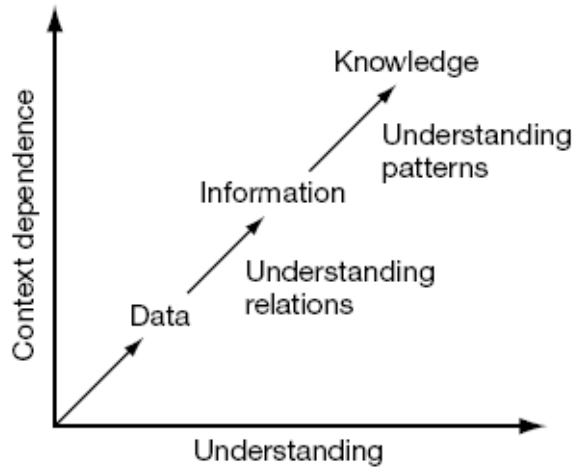


FIGURA 4 – Relacionamento entre Contexto e Compreensão
Fonte: (WATSON, 2003)

Por meio da Figura 4 observa-se a relação direta entre a dependência de domínio e o grau de entendimento de um conceito, ou seja, aumentando-se o contexto no qual um determinado elemento está inserido, inclui-se mais sentido ou significado ao seu conteúdo.

Esses diferentes níveis que o dado sofre até alcançar o conhecimento podem ser obtidos através de correlações, interpretações ou regras entre conceitos de um dado domínio. Para que esse desígnio seja atingido pelos sistemas informatizados, pode-se contar com o auxílio de ontologias. Araújo (2003) alega que ontologias são aplicáveis na representação computacional de um domínio de conhecimento de modo que a comunicação entre pessoas e computadores se realize automaticamente.

2.2.1 Definição de Ontologias

Uma definição que, apesar de ter mais de dez anos, ainda é muito empregada na área de tecnologia de informação (TI) é a de Thomas Gruber (1993). Ele afirma que uma ontologia é uma especificação explícita e formal de

uma conceitualização compartilhada. O termo “conceitualização” refere-se a um modelo abstrato de algum fenômeno que identifique seus conceitos relevantes. A palavra “explícita” significa que os tipos de conceitos usados e as limitações do uso desses conceitos devem ser definidos explicitamente. A terminologia “formal” refere-se à ontologia que deve ser passível de ser processada por máquinas. Já o termo “compartilhada” estabelece que a ontologia está ligada a um conhecimento comum, isto é, esse conhecimento não deve ser restrito a alguns indivíduos, mas aceito por um grupo de pessoas (STUDER; BENJAMINS; FENSEL, 1998).

Uma ontologia define um vocabulário comum e os conceitos (ou significados) usados para descrever e representar uma área de conhecimento (FENSEL; VAN HARMELEN, 2003). Pérez e Benjamins (1999) consideram as ontologias como uma composição de conceitos ou classes, instâncias, relacionamentos, funções e axiomas. Noy e McGuinness (2000) articulam que as ontologias se assemelham à modelagem orientada a objetos. Em contrapartida, o modo de desenvolvimento é diferente, pois a construção do modelo ontológico fundamenta-se no significado das estruturas de classes, e o modelo orientado a objetos baseia-se nos métodos ou nas operações que as classes têm para praticar uma ação.

2.2.2 Tipos de Ontologias

Segundo Guarino (1998), as ontologias são classificadas quanto à sua generalidade ou ao nível de representação. A Figura 5 exibe os quatro tipos de ontologias que são estabelecidos por esse autor.

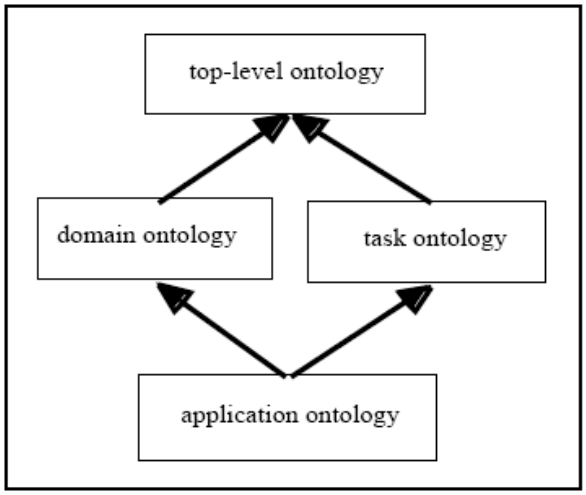


FIGURA 5 – Tipos de Ontologias e suas relações
FONTE: (GUARINO , 1998)

- **Ontologia de Alto Nível** (top-level ontology): descrevem conceitos gerais que independem de um problema particular, tais como tempo, espaço, matéria, eventos, etc. (GUARINO, 1998).
- **Ontologia de Domínio** (domain ontology): representa um conhecimento válido para um tipo de domínio particular, como medicina e indústria farmacêutica. São especializações das ontologias de Alto Nível (FENSEL, 2001; GUARINO, 1998).
- **Ontologia de Tarefa** (task ontology): oferece as terminologias para se modelarem tarefas ou atividades. Está ligada à resolução de problemas descrevendo o vocabulário de uma tarefa ou atividade, como vendas (ARAÚJO, 2003; GUARINO, 1998).
- **Ontologia de Aplicação** (application ontology): são utilizadas para descrever os conceitos dependentes entre a Ontologia de Domínio e a Ontologia de Tarefa. Representam os papéis desempenhados pelas entidades de domínio (ARAÚJO, 2003; GUARINO, 1998).

Pérez e Benjamins (1999) adicionam mais um tipo de ontologia a essa classificação: a ontologia de método, que fornece mais relações para definir os processos da Ontologia de Tarefa, como explanado acima.

Este projeto em particular envolve uma ontologia de domínio no contexto de ensino superior, em que as entidades conceituais que figuram são universidade, professores, estudantes, pesquisadores, entre outras.

As ontologias são objetos de estudo de muitas áreas e oferecem diversos benefícios como compartilhamento de conhecimento, representação, recuperação e tratamento da informação na Web, que favoreceu boa parte de sua popularidade.

A Web Semântica é uma das pesquisas que se fundamentam em ontologias para oferecer esses benefícios.

2.2.3 Web Semântica

Através da Internet, milhares de recursos eletrônicos, distribuídos em todas as partes do mundo, estão acessíveis para leitura e manipulação por pessoas e máquinas. No entanto, a busca por esses recursos é prejudicada devido não só ao volume de documentos presentes na Web mas porque boa parte dos seus elementos apresentam-se de modo semi-estruturado (MELLO et al., 2006; ARAÚJO, 2003).

As buscas por palavras-chave efetuadas por sites como Alta-Vista, Yahoo e Google são problemáticas. Antoniou e Harnelen (2004) levantam quatro limitações desse tipo de busca: 1) a possibilidade de poucas ou quase nenhuma ocorrência encontrada; 2) a baixa precisão no retorno das consultas; 3) os resultados são sensíveis às palavras-chave; e 4) a integração do conteúdo que está disperso em vários documentos necessita de várias consultas.

Uma iniciativa que pretende solucionar os problemas da Web atual, idealizada por Berneers-Lee (2001a), baseia-se no uso de ontologias para incorporar mais semântica ao conteúdo das páginas e aumentar a capacidade de processamento, intercâmbio e procura por informações – a Web Semântica. Berneers-lee define a Web Semântica como um extensão da presente Web, em que as máquinas possam cooperar com as pessoas de forma automática. Essa nova versão da Web foi dividida em várias camadas para dar suporte ao desenvolvimento e pode ser visualizada na Figura 6 a seguir.

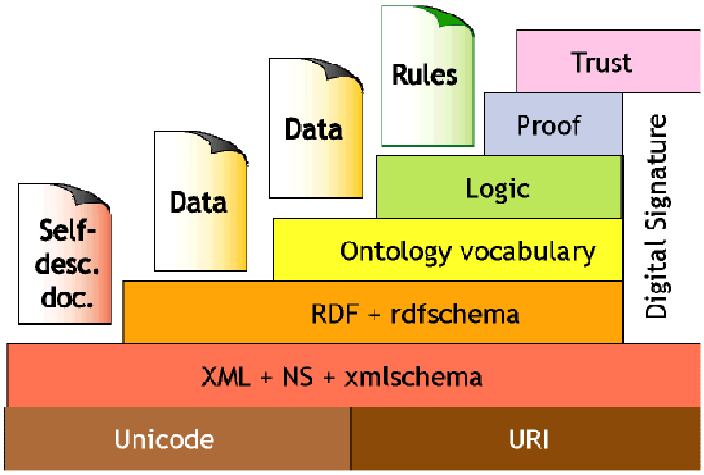


FIGURA 6 – Arquitetura da Web Semântica
(Fonte: BERNEERS-LEE , 2001a)

Na Figura 6 cada camada inferior fornece subsídio para a aplicação das camadas acima. Na prática, as camadas superiores ainda são alvo de estudo e não se encontram totalmente desenvolvidas.

A camada base da Web Semântica serve para padronizar os conteúdos (Unicode) e para identificar de modo unívoco os recursos da Web (URI). A camada XML[11] (eXtensible Markup Language), NameSpace[12] e XML Schema[13] possibilita a estruturação, a troca e a validação dos documentos eletrônicos.

Imediatamente acima, RDF[14] (Resource Description Framework) e RDF Schema[15] permitem definir metadados (dados sobre dados) e adicionar mais semântica a um documento, com a vantagem de não precisar referir-se à sua estrutura (FENSEL, 2001). Na camada Ontologias, determinam-se os vocabulários compartilhados que representam um certo domínio para dar mais metainformação e interoperabilidade (BERNEERS-LEE, 2001b). A camada Logic realça as linguagens de ontologias e permite a adição de regras de dedução, as conversões de termos e a lógica de predicado (BERNEERS-LEE, 2001b). A camada Proof responsabiliza-se pela execução dos processos dedutivos e pela validação das camadas inferiores (BERNEERS-LEE, 2001b). Por fim, a última camada (Trust)

[11] XML <http://www.w3.org/XML>.
[12] Namespace <http://www.w3.org/TR/xml-names11>.
[13] XML Schema <http://www.w3.org/XML/Schema>.
[14] RDF <http://www.w3.org/RDF>.
[15] RDF Schema <http://www.w3.org/TR/rdf-schema>.

será auxiliada pelas assinaturas e pelos certificados digitais para garantir veracidade e integridade no uso e no compartilhamento de ontologias.

Os vários níveis de formalismos propostos pela Web Semântica podem contribuir para a representação de conhecimento, e são usados como base nesta pesquisa.

2.3 Representação de Conhecimento

Para que possa haver representação de conhecimento, é necessário que exista uma linguagem formal que o especifique. Este capítulo, além de relatar o motivo por que a linguagem OWL foi escolhida para a representação de ontologias, propõe-se também a realizar um apanhado dos principais formalismos recomendados para essa finalidade. Os critérios utilizados na seleção das linguagens são:

- expressividade semântica – deve ser capaz de possuir construções suficientes para representar qualquer tipo de ontologia e área de interesse;
- padrão aberto – segundo a União Européia (EIF, 2004), um padrão aberto deve fornecer disponibilidade, liberdade de uso em todo o mundo, extensibilidade por parte do usuário final e uma organização que dê suporte ao padrão;
- disponibilidade de ferramentas open source - devem existir softwares livres para manuseio da linguagem de modo que se possa escolher qual o mais adequado pra ser aproveitado no desenvolvimento; e
- adequação com a Web Semântica – a arquitetura SBI prevê um módulo relacionado a serviços Web semânticos (*Web Semantic Services*). Desta maneira, a linguagem deve oferecer um fácil intercâmbio e integração com o contexto Web, no qual boa parte dos recursos são construídos com linguagens de marcação. Por essa razão, as linguagens de estudo são baseadas na pilha de recomendações do W3C para a Web Semântica.

Vale lembrar que a arquitetura está fracamente acoplada a uma linguagem específica. Como sugerido por Sell (2006):

A arquitetura poderá suportar soluções de BI utilizando a linguagem de preferência de uma organização, no entanto, quanto mais expressivo for o formalismo adotado, maiores as potencialidades de recomendações e inferências sobre os resultados de uma consulta.

2.3.1 eXtensible Markup Language (XML)

O XML (eXtensible Markup Language) é uma linguagem de marcação oriunda da SGML, recomendada pelo W3C (World Wide Web Consourtium), que visa obter uma maneira flexível de estruturar documentos para compartilhamento de informações.

Para que haja troca de informações, em geral, aconselha-se que o documento tenha um vocabulário comum e válido entre as entidades envolvidas. XML por si só não oferece esse recurso, apenas permite que marcações ou tags sejam criadas conforme a pretensão do usuário. Para tanto, o W3C recomenda o uso conjunto de outros tipos de documentos, como o DTD (Document Type Definition) ou ainda o XML Schema.

2.3.1.1 XML Document Type Definition (XML DTD)

A validade de um documento XML pode ser obtida associando-o a uma definição denominada XML Document Type Definition (DTD). Através de seu emprego, consegue-se atribuir formas gramaticais ao documento. Portanto, o XML deve seguir o padrão de sintaxe estabelecido pelo DTD.

Mesmo que forneça um certo nível de validação para XML, o DTD, além de possuir limitações sintáticas, não oferece expressividade semântica, que é o principal problema relacionado ao intuito da pesquisa. Entre algumas limitações sintáticas, pode-se destacar a cardinalidade limitada a três tipos (0 a 1, 0 a N e 1 a N) e a validação aplicável ao tipo “String”.

2.3.1.2 XML Schema Definition (XML(S))

Para suprir as deficiências do DTD, conferindo mais minúcias ao documento XML, pode-se utilizar o XML Schema[16] como alternativa. Esse, por sua vez, tem a vantagem de ser escrito com a própria sintaxe de XML e permite extensões e uso de tipos de dados, primitivos e derivados.

Embora seja possível garantir mais restrições e detalhes à estrutura do arquivo, o XML Schema tem uma capacidade limitada para descrever relações entre os recursos representados e, conseqüentemente, não possui forte expressividade semântica (DAML, 2002a). Então, mesmo sendo válido e tendo viabilidade para o intercâmbio e para a estruturação da informação, o XML possui várias lacunas que impedem sua escolha, haja vista que:

a) apenas oferece uma especificação sintática e em forma hierárquica, sem definir um vocabulário particular para declaração de ontologias. Logo, cabe a um usuário especialista conceber o modelo e o vocabulário necessários para a definição da ontologia;
b) é possivel criar algumas formas semanticamente equivalentes para expressar os conceitos de interesse, ou seja, uma relação de muitos para um (W3C, 2004a). Como exemplo permite-se estruturar relações de atividades profissionais entre pessoa e empresa das seguintes formas não ambíguas:

Listagem 1 – Primeira forma de formatar XML para relação profissional

```
<Instituicao sigla = “UFSC” >
    <Pessoa nome = “João” vinculo= “Celetista” />
    <Pessoa nome = “Pedro” vinculo= “Servidor Público” />
</ Instituicao >
```

Listagem 2 – Segunda forma de formatar XML para relação profissional

```
<Pessoa nome = “João” >
        < Instituicao sigla = “UFSC” >
                <Vinculo> Celetista </Vinculo>
        </ Instituicao >
</Pessoa>
<Pessoa nome = “Pedro” >
```

[16] XML Schema <http://www.w3.org/XML/Schema>.

```
        < Instituicao sigla = “UFSC” >
                <Vinculo> Servidor Público </Vinculo>
        </ Instituicao >
</Pessoa>
```

c) faz-se necessária a elaboração de outro tipo de documento, DTD ou XML Schema, como visto anteriormente, para a definição de um vocabulário específico e as combinações de valores, marcações ou tags válidas. Embora XML Schema ou DTD sejam suficientes para trocar dados entre as partes que já tenham acordado de antemão as definições, sua falta de semântica impede que as máquinas executem confiantemente essa tarefa dada a novos vocabulários (W3C, 2004b); e
d) como conseqüência dos itens supracitados, tornam-se dificultosos o seu reuso e o compartilhamento de ontologias entre sistemas.

XML(S) atua no segundo nível da arquitetura da Web Semântica, juntamente com o NameSpace, como apresentado na Figura 6. Para complementar suas características, o W3C recomenda RDF para o terceiro nível.

2.3.2 Resource Description Framework (RDF)

Com o objetivo de oferecer mais significado e descrição aos recursos disponíveis na Web, de modo que aplicações possam compreendê-los e efetuar o intercâmbio desses recursos, o W3C propõe um framework conhecido como Resource Description Framework (RDF). Esses recursos podem abranger não somente elementos da Web mas também qualquer conceito que se deseja modelar.

O RDF situa-se na terceira camada da arquitetura Web Semântica e é um modelo baseado em grafos que utiliza a sintaxe XML para representação da informação de qualquer domínio. A estrutura de grafos do RDF é composta de uma coleção de triplas que consistem em sujeitos, predicados e objetos (W3C, 2004b). Assim, uma parte das listagens da seção anterior pode ser representada através da seguinte notação, conforme vista na Figura 7 a seguir.

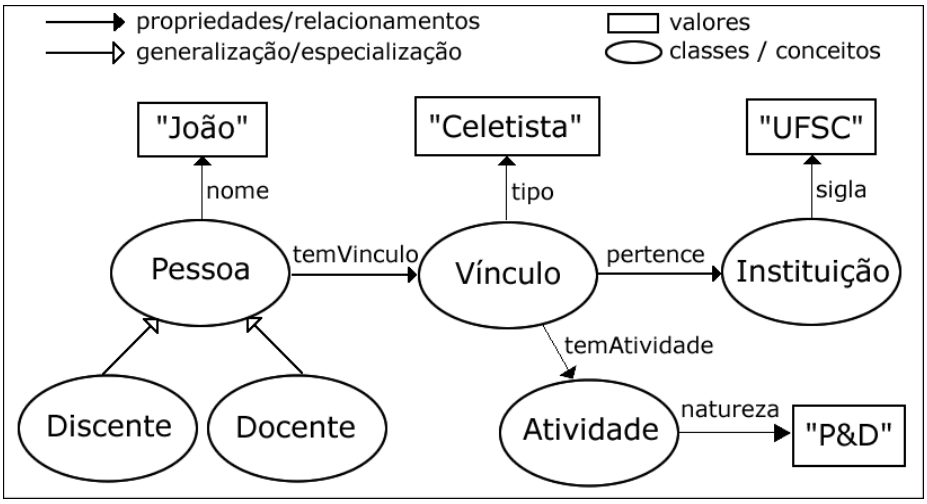


FIGURA 7 – Exemplo de uma notação do modelo RDF

A Figura 7 acima possui algumas triplas RDF. Dentre elas, nota-se que o nodo que representa o conceito *Pessoa* (Sujeito) tem uma propriedade *nome* (Predicado) cujo valor é *João* (Objeto). Ainda com relação a esse conceito *Pessoa*, tem-se mais uma propriedade *temVinculo* que referencia outro conceito chamado *Vinculo*, e assim sucessivamente. Pode-se ter ainda subconceitos ou subclasses como *Discente* e *Docente*, que são especializações da classe *Pessoa*.

RDF, diferentemente da estrutura de árvore de XML, permite navegar por vários caminhos do grafo que formam uma rede semântica para obter uma informação contida em um nodo (W3C, 2004a).

2.3.2.1 Resource Description Framework Schema (RDF(S))

O RDF Schema, assim como o XML Schema, impõe a validação do modelo RDF e, além disso, permite a definição de vocabulários, hierarquia de classes, propriedades e alguns tipos de restrições.

Embora útil para descrever metadados e recursos na Web, o RDF(S) possui algumas implicações que dificultam sua escolha. Conforme Broeksta (2001), há uma semântica formal limitada para as primitivas do RDF Schema, e sua expressividade é insuficiente para desenvolver modelos ontológicos e de

raciocínio. De fato, essa insuficiência é relatada pelo próprio W3C, quando afirma que o RDF perde em termos de expressividade para dar suporte à escalabilidade de recursos na internet. Essa perda de expressividade pode ser indicada pela ausência de algumas relações entre classes (disjunção e igualdade), algumas características de propriedades (inversão, cardinalidade, transitividade), entre outras.

2.3.3 Web Ontology Language (OWL)

Para tratar dos problemas mencionados acima, permitindo construções semânticas mais ricas, o W3C propõe uma extensão de RDF denominada de Web Ontology Language (OWL).

OWL surgiu de iniciativas européia e americana, respectivamente, Darpa Agent Markup Language (DAML)[17] e Ontology Inference Layer (OIL)[18], e é a parte mais recente da pilha de recomendações do W3C relativa à Web Semântica.

Para viabilizar a representação de ontologias, a linguagem é composta de três sublinguagens:

- *OWL Lite* – subconjunto mais limitado de construções de OWL necessário para a definição de ontologias, com hierarquias e restrições simples. Acrescenta ao modelo RDF as igualdades e as disjunções entre classes e a cardinalidade 0 e 1;
- *OWL DL* - dá subsídio a OWL para a obtenção de mais expressividade, com completude (garantia de que todas as conclusões serão tomadas) e decibilidade (todas as conclusões serão efetuadas em tempo finito). OWL DL acrescenta todas as demais construções possíveis da linguagem não contidas em OWL Lite. O termo DL refere-se à área de lógica de descrição (em inglês *Description Logics*); e
- *OWL Full* – apresenta todas as características de OWL DL com toda a liberdade sintática de RDF.

Apesar de ser uma revisão de DAML e OIL, OWL não traz completamente as estruturas para a representação de ontologias desses dois formalismos. A

[17] DAML <http://www.daml.org>.
[18] OIL <http://www.ontoknowledge.org/oil>.

Tabela 2 mostra um comparativo de três autores entre os formalismos vistos até aqui, destacando as características (dimensões) que cada um oferece. Essa tabela está atualizada com as versões mais recentes dessas linguagens (XML1.1, RDF 1.0, DAML+OIL 2001 e OWL 2004).

TABELA 2 – Comparativo entre os formalismos de representação de ontologias

Dimensão / Formalismo		XML(S)	RDF(S)	DAML+OIL	OWL
Definição do Contexto		**X**	**X**	**X**	**X**
Definições de Classes	**Objetos e Propriedades**		**X**	**X**	**X**
	Herança		**X**	**X**	**X**
Restrições de Propriedades e Elementos	**Faixa ou escala dos tipos de valores**	**X**	**X**	**X**	**X**
	Domínio	**X**	**X**	**X**	**X**
	Cardinalidade	**X**		**X**	**X**
	Propriedades qualificadas			**X**	
Tipos de Dados e Instâncias	**Tipos Básicos**	**X**		**X**	**X**
	Enumeração de propriedades e elementos	**X**		**X**	**X**
	Instâncias	**X**	**X**	**X**	**X**
Coleções de Dados	**Listas limitadas**		**X**	**X**	**X**
	Ordenação dos elementos	**X**	**X**	**X**	**X**
Expressões Lógicas e Conjuntos	**Negação**			**X**	**X**
	Disjunção			**X**	**X**
	Conjunção		**X**	**X**	**X**
	Equivalência			**X**	**X**
Tipos de Propriedades	**Inversa**			**X**	**X**
	Transitiva			**X**	**X**
	Simétrica				**X**
Reificações			**X**	**X**	**X**

(Fonte adaptada: DAML, 2002b; GIL; RATNAKAR, 2002)

Definição de Contexto: a utilização do NameSpace é uma maneira para definir o contexto em que as diferentes terminologias empregadas na ontologia estão situadas (GIL; RATNAKAR, 2002). Todos os formalismos relatados apresentam esse recurso, e, assim, pode-se diferenciar e interpretar um determinado termo de acordo com o seu contexto.

Definições de Classes: o XML(S) não define explicitamente construções de classes e subclasses. Existe a possibilidade de derivação de elementos por extensão ou por restrição de tipos na qual se pode interpretar como classes e subclasses. Entretanto, a ausência de um padrão para representar classes e propriedades resulta na ambigüidade ao traduzir o modelo XML para um modelo semântico (GIL; RATNAKAR, 2002), um exemplo disso é que, em vez de definir uma classe, poderia-se definir uma propriedade de classe. Já os demais formalismos, RDF(S), DAML+OIL e OWL, apresentam explicitamente construtores para classes/subclasses e propriedades/subpropriedades.

Restrições de Propriedades e Elementos: todas as linguagens possuem determinação dos tipos de dados para as propriedades e o domínio ao qual pertencem, somente RDF(S) não impõem restrições de cardinalidade em sua especificação. Dessa forma, a propriedade (ou predicado) *nome*, cujo domínio é o conceito *Pessoa* visto anteriormente na Figura 7, poderia ter múltiplos valores. OWL não trouxe em sua especificação as restrições qualificadas existentes em DAML+OIL (DAML, 2002b).

Tipos de Dados e Instâncias: o XML(S) dispõe de tipos de dados que são utilizados também em DAML+OIL e OWL, em que são compatíveis com Banco de Dados. Já os tipos de dados em RDF(S) são essencialmente literais (string). O RDF(S) não possui enumerações de propriedades e de valores. Dessa forma, não se pode dizer que a propriedade *tipo* do conceito V*ínculo***,** visualizado na Figura 7, tem somente os valores “Celetista”, “Estatutário” e “Bolsista”, por exemplo. Todos os formalismos possuem definições de indivíduos ou instâncias do seu modelo.

Coleções de Dados: todas as linguagens permitem especificar a ordem dos elementos de uma coleção determinada. O XML(S) não oferece construtores para a definição de coleções restritas ou limitadas de elementos. Esse recurso poderia

ser utilizado, por exemplo, para declarar que uma instância do conceito *Instituição* tem exatamente 10 instâncias do conceito *Vinculo* do tipo “Celetista”.

Expressões Lógicas e Conjuntos: apenas DAML+OIL e OWL possuem todas as expressões lógicas relatadas na Tabela 2. O RDF(S) oferece apenas a conjunção através da combinação de herança múltipla.

Tipos de Propriedades: as propriedades transitivas e inversas são vistas somente nas especificações de DAML+OIL e OWL. Além dessas características, a OWL ainda suporta a simetria entre propriedades.

Reificações: a partir da estrutura de grafos proporcionada pelo framework RDF(S), os sujeitos de uma determinada sentença podem ser também utilizados como objetos de outra sentença. Na Figura 7, o conceito *Vinculo* exerce a função de objeto (ver predicado *temVinculo* do conceito *Pessoa*) e, simultaneamente, desempenha a função de sujeito de outra sentença (ver predicado *pertence* desse conceito).

OWL define os termos da ontologia pelo uso de hierarquias, uniões, disjunções, intersecções, complementos e outras operações e tipos de estruturas sobre classes e propriedades. Todavia, não permite produzir regras ou sentenças axiomáticas entre classes e propriedades para a realização de outros tipos de inferência. Pode-se pegar como exemplo uma ontologia de um domínio que expressa relações familiares, na qual se tem conceitos que representam *Pai*, *Mãe* e *Filho.* Para que se possa deduzir uma nova relação tal como *Irmão*, através das construções de OWL, uma das soluções possíveis é definir que *Irmão* é uma subclasse do conceito *Filho*. Assim, determina-se uma hierarquia entre o conceito *Filho* e *Irmão* que na realidade não deveria existir, pois, dado um novo grau de parentesco, novas classes e subclasses deveriam ser criadas.

Uma saída é estabelecer uma regra que assegure que todo indivíduo filho da mesma mãe, exceto ele próprio, seja um *Irmão*. Da mesma forma, poderia ser utilizada a mesma regra que define o conceito *Irmão* para deduzir o conceito *Tio* (qualquer indivíduo que seja *Irmão* do *Pai* ou *Irmão* da *Mãe*), e assim sucessivamente. OWL exclusivamente não consegue resolver essa problemática sem utilizar taxonomias e operações de conjuntos, pois não oferece recursos explícitos para a aplicação de regras.

Duas linguagens de marcação que podem ser combinadas à OWL para dar essa expressividade adequada são: 1) RuleML (Rule Markup Language) (RULEML, 2000); e 2) SWRL (Semantic Web Rule Language) (SWRL, 2004) futuramente. Ambas as linguagens oferecem um conjunto de declarações antecedentes, que definem as premissas das regras e um conjunto de fatos conseqüentes, caso essas regras sejam satisfeitas.

Tanto RuleML como SWRL atuam no nível lógico, isto é, na camada Logic da arquitetura da Web Semântica e adicionam mais representatividade à OWL, que, por sua vez, age na camada de Ontologias vista na Figura 5.

Por se tratar de uma proposta recente e que ainda não está totalmente concluída pelo W3C, atualmente SWRL praticamente não apresenta muitos mecanismos de inferência *open source* para suporte. Dentre as ferramentas disponíveis no mercado mais referenciadas que trabalham com OWL e possuem base para aplicação de regras, indica-se o Jena Semantic Web Framework (JENA, 2006). Mesmo que o Jena ainda não dê suporte à SWRL, ele possui uma linguagem própria e de fácil entendimento que permite a composição de regras sobre a ontologia. Por esse motivo, esse mecanismo de inferência foi selecionado para auxiliar no desenvolvimento do sistema de informação relacionado a este trabalho.

2.4 Considerações sobre o Capítulo

O conhecimento adequado para a gestão dos negócios organizacionais pode ser mais bem adquirido caso existam meios ou tecnologias que disponibilizem as informações necessárias. As mudanças de enfoque oferecidas pelos processos e recursos da arquitetura de BI assessoram a tomada de decisão e introduzem visões analíticas a qualquer área de interesse. Com isso, a inteligência de negócio, que pode ser entendida como um conjunto de processos, estratégias e ferramentas associados à administração e à tomada de decisão, pode ser continuamente atingida e aperfeiçoada nas organizações.

Entre os componentes da arquitetura de BI tradicional, o DW é visto como elemento central e divide a arquitetura em duas partes: 1) Back-End (ou ainda

Back-Room); e 2) Front-End (ou Front-Room). O DW contém informações históricas organizadas em áreas de negócio e integradas em um repositório comum. Todas essas informações são extraídas, preparadas e armazenadas no DW através de três processos contínuos denominados Extração, Transformação e Carga, situados na região Back-End. Os recursos de Front-End como as ferramentas OLAP são responsáveis pelas consultas cometidas ao DW. Com o objetivo de agilizar essas consultas, o DW apresenta um modelo diferente do usado em base de dados transacionais, chamado de modelo dimensional ou modelo estrela. Esse modelo é caracterizado pela denormalização e, através de dimensões e de tabelas de fatos, provê fácil navegabilidade sobre as informações em troca de redundância dos dados.

Além das possíveis análises estratégicas que se pode alcançar com o DW, é necessário expressar de alguma forma o conhecimento organizacional para que tecnologias possam manipulá-lo. Uma possível solução é o uso de ontologias, as quais muitos projetos e pesquisas adotam. As ontologias possuem um conjunto de propriedades e relacionamentos entre conceitos que dão semântica ao universo de discurso tratado. Para representar o conhecimento através das ontologias, este trabalho segue a pilha de formalismo da Web Semântica, cuja linguagem mais propícia atualmente é a OWL.

A linguagem OWL não oferece suporte para regras, e por esse motivo dificulta que deduções semânticas possam ser feitas. Outras linguagens devem ser utilizadas para esse propósito, como SWRL ou ainda RuleML. Porém, na prática, essas linguagens ainda oferecem poucos mecanismos de inferência desenvolvidos para seu uso. Adotou-se um mecanismo de inferência, denominado Jena Semantic Web Framework, que, apesar de não ter suporte para SWRL e para RuleML, possui uma linguagem nativa para a definição de regras.

3 Uso de Ontologias no processo de Business Intelligence

Neste capítulo é relatada a arquitetura SBI, juntamente com algumas contribuições e tecnologias aplicadas aos componentes pertencentes ao escopo deste trabalho. Após esse relato são descritos o sistema desenvolvido e sua aplicabilidade em uma ontologia e em um DW de contexto acadêmico.

3.1 Arquitetura SBI e Tecnologias Utilizadas

A arquitetura SBI (SELL, 2006), a qual é visualizada novamente na Figura 8 abaixo, possui diversos módulos fracamente acoplados e dispersos em cinco camadas. Visto que uma nova abordagem é empregada sobre essa arquitetura, adicionou-se um sexta camada que já pode ser vista também nessa figura. As camadas são:

- **Fonte de Dados** – corresponde ao DW da organização;
- **Acesso e Persistência** – adicionada para facilitar a comunicação com as fontes de dados e desacoplar ainda mais os módulos que se comunicavam diretamente com o DW.
- **Repositórios de Ontologias** – possui três repositórios ontológicos para análises e deduções semânticas sobre a camada inferior: Ontologia do Domínio, Ontologia BI e Ontologia de Serviços. A Ontologia de Serviços não faz parte do foco deste trabalho e não é explanada;
- **Mecanismos de Inferência** – envolvem os aplicativos de manipulação dos formalismos usados na representação de ontologias;
- **Módulos Funcionais** – responsáveis pela coordenação das inferências executadas na camada Mecanismos de Inferência e Repositório de Ontologias, e pela acessibilidade por parte das ferramentas-cliente da camada superior; e
- **Clientes** – onde as consultas *ad hoc* e as diversas formas de realizar OLAP são feitas na arquitetura SBI.

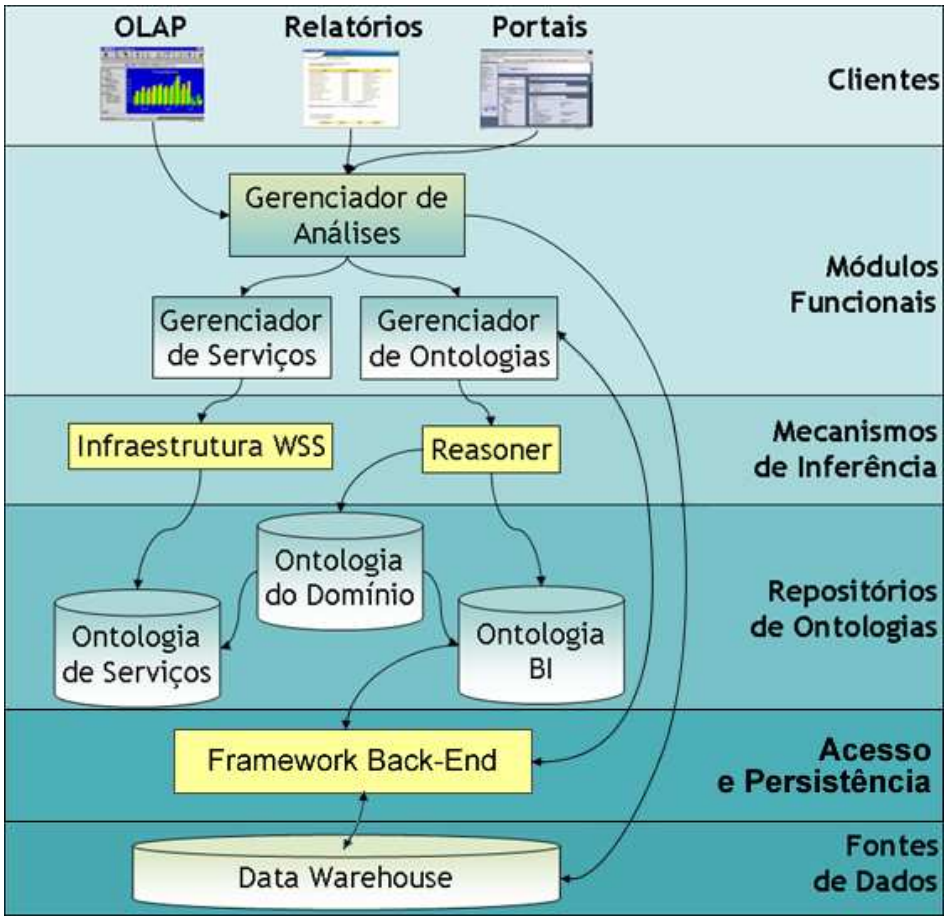


FIGURA 8 – Arquitetura Semantic Business Intelligence (SBI) adaptada
(Fonte adaptada: SELL, 2006)

Somente os elementos do foco desse trabalho foram adpatados na Figura 8 acima. A principal mudança está relacionada ao Gerencidor de Ontologias, que deve ter interação com o componente da camada Acesso e Persistência. A Ontologia BI, que antes era associada diretamente às fontes de dados, agora possui o intermédio desse componente, que acessa as fontes de dados pelo o mapeamento contido nessa ontologia.

A seguir, serão descritos as camadas da arquitetura SBI com seus respectivos módulos e tecnologias utilizados no projeto.

3.1.1 Fonte de Dados

Contém as informações de negócio organizadas em Data Marts, utilizados para compor a Ontologia do Domínio. Ou seja, as fontes de dados possuem o conjunto de valores necessários para a criação de instâncias da Ontologia de Domínio. Neste trabalho, essa camada também é utilizada para armazenamento de conclusões baseadas em regras sobre as instâncias da ontologia. Logo, dois tipos de modelos devem estar presentes nessa região: o modelo dimensional, projetado pela organização na criação do DW; e o modelo tripla, uma estrutura adicional representada por uma tabela de fato, na qual se propõe para armazenar as inferências semânticas.

O modelo dimensional está ilustrado na seção que aborda o estudo de caso neste capítulo. O modelo tripla proposto, cuja finalidade é guardar as derivações da ontologia, é apresentado na Figura 9 a seguir. Esse modelo é composto de uma tabela de fato que associa quaisquer duas dimensões a um predicado, analogamente ao modelo sujeito-predicado-objeto de RDF (W3C, 2004b) descrito no capítulo 2. O Sujeito e o Objeto do modelo tripla são referências para qualquer uma das chaves artificiais das dimensões do DW. Ora uma determinada chave artificial pode ser sujeito, ora ela pode ser objeto de outra tripla. O predicado possui os termos descritivos que são gerados nas conclusões das regras de negócio sobre a ontologia. Esses predicados são equivalentes às medidas não aditivas de um modelo dimensional.

Já que qualquer dimensão pode ser associada ao fato tripla, todas as restrições de integridade ou relações entre as dimensões e o fato tripla são garantidas pela aplicação. Isso faz com que o modelo tripla tenha um acoplamento fraco com modelo estrela do DW, que é preservado. As ligações da Figura 9 simbolizam as associações entre as dimensões e o fato tripla.

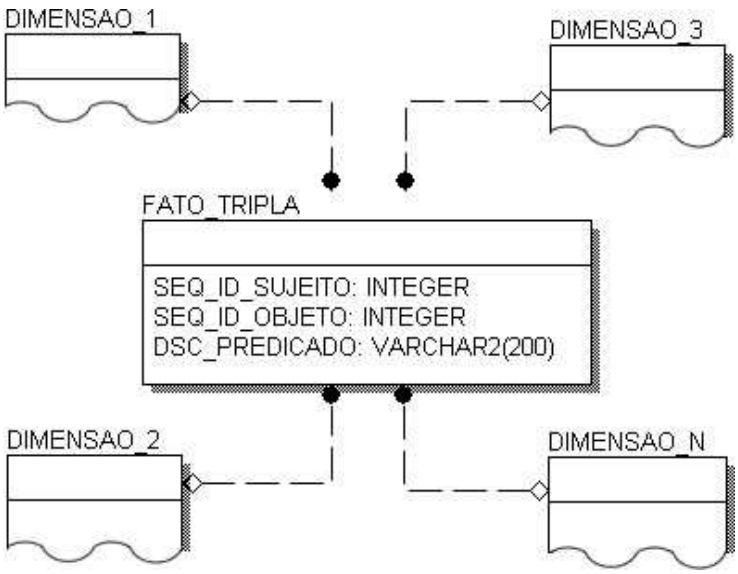


FIGURA 9 – Modelo tripla para armazenamento de inferências semânticas

Todas as derivações devem ser salvas previamente no modelo tripla, da Figura 9, em paralelo com o processamento ETL. Deste modo, as ferramentas OLAP podem buscar as informações do DW definidas no processamento ETL tradicional e, ainda, ter uma nova visão dessas informações consoante os termos determinados na ontologia. Por exemplo, uma instituição analisa um indicador relacionado ao total de pessoas segundo os níveis de formação, como graduação, especialização, mestrado e doutorado (supondo que exista uma dimensão relacionada à formação). Porém, essa instituição deseja realizar agora um *Drill-Down* a fim de identificar os pesquisadores, orientadores e egressos do nível de mestrado. Já que a definição de *Pesquisador*, *Orientador* e *Egresso* não foi definida no processo ETL, nas abordagens clássicas de BI a operação *Drill* desejada dificilmente consegue ser concretizada. Com a solução proposta, alteram-se ou criam-se apenas as regras na ontologia de domínio para que as fontes de dados passem também a armazenar esses três conceitos. Para que isso ocorra, pressupõe-se que haja maneiras de deduzir esses termos. O termo *Egresso*, por exemplo, poderia ser deduzido e salvo no modelo tripla através da dimensão de formações do DW quando o ano de conclusão no nível de mestrado está devidamente informado. Vale lembrar que o Drill-Down desejado é realizado na abordagem dada por Sell, a diferença é que neste trabalho os conceitos

deduzidos são inseridos num modelo complementar ao modelo estrela para otimizar as consultas.

No exemplo supracitado o modelo tripla, o Sujeito poderia estar associado a uma pessoa (supondo que exista uma dimensão relacionada às pessoas); o Predicado poderia referir-se aos conceitos *Pesquisador*, *Orientador* ou *Egresso* e o Objeto poderia ser a instituição da formação (supondo que exista um dimensão de Instituições). Basta que a ferramenta OLAP faça um cruzamento das informações do DW com as informações do modelo tripla para fazer o *Drill-Down* solicitado. Com isso, pode-se inferir que qualquer indivíduo é um pesquisador, um orientador ou um egresso de uma instituição.

As fontes de dados deste projeto são Data Marts, com predomínio da modelagem dimensional. Entretanto também podem ser utilizados os sistemas operacionais, cuja modelagem característica é o MER. Neste caso, visto que o modelo tripla tem uma única chave simples de referência para as dimensões, ele deve ser criado para que tanto o Sujeito quanto o Objeto tenham chaves compostas. Isso pode ocorrer devido ao MER nem sempre ter chaves simples (com um único atributo pertencente a chave primária) para identificar uma pessoa ou instituição como no exemplo descrito acima. Logo, a identificação do Sujeito ou Objeto, que antes se fazia com uma chave artificial simples, pode agora ser feita com chaves compostas. O sofware desenvolvido já possui essa capacidade de configurar o modelo tripla, porém toda a criação física nas fontes de dados deve ser feita pelo DBA.

Na versão da arquitetura SBI adotada por Sell (2006), as deduções semânticas não são salvas e todo o resultado deve ser reprocessado para dentro da Ontologia do Domínio. Dependendo do grau de complexidade e do tamanho da ontologia e da fonte de dados, pode haver um baixo desempenho nas consultas, além de permitir a replicação de instâncias da ontologia.

Com a inserção das deduções semânticas no DW, têm-se as seguintes vantagens:

1) melhoria no desempenho e na facilidade de consultas pelos sistemas OLAP: já que as deduções semânticas se encontram todas

armazenadas previamente no DW e só ocorre processamento para busca desses resultados;

2) acessibilidade das inferências por inúmeras ferramentas-cliente simultaneamente: o armazenamento das informações também possibilita que muitos clientes, em ambientes distribuídos, façam as suas análises;
3) possibilidade de conferência de alguns indicadores do DW: quando o valor de um indicador produzido pelo processo ETL é comparado com o número de deduções geradas para esse indicador. Como exemplo, poderia-se verificar se o valor do indicador que se refere ao total de pesquisadores gerado pela ferramenta proposta é exatamente igual ao valor determinado pelo processo ETL.
4) o código-fonte da ferramenta ETL não necessita sofrer modificações – a modificação da ontologia de domínio e as regras de negócio são suficientes para adequar todo o processo de Data Warehousing; e
5) o modelo dimensional não precisa ser remodelado – o modelo tripla introduzido pode ser combinado com as informações do modelo dimensional para extrair as novas variáveis de interesse.

As vantagens 4 e 5 somente são aplicadas quando as modificações desejadas podem ser deduzidas do DW e posteriormente persistidas no modelo tripla.

Em contrapartida, existem desvantagens, como o grande volume do fato tripla e o aumento do espaço para armazenamento. Conforme Inmon (2005), a economia de espaço comparada ao valor do DW para a corporação é insignificante. Mesmo que haja um grande volume de dados no modelo tripla com as técnicas utilizadas em banco de dados, como particionamento e indexação, consegue-se obter bons desempenhos nas consultas.

3.1.2 Acesso e Persistência

Para promover uma interface de comunicação com as fontes de dados e para facilitar a persistência de inferências da ontologia, adicionou-se à arquitetura SBI mais uma camada, localizada um nível imediatamente acima das fontes de dados.

A comunicação entre o Gerenciador de Ontologias desenvolvido e as fontes de dados se dá por meio dessa nova camada. Assim como os Mecanismos de Inferências, essa camada permite que ferramentas desenvolvidas por terceiros possam ser introduzidas na arquitetura. Dentre as ferramentas que podem ser inclusas, pode-se citar: o framework Hibernate[19], que realiza o mapeamento do modelo Orientado-Objeto para o modelo relacional; o servidor multidimensional Mondrian[20], que simula um banco de dados multidimensional sob uma base de dados relacional e o Sesame[21], um framework para inferência e armazenamento de RDF.

Na atual versão do projeto, utiliza-se um framework para ETL que se encarrega de todas as requisições SQL submetidas à base de dados. Outra vantagem é que a comunicação entre as fontes de dados e os demais módulos da arquitetura SBI pode ser feita de modo transparente de modo que outras fontes de dados, estruturadas ou não, possam ser usadas. Essa nova camada denominada Acesso e Persistência também é visualizada na Figura 11, juntamente com a estrutura do aplicativo desenvolvido.

3.1.3 Repositórios de Ontologias

Todos os módulos dessa camada são contribuições do doutorado de Sell (2006) e contêm a base de conhecimento formada pelo conjunto de ontologias: Ontologia de Domínio, Ontologia de BI e Ontologia de Serviços. A Ontologia de

[19] Hibernate <http://www.hibernate.org/>.
[20] Mondrian <http://mondrian.pentaho.org/>.
[21] Sesame <http://www.openrdf.org/>

Serviços não é descrita neste documento porque foge ao escopo do problema proposto.

3.1.3.1 Ontologia do Domínio

Segundo Sell (2006), a Ontologia de Domínio provê a terminologia formalmente especificada do negócio que está sendo suportada pela arquitetura. Esse módulo deve ser utilizado pelo usuário para a inserção de seus termos e regras, isto é, o usuário deve determinar a hierarquia de classes e propriedades correspondentes às terminologias de seu negócio bem como as regras de inferência e expressões lógicas.

A Ontologia de Domínio é representada através da linguagem OWL nesta pesquisa. Já que OWL não possui suporte para regras, foi aplicada a linguagem do próprio Mecanismo de Inferência, que é mencionado à frente.

Para a criação de qualquer ontologia de domínio, é aconselhável que o usuário utilize editores disponíveis no mercado, como Prótégé [22], OilEd [23], SWOOP[24] e outros que façam manuseio de OWL. Uma vez completa, a Ontologia de Domínio dá suporte para que determinados módulos funcionais e os Mecanismos de Inferência possam tomar conclusões. Todavia, a Ontologia de Domínio oferece apenas um modelo ou esquema conceitual do negócio da organização e necessita do conjunto de instâncias para que as inferências sejam efetuadas. Para esse propósito, faz-se necessário o uso de mais outra ontologia, a Ontologia BI.

3.1.3.2 Ontologia BI

A Ontologia BI é a responsável por dar suporte ao mapeamento entre toda a conceitualização estabelecida na Ontologia de Domínio e as Fontes de Dados,

[22] Prótégé <http://protege.stanford.edu>.
[23] OILEd <http://oiled.man.ac.uk>.
[24] SWOOP <http://www.mindswap.org/2004/SWOOP>.

isto é, trata a correspondência de cada classe ou propriedade modelada na Ontologia de Domínio com as dimensões, tabelas de fato e atributos do DW.

As informações contidas no DW, especificamente nas dimensões, servem como valores na criação de instâncias (indivíduos) dos conceitos da Ontologia do Domínio. As tabelas de fatos, por apresentarem medidas numéricas e sumarizadas, dificilmente são associadas a propriedades e classes da ontologia. Portanto, a Ontologia BI serve como um guia na criação de instâncias da Ontologia de Domínio a partir das dimensões do DW.

Além de permitir um vínculo entre DW e a Ontologia de Domínio, a Ontologia BI possui outras funcionalidades que não estão relacionadas à problemática do presente estudo, como, por exemplo, a redefinição de consultas com base no perfil do cliente.

Assim como a Ontologia do Domínio, a Ontologia BI é projetada com as construções OWL. Para realizar a correta associação entre modelo ontológico e modelo dimensional, Sell (2006) propõe algumas classes e propriedades que são exibidas na Figura 10 a seguir. Algumas dessas classes e propriedades sofreram alterações devido à OWL possuir características que auxiliam no papel da Ontologia BI, tais como a declaração de propriedades inversas, transitivas e funcionais.

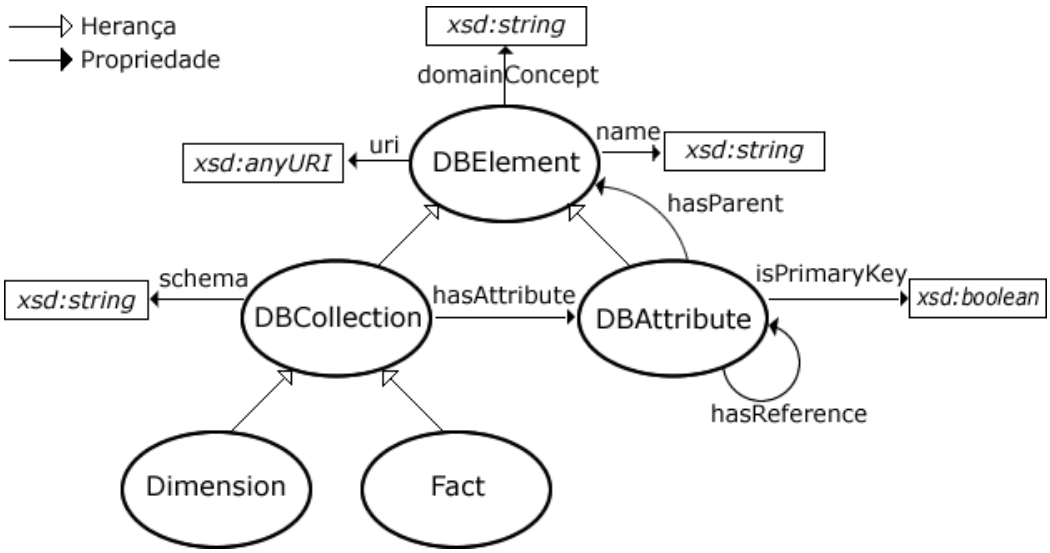


FIGURA 10 – Visão parcial da Ontologia BI

A Figura 10 contém somente a parte da Ontologia BI que é usada na pesquisa. Essa figura apresenta duas classes responsáveis por mapear dimensões e tabelas de fato, Dimension e Fact, respectivamente. Essas classes são especializações da classe DBCollection, que representa uma tabela da base de dados. DBCollection é composta de um ou mais indivíduos da classe DBAttribute, que, por sua vez, representam os atributos da tabela. Todas as classes são subclasses de DBElement, que é a representação de qualquer elemento da base de dados, como tabelas, visões e atributos. A Tabela 3 a seguir descreve as propriedades de cada classe da Ontologia BI.

TABELA 3 – Descrição das propriedades da Ontologia BI

Classe	Propriedade	Descrição
DBElement	**uri**	Propriedade oferecida por OWL que identifica unicamente um elemento.
	domainConcept	Identifica o conceito/classe da Ontologia do Domínio.
	name	O nome do elemento (tabela ou atributo) da base de dados.
DBCollection	**hasAttribute**	Lista de atributos pertencentes à tabela.
	schema	O nome do esquema da base de dados da qual DBCollection faz parte.
DBAttribute	**isPrimaryKey**	Indica se o atributo faz parte da chave primária ou é chave artificial.
	hasReference	Faz referência a um atributo de outra DBCollection. É análogo a uma chave estrangeira e também utilizado para realizar junções entre as tabelas da base de dados.
	hasParent	Representa a DBCollection pai da qual o atributo faz parte.

As classes Dimension e Fact não estão na Tabela 3 acima, pois não incluem nenhuma outra propriedade, além das já herdadas pelas classes-pai. Sell (2006) utiliza diversas outras propriedades e classes, porém são utilizadas em outras funcionalidades que não abrangem o escopo do estudo.

O mapeamento das Fontes de Dados com a ontologia de negócio é feito através da criação das instâncias da Ontologia BI vista na Figura 10. Cada classe

definida na Ontologia do Domínio deve ter uma correspondência com alguma dimensão do DW. Por sua vez, as propriedades dessas classes devem estar associadas a atributos das dimensões. Deste modo, uma instância da classe DBCollection associa-se a uma classe da Ontologia do Domínio. Já uma instância da classe DBAttribute corresponde a uma propriedade ou um relacionamento de uma classe da Ontologia do Domínio. Quando há um relacionamento entre classes, e, conseqüentemente, uma ligação entre as tabelas do DW, a propriedade *hasReference* de *DBAttribute* deve indicar o atributo desse relacionamento, como uma chave estrangeira (ou em inglês *foreign key*).

A definição da ontologia de domínio e a sua correlação com a base de dados, através da criação de instâncias da Ontologia de BI, exigem da organização dois papéis:

- analista de negócio: pessoa que entende sobre as regras de negócio e dos processos da organização; e
- engenheiro de ontologias: pessoa que deve modelar as regras e a ontologia do domínio da organização e associá-las à base de dados.

O conjunto de construções OWL que representam a modelagem da Ontologia BI juntamente com as instâncias criadas podem ser vistos no Apêndice D deste documento.

3.1.4 Mecanismos de Inferência

Os mecanismos de inferência executam as deduções semânticas sobre a ontologia através de regras definidas pelo usuário. Diversos sistemas externos à arquitetura, desenvolvidos por terceiros, podem ser utilizados para tal propósito.

O mecanismo de inferência agregado à arquitetura é o Jena Semantic Web Framework (JENA, 2006), que, além de manipular RDF e OWL, possui capacidade de processar inferências a partir de regras definidas segundo uma linguagem própria. O motor de inferência do próprio Jena é utilizado no projeto. Todas as regras utilizadas nas derivações foram criadas manualmente e estão no Apêndice C.

3.1.5 Módulos Funcionais

Dentre os módulos funcionais presentes na arquitetura SBI, o Gerenciador de Ontologias é o único que abrange o enfoque deste trabalho. Ele realiza a comunicação com o repositório de ontologias e as fontes de dados, além de coordenar a ação do mecanismo de inferência.

O Gerenciador de Ontologias utiliza o recurso de acesso às fontes de dados para comandar a criação de instâncias da Ontologia de domínio através da Ontologia BI. Dispõe-se desse recurso de acesso devido ao acréscimo da nova camada entre as fontes de dados e o repositório de ontologias. Essa nova camada, que é ilustrada com a aplicação desenvolvida, isola o Gerenciador de Ontologias de qualquer tipo de base de dados, como base de dados relacionais, orientado-objeto, textuais e outras.

O módulo funcional abordado é implementado através das APIs Java e corresponde à parte central do desenvolvimento da ferramenta. Todo o projeto de software foi idealizado para que diferentes mecanismos de inferência e outros repositórios de ontologias possam ser adaptados à arquitetura SBI.

3.1.6 Clientes

A camada Clientes é a única zona da arquitetura que não é compreendida pelo software desenvolvido. Nela encontram-se os elementos de Front-End, que possuem contato direto com o cliente, situados no topo da arquitetura SBI. Portais, ferramentas OLAP, *Dashboards*, aplicações de Data Mining, dentre outros recursos, devem compor essa região para intervenção do usuário.

Para que as análises multidimensionais executadas sobre o DW possam ser acompanhadas das inferências do modelo tripla, faz-se necessário que a ferramenta OLAP conheça a correspondência entre o sujeito, predicado e objeto, isto é, a ferramenta OLAP deve saber, por exemplo, que o predicado *Pesquisador*, definido no modelo tripla, tem o sujeito relacionado à dimensão de pessoas e o objeto associado à dimensão de instituições. Portanto, para conhecer os

indivíduos que são pesquisadores, a aplicação analítica deve fazer uma junção com a dimensão de pessoas através do valor do sujeito e também uma junção com a dimensão de instituições por meio do valor do objeto.

O reconhecimento das dimensões associadas a um predicado pode ser feito por meio do repositório de Ontologias, que deve ser acessado pelo sistema de Front-End. Outra possível solução seria a criação de um outro fato tripla, na qual os valores do sujeito e objeto descreveriam as dimensões de acordo com o predicado.

3.2 Ferramenta desenvolvida

A aplicação está estruturada em cinco camadas, que obedecem à arquitetura SBI, da qual a camada Acesso e Persistência é inédita.

A Figura 11 abaixo dá uma visão global do projeto do aplicativo construído bem como a disposição dos componentes na arquitetura SBI. Os elementos em azul na Figura 11 correspondem aos desenvolvidos por terceiros ou são externos ao software proposto. Já os elementos em laranja são membros da solução de BI criada e compõem o módulo Gerenciador de Ontologias. As setas da Figura 11 representam as interações e o fluxo de informações entre os elementos da arquitetura.

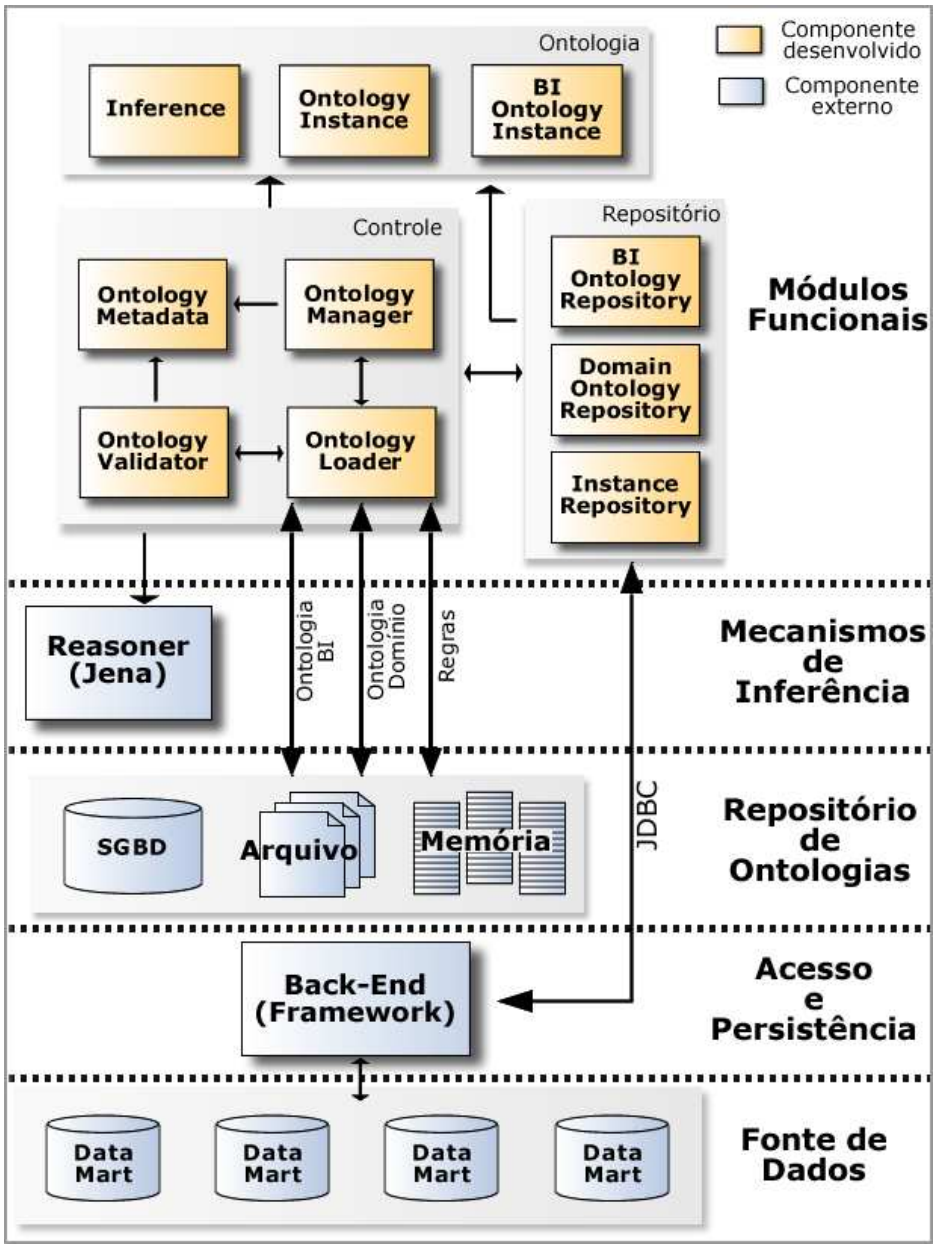


FIGURA 11 – Estrutura da ferramenta desenvolvida

Cada um dos elementos, vistos em laranja na Figura 11, possui diversas classes e componentes Java usados na sua implementação. Essas bibliotecas de classes, assim como um framework, foram concebidas com o objetivo de serem reusáveis e de permitirem transparência no desenvolvimento da arquitetura SBI, independentemente dos mecanismos de inferência, dos repositórios de ontologias e das fonte de dados utilizados. Na prática, tem-se uma modelagem de classes destinada a atender às funcionalidades do Gerenciador de Ontologias, mesmo que

haja alterações nos componentes externos. Com base nessa modelagem feita, projetou-se uma solução específica para a linguagem OWL com integração ao mecanismo de inferência Jena e a uma fonte de dados relacional. Uma diagramação UML sucinta pode ser encontrada no Apêndice A do trabalho.

O Gerenciador de Ontologias tem suas funcionalidades divididas entre alguns dos elementos da região Módulos Funcionais, como visualizado na Figura 11. Esses elementos estão dispostos em três grupos:

- **Controle:** compreende o núcleo do Gerenciador de Ontologias, no qual se encontram os elementos responsáveis pela execução das atividades do mecanismo de inferência, pela leitura e validação do repositório de ontologias e também pela coordenação entre esses elementos;
- **Repositório:** composto dos elementos que manipulam os repositórios de ontologias e as fontes de dados. Possui elementos específicos destinados a tratar da Ontologia do Domínio, da Ontologia BI e do DW, que contêm as instâncias da ontologia; e
- **Ontologia:** envolve os componentes que representam as inferências e regras aplicadas na ontologia, e ainda as instâncias da Ontologia do Domínio e da Ontologia BI.

3.2.1 Controle

O bloco Controle é composto de quatro elementos: *OntologyLoader*, *OntologyValidator*, *OntologyMetadata* e *OntologyManager*.

O *OntologyLoader* é encarregado de realizar a leitura do repositório de ontologias. As ontologias podem estar formalizadas e estruturadas de diversas maneiras, salvas em arquivos OWL, armazenadas em banco de dados ou ainda contidas na memória de trabalho, como mostra a Figura 11. Através do *OntologyLoader* pode-se utilizar essas opções para leitura ou importação de ontologias, assim como das regras de negócio que são aplicadas.

Algumas classes Java formam *OntologyLoader* para que as ontologias possam ser lidas de acordo com o formato no qual se encontram. Uma especialização ou extensão dessas classes deve ser feita para suportar outros

formatos. A versão atual inclui uma especialização para o formato OWL, que é utilizado para representação da Ontologia do Domínio e Ontologia BI. O framework Jena possui opções para que as ontologias possam ser armazenadas em base de dados relacionais bem como construídas e mantidas em memória diretamente através da programação Java.

Para verificar a existência de erros nas ontologias e nas regras de negócio, o *OntologyLoader* conta com o auxílio de outro elemento denominado *OntologyValidator*. Dentre as validações exercidas pelo *OntologyValidator*, pode-se mencionar a identificação de valores obrigatórios para as propriedades, a garantia de que as restrições definidas no modelo ontológico estão respeitadas, a verificação do correto mapeamento entre a Ontologia de Domínio e as fontes de dados feita pela Ontologia BI e a busca de reificações ou heranças cíclicas na Ontologia do Domínio e BI. As heranças cíclicas devem ser evitadas na ontologia, pois podem implicar problemas de decibilidade e recursividade infinita. As regras de negócio são validadas junto ao próprio mecanismo de inferência, cujos seus recursos e sua sintaxe nativa são aproveitados.

A validação feita pelo componente *OntologyValidator* requer informações das propriedades e restrições da Ontologia BI. Essas informações são especificadas no elemento *OntologyMetadata*, que provê os metadados da Ontologia BI. *OntologyMetadata* é responsável também pelas configurações de entrada do sistema, tais como os parâmetros para conexão com a base de dados, definição do modelo tripla (atributos e tabela) e o caminho dos repositórios de ontologias (arquivos OWL). Por ter atuação na camada Back-End das arquiteturas de BI e ser executada em modo batch não há interações entre o usuário e a ferramenta, uma vez definidas as entradas. O arquivo de propriedades que contém todas as entradas do sistema está no Apêndice F deste documento.

A orquestração de toda a parte de Controle com as demais é conduzida pelo *OntologyManager*, componente principal do Gerenciador de Ontologias. Toda seqüência funcional, desde a leitura das ontologias até a persistência das inferências, é coordenada por esse elemento. Resumidamente, a série de ações desempenhadas por *OntologyManager* é:

1) leitura das instâncias da Ontologia BI - a partir das instâncias da Ontologia BI, que são obtidas pelo *OntologyLoader* e *OntologyValidator*, tem-se o mapeamento entre Ontologia do Domínio e dimensões e tabelas de fato.
2) leitura da Ontologia do Domínio e regras de negócio – a ontologia de domínio, ainda sem instâncias criadas, é lida pelo OntologyLoader, e a sua correspondência com a Ontologia BI é analisada pelo *OntologyValidator*. Após os passos 1 e 2, o mecanismo de inferência já possui as ontologias em sua memória de trabalho por meio do *OntologyManager;*
3) emissão da consulta para obtenção das instâncias da Ontologia do Domínio – os valores necessários para a criação das instâncias devem ser buscados das fontes de dados. Através das instâncias da Ontologia BI, uma requisição é emitida à camada de Acesso e Persistência, que se destina a trazer esses valores;
4) criação das instâncias da Ontologia do Domínio – as instâncias das classes e os valores das propriedades e dos relacionamentos são construídos e adicionados ao Jena, conforme o resultado do passo 3. Nesse passo, o OntologyValidator garante as restrições e as obrigatoriedades do modelo ontológico;
5) aplicação das regras de negócio - uma vez criadas as instâncias da Ontologia do Domínio, todas as conclusões semânticas podem ser praticadas a partir das regras de negócio definidas pelo usuário. Este passo é totalmente desempenhado pelo mecanismo de inferência; e
6) armazenamento das inferências semânticas – por fim, cada inferência da ontologia retornada pelo mecanismo de inferência é inserida no fato tripla.

Os repositórios de Ontologias que são lidos nos passos 1 e 2 devem estar devidamente informados no elemento *OntologyMetadata* através de um arquivo de propriedades.

A realização do passo 3 requer que uma classe ou um conceito de negócio principal da ontologia de domínio seja estabelecido como mais uma informação entrada do aplicativo. Essa classe ou conceito é usada como um ponto inicial para a navegação no grafo RDF/OWL. Dessa forma, o *OntologyManager* coordena a navegação pelos relacionamentos e pelas propriedades da Ontologia do Domínio

e orienta a formulação da consulta à base de dados com as tabelas mapeadas na Ontologia BI.

Os passos 4, 5 e 6 são repetidos até que os resultados da consulta realizada no passo 3 terminem. A cada iteração as instâncias são sobrepostas no passo 4, sem haver acúmulo ou replicação em memória de trabalho.

Como exemplo dos seis passos do *OntologyManager* supõe-se que a ontologia de domínio seja formada simplesmente pelas classes *Pessoa* e *Instituição*, citadas em outros exemplos anteriores. No passo 1, então, a associação entre a fonte de dados e os conceitos *Pessoa* e *Instituição* é reconhecida. Isto é, a correspondência entre tabelas e conceitos ou atributos e propriedades são lidos pelo sistema. Posteriormente no passo 2, a Ontologia do Domínio é lida e as classes e as propriedades são verificadas com os mapeamentos da Ontologia BI. Com o mapeamento realizado, uma consulta às fontes de dados deve ser submetida no passo 3. Então, neste exemplo duas dimensões estariam envolvidas nessa consulta, uma relacionada às pessoas e outra às instituições. Os valores retornados, usados na criação das instâncias da Ontologia do Domínio, são definidos nas classes *Pessoa* e *Instituição* no passo 4. As propriedas da classe *Pessoa* como nome, sexo, data de nascimento, etc e as propriedades da classe *Instituição* como nome, sigla, etc, poderiam ser atribuídas. Com as instâncias criadas, aplicação as regras de inferência no passo 5. Como por exemplo, poderia-se inferir se as pessoas são jovens ou adultas através de sua data de nascimento. No sexto e último passo, todas as inferências devem ser salvas no modelo tripla proposto. Neste exemplo, o sujeito seria uma pessoa, o predicado seria composto dos conceitos *Jovem* ou *Adulto*, e o objeto seria um instituição, na qual a pessoa trabalha ou estuda. Logo, poderia-se saber qual relação de jovens e adultos na instituição.

3.2.2 Repositório

Essa região do Gerenciador de Ontologias contém três elementos que representam os repositórios de Ontologia do Domínio, a Ontologia BI e as fontes de dados. Esses elementos são necessários para que o *OntologyManager* possa

manipular e conduzir as ações de forma transparente, visto que os repositórios de ontologias e as fontes de dados podem estar estruturadas de distintas maneiras. Os três elementos do grupo Repositório são: *BIOntologyRepository*; *DomainOntologyRepository* e *InstanceRepository*.

BIOntologyRepository é o responsável exclusivamente pelo repositório das instâncias da Ontologia BI, logo após carregadas pelo *OntologyLoader*. As buscas por conceitos, propriedades e relações com as fontes de dados são tratadas diretamente por esse membro. Logo, as tabelas do modelo dimensional e as junções, usadas para a construção das instâncias da ontologia de domínio, são obtidas por *BIOntologyRepository*. Ele auxilia o *OntologyManager* no passo 3, descrito na seção anterior.

Assim como o *BIOntologyRepository*, o papel do *DomainOntologyRepository* é manipular as instâncias contidas no repositório, porém da Ontologia do Domínio. Dessa forma, a criação e a navegação das instâncias da Ontologia do Domínio são cumpridas por esse item. Além disso, ele retorna as deduções semânticas da ontologia de domínio feitas pelo mecanismo de inferência. *DomainOntologyRepository* participa no passo 4 e 5 do *OntologyManager* exposto anteriormente.

InstanceRepository tem a função de se comunicar com as fontes de dados através da camada Acesso e Persistência. Sua finalidade é extrair as informações necessárias para a criação das instâncias da Ontologia do Domínio e também persistir as conclusões semânticas. Algumas interfaces de componentes Java e padrões de projeto (ou em inglês *design patterns*) foram usados para adequar outros recursos da camada Acesso e Persistência. Portanto, a complexidade da comunicação com as fontes de dados fica atrelada aos recursos do nível de Acesso e Persistência.

O framework Back-End para o processamento ETL, situado na camada Acesso e Persistência, é utilizado neste trabalho para estabelecer a conexão JDBC (Java Database Connectivity) com o DW. Ele faz a consulta SQL e retorna os valores para o *InstanceRepository*, e também o auxilia na inserção das deduções semânticas no fato tripla.

As dimensões e as suas ligações com as demais tabelas do modelo estrela, mapeadas na Ontologia BI e necessárias para montagem da consulta SQL, são disponibilizadas pelo *BIOntologyRepository*. Essas ligações estão definidas pela propriedade *hasReference* da classe *DBAttribute* da Ontologia BI (vide Tabela 3). A partir de uma classe ou um conceito da Ontologia de domínio, *BIOntologyRepository* navega pelas instâncias de *DBAttribute* através da propriedade *hasReference*. Dessa forma, as tabelas e os atributos usados nas junções são obtidos pelo *InstanceRepository*, que os repassa para o framework Back-End.

Tecnicamente, o framework Back-End elabora um SQL com junções externas e internas de acordo com a obrigatoriedade dos relacionamentos das classes. Para as classes que possuem relacionamentos obrigatórios com outras classes, são montadas junções internas (ou em inglês *inner joins*) entre as tabelas. Caso o relacionamento entre as classes não seja obrigatório, junções externas (ou em inglês *outer joins*) são criadas.

3.2.3 Ontologia

Dados os diferentes mecanismos de inferência e os distintos repositórios de ontologias, faz-se necessários elementos para abstrair como as inferências semânticas e as instâncias das ontologias estão representadas no software. O Jena e outros frameworks possuem implementações específicas para instâncias, ontologias e inferências via classes Java. Logo, para fornecer uma representação comum desses itens à arquitetura, foram utilizados os padrões de projeto *Adapter* e *Bridge*. Esses padrões estão reunidos em três componentes: *Inference*, *OntologyInstance* e *BIOntologyInstance*, do grupo denominado *Ontologia.*

Cada inferência elaborada pelo Jena é associada a um elemento *Inference* através do intermédio do *DomainOntologyRepository*. *Inference* corresponde às conclusões tomadas pela aplicação das regras de negócio sobre a ontologia de domínio. Esse elemento possui as referências para o sujeito, predicado e objeto, e, ainda, para a regra que os originou. Uma vez definido, o objeto *Inference* deve ser inserido no fato tripla. O *OntologyManager* recebe todas as inferências de

DomainOntologyRepository e as entrega ao *InstanceRepository*, que persiste às inferências. Para a persistência das inferências, as chaves artificiais das dimensões, que estão relacionadas ao sujeito e ao objeto da tripla, devem ser passadas para o framework Back-End. Da mesma maneira que são obtidas as junções, existe uma propriedade chamada *isPrimaryKey* na Ontologia BI que define os atributos das tabelas que fazem parte da chave primária.

OntologyInstance e *BIOntologyInstance* desempenham as funções das instâncias da Ontologia do Domínio e da Ontologia BI, respectivamente. Todas as instâncias da Ontologia do Domínio estão contidas no *DomainOntologyRepository*, assim como as instâncias da Ontologia BI estão no *BIOntologyRepository*. Para cada instância de uma classe da Ontologia do Domínio tem um objeto *OntologyInstance* que possui todas as suas propriedades e valores. A *BIOntologyInstance* funciona como uma especialização de *OntologyInstance*, e ainda apresenta as propriedades e as correlações com as fontes de dados.

A aplicabilidade do Gerenciador de Ontologias, desenvolvido nos três grupos supracitados, e as demais regiões da arquitetura são demonstradas na seção a seguir.

3.3 Estudo de Caso: Uso de ontologia de contexto acadêmico

Esta seção apresenta a aplicação da ferramenta projetada conforme a arquitetura SBI. O contexto deste trabalho envolve um estudo de caso associado à Plataforma Lattes Institucional (PLATAFORMA LATTES INSTITUCIONAL, 2001), do CNPq (PLI). A PLI é um sistema de gestão curricular relacionada à área de Ciência e Tecnologia. Todas as informações curriculares são amostras reais e foram concedidas pela Universidade Federal de Santa Catarina (UFSC).

Adotou-se um Data Mart componente da PLI que contempla os vínculos e o histórico das atuações acadêmicas e profissionais desempenhadas pelas pessoas da instituição. O processo de Data Warehousing deste estudo possibilita a condução de novas análises sobre esse Data Mart. A parte do modelo dimensional utilizada pode ser verificada na Figura 12.

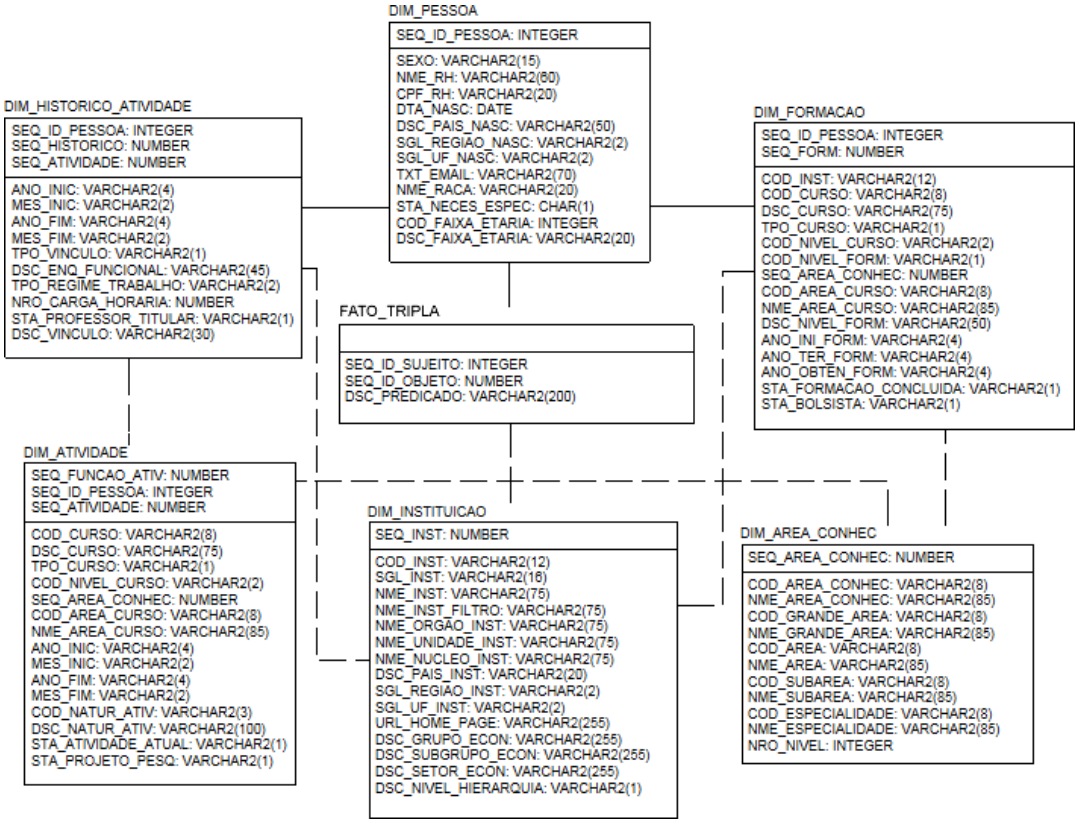


FIGURA 12 – Parte do modelo dimensional da Plataforma Lattes Institucional

O modelo dimensional da Figura 12 apresenta seis dimensões, além da tabela de fato tripla utilizada para o armazenamento das deduções semânticas. Verifica-se a presença de um modelo Snowflake com algumas dimensões interligadas. A Tabela 4 a seguir apresenta uma breve descrição e o número total de tuplas de cada dimensão.

TABELA 4 – Descrição das dimensões utilizadas do Data Mart

Dimensão	Total de Registros	Descrição
DIM_PESSOA	11.229	Contém os dados pessoais dos currículos.
DIM_FORMACAO	24.268	Possui as formações e os cursos das pessoas.
DIM_INSTITUICAO	55	Possui os grupos de pesquisa, centros e órgãos.
DIM_HISTORICO_ATIVIDADE	648	Possui o histórico dos vínculos acadêmicos ou profissionais das pessoas.
DIM_ATIVIDADE	15.532	Contém as atividades das pessoas.
DIM_AREA_CONHEC	1.328	Armazena as áreas de conhecimento.

O número de dimensões e a quantidade total de registros exibidos na Tabela 4 acima são determinantes no tempo de processamento da ferramenta. Quanto maior o número de dimensões utilizadas, maior o número de junções que são feitas pela ferramenta, o que pode diminuir o seu desempenho. A quantidade de junções está associada também aos relacionamentos entre classes da ontologia, pois quanto maior o relacionamento entre as classes, maior será o número de junções entre as dimensões. O total de registros das dimensões determina o número de construções de instâncias e de iterações para processamento das regras. Quanto maior o volume de dados, mais instâncias devem ser criadas na ontologia de domínio. Vale lembrar que as instâncias da ontologia são substituídas a cada iteração, e não há acúmulo em memória.

Semelhante ao processamento ETL, espera-se que os processos de inferências sobre a ontologia sejam realizados numa transação com grande volume de dados. Por isso, recomenda-se também a execução em períodos que não afetem os sistemas transacionais da organização, já que os procedimentos podem ocupar de forma significativa a plataforma de hardware e software.

Para realizar as conclusões semânticas com base nas regras de negócio da organização e propor novas análises sobre o Data Mart, criou-se a ontologia de domínio, que é visualizada na Figura 13 adiante. O editor de ontologias *Prótégé* auxiliou nesse desenvolvimento, e o arquivo OWL gerado pode ser visto no Apêndice B. As classes e as propriedades encontram-se nos retângulos em azul, e os relacionamentos estão ilustrados pelas setas.

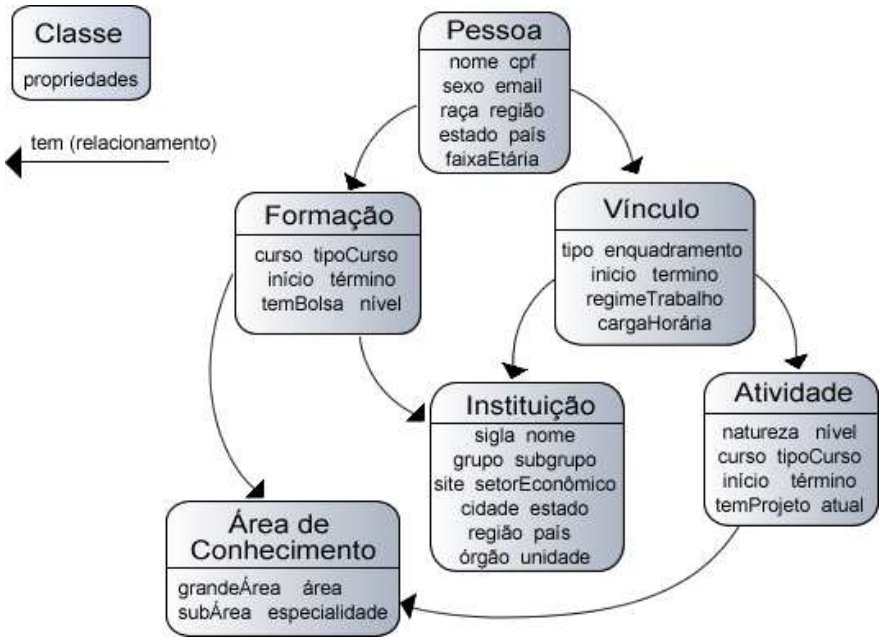


FIGURA 13 – Ilustração da Ontologia de Domínio de contexto acadêmico

Para facilitar o mapeamento com as fontes de dados, a ontologia de domínio da Figura 13 possui uma correspondência direta (um para um) entre suas classes e as dimensões do DW. Embora o estudo de caso não apresente, inúmeros conceitos e classes da Ontologia do Domínio podem estar associados a uma única dimensão do DW, e vice-versa. Evidentemente, a relação um para um entre uma classe da ontologia e uma dimensão do DW deve ocorrer com menor freqüência. Isso acontece porque um modelo ontológico geralmente tem muitas classes e subclasses, diferentemente da modelagem denormalizada do DW, que engloba muitos atributos em uma única dimensão. O modelo da Ontologia de Domínio não precisa estar atrelado ao modelo estrela porque a Ontologia BI faz toda a correlação.

Com base na Ontologia do Domínio, estabeleceram-se as regras de negócio para concluir os seguintes conceitos:

a) *Funcionário Vinculado*: pessoa que possui algum tipo de vínculo empregatício em andamento na instituição;
b) *Docente*: ministrante de cursos da graduação ou da pós-graduação na instituição. O ministrante de cursos técnicos ou de treinamentos não está incluído;

c) *Egresso*: indivíduo que concluiu sua formação na organização acadêmica.
d) *Estudante:* aluno devidamente matriculado em algum curso de graduação, especialização, mestrado ou doutorado;
e) *Estagiário*: estudante que realiza qualquer tipo de estágio na universidade;
f) *Pesquisador*: funcionário vinculado que tem participação em algum projeto de pesquisa ou que ainda não terminou o pós-doutorado;
g) *Administrador*: qualquer pessoa que realiza atividades na administração de cursos, centros ou departamentos da universidade. Estão inclusos os coordenadores de cursos; diretores de centros e departamentos, os pró-reitores e reitores da universidade;
h) *Trainee*: aluno que tem treinamentos ainda não concluídos na instituição;
i) *Servidor Técnico*: funcionário vinculado e que ainda realiza serviços técnicos especializados dentro da instituição; e
j) *Extensão*: sujeito que faz algum curso de extensão universitária.

Os dez conceitos determinados acima formam os predicados do fato tripla, cujos sujeitos são referentes à dimensão DIM_PESSOA e os objetos estão relacionados à dimensão DIM_INSTITUICAO neste estudo de caso, isto é, a chave artificial da DIM_PESSOA e da DIM_INSTITUICAO correspondem aos atributos SEQ_ID_SUJEITO e SEQ_ID_OBJETO do fato tripla, respectivamente (veja Figura 12). Os predicados, que são armazenados no atributo DSC_PREDICADO do fato tripla, fornecem novas variáveis de análise ou filtros para quaisquer consultas sobre essas duas dimensões.

Cada instituição de ensino pode representar diferentes ontologias de domínio e regras de acordo com a semântica dos seus termos. Para uma universidade pública, por exemplo, um pesquisador pode ser caracterizado como um docente que é membro de algum grupo de pesquisa. Agora, para uma universidade particular, um pesquisador pode referir-se a qualquer funcionário que coordena os projetos patrocinados por empresas. Independente de como a organização define a semântica de sua terminologia, essa flexibilidade possibilita uma extensão analítica sobre o DW sem que o modelo dimensional e a ferramenta ETL sofram modificações.

As regras de negócio usadas para inferir os conceitos estão representadas pela linguagem nativa do Jena. Um exemplo da sintaxe que define o conceito *Estagiário* é mostrada na listagem 3 a seguir. Optou-se pelo uso da língua inglesa na especificação dos conceitos da Ontologia em OWL, e, por isso, as classes e as propriedades aparecem em inglês na listagem. As demais regras estão no Apêndice C.

Listagem 3 – Sintaxe da regra para a definição do conceito Estagiário

```
[R9.Estagiario:
      (?person rdf:type is:Person)
      (?person is:hasAffiliation ?affiliation)
      (?affiliation rdf:type is:Affiliation)
      (?affiliation is:hasInstitution ?institution)
      (?institution rdf:type is:Institution)
      (?affiliation is:hasActivity ?activity)
      (?activity rdf:type is:Activity
      (?activity is:nature 'Estágio')
      (?activity is:endYear 'rdf:nil')
    ->
      (?person ESTAGIÁRIO ?institution)
]
```

A regra da listagem 3 estabelece que toda instância (do tipo Pessoa ou *Person*) que possui um vínculo (do tipo Vínculo ou *Affiliation*) com uma atividade (do tipo Atividade ou *Activity*) cuja natureza é ‘Estágio’ e sem término numa instituição (do tipo Instituição ou *Institution*) é um *Estagiário*. Como se percebe na Listagem 3, cada premissa e conclusão das regras de negócio seguem o modelo Sujeito-Predicado-Objeto de RDF/OWL. A função do mecanismo de inferência é analisar se as instâncias atuais da ontologia, contidas na sua memória de trabalho, atendem às premissas da regra. Atualmente todas as regras devem ser especificadas manual. Para mais informações sobre a sintaxe e o motor de inferência, consulte a documentação do Jena (JENA, 2006).

Após a modelagem da Ontologia do Domínio e a determinação das regras, criaram-se as instâncias da Ontologia BI também com o auxílio do *Prótégé*. Toda Ontologia BI em OWL pode ser vista no Apêndice D.

Posteriormente à definição da base de conhecimento, formada pelas ontologias, as entradas do software devem ser configuradas. Portanto, as propriedades de conexão com o Data Mart da PLI, o conceito inicial necessário

para a navegação sobre a ontologia (no caso o conceito *Pessoa*) e os repositórios de ontologias foram definidos conforme o estudo de caso. Todas as configurações estão no Apêndice F.

3.3.1 Resultados Obtidos no Estudo de Caso

Os resultados e as principais métricas da aplicação são descritos nesta seção bem como a quantidade de deduções semânticas e o método usado para analisar a coerência com a amostra de dados.

O estudo de caso foi executado em duas plataformas computacionais distintas para verificar a influência do desempenho. A Tabela 5 demonstra as configurações, tanto de hardware quanto de software, de cada um dos computadores empregados nos testes da ferramenta.

TABELA 5 – Configurações de hardware e software das máquinas usadas

Configuração		**Desktop**	**Laptop**
Hardware	**Processador**	Pentium IV HT 3.2 GHz	Celeron Centrino 1.6GHz
	Memória principal	1GB RAM	1GB RAM
	Memória secundária	80GB	120GB
Software	**Máquina Virtual Java**	Java 2 SDK 1.5.0_09	Java 2 SDK 1.5.0_10
	Mecanismo de Inferência	Jena SWF 2.4	Jena SWF 2.4
	Sistema Operacional	Windows XP Professional service Pack 2	Windows XP Professional service Pack 2
	SGBD	Oracle Enterprise Edition 10.2.0.1.0g	MySQL 5.0.27

O SGBD Oracle, usado pela máquina Desktop, encontra-se em um servidor dedicado na rede local. Já o MySQL está instalado no próprio Laptop. Ambos foram devidamente indexados para dar mais desempenho em consultas.

Essas configurações são importantes para a execução de todo o processo de deduções semânticas sobre a ontologia, principalmente o tamanho da memória principal. O motor de inferência utilizado pelo Jena tem a característica de consumir muita memória durante a sua ação, fato que foi observado nos testes

iniciais. Recomenda-se que pelo menos 256MB de memória principal sejam destinados à máquina virtual Java para suportar todas as inferências.

As instâncias da Ontologia BI mapearam ao todo sete relacionamentos ou junções entre as dimensões do Data Mart, que implicaram 48.591 linhas retornadas na requisição SQL. A consulta gerada automaticamente pode ser vista no Apêndice E. Constroem-se apenas as instâncias das seis classes da ontologia de domínio vistas na Figura 13; e a cada registro retornado na consulta, as propriedades e os relacionamentos são definidos com os valores das fontes de dados.

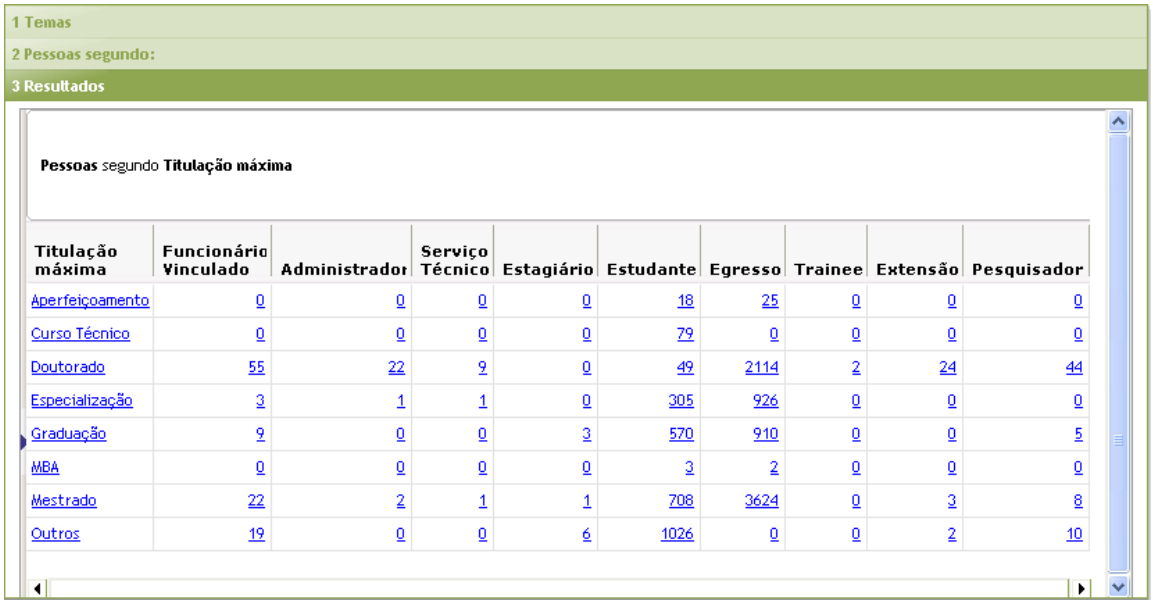
Todo conjunto de dados foi processado pela máquina Desktop em 2 horas, 15 minutos e 27 segundos, e resultou em 22.260 conclusões. O tempo gasto por essa máquina para criar uma instância de cada classe da ontologia de domínio e posteriormente inferir os dez conceitos estipulados foi de 365 milissegundos em média. Isso implica aproximadamente três inferências semânticas por segundo. O SGBD Oracle demorou apenas 219 milissegundos para retornar os resultados da consulta SQL.

Já no Laptop, o mesmo processamento de 22.260 deduções durou 4 horas, 35 minutos e 42 segundos, que corresponde em média a 1,5 inferências por segundo. O período gasto desde a criação de cada instância da ontologia até a aplicação das dez regras de negócio é de 740 milissegundos. Gastaram-se 4,3 segundos na requisição SQL com o SGBD MySQL.

Notou-se uma redução do desempenho à medida que acontecia o processamento de inferências. Nos melhores casos, 49 inferências por segundo chegaram a ser processadas no Desktop e 19 inferências por segundo no Laptop. Atribui-se a queda de desempenho ao mecanismo de inferência, que acumula algumas instâncias em memória. O Jena oferece um modo de descarte de seu cache, porém ocasiona uma queda ainda maior da performance, e, por isso, esse modo não foi aplicado.

A quantidade de deduções para cada um dos conceitos definidos nas regras de negócio pode ser vista na Figura 14 a seguir. Essa figura exibe as pessoas da instituição de ensino segundo a titulação máxima. Uma ferramenta OLAP foi configurada para realizar essa análise, que une as informações do

modelo dimensional e as inferências do modelo tripla. A exibição e a navegabilidade entre o modelo dimensional e o modelo tripla são de responsabilidade da ferramenta OLAP e não fazem parte do foco do problema.



Titulação máxima	Funcionário Vinculado	Administrador	Serviço Técnico	Estagiário	Estudante	Egresso	Trainee	Extensão	Pesquisador
Aperfeiçoamento	0	0	0	0	18	25	0	0	0
Curso Técnico	0	0	0	0	79	0	0	0	0
Doutorado	55	22	9	0	49	2114	2	24	44
Especialização	3	1	1	0	305	926	0	0	0
Graduação	9	0	0	3	570	910	0	0	5
MBA	0	0	0	0	3	2	0	0	0
Mestrado	22	2	1	1	708	3624	0	3	8
Outros	19	0	0	6	1026	0	0	2	10

FIGURA 14 – Visualização do resultado em uma ferramenta OLAP

Devido ao número de junções e de possíveis repetições de valores para determinadas instâncias, algumas inferências podem ser reproduzidas na base de dados, isto é, mais de uma ocorrência dos valores da tripla (Sujeito–Predicado-Objeto) pode ocorrer. Das 22.260 deduções armazenadas pelo fato tripla, ao final da execução do aplicativo, 10.648 são distintas. Embora ocorram repetições das inferências, elas já são eliminadas pelo sistema e só foram utilizadas apenas para estabelecer as métricas de desempenho. Visto que há igualdade na amostra de dados, Desktop e Laptop produziram a mesma quantidade de derivações para cada um dos conceitos, condição já esperada.

Na Figura 14 é feita uma operação drill que mostra a disposição dos indicadores inferidos em oito tipos de formações. A consulta feita pela ferramenta OLAP agrega as dimensões DIM_PESSOA, DIM_FORMACAO, DIM_INSTITUICAO e ainda o fato tripla (FATO_TRIPLA). Nos cabeçalhos de linha estão todos os níveis de formação, e nos cabeçalhos de coluna estão os conceitos inferidos.

Por causa da extensão do modelo dimensional e da semântica da terminologia usada pela instituição, muitas pesquisas sobre o DW podem ser mais eficientemente conduzidas e decisões podem ser mais bem tomadas. Por exemplo, um gestor acadêmico pode conhecer quais serão os possíveis candidatos ao curso de doutorado caso saiba quem são os egressos do curso de mestrado da sua instituição. Em outro cenário, um proprietário de uma empresa de eventos de formaturas poderia focar seus serviços nos alunos que ainda não concluíram a graduação, ou seja, no termo denotado semanticamente como *Estudante*. Enfim, a nova perspectiva de análise oferecida pela arquitetura SBI com o fato tripla pode auxiliar inúmeros atores em sua tomada de decisão.

Para constatar a veracidade das deduções semânticas, fizeram-se consultas manuais diretamente ao SGBD. Uma porção dos conceitos produzidos foi pontualmente verificada, ou seja, boa parte dos sujeitos, predicados e objetos do fato tripla foi conferida com as informações do Data Mart. Além disso, o valor sumarizado de cada conceito deduzido foi comparado também com as informações das fontes de dados. Em 100% dos casos, no Desktop e no Laptop, houve coincidência com os resultados das regras de negócio, o que viabiliza a aplicação desenvolvida.

4 Conclusão

Juntamente com a Web Semântica, muitas pesquisas são desenvolvidas para que os recursos computacionais interajam com as pessoas e possam, de maneira automática, extrair e interpretar informações. É evidente que essas informações e as tecnologias de informação alinhadas aos processos de negócio contribuem para o sucesso das organizações, e é o que ocasiona muitos investimentos na área de BI. Entretanto, a maioria das soluções de BI presente oferece pouca ou mesmo nenhuma flexibilidade para mudanças de regras de negócio e não permite realizar novas análises, além das já definidas nas fases iniciais dos projetos.

O uso de semântica na informação subsidiado por ontologias permite que os executivos possam ter uma nova perspectiva para as análises estratégicas, de acordo com terminologias comuns de seu negócio. O presente estudo foca em uma arquitetura do contexto de BI, denominada arquitetura Semântica Business Intelligence (SBI) proposta por Sell (2006) para agilizar o processamento OLAP e adequar o processo de Data Warehousing às regras definidas pela organização.

Com base em alguns módulos da arquitetura SBI, projetou-se um sistema com recursos em estado-da-arte capaz de produzir quaisquer variáveis de interesse usadas como filtros nas consultas das ferramentas OLAP. Essas variáveis são resultados de deduções semânticas feitas por regras de negócio definidas pela própria instituição. O sistema desenvolvido aplica as regras de negócio sobre a ontologia determinada pelo usuário. As informações contidas no DW são utilizadas para a criação de instâncias da ontologia de negócio.

Uma das principais contribuições do presente estudo está na adoção de um modo diferenciado da pesquisa de Sell (2006) no que diz respeito à materialização dos resultados das inferências. Todas as deduções semânticas são armazenadas no próprio DW numa estrutura adicional ao modelo estrela. Essa estrutura é denominada de modelo tripla, no qual qualquer dimensão se relaciona com a outra por meio de um predicado, analogamente ao modelo RDF/OWL. Dessa forma, as análises sobre o modelo dimensional e o modelo tripla podem ser combinadas de modo que novas visões e novas informações possam ser obtidas. Outras

vantagens proporcionadas pelo fato tripla são: o desempenho das análises por parte dos sistemas de Front-End; acessibilidade e escalabilidade de muitas ferramentas OLAP; validação de alguns indicadores do DW; e flexibilidade para produzir novos indicadores sem alterar as ferramentas ETL ou o modelo dimensional.

Um cenário relacionado à Plataforma Lattes Institucional, com enfoque nas atividades acadêmicas e profissionais em uma instituição de ensino, é aplicado como estudo de caso. Todas as informações curriculares usadas no estudo de caso são reais e fornecidas pela UFSC. Dez novos conceitos estabelecidos nas ontologias foram adicionados ao DW, e uma ferramenta OLAP foi configurada para demonstrar a aplicabilidade do software desenvolvido. Com isso, novas operações drill puderam ser cometidas por essa ferramenta sobre o DW, que passa a auxiliar ainda mais o processo de tomada de decisão.

4.1 Trabalhos Futuros

Visto que recursos externos foram utilizados na construção do aplicativo, muitos experimentos com outras soluções ainda podem ser realizados. O framework Jena possui uma máquina de inferência própria e há outros dispositivos de inferência disponíveis do mercado, como Racer[25], Pellet[26], FaCT[27], que poderiam ser comparados e acrescidos a ele. A linguagem de regras nativa do Jena pode ser também substituída pelas propostas do W3C, SWRL ou, ainda, RuleML. Isso implicaria no desenvolvimento de um parser para essas linguagens e o uso conjunto ao mecanismo de inferência. Com essa substituição, eliminar-se-ia a elaboração manual das regras, visto que já existem editores para essas duas linguagens.

Nesta pesquisa adotou-se o DW como fonte de informações para criação das instâncias da ontologia e também como destino das deduções semânticas. As fontes de dados operacionais não são utilizadas para extrair os valores das

[25] Racer <http://www.racer-systems.com>.
[26] Pellet <http://pellet.owldl.com>.
[27] FaCT <http://www.cs.man.ac.uk/~horrocks/FaCT>.

instâncias da ontologia, embora a ferramenta desenvolvida não tenha essa limitação. As ontologias podem ser úteis para guiar todo o processo de ETL dos dados, desde sua extração das fontes operacionais até sua integração no DW. O processo ETL guiado por ontologias, na qual envolve a adptação do modelo dimensional ao invés do modelo tripla, é um trabalho ainda inexplorado e ao mesmo tempo útil, pois o torna maleável às mudanças de requisitos ou regras de negócio.

Referências

ANTONIOU, G.; HARMELEN, F**. A Semantc Web Primer**. The MIT Press Cambridge, Inglaterra, 2004.

ARAUJO, M. **Educação a Distância e a Web Semântica**: Modelagem Ontológica de materiais e Objetos de Aprendizagem para a Plataforma COL. São Paulo, 2003. Tese – Escola Politécnica da Universidade de São Paulo.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. **The Semantic Web**. Scientific American, 2001a.

BERNEERS-LEE T. **Design Issues**: Web Architecture. 2001b. Disponível na Internet: <http://www.w3.org/DesignIssues>. Acesso em 30 de junho de 2006.

BERGAMASCHI, S.; QUIX, C.; JARKE, M. The SEWASIE EU IST Project. **SIG SEMIS Bulletin**, 2005. <Disponível em: <http://www.sewasie.org/documents/sewasie-sigsemis.pdf>. Acesso em 11 de janeiro de 2007.

BIERE, M. **Business Intelligence for Enterprise**. Ed. Prentice Hall, 2003

BROEKSTA, J. Enabling **Knowledge Representation on the Web by Extending RDF Schema**, 2001. Disponível em: <http://citeseer.ist.psu.edu/451666.html>. Acesso em 10 de julho de 2006.

CODY W. F; KREULEN J. T.; KRISHNA V.;SPANGLER W. S. **The integration of business intelligence and knowledge management**. IBM Systems Journal, 2002. Disponível em: <http://www.research.ibm.com/journal/sj/414/cody.pdf>. Acesso em 11 de janeiro 2007.

COMPUTER WORLD, 2006. **Maturidade levará mercado de BI à consolidação** Disponível em: http://computerworld.uol.com.br/gestao/2006/04/11/idgnoticia.2006-04-11.3756894261/IDGNoticia_view . Acesso em 22 de julho de 2006.

DAML Language, 2002a. **The DARPA Agent Markup Language Homepage** Disponível em: <http://www.daml.org> - Acesso em 12 de julho de 2006.

DAML Language Feature Comparison, 2002b. Disponível em <http://www.daml.org/language/features.html>. Acesso em 15 de julho de 2006.

ECKERSON, W. - **Performance Dashboards**: measuring, monitoring and managing your business. John Wiley & Sons: 2006.

EIL, 2004. **European interoperability framework for Pan-European E-Government Services**.. Disponível

em:<http://europa.eu.int/idabc/en/document/3761>. Acesso em: 11 de julho de 2006.

FENSEL, D.; VAN HARMELEN, F**. Towards the semantic web**: ontology-driven knowledge management. 2003

FENSEL, D. Ontologies: **Silver Bullet for knowledge Management and Eletronic Commerce.** Springer-Verlag: Berlin, 2001.

F-OWL, 2003. **An OWL Inference Engine in Flora-2** – Disponível em <http://fowl.sourceforge.net>. Acesso em 21 de julho de 2006.

GARTNER GROUP. **Gartner says more than 50 Percent of data warehouse projects will have limited acceptance or will be failures through 2007**. Disponível em: <http://www.gartner.com/press_releases/asset_121817_11.html>. Acesso em 30 de novembro de 2006>

GIL Y.; RATNAKAR V. **A Comparison of (Semantic) Markup Languages**, 2002. USC Information Sciences Institute and Computer Science Department. Disponível em: <http://trellis.semanticweb.org/expect/web/semanticweb/paper.pdf> Acesso em 04 de maio de 2006.

GRUBER, T. R. **A Translation Approach to Portable Ontology Specifications**. 1993. Disponível na Internet http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html. Acesso em 06 de abril de 2006.

GUARINO, N. **Formal ontology and information systems**. Amsterdam, 1998.

INMON, W. H. **Building the Data Warehouse**. 4. ed. New York: John Wiley & Sons, 2005.

INMON, W. H. **Como construir o Data Warehouse**. Rio de Janeiro: Campus, 1997.

JENA 2 WEB SEMANTIC FRAMEWORK - Hewlett-Packard Development Company, LP. Disponível em <http://jena.sourceforge.net>. Acesso em 26 de março de 2006.

KIMBALL, R.; ROSS, M.**The data warehouse toolkit**: The Complete Guide to Dimensional Modeling. 2.ed. John Wiley & Sons, Inc., 2002.

KIMBALL, R. **The Data Warehouse Lifecycle Toolkit:** Expert Methods for Designing, Developing Data Warehouses. New York: Wiley Computer Publishing, 1998.

KIMBALL, R.; CASERTA, J. **The Data Warehouse ETL Toolkit**: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. Wiley Publishing, 2004.

MANDARAX PROJECT, 2006. Disponível em <http://mandarax.sourceforge.net>. Acesso em 26 de março de 2006.

MELLO, R.; DORNERLES, C.; KADE;A.; BRAGANHOLO, V.; HEUSER; C. **Dados Semi-Estruturados**. Universidade Federal de Santa Catarina e Universidade Federal do Rio Grande do Sul.

NOY, N.. MCGUINNESS, D. **Ontology Development 101**: A Guide to Creating Your First Ontology, 2000. Universidade de Stanford.

OWL API, 2006. Disponível em <http://owl.man.ac.uk>. Acesso em 04 de abril de 2006.

PÉREZ, G; BENJAMINS, V. **Overview of Knowledge Sharing and Reuse Components**: Ontologies and Problem-Solving Methods, 1999.

PRIEBE, T.; PERNUL, G. **Ontology-based integration of OLAP and information retrieval**. In: THE DEXA 2003 WORKSHOP ON WEB SEMANTICS. WebS 2003). Prague, Czech Republic.

PLATAFORMA LATTES INSTITUCIONAL, CNPq. Desenvolvido pelo Grupo Stela, 2001. Disponível em <http://lattes.ufsc.br>. Acesso em 20 de novembro de 2006.

Rule Markup Language (RuleML), 2000. Disponível em <http://www.ruleml.org>. Acesso em 5 de maio de 2006.

SELL, D. **Uma Arquitetura para Business Intelligence baseada em Tecnologias Semânticas para Suporte a Aplicações Analíticas**. Florianópolis, 2006. Tese – Engenharia de Produção e Sistemas, UFSC.

SELL, D. **Uma Arquitetura para Distribuição dos Componentes Tecnológicos de Sistemas de Informações Baseados em Data Warehouse**. Florianópolis, 2001. Dissertação – Engenharia de Produção e Sistemas, UFSC

STUDER, R.; BENJAMINS, V.; FENSEL, D. **Knowledge Engineering**: Principles and Method. 1998. Disponível em <http://citeseer.ist.psu.edu/article/studer98knowledge.html>. Acesso em 13 de maio de 2006

SweetRules, 2005. Disponível em <http://sweetrules.projects.semwebcentral.org>. Acessado em 11 de junho de 2006.

Semantic Web Rule Language (SWRL). Disponível em <http://www.w3.org/Submission/SWRL>. Acesso em 5 de maio de 2006.

WATSON I.Applying **Knowledge Management**: Techniques for Building Corporate Memories,2003 University of Auckland.

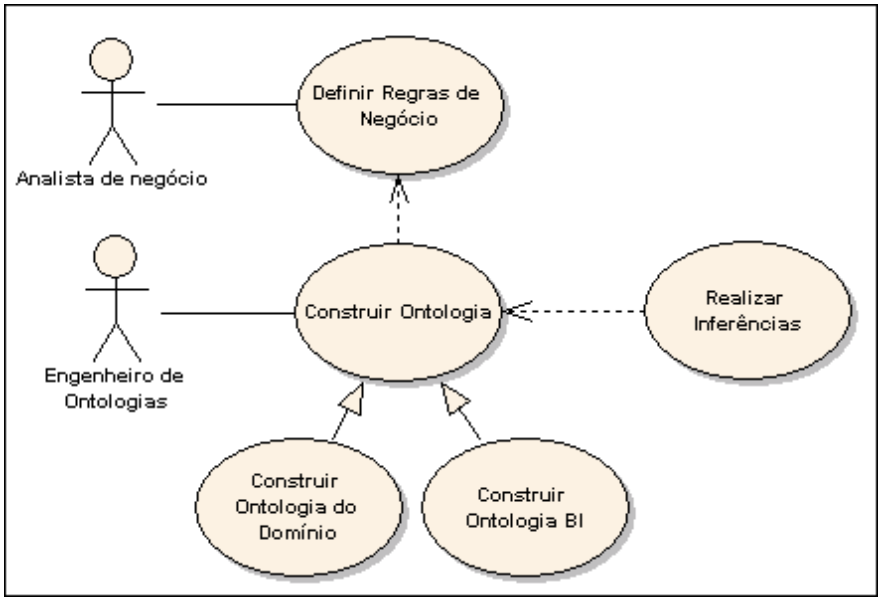
W3C. Resource Description Framework (RDF) Semantic. W3C Recommendation 10 February 2004a. Disponível em <http://www.w3.org/TR/2004/REC-rdf-mt-20040210>. Acesso em 24 de junho de 2006.

W3C. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004b. Disponível em: <http://www.w3.org/TR/rdf-concepts/>. Acesso em 24 de junho de 2006.

W3C Web Services Glossary, 2004c. Disponível em <http://www.w3.org/TR/ws-gloss>. Acesso em 18 de maio de 2006.

Apêndice A – Diagramas UML

Diagrama de Casos de Uso



Caso de Uso: realizar inferências.

Ator: analista de negócio, engenheiro de ontologias e Sistema.

Nível: real e estendido.

Descrição: o analista de negócio deve definir todas as regras baseadas no conhecimento dos processos de seu negócio. Com base nisso, o engenheiro de ontologias projeta as ontologias do negócio tratado e associa com as fontes de dados. Posteriormente, ocorre o processamento de inferências semânticas sobre a ontologia.

Fluxo principal

1. O Analista de negócio informa ao engenheiro de ontologias uma concepção de termos pertencentes ao seus processos de negócio.
2. O engenheiro de ontologias modela classes, relacionamentos e regras com os termos definidos pelo analista de negócio para formar a Ontologia de Domínio.
3. O engenheiro de ontologias informa ao sistema o mapeamento da ontologia de domínio e as fontes de dados através da Ontologia BI.
4. O Sistema confere e valida as informações do repositório de ontologias.
5. O sistema aplica as regras de negócio sobre a ontologia e salva os resultados.
   5.1. O sistema busca as informações das fontes de dados.
   5.2. O sistema cria as instâncias da ontologia de domínio.
   5.3. O sistema realiza as deduções semânticas com o auxílio do Mecanismo de inferência.
   5.4. O sistema salva todas as deduções feitas nas fontes de dados.

Seqüências Alternativas

4a: A Ontologia de Domínio possui inconsistências de valores.
4a.1: O sistema informa a inconsistência ocorrida.
4a.2: Retorna ao passo 2 do fluxo principal.

4b: A Ontologia BI não está em conformidades com a Ontologia do Domínio.
4b.1: O sistema informa os erros de mapeamento encontrados.
4b.2: Retorna ao passo 2 do fluxo principal.

5.1a: As fontes de dados não estão acessíveis.
5.1a.1: O sistema é abortado.

5.1b: As tabelas ou atributos do DW estão especificados incorretamente.
5.1b.1: O sistema informa o errro na consulta às fontes de dados.
5.1b.2: Retorna ao passo 3 do fluxo principal.

5.3a: A camada do Mecanismo de Inferência não está acessível.
5.3a.1: O sistema informa o erro de acessibilidade ou a má configuração do mecanismo de inferência.
5.3a.2: O sistema é abortado.

5.4a: O modelo de persistência de inferência não está corretamente definido.
5.4a.1: O sistema informa que o modelo tripla deve ser informado corretamente.
5.4a.2: O sistema é abortado.

Diagrama de Classes do Framework SBI

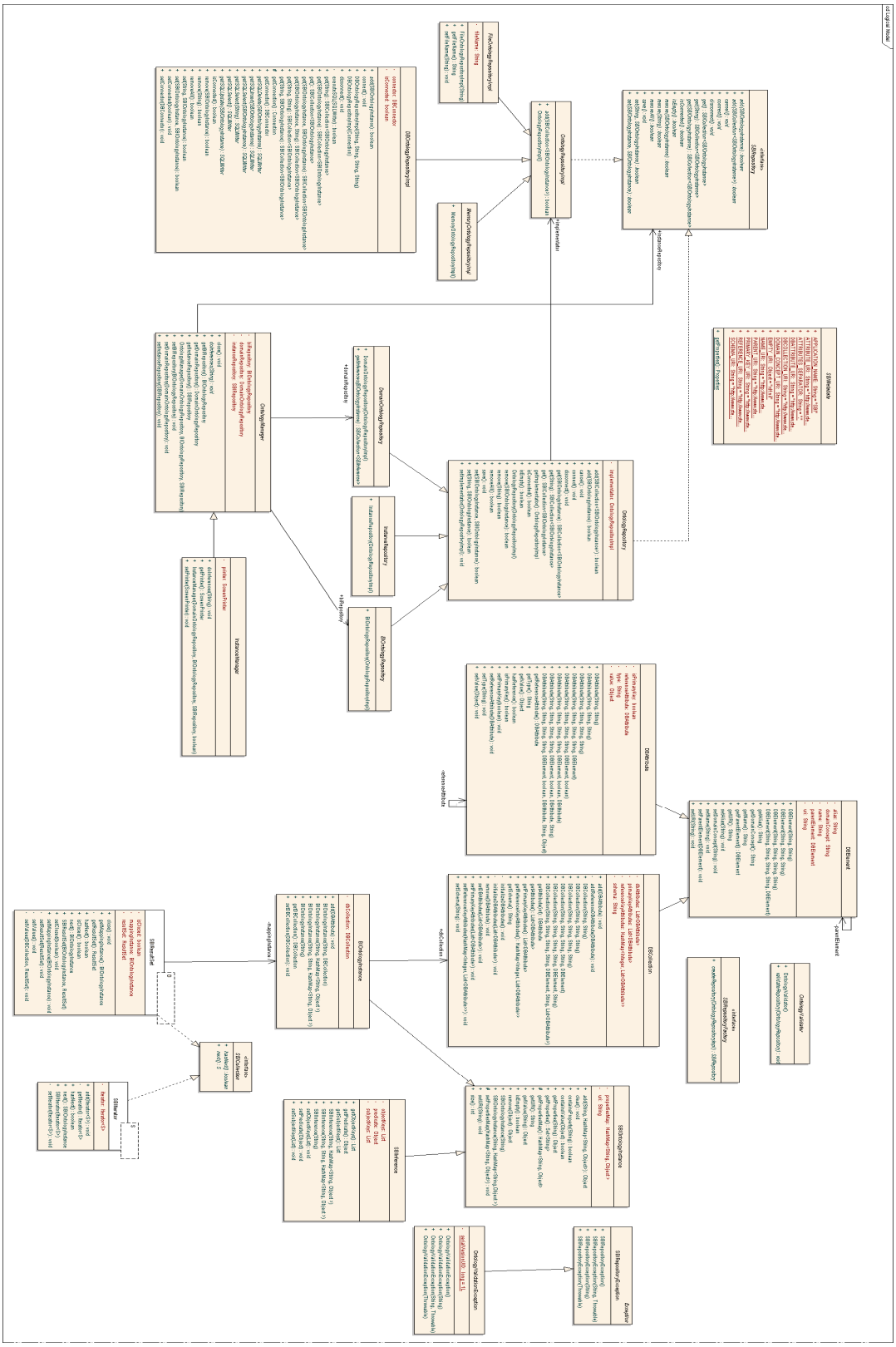
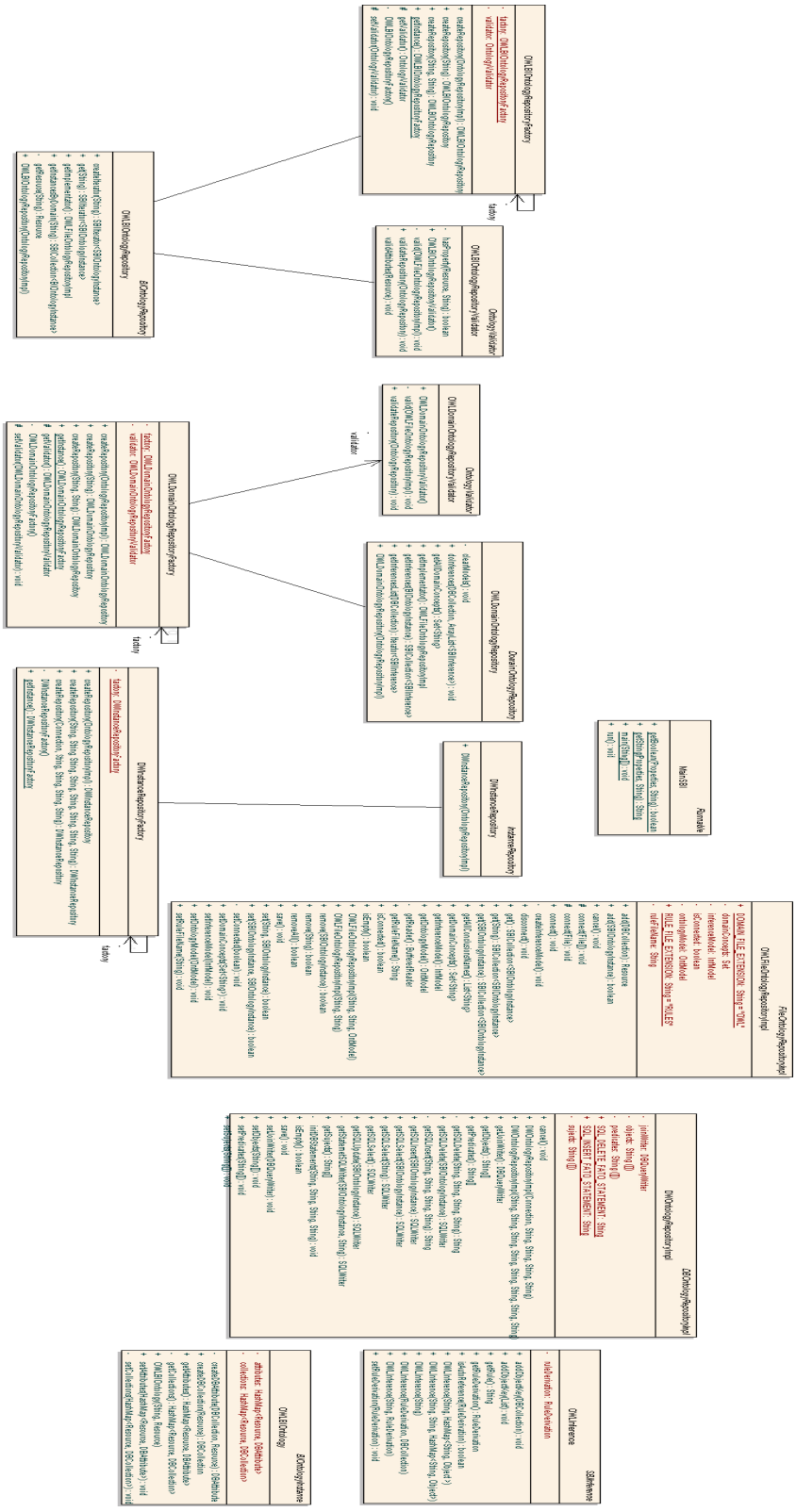


Diagrama de Classes da extensão do Framework SBI
(implementação de OWL com Jena)



Apêndice B – Especificação da Ontologia do Domínio em OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.stela.info#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.stela.info">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Institution">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="regionAcronym"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="department"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="country"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="unit"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="name"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="site"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="acronym"/>
```

```
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="stateAcronym"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="economicGroup"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="KnowledgeArea">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="generalName"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="bitrhDate"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#stateAcronym"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
```

```
        <owl:DatatypeProperty rdf:ID="race"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="email"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasFormation"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Formation"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Affiliation"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasAffiliation"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#country"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Activity">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="isResearchProject"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#KnowledgeArea"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasKnowledgeArea"/>
      </owl:onProperty>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="beginMonth"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="beginYear"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="isActual"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="courseName"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="level"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="courseType"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="nature"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Affiliation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="affiliationType"/>
```

```
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Activity"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasActivity"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="endYear"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="functionalCareer"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#beginMonth"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>vínculo</rdfs:comment>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:about="#beginYear"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:InverseFunctionalProperty rdf:ID="hasInstitution"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="#Institution"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:DatatypeProperty rdf:ID="workingRegime"/>
    </owl:onProperty>
    <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
```

```
        <owl:DatatypeProperty rdf:ID="endMonth"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Formation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#KnowledgeArea"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasKnowledgeArea"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#level"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#courseType"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#beginYear"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#courseName"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#endYear"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Institution"/>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:about="#hasInstitution"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
  <owl:ObjectProperty rdf:about="#hasKnowledgeArea">
    <rdfs:range rdf:resource="#KnowledgeArea"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Formation"/>
          <owl:Class rdf:about="#Activity"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasActivity">
    <rdfs:range rdf:resource="#Activity"/>
    <rdfs:domain rdf:resource="#Affiliation"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#hasAffiliation">
    <rdfs:range rdf:resource="#Affiliation"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="author">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#beginMonth">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Activity"/>
          <owl:Class rdf:about="#Affiliation"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#endMonth">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Activity"/>
          <owl:Class rdf:about="#Affiliation"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#email">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#endYear">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Formation"/>
          <owl:Class rdf:about="#Activity"/>
          <owl:Class rdf:about="#Affiliation"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#economicGroup">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Grupo econômico em que a Instituição se classifica
</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
  <rdfs:domain rdf:resource="#Institution"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#workingRegime">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >regime de trabalho</rdfs:comment>
  <rdfs:domain rdf:resource="#Affiliation"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="city">
  <rdfs:domain rdf:resource="#Institution"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#beginYear">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Formation"/>
        <owl:Class rdf:about="#Activity"/>
        <owl:Class rdf:about="#Affiliation"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="month">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#functionalCareer">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Affiliation"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >enquadramento funcional</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#race">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Person"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#level">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >nível do curso</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#courseType">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#isResearchProject">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#nature">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Activity"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#acronym">
  <rdfs:domain rdf:resource="#Institution"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="subgroup">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="type">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="language">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="subtype">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="group">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#generalName">
  <rdfs:domain rdf:resource="#KnowledgeArea"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#site">
  <rdfs:domain rdf:resource="#Institution"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="gender">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Masculino</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >\Não Informado</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Feminino</rdf:first>
        </rdf:rest>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
  <rdfs:domain rdf:resource="#Person"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#regionAcronym">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Person"/>
        <owl:Class rdf:about="#Institution"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#name">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Person"/>
        <owl:Class rdf:about="#Institution"/>
        <owl:Class rdf:about="#KnowledgeArea"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#isActual">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#courseName">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:about="#bitrhDate">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#country">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Descricao do pais de nascimento</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Person"/>
          <owl:Class rdf:about="#Institution"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#department">
    <rdfs:domain rdf:resource="#Institution"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#stateAcronym">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Person"/>
          <owl:Class rdf:about="#Institution"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#affiliationType">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Affiliation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="cityName">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#unit">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Unidade da Instituição</rdfs:comment>
    <rdfs:domain rdf:resource="#Institution"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="year">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:InverseFunctionalProperty rdf:about="#hasFormation">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Formation"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:InverseFunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#hasInstitution">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu -->
```

Apêndice C – Especificação das Regras de Negócio

```
[R1:        (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:endYear 'rdf:nil' )
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
       ->
            (?person FUNCIONARIO_VINCULADO ?institution)
]

[R2.1:      (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?affiliation rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:endYear 'rdf:nil')
            (?activity is:nature 'Graduação')
      ->
            (?person DOCENTE ?institution)
]

[R2.2:      (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:endYear 'rdf:nil')
            (?activity is:nature 'Pós-graduação')
      ->
            (?person DOCENTE ?institution)
]

[R3:        (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:nature 'Extensão Universitária')
            (?activity is:endYear 'rdf:nil' )
      ->
            (?person EXTENSÃO ?institution)
]


[R4.1:      (?person FUNCIONARIO_VINCULADO ?institution)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
```

```
            (?activity is:nature 'Pesquisa e Desenvolvimento')
      ->
            (?person PESQUISADOR ?institution)
]
[R4.2:      (?person FUNCIONARIO_VINCULADO ?institution)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:isResearchProject 'S')
      ->
            (?person PESQUISADOR ?institution)
]
[R4.3:      (?person FUNCIONARIO_VINCULADO ?institution)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:level 'Pós-Doutorado' )
            (?formation is:endYear 'rdf:nil')
      ->
            (?person PESQUISADOR ?institution)
]

[R5.1:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear 'rdf:nil')
            (?formation is:level 'Graduação')
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person ESTUDANTE ?institution)
]
[R5.2:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear 'rdf:nil')
            (?formation is:level 'Especialização')
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person ESTUDANTE ?institution)
]

[R5.3:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear 'rdf:nil')
            (?formation is:level 'Mestrado')
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person ESTUDANTE ?institution)
]

[R5.4:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear 'rdf:nil')
            (?formation is:level 'Doutorado')
```

```
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person ESTUDANTE ?institution)
]

[R6.1:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear ?endYear)
            notEqual(?endYear, 'rdf:nil' )
            (?formation is:level 'Graduação' )
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person EGRESSO ?institution)
]

[R6.2:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear ?endYear)
            notEqual(?endYear, 'rdf:nil' )
            (?formation is:level 'Especialização' )
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person EGRESSO ?institution)
]

[R6.3:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear ?endYear)
            notEqual(?endYear, 'rdf:nil' )
            (?formation is:level 'Mestrado' )
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person EGRESSO ?institution)
]

[R6.4:      (?person rdf:type is:Person)
            (?person is:hasFormation ?formation)
            (?formation rdf:type is:Formation)
            (?formation is:endYear ?endYear)
            notEqual(?endYear, 'rdf:nil' )
            (?formation is:level 'Doutorado' )
            (?formation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
      ->
            (?person EGRESSO ?institution)
]

[R7:        (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
```

```
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:nature 'Treinamento')
            (?activity is:endYear 'rdf:nil' )
      ->
            (?person TRAINEE ?institution)
]

[R8:        (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:nature 'Direção e Administração')
            (?activity is:endYear 'rdf:nil' )
      ->
            (?person ADMINISTRADOR ?institution)
]

[R9:        (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:nature 'Estágio')
            (?activity is:endYear 'rdf:nil' )
      ->
            (?person ESTAGIÁRIO ?institution)
]

[R10:       (?person rdf:type is:Person)
            (?person is:hasAffiliation ?affiliation)
            (?affiliation rdf:type is:Affiliation)
            (?affiliation is:hasInstitution ?institution)
            (?institution rdf:type is:Institution)
            (?affiliation is:hasActivity ?activity)
            (?activity rdf:type is:Activity)
            (?activity is:nature 'Serviço Técnico Especializado')
            (?activity is:endYear 'rdf:nil' )
      ->
            (?person SERVIDOR_TÉCNICO ?institution)
]
```

Apêndice D – Especificação da Ontologia BI em OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.stela.info#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.stela.info">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
  </owl:Ontology>
  <owl:Class rdf:ID="DBAttribute">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasReference"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasParent"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DBElement"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="isPrimaryKey"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DBCollection">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="schema"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DBElement"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
```

```
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#DBElement">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="domainConcept"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="name"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="Dimension">
  <rdfs:subClassOf rdf:resource="#DBCollection"/>
</owl:Class>
<owl:Class rdf:ID="Fact">
  <rdfs:subClassOf rdf:resource="#DBCollection"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#hasReference">
  <rdfs:domain rdf:resource="#DBAttribute"/>
  <rdfs:range rdf:resource="#DBAttribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasAttribute">
  <rdfs:domain rdf:resource="#DBCollection"/>
  <rdfs:range rdf:resource="#DBAttribute"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasParent"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDBCollection"/>
<owl:ObjectProperty rdf:about="#hasParent">
  <owl:inverseOf rdf:resource="#hasAttribute"/>
  <rdfs:range rdf:resource="#DBCollection"/>
  <rdfs:domain rdf:resource="#DBAttribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDimension">
  <rdfs:domain rdf:resource="#Fact"/>
  <rdfs:range rdf:resource="#Dimension"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="#name">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#DBAttribute"/>
        <owl:Class rdf:about="#DBElement"/>
      </owl:unionOf>
```

```
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isForeignKey">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#domainConcept">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#DBElement"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#isPrimaryKey">
  <rdfs:domain rdf:resource="#DBAttribute"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#schema">
  <rdfs:domain rdf:resource="#DBCollection"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<DBAttribute rdf:ID="DIM_FORMACAO.ANO_TERMINO">
  <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</isPrimaryKey>
  <hasParent>
    <Dimension rdf:ID="DIM_FORMACAO">
      <hasAttribute>
        <DBAttribute rdf:ID="DIM_FORMACAO.ANO_INICIO">
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isPrimaryKey>
          <hasParent rdf:resource="#DIM_FORMACAO"/>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >ANO_INI_FORM</name>
          <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.stela.info#beginYear</domainConcept>
        </DBAttribute>
      </hasAttribute>
      <hasAttribute rdf:resource="#DIM_FORMACAO.ANO_TERMINO"/>
      <hasAttribute>
        <DBAttribute rdf:ID="DIM_FORMACAO.SEQ_ID_PESSOA">
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >SEQ_ID_PESSOA</name>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</isPrimaryKey>
          <hasParent rdf:resource="#DIM_FORMACAO"/>
        </DBAttribute>
      </hasAttribute>
      <hasAttribute>
        <DBAttribute rdf:ID="DIM_FORMACAO.TPO_CURSO">
          <hasParent rdf:resource="#DIM_FORMACAO"/>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isPrimaryKey>
          <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.stela.info#courseType</domainConcept>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >TPO_CURSO</name>
        </DBAttribute>
      </hasAttribute>
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#Formation</domainConcept>
      <hasAttribute>
        <DBAttribute rdf:ID="DIM_FORMACAO.COD_INST">
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >COD_INST</name>
          <hasParent rdf:resource="#DIM_FORMACAO"/>
          <hasReference>
            <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.COD_INST">
              <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
>COD_INST</name>
<hasParent>
 <Dimension rdf:ID="DIM_INSTITUICAO_FORMACAO">
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.NME_INST">
    <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >NME_INST</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#name</domainConcept>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute rdf:resource="#DIM_INSTITUICAO_FORMACAO.COD_INST"/>
  <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >SBI</schema>
  <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >DIM_INSTITUICAO</name>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.DSC_PAIS_INST">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DSC_PAIS_INST</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#country</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.DSC_GRUPO_ECONOMICO">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DSC_GRUPO_ECON</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#economicGroup</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.NME_ORGAO_INST">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >NME_ORGAO_INST</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#department</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.SGL_INST">
    <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SGL_INST</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#acronym</domainConcept>
   </DBAttribute>
  </hasAttribute>
  <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >http://www.stela.info#Institution</domainConcept>
```

```
          <hasAttribute>
           <DBAttribute rdf:ID="DIM_INSTITUICAO_FORMACAO.SEQ_INST">
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >SEQ_INST</name>
            <hasParent rdf:resource="#DIM_INSTITUICAO_FORMACAO"/>
            <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</isPrimaryKey>
           </DBAttribute>
          </hasAttribute>
         </Dimension>
        </hasParent>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
       </DBAttribute>
     </hasReference>
     <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >http://www.stela.info#hasInstitution</domainConcept>
     <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
     >false</isPrimaryKey>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_FORMACAO.DSC_NIVEL">
     <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >DSC_NIVEL_FORM</name>
     <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >http://www.stela.info#level</domainConcept>
     <hasParent rdf:resource="#DIM_FORMACAO"/>
     <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
     >false</isPrimaryKey>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_FORMACAO.DSC_CURSO">
     <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >DSC_CURSO</name>
     <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
     >false</isPrimaryKey>
     <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >http://www.stela.info#courseName</domainConcept>
     <hasParent rdf:resource="#DIM_FORMACAO"/>
   </DBAttribute>
  </hasAttribute>
  <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >SBI</schema>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_FORMACAO.SEQ_FORM">
     <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
     >true</isPrimaryKey>
     <hasParent rdf:resource="#DIM_FORMACAO"/>
     <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >SEQ_FORM</name>
   </DBAttribute>
  </hasAttribute>
  <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >DIM_FORMACAO</name>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_FORMACAO.COD_AREA_CURSO">
     <hasReference>
      <DBAttribute rdf:ID="DIM_AREA_CONHEC_FORMACAO.COD_AREA_CONHEC">
       <hasParent>
        <Dimension rdf:ID="DIM_AREA_CONHEC_FORMACAO">
         <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >DIM_AREA_CONHEC</name>
         <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
            >http://www.stela.info#KnowledgeArea</domainConcept>
            <hasAttribute>
             <DBAttribute rdf:ID="DIM_AREA_CONHEC_FORMACAO.SEQ_AREA_CONHEC">
              <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
              >true</isPrimaryKey>
              <hasParent rdf:resource="#DIM_AREA_CONHEC_FORMACAO"/>
              <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >SEQ_AREA_CONHEC</name>
             </DBAttribute>
            </hasAttribute>
            <hasAttribute rdf:resource="#DIM_AREA_CONHEC_FORMACAO.COD_AREA_CONHEC"/>
            <hasAttribute>
             <DBAttribute rdf:ID="DIM_AREA_CONHEC_FORMACAO.NME_GRANDE_AREA">
              <hasParent rdf:resource="#DIM_AREA_CONHEC_FORMACAO"/>
              <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
              >false</isPrimaryKey>
              <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >http://www.stela.info#generalName</domainConcept>
              <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >NME_GRANDE_AREA</name>
             </DBAttribute>
            </hasAttribute>
            <hasAttribute>
             <DBAttribute rdf:ID="DIM_AREA_CONHEC_FORMACAO.NME_AREA_CONHEC">
              <hasParent rdf:resource="#DIM_AREA_CONHEC_FORMACAO"/>
              <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
              >false</isPrimaryKey>
              <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >http://www.stela.info#name</domainConcept>
              <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >NME_AREA_CONHEC</name>
             </DBAttribute>
            </hasAttribute>
            <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >SBI</schema>
           </Dimension>
          </hasParent>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isPrimaryKey>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >COD_AREA_CONHEC</name>
         </DBAttribute>
        </hasReference>
        <hasParent rdf:resource="#DIM_FORMACAO"/>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#hasKnowledgeArea</domainConcept>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >COD_AREA_CURSO</name>
       </DBAttribute>
      </hasAttribute>
     </Dimension>
    </hasParent>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#endYear</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ANO_TER_FORM</name>
   </DBAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.SEQ_ATIVIDADE">
    <hasParent>
     <Dimension rdf:ID="DIM_ATIVIDADE">
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_ATIVIDADE.TPO_CURSO">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >TPO_CURSO</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#courseType</domainConcept>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
   </DBAttribute>
  </hasAttribute>
  <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >SBI</schema>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.COD_AREA_CURSO">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >COD_AREA_CURSO</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#hasKnowlegeArea</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <hasReference>
     <DBAttribute rdf:ID="DIM_AREA_CONHEC_ATIVIDADE.COD_AREA_CONHEC">
      <hasParent>
       <Dimension rdf:ID="DIM_AREA_CONHEC_ATIVIDADE">
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#KnowledgeArea</domainConcept>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DIM_AREA_CONHEC</name>
        <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SBI</schema>
        <hasAttribute rdf:resource="#DIM_AREA_CONHEC_ATIVIDADE.COD_AREA_CONHEC"/>
        <hasAttribute>
         <DBAttribute rdf:ID="DIM_AREA_CONHEC_ATIVIDADE.SEQ_AREA_CONHEC">
          <hasParent rdf:resource="#DIM_AREA_CONHEC_ATIVIDADE"/>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</isPrimaryKey>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >SEQ_AREA_CONHEC</name>
         </DBAttribute>
        </hasAttribute>
        <hasAttribute>
         <DBAttribute rdf:ID="DIM_AREA_CONHEC_ATIVIDADE.NME_GRANDE_AREA">
          <hasParent rdf:resource="#DIM_AREA_CONHEC_ATIVIDADE"/>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isPrimaryKey>
          <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.stela.info#generalName</domainConcept>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >NME_GRANDE_AREA</name>
         </DBAttribute>
        </hasAttribute>
        <hasAttribute>
         <DBAttribute rdf:ID="DIM_AREA_CONHEC_ATIVIDADE.NME_AREA_CONHEC">
          <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >http://www.stela.info#name</domainConcept>
          <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >NME_AREA_CONHEC</name>
          <hasParent rdf:resource="#DIM_AREA_CONHEC_ATIVIDADE"/>
          <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</isPrimaryKey>
         </DBAttribute>
        </hasAttribute>
       </Dimension>
      </hasParent>
      <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</isPrimaryKey>
```

```
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >COD_AREA_CONHEC</name>
     </DBAttribute>
    </hasReference>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.STA_PROJETO_PESQ">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#isResearchProject</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >STA_PROJETO_PESQ</name>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.MES_TERMINO">
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >MES_FIM</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#endMonth</domainConcept>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.ANO_INICIO">
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ANO_INIC</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#beginYear</domainConcept>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.MES_INICIO">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >MES_INIC</name>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#beginMonth</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.DSC_CURSO">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#courseName</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DSC_CURSO</name>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_ATIVIDADE"/>
   </DBAttribute>
  </hasAttribute>
  <hasAttribute>
   <DBAttribute rdf:ID="DIM_ATIVIDADE.STA_ATIVIDADE_ATUAL">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#isActual</domainConcept>
```

```
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >STA_ATIVIDADE_ATUAL</name>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <hasParent rdf:resource="#DIM_ATIVIDADE"/>
       </DBAttribute>
      </hasAttribute>
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#Activity</domainConcept>
      <hasAttribute rdf:resource="#DIM_ATIVIDADE.SEQ_ATIVIDADE"/>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_ATIVIDADE.DSC_NATUREZA">
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <hasParent rdf:resource="#DIM_ATIVIDADE"/>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DSC_NATUR_ATIV</name>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#nature</domainConcept>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_ATIVIDADE.SEQ_FUNCAO_ATIV">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SEQ_FUNCAO_ATIV</name>
        <hasParent rdf:resource="#DIM_ATIVIDADE"/>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >true</isPrimaryKey>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_ATIVIDADE.ANO_TERMINO">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >ANO_FIM</name>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#endYear</domainConcept>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <hasParent rdf:resource="#DIM_ATIVIDADE"/>
       </DBAttribute>
      </hasAttribute>
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >DIM_ATIVIDADE</name>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_ATIVIDADE.SEQ_ID_PESSOA">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SEQ_ID_PESSOA</name>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >true</isPrimaryKey>
        <hasParent rdf:resource="#DIM_ATIVIDADE"/>
       </DBAttribute>
      </hasAttribute>
     </Dimension>
    </hasParent>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</isPrimaryKey>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SEQ_ATIVIDADE</name>
   </DBAttribute>
   <DBAttribute rdf:ID="DIM_HISTORICO.MES_FIM">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#endMonth</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >MES_FIM</name>
    <hasParent>
```

```
<Dimension rdf:ID="DIM_HISTORICO_ATIVIDADE">
 <hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.SEQ_ATIVIDADE_FK">
   <hasReference rdf:resource="#DIM_ATIVIDADE.SEQ_ATIVIDADE"/>
   <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
   >false</isPrimaryKey>
   <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
   <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >http://www.stela.info#hasActivity</domainConcept>
   <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >SEQ_ATIVIDADE</name>
  </DBAttribute>
 </hasAttribute>
 <hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.COD_INST">
   <hasReference>
    <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.COD_INST">
     <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >COD_INST</name>
     <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
     >false</isPrimaryKey>
     <hasParent>
      <Dimension rdf:ID="DIM_INSTITUICAO_HISTORICO">
       <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >SBI</schema>
       <hasAttribute>
        <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.DSC_GRUPO_ECONOMICO">
         <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >DSC_GRUPO_ECON</name>
         <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >http://www.stela.info#economicGroup</domainConcept>
         <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
         >false</isPrimaryKey>
         <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
        </DBAttribute>
       </hasAttribute>
       <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >http://www.stela.info#Institution</domainConcept>
       <hasAttribute>
        <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.NME_INST">
         <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >NME_INST</name>
         <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >http://www.stela.info#name</domainConcept>
         <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
         <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
         >false</isPrimaryKey>
        </DBAttribute>
       </hasAttribute>
       <hasAttribute>
        <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.SGL_INST">
         <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
         <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
         >false</isPrimaryKey>
         <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >http://www.stela.info#acronym</domainConcept>
         <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
         >SGL_INST</name>
        </DBAttribute>
       </hasAttribute>
       <hasAttribute rdf:resource="#DIM_INSTITUICAO_HISTORICO.COD_INST"/>
       <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >DIM_INSTITUICAO</name>
       <hasAttribute>
        <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.NME_ORGAO_INST">
```

```
            <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</isPrimaryKey>
            <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
            <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >http://www.stela.info#department</domainConcept>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >NME_ORGAO_INST</name>
           </DBAttribute>
          </hasAttribute>
          <hasAttribute>
           <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.DSC_PAIS_INST">
            <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
            <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</isPrimaryKey>
            <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >http://www.stela.info#country</domainConcept>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >DSC_PAIS_INST</name>
           </DBAttribute>
          </hasAttribute>
          <hasAttribute>
           <DBAttribute rdf:ID="DIM_INSTITUICAO_HISTORICO.SEQ_INST">
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >SEQ_INST</name>
            <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >true</isPrimaryKey>
            <hasParent rdf:resource="#DIM_INSTITUICAO_HISTORICO"/>
           </DBAttribute>
          </hasAttribute>
         </Dimension>
        </hasParent>
       </DBAttribute>
      </hasReference>
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >COD_INST</name>
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#hasInstitution</domainConcept>
      <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</isPrimaryKey>
      <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
     </DBAttribute>
    </hasAttribute>
    <schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SBI</schema>
    <hasAttribute>
     <DBAttribute rdf:ID="DIM_HISTORICO.SEQ_ID_PESSOA">
      <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
      <hasReference rdf:resource="#DIM_ATIVIDADE.SEQ_ID_PESSOA"/>
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#hasActivity</domainConcept>
      <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >true</isPrimaryKey>
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >SEQ_ID_PESSOA</name>
     </DBAttribute>
    </hasAttribute>
    <hasAttribute>
     <DBAttribute rdf:ID="DIM_HISTORICO.ANO_FIM">
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >ANO_FIM</name>
      <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#endYear</domainConcept>
      <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</isPrimaryKey>
```

```
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.TPO_REGIME_TRABALHO">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >TPO_REGIME_TRABALHO</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#workingRegime</domainConcept>
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.DSC_ENQ_FUNCIONAL">
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#functionalCareer</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DSC_ENQ_FUNCIONAL</name>
  </DBAttribute>
</hasAttribute>
<domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://www.stela.info#Affiliation</domainConcept>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.TPO_VINCULO">
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#affiliationType</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >TPO_VINCULO</name>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.ANO_INIC">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ANO_INIC</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#beginYear</domainConcept>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.SEQ_ATIVIDADE">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SEQ_ATIVIDADE</name>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</isPrimaryKey>
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
  </DBAttribute>
</hasAttribute>
<hasAttribute rdf:resource="#DIM_HISTORICO.MES_FIM"/>
<name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>DIM_HISTORICO_ATIVIDADE</name>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_HISTORICO.MES_INIC">
    <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
```

```
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >MES_INIC</name>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#beginMonth</domainConcept>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_HISTORICO.SEQ_ID_PESSOA_ATIVIDADE">
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#hasActivity</domainConcept>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SEQ_ID_PESSOA</name>
        <hasParent rdf:resource="#DIM_HISTORICO_ATIVIDADE"/>
        <hasReference rdf:resource="#DIM_ATIVIDADE.SEQ_ID_PESSOA"/>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
       </DBAttribute>
      </hasAttribute>
     </Dimension>
    </hasParent>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
   </DBAttribute>
   <DBAttribute rdf:ID="DIM_PESSOA.SGL_UF">
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent>
     <Dimension rdf:ID="DIM_PESSOA">
      <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >http://www.stela.info#Person</domainConcept>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_PESSOA.SGL_REGIAO">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SGL_REGIAO_NASC</name>
        <hasParent rdf:resource="#DIM_PESSOA"/>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#regionAcronym</domainConcept>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_PESSOA.SEXO">
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#gender</domainConcept>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >SEXO</name>
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <hasParent rdf:resource="#DIM_PESSOA"/>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_PESSOA.DSC_PAIS">
        <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</isPrimaryKey>
        <hasParent rdf:resource="#DIM_PESSOA"/>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DSC_PAIS_NASC</name>
        <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >http://www.stela.info#country</domainConcept>
       </DBAttribute>
      </hasAttribute>
      <hasAttribute>
       <DBAttribute rdf:ID="DIM_PESSOA.NME_PESSOA">
```

```
    <hasParent rdf:resource="#DIM_PESSOA"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#name</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >NME_RH</name>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_PESSOA.DTA_NASC">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#birthDate</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DTA_NASC</name>
    <hasParent rdf:resource="#DIM_PESSOA"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_PESSOA.SEQ_ID_PESSOA_FORM">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#hasFormation</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SEQ_ID_PESSOA</name>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasReference rdf:resource="#DIM_FORMACAO.SEQ_ID_PESSOA"/>
    <hasParent rdf:resource="#DIM_PESSOA"/>
  </DBAttribute>
</hasAttribute>
<schema rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>SBI</schema>
<hasAttribute rdf:resource="#DIM_PESSOA.SGL_UF"/>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_PESSOA.SEQ_ID_PESSOA">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SEQ_ID_PESSOA</name>
    <hasParent rdf:resource="#DIM_PESSOA"/>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</isPrimaryKey>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_PESSOA.EMAIL">
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#email</domainConcept>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >TXT_EMAIL</name>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <hasParent rdf:resource="#DIM_PESSOA"/>
  </DBAttribute>
</hasAttribute>
<hasAttribute>
  <DBAttribute rdf:ID="DIM_PESSOA.SEQ_ID_PESSOA_FUNCAO">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SEQ_ID_PESSOA</name>
    <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isPrimaryKey>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#hasAffiliation</domainConcept>
    <hasParent rdf:resource="#DIM_PESSOA"/>
    <hasReference rdf:resource="#DIM_HISTORICO.SEQ_ID_PESSOA"/>
```

```
          </DBAttribute>
        </hasAttribute>
        <hasAttribute>
          <DBAttribute rdf:ID="DIM_PESSOA.RACA">
            <hasParent rdf:resource="#DIM_PESSOA"/>
            <isPrimaryKey rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</isPrimaryKey>
            <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >NME_RACA</name>
            <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >http://www.stela.info#race</domainConcept>
          </DBAttribute>
        </hasAttribute>
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >DIM_PESSOA</name>
      </Dimension>
    </hasParent>
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SGL_UF_NASC</name>
    <domainConcept rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >http://www.stela.info#stateAcronym</domainConcept>
  </DBAttribute>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)  http://protege.stanford.edu -->
```

Apêndice E - Consulta SQL gerada automaticamente no estudo de caso

```
SELECT   C1.SGL_REGIAO_NASC A1, C1.SEXO A2, C1.DSC_PAIS_NASC A3, C1.NME_RH
         A4, C1.DTA_NASC A5, C1.SEQ_ID_PESSOA A6, C1.SGL_UF_NASC A7,
         C1.SEQ_ID_PESSOA A8, C1.TXT_EMAIL A9, C1.SEQ_ID_PESSOA A10,
         C1.NME_RACA A11, C2.ANO_INI_FORM A12, C2.ANO_TER_FORM A13,
         C2.SEQ_ID_PESSOA A14, C2.TPO_CURSO A15, C2.COD_INST A16,
         C2.DSC_NIVEL_FORM A17, C2.DSC_CURSO A18, C2.SEQ_FORM A19,
         C2.COD_AREA_CURSO A20, C3.NME_AREA_CONHEC A21,
         C3.NME_GRANDE_AREA A22, C3.COD_AREA_CONHEC A23,
         C3.SEQ_AREA_CONHEC A24, C4.NME_INST A25, C4.COD_INST A26,
         C4.DSC_PAIS_INST A27, C4.DSC_GRUPO_ECON A28, C4.NME_ORGAO_INST
         A29, C4.SGL_INST A30, C4.SEQ_INST A31, C5.SEQ_ATIVIDADE A32,
         C5.COD_INST A33, C5.SEQ_ID_PESSOA A34, C5.ANO_FIM A35,
         C5.TPO_REGIME_TRABALHO A36, C5.DSC_ENQ_FUNCIONAL A37,
         C5.TPO_VINCULO A38, C5.ANO_INIC A39, C5.SEQ_ATIVIDADE A40,
         C5.MES_FIM A41, C5.MES_INIC A42, C5.SEQ_ID_PESSOA A43,
         C6.DSC_GRUPO_ECON A44, C6.NME_INST A45, C6.SGL_INST A46,
         C6.COD_INST A47, C6.NME_ORGAO_INST A48, C6.DSC_PAIS_INST A49,
         C6.SEQ_INST A50, C7.TPO_CURSO A51, C7.COD_AREA_CURSO A52,
         C7.STA_PROJETO_PESQ A53, C7.MES_FIM A54, C7.ANO_INIC A55,
         C7.MES_INIC A56, C7.DSC_CURSO A57, C7.STA_ATIVIDADE_ATUAL A58,
         C7.SEQ_ATIVIDADE A59, C7.DSC_NATUR_ATIV A60, C7.SEQ_FUNCAO_ATIV
         A61, C7.ANO_FIM A62, C7.SEQ_ID_PESSOA A63, C8.NME_AREA_CONHEC A64,
         C8.NME_GRANDE_AREA A65, C8.SEQ_AREA_CONHEC A66,
         C8.COD_AREA_CONHEC A67
FROM     DIM_PESSOA C1
          LEFT OUTER JOIN DIM_FORMACAO C2
            ON C1.SEQ_ID_PESSOA = C2.SEQ_ID_PESSOA
          LEFT OUTER JOIN DIM_AREA_CONHEC C3
            ON C2.COD_AREA_CURSO = C3.COD_AREA_CONHEC
          LEFT OUTER JOIN DIM_INSTITUICAO C4
            ON C2.COD_INST = C4.COD_INST
          LEFT OUTER JOIN DIM_HISTORICO_ATIVIDADE C5
            ON C1.SEQ_ID_PESSOA = C5.SEQ_ID_PESSOA
          LEFT OUTER JOIN DIM_INSTITUICAO C6
            ON C5.COD_INST = C6.COD_INST
          LEFT OUTER JOIN DIM_ATIVIDADE C7
            ON C5.SEQ_ATIVIDADE = C7.SEQ_ATIVIDADE AND
               C5.SEQ_ID_PESSOA = C7.SEQ_ID_PESSOA
          LEFT OUTER JOIN DIM_AREA_CONHEC C8
            ON C7.COD_AREA_CURSO = C8.COD_AREA_CONHEC
```

OBS: Todos os *alias* (apelidos) das tabelas e atributos são estabelecidos pela ferramenta. Utiliza-se o prefixo C para as tabelas e o prefixo A para os atributos.

Apêndice F – Configurações utilizadas no aplicativo

```
############################################################
#
#           ARQUIVO DE CONFIGURAÇÕES do SBI
#
#           Autor: Dhiogo Cardoso da Silva
#
############################################################
# REPOSITÓRIOS DE ONTOLOGIAS
sbi.repository.domain   =     repository/domain
sbi.repository.bi       =     repository/bi
sbi.repository.rules    =     repository/rules
############################################################
# CONCEITO OU PONTO INICIAL PARA NAVEGAÇÃO NA ONTOLOGIA
sbi.ontology.start      = http://www.stela.info#Person

############################################################
# PARÂMETRO PARA IMPRESSÃO NA TELA
sbi.print.enable  = false

############################################################
# FONTE DE DADOS

#Configuraçõs do Modelo Tripla
sbi.db.tripleFact.name = FATO_TRIPLA
sbi.db.tripleFact.subject = SEQ_ID_SUJEITO
sbi.db.tripleFact.predicate = DSC_PREDICADO
sbi.db.tripleFact.object = SEQ_ID_OBJETO

#Configurações do SGBD

#ORACLE
#sbi.db.jdbcDriver =oracle.jdbc.driver.OracleDriver
#sbi.db.url =jdbc:oracle:thin:@192.168.0.247:1521:orades
#sbi.db.user =
#sbi.db.pass =

#MYSQL
sbi.db.jdbcDriver = com.mysql.jdbc.Driver
sbi.db.url = jdbc:mysql://localhost/sbi
sbi.db.user = root
sbi.db.pass = root

#SYBASE
#sbi.db.jdbcDriver: com.sybase.jdbc2.jdbc.SybDriver
#sbi.db.url: jdbc:sybase:Tds:192.168.0.244:5000
#sbi.db.user:
#sbi.db.pass:
```

Anexo A – Código-fonte do aplicativo

```
package info.stela.sbi.repository;

import info.stela.sbi.repository.impl.OntologyRepositoryImpl;


public interface SBIRepositoryFactory {

	/**
	 * Método utilizado para criar uma instância de SBIRepository
	 * @param implementator Responsável por implementar as operações do
	repositório
	 * @return Repositório de instâncias da ontologia
	 * @throws Exception
	 */
	public abstract SBIRepository createRepository(OntologyRepositoryImpl
	implementator) throws Exception;

}
package info.stela.sbi.repository;

import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public abstract class BIOntologyRepository extends OntologyRepository{

	public BIOntologyRepository(OntologyRepositoryImpl implementator) {
		super(implementator);
	}


}
package info.stela.sbi.repository;

import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIInference;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public abstract class DomainOntologyRepository extends OntologyRepository{

	public DomainOntologyRepository(OntologyRepositoryImpl implementator) {
		super(implementator);
	}

	/**
	 * Método utilizado para buscar os resultados das inferências de uma
	instância da ontologia
	 * @param biInstance Instância de domínio em que as regras serão
	aplicadas
	 * @return A coleção dos resultados de inferências aplicadas na instância
	da ontologia de domínio
	 * @throws Exception Caso algum problema na consulta ao repositório ou
	realização das inferências ocorrer
	 */
	public abstract SBICollection<SBIInference>
	getInferences(BIOntologyInstance biInstance) throws Exception;
```

```
}
package info.stela.sbi.repository;

import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public class InstanceRepository extends OntologyRepository {

	public InstanceRepository(OntologyRepositoryImpl implementator) {
		super(implementator);
	}

}
package info.stela.sbi.repository;

import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public class OntologyRepository implements SBIRepository{

	private OntologyRepositoryImpl implementator;

	public OntologyRepository(OntologyRepositoryImpl implementator){
		this.setImplementator(implementator);
	}


	/**
	 * @param instances
	 * @return
	 * @throws Exception
	 * @see
	info.stela.sbi.repository.SBIRepository#add(info.stela.sbi.ontology.colle
	ction.SBICollection)
	 */
	public boolean add(SBICollection<SBIOntologyInstance> instances) throws
	SBIRepositoryException{
		return implementator.add(instances);
	}



	/**
	 * @param instance
	 * @return
	 * @throws Exception
	 * @see
	info.stela.sbi.repository.SBIRepository#add(info.stela.sbi.ontology.SBIOn
	tologyInstance)
	 */
	public boolean add(SBIOntologyInstance instance) throws
	SBIRepositoryException {
		return implementator.add(instance);
	}
```

```
/**
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#connect()
 */
public void connect() throws SBIRepositoryException {
      implementator.connect();
}



/**
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#disconnect()
 */
public void disconnect() throws SBIRepositoryException {
      implementator.disconnect();
}



/**
 * @param instance
 * @return
 * @throws Exception
 * @see
info.stela.sbi.repository.SBIRepository#get(info.stela.sbi.ontology.SBIOn
tologyInstance)
 */
public SBICollection<SBIOntologyInstance> get(SBIOntologyInstance
instance) throws SBIRepositoryException {
      return implementator.get(instance);
}



/**
 * @param uri
 * @return
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#get(java.lang.String)
 */
public SBICollection<SBIOntologyInstance> get(String uri) throws
SBIRepositoryException {
      return implementator.get(uri);
}



/**
 * @return
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#get()
 */
public SBICollection<SBIOntologyInstance> get() throws
SBIRepositoryException {
      return implementator.get();
}
```

```
/**
 * @return
 * @see info.stela.sbi.repository.SBIRepository#isConnected()
 */
public boolean isConnected() {
      return implementator.isConnected();
}



/**
 * @param instance
 * @return
 * @throws Exception
 * @see
info.stela.sbi.repository.SBIRepository#remove(info.stela.sbi.ontology.SB
IOntologyInstance)
 */
public boolean remove(SBIOntologyInstance instance) throws
SBIRepositoryException {
      return implementator.remove(instance);
}



/**
 * @param uri
 * @return
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#remove(java.lang.String)
 */
public boolean remove(String uri) throws SBIRepositoryException {
      return implementator.remove(uri);
}



/**
 * @return
 * @throws Exception
 * @see info.stela.sbi.repository.SBIRepository#removeAll()
 */
public boolean removeAll() throws SBIRepositoryException {
      return implementator.removeAll();
}



/**
 * @param instance
 * @param newInstance
 * @return
 * @throws Exception
 * @see
info.stela.sbi.repository.SBIRepository#set(info.stela.sbi.ontology.SBIOn
tologyInstance, info.stela.sbi.ontology.SBIOntologyInstance)
 */
```

```
	public boolean set(SBIOntologyInstance instance, SBIOntologyInstance
	newInstance) throws SBIRepositoryException {
		return implementator.set(instance, newInstance);
	}

	/**
	 * @param uri
	 * @param instance
	 * @return
	 * @throws Exception
	 * @see info.stela.sbi.repository.SBIRepository#set(java.lang.String,
	info.stela.sbi.ontology.SBIOntologyInstance)
	 */
	public boolean set(String uri, SBIOntologyInstance instance) throws
	SBIRepositoryException {
		return implementator.set(uri, instance);
	}


	public void cancel() throws SBIRepositoryException {
		implementator.cancel();
	}


	public void save() throws SBIRepositoryException {
		implementator.save();
	}

	////////////////////////////////////////////////////////////////////
	//GET's and SET's


	public OntologyRepositoryImpl getImplementator() {
		return implementator;
	}

	public void setImplementator(OntologyRepositoryImpl implementator) {
		this.implementator = implementator;
	}


	public boolean isEmpty() throws SBIRepositoryException {
		return implementator.isEmpty();
	}

}


package info.stela.sbi.repository;

import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;

/**
 * SBIRepository representa um repositório da arquitetura SBI.
 * Esse repositório pode conter ontologias, conclusões ou regras de inferências.
 *
 * @author Dhiogo
 *
```

```
 */
public interface SBIRepository {


        /**
         * Método utilizado para realizar a conexão com o repositório
         * @throws Exception Caso ocorrer algum problema ao realizar a conexão
        com o repositório
         */
        public abstract void connect() throws SBIRepositoryException;

        /**
         * Método utilizado para liberar a conexão do repositório
         * @throws Exception Caso ocorrer algum problema ao realizar a desconexão
         */
        public abstract void disconnect() throws SBIRepositoryException;

        /**
         * Método que indica se está conectado com o repositório
         * @return <tt>True</tt> está conectado ou <tt>False</tt> caso não esteja
        conectado
         */
        public abstract boolean isConnected();

        /**
         * Método que adiciona uma nova instância ao repositório
         * @param instance Instância que será adicionada ao repositório
         * @return <tt>True</tt> caso a instância for adicionada com êxito ou
        <tt>False</tt> caso contrário
         * @throws Exception
         */
        public abstract boolean add(SBIOntologyInstance instance) throws
        SBIRepositoryException;

        /**
         * Método que adiciona uma coleção de instâncias ao repositório
         * @param instances Coleção de instâncias que serão adicionadas ao
        repositório
         * @return <tt>True</tt> caso a instância for adicionada com êxito ou
        <tt>False</tt> caso contrário
         */
        public abstract boolean add(SBICollection<SBIOntologyInstance> instances)
        throws SBIRepositoryException;

        /**
         * Método que remove uma instância do repositório
         * @param instance Instância a ser removida
         * @return <tt>True</tt> caso a instância for removida com êxito ou
        <tt>False</tt> caso contrário
         */
        public abstract boolean remove(SBIOntologyInstance instance)  throws
        SBIRepositoryException;

        /**
         * Método que remove uma instância do repositório
         * @param uri Identificador (URI) da instância a ser removida
         * @return <tt>True</tt> caso a instância for removida com êxito ou
        <tt>False</tt> caso contrário
         */
```

```
public abstract boolean remove(String uri)  throws
SBIRepositoryException;

/**
 * Método que remove todas as instâncias do repositório
 * @return <tt>True</tt> caso todas instâncias forem removidas com êxito
ou <tt>False</tt> caso contrário
 */
public abstract boolean removeAll()  throws SBIRepositoryException;

/**
 * Método que substitui a instância por outra no repositório.
 * @param uri Identificador (URI) da instância a ser subsituída ou
adicionada
 * @param instance A instância a ser substituída
 * @return <tt>True</tt> caso encontre o uri e a instância for
substituída com êxito ou <tt>False</tt> caso contrário
 */
public abstract boolean set(String uri, SBIOntologyInstance instance)
throws SBIRepositoryException;

/**
 * Método que substitui a instância por outra no repositório.
 * @param instance instância a ser substituída
 * @param newInstance Nova instância a ser armazenada no repositório
 * @return <tt>True</tt> caso encontre a instância a ser substituída e a
nova instância for inserida com êxito ou <tt>False</tt> caso contrário
 */
public abstract boolean set(SBIOntologyInstance instance,
SBIOntologyInstance newInstance)  throws SBIRepositoryException;

/**
 * Método que retorna todas as instâncias do repositório
 * @return Coleção que contém todas as instâncias do repositório
 */
public abstract SBICollection<SBIOntologyInstance> get()  throws
SBIRepositoryException;

/**
 * Método utilizado para buscar todas as instâncias através de um
identificador (URI)
 * @param uri Identificador da instância
 * @return Conjunto de instâncias identificadas pelo URI
 */
public abstract SBICollection<SBIOntologyInstance> get(String uri)
throws SBIRepositoryException;


/**
 * Método utilizado para buscar todas as instâncias através de uma outra
instância de referência
 * @param instance Instância de referência
 * @return Conjunto de instâncias associadas a instância de referência
 */
public abstract SBICollection<SBIOntologyInstance>
get(SBIOntologyInstance instance) throws SBIRepositoryException;


/**
```

```
         * Método utilizado para cancelar as operações realizadas que ainda não
        foram salvas
         */
        public abstract void cancel() throws SBIRepositoryException;

        /**
         * Método utilizado para salvar as operações efetuadas
         * @throws Exception Caso algum problema no salvamento das operações
        ocorrer
         */
        public abstract void save() throws SBIRepositoryException;

        /**
         * Método que determina se o repositório possui ou nã instâncias
         * @return <tt>True</tt> caso o repositório tenha uma ou mais instâncias
        ou <tt>False</tt> caso constrário
         * @throws Exception Caso algum problema no acesso ao repositório ocorrer
         */
        public boolean isEmpty() throws SBIRepositoryException;




}


package info.stela.sbi.repository;

public class SBIRepositoryException extends Exception{

        public SBIRepositoryException() {
              super();
        }

        public SBIRepositoryException(String message, Throwable cause) {
              super(message, cause);
              // TODO Auto-generated constructor stub
        }

        public SBIRepositoryException(String message) {
              super(message);
              // TODO Auto-generated constructor stub
        }

        public SBIRepositoryException(Throwable cause) {
              super(cause);
              // TODO Auto-generated constructor stub
        }

}

package info.stela.sbi.repository.impl;

import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.repository.SBIRepositoryException;
import info.stela.sbi.repository.SBIRepository;

/**
```

```
 * Classe que fornece a implementação do SBIRepository, baseado no padrão de
        projeto conhecido como Bridge.
 * Cada OntologyRepository está associado a uma instância de
        OntologyRepositoryImpl que é responsável por realizar todas
 * as operações ou métodos do repositório. Dessa forma, a abstração do repositório
        fica desacoplada de sua implementação.
 *
 * @author Dhiogo
 *
 */
public abstract class OntologyRepositoryImpl implements SBIRepository{

        /**
         * Construtor padrão de OntologyRepositoryImpl
         */
        public OntologyRepositoryImpl(){
        }

        public boolean add(SBICollection<SBIOntologyInstance> instances) throws
        SBIRepositoryException {
              boolean added = false;
              try {
                    while ( instances.hasNext() )
                          this.add(instances.next());
                    added = true;
              } catch (SBIRepositoryException e) {
                    throw e;
              } catch (Exception e) {
                    throw new SBIRepositoryException(e);
              }
              return added;
        }



}
package info.stela.sbi.repository.impl;

import info.stela.backend.general.database.DBConnector;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.ontology.collection.SBIResultSet;
import info.stela.sbi.repository.SBIRepositoryException;
import info.stela.utils.SQLStatementMgr;
import info.stela.writer.SQLWriter;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;

public abstract class DBOntologyRepositoryImpl extends OntologyRepositoryImpl{

        private DBConnector connector;
        private boolean isConnected;

        public DBOntologyRepositoryImpl(String driverJDBC, String url, String
        username, String password){
```

```
	this.setConnector(new DBConnector(driverJDBC, url, username,
password));
	this.setConnected(false);
}
public DBOntologyRepositoryImpl(Connection connection) throws
SQLException{
	this.setConnector(new DBConnector(connection));
	this.setConnected(false);
}


public abstract SQLWriter getSQLInsert(SBIOntologyInstance instance);
public abstract SQLWriter getSQLDelete(SBIOntologyInstance instance);
public abstract SQLWriter getSQLUpdate(SBIOntologyInstance instance);
public abstract SQLWriter getSQLSelect(SBIOntologyInstance instance);
public abstract SQLWriter getSQLSelect(String uri);
public abstract SQLWriter getSQLSelect();

public void connect() throws SBIRepositoryException {
	try {
		this.getConnector().initConnection();
		this.setConnected(true);
	} catch (Exception e) {
		throw new SBIRepositoryException(e);
	}

}
public void disconnect() throws SBIRepositoryException {
	try {
		this.getConnector().closeConnection();
		this.setConnected(false);
	} catch (SQLException e) {
		throw new SBIRepositoryException(e);
	}
}
public boolean isConnected() {
	return this.isConnected;
}

/**
 * @return the connector
 */
public DBConnector getConnector() {
	return connector;
}

/**
 * @param connector the connector to set
 */
public void setConnector(DBConnector connector) {
	this.connector = connector;
}

/**
 * @param isConnected the isConnected to set
 */
public void setConnected(boolean isConnected) {
	this.isConnected = isConnected;
}
```

```
protected Connection getConnection(){
      return this.getConnector().getConnection();
}

public boolean add(SBIOntologyInstance instance) throws
SBIRepositoryException {
      //this.executeSQL(this.getSQLDelete(instance));
      return this.executeSQL(this.getSQLInsert(instance));
}

private boolean executeSQL(SQLWriter sqlWriter) throws
SBIRepositoryException{
      boolean added = false;
      try {
            if ( sqlWriter != null ){

      SQLStatementMgr.executeSql(this.getConnection(),sqlWriter, false);
                  added = true;
            }
      } catch (Exception e) {
            throw new SBIRepositoryException(e);
      }
      return added;
}

public SBICollection<SBIOntologyInstance> get(String uri) throws
SBIRepositoryException{
      SBICollection resultset = null;
      PreparedStatement pstSelect = null;
      ArrayList params = new ArrayList();
      try {
            String sql = this.getSQLSelect(uri).getSQLParsed(params);
            pstSelect = this.getConnection().prepareStatement(sql);
            String name;
            Object objValue;
            for (int i = 0; i < params.size(); i++) {
                  name = (String) params.get(i);
                  objValue = this.getSQLSelect(uri).getParam(name);
                  pstSelect.setObject(i + 1, objValue);
            }
            resultset = new SBIResultSet(null, pstSelect.executeQuery());

      }catch(Exception e){
            throw new SBIRepositoryException(e);
      }
      return resultset;
}

public SBICollection<SBIOntologyInstance> get(SBIOntologyInstance
instance) throws SBIRepositoryException {
      SBICollection resultset;
      PreparedStatement pstSelect = null;
      ArrayList params = new ArrayList();
      try {
            String sql =
this.getSQLSelect(instance).getSQLParsed(params);
            pstSelect = this.getConnection().prepareStatement(sql);
            String name;
```

```
            Object objValue;
            for (int i = 0; i < params.size(); i++) {
                  name = (String) params.get(i);
                  objValue = this.getSQLSelect(instance).getParam(name);
                  pstSelect.setObject(i + 1, objValue);
            }

            resultset = new SBIResultSet((BIOntologyInstance) instance,
pstSelect.executeQuery());

      }catch(Exception e){
            throw new SBIRepositoryException(e);
      }
      return resultset;
}


public SBICollection<SBIOntologyInstance> get() throws
SBIRepositoryException {
      SBICollection<SBIOntologyInstance> resultset;
      PreparedStatement pstSelect = null;
      ArrayList params = new ArrayList();
      try {
            String sql = this.getSQLSelect().getSQLParsed(params);
            pstSelect = this.getConnection().prepareStatement(sql);
            String name;
            Object objValue;
            for (int i = 0; i < params.size(); i++) {
                  name = (String) params.get(i);
                  objValue = this.getSQLSelect().getParam(name);
                  pstSelect.setObject(i + 1, objValue);
            }
            resultset = new SBIResultSet<SBIOntologyInstance>(null,
pstSelect.executeQuery());

      }catch(Exception e){
            throw new SBIRepositoryException(e);
      }
      return resultset;
}


public boolean remove(SBIOntologyInstance instance)throws
SBIRepositoryException {
      // TODO Auto-generated method stub
      return false;
}


public boolean remove(String uri)throws SBIRepositoryException {
      // TODO Auto-generated method stub
      return false;
}


public boolean removeAll()throws SBIRepositoryException {
      // TODO Auto-generated method stub
      return false;
}
```

```
	public boolean set(String uri, SBIOntologyInstance instance)throws
	SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}


	public boolean set(SBIOntologyInstance instance, SBIOntologyInstance
	newInstance) throws SBIRepositoryException{
		// TODO Auto-generated method stub
		return false;
	}


	public SBICollection<SBIOntologyInstance> get(SBIOntologyInstance
	instance, SBIOntologyInstance referenceInstance) throws
	SBIRepositoryException {
		// TODO Auto-generated method stub
		return null;
	}


	public SBICollection<SBIOntologyInstance> get(SBIOntologyInstance
	instance, String referenceURI)throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return null;
	}


	public SBICollection<SBIOntologyInstance> get(String uri, String
	referenceURI) throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return null;
	}


	public SBICollection<SBIOntologyInstance> get(String uri,
	SBIOntologyInstance referenceInstance)throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return null;
	}




}

package info.stela.sbi.repository.impl;

import java.io.File;
import java.io.FileNotFoundException;

/**
 * Classe responsável pela implementação de OntologyRepository que utiliza o
	armazenamento em arquivo
 *
```

```
 * @author Dhiogo
 */
public abstract class FileOntologyRepositoryImpl extends OntologyRepositoryImpl {

	private String fileName;

	public FileOntologyRepositoryImpl(String fileName) throws
	FileNotFoundException {
		this.setFileName(fileName);
	}


	public String getFileName() {
		return fileName;
	}

	public void setFileName(String fileName) throws FileNotFoundException {
		this.fileName = fileName;
	}



}
package info.stela.sbi.repository.impl;


public abstract class MemoryOntologyRepositoryImpl extends OntologyRepositoryImpl{

	public MemoryOntologyRepositoryImpl(){

	}

}

package info.stela.sbi.ontology;

import java.util.HashMap;
import java.util.Set;

public class SBIOntologyInstance{

	private String uri;
	private HashMap<String, Object > propertiesMap;

	public SBIOntologyInstance(String uri){
		this.setURI(uri);
		this.setPropertiesMap( new HashMap<String,Object >() );
	}

	public SBIOntologyInstance(String uri, HashMap<String,Object >
	properties){
		this.setURI(uri);
		this.setPropertiesMap(properties);
	}

	public Object getProperties(String domainURI){
		return this.getPropertiesMap().get(domainURI);
	}
```

```
public void clear() {
      propertiesMap.clear();
}

public boolean containsProperty(String property) {
      return propertiesMap.containsKey(property);
}

public boolean isEmpty() {
      return propertiesMap.isEmpty();
}

public  Object getValue( String property) {
      return propertiesMap.get(property);
}

public Set<String> getProperties(){
      return this.propertiesMap.keySet();
}

/**
 * @param value
 * @return
 * @see java.util.HashMap#containsValue(java.lang.Object)
 */
public boolean containsValue(Object value) {
      return propertiesMap.containsValue(value);
}

/**
 * @param key
 * @param value
 * @return
 * @see java.util.HashMap#put(java.lang.Object, java.lang.Object)
 */
public Object add(String property, HashMap<String, Object> value ) {
      return propertiesMap.put(property, value);
}

/**
 * @param key
 * @return
 * @see java.util.HashMap#remove(java.lang.Object)
 */
public Object remove(Object key) {
      return propertiesMap.remove(key);
}

/**
 * @return
 * @see java.util.HashMap#size()
 */
public int size() {
      return propertiesMap.size();
}

/**
```

```
	 * Retorna o URI (Universal Resource Identifier) para a instância da
	ontologia
	 * @return Identificador da instância da Ontologia
	 */
	public String getURI(){
		return this.uri;
	}

	/**
	 * @param uri the uri to set
	 */
	public void setURI(String uri) {
		this.uri = uri;
	}

	/**
	 * @return the propertiesMap
	 */
	protected HashMap<String, Object> getPropertiesMap() {
		return propertiesMap;
	}

	/**
	 * @param propertiesMap the propertiesMap to set
	 */
	protected void setPropertiesMap(HashMap<String, Object> properties) {
		this.propertiesMap = properties;
	}



}
package info.stela.sbi.ontology;

import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;

import java.util.HashMap;

public class BIOntologyInstance extends SBIOntologyInstance{

	private DBCollection dbCollection;

	public BIOntologyInstance(String uri, DBCollection dbcollection) {
		super(uri);
		this.setDBCollection(dbcollection);
	}

	public BIOntologyInstance(String uri, HashMap<String, Object >
	properties) {
		super(uri, properties);
	}

	public BIOntologyInstance(String uri, String id, HashMap<String, Object >
	properties) {
		super(uri, properties);
	}

	public BIOntologyInstance(String uri) {
```

```
		super(uri);
	}


	public void add(DBAttribute dbAttribute) {
		dbCollection.add(dbAttribute);
		dbAttribute.setParentElement(this.getDBCollection());
	}


	public DBCollection getDBCollection() {
		return dbCollection;
	}

	public void setDBCollection(DBCollection dbcollection) {
		this.dbCollection = dbcollection;
	}


}
package info.stela.sbi.ontology;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;


public class SBIInference extends SBIOntologyInstance {

	//Representa as chaves ou identificadores do sujeito
	//(PK simples ou composta)
	private List subjectKeys;

	private Object predicate;

	//Representa as chaves ou identificadores do objeto
	//(PK simples ou composta)
	private List  objectKeys;

	public SBIInference(String uri, HashMap<String, Object > properties) {
		super(uri, properties);
		this.setSubjectKeys(new ArrayList());
		this.setObjectKeys(new ArrayList());
	}

	public SBIInference(String uri, String id, HashMap<String, Object >
	properties) {
		super(uri, properties);
		this.setSubjectKeys(new ArrayList());
		this.setObjectKeys(new ArrayList());
	}

	public SBIInference(String uri) {
		super(uri);
		this.setSubjectKeys(new ArrayList());
		this.setObjectKeys(new ArrayList());
	}

	public List getObjectKeys() {
```

```
		return objectKeys;
	}

	public void setObjectKeys(List objectKeys) {
		this.objectKeys = objectKeys;
	}

	public Object getPredicate() {
		return predicate;
	}

	public void setPredicate(Object predicate) {
		this.predicate = predicate;
	}

	public List getSubjectKeys() {
		return subjectKeys;
	}

	public void setSubjectKeys(List subjectKeys) {
		this.subjectKeys = subjectKeys;
	}


}

package info.stela.sbi.ontology.collection;

import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;

import java.sql.ResultSet;
import java.sql.SQLException;

public class SBIResultSet<D> implements SBICollection {

	private ResultSet resultSet;
	private boolean isClosed;
	private BIOntologyInstance mappingInstance;

	public SBIResultSet(BIOntologyInstance mappingInstance, ResultSet
	resultSet){
		this.setResultSet(resultSet);
		this.setClosed(false);
		this.setMappingInstance(mappingInstance);
	}


	/**
	 * Método que verifica se há novas instâncias retornadas pela consulta à
	base de dados (ResultSet).
	 * Caso não haja mais instâncias, o cursor da consulta é automaticamente
	fechado.
	 */
	public boolean hasNext() throws SQLException{
		boolean hasNext = false;
		try {
			if ( this.isClosed() == false ){
```

```
                  hasNext = this.getResultSet().next();
                  if ( hasNext == false)
                        this.close();
            }
      } catch (SQLException e) {
            throw e;
      }
      return hasNext;
}

public void close() throws SQLException {
      this.getResultSet().close();
      this.setResultSet(null);
      this.setClosed(true);
}


public BIOntologyInstance next() throws SQLException {
      this.setValues();
      return this.getMappingInstance();

}



private void setValues() throws SQLException{
      setValues(this.getMappingInstance().getDBCollection(),
this.getResultSet() );
}

private void setValues( DBCollection dbCollection , ResultSet resultset)
throws SQLException{
      DBAttribute dbAttribute;
      for (int i = 0 ; i < dbCollection.getAttributes().size(); i++ ){
            dbAttribute = dbCollection.getAttribute(i);
            dbAttribute.setValue(
resultset.getObject(dbAttribute.getAlias()) );
            if ( dbAttribute.hasReference() )
                  setValues((DBCollection)
dbAttribute.getReferenceAttribute().getParentElement(), resultset);
            else{
                  dbAttribute.setValue(
resultset.getObject(dbAttribute.getAlias()) );
            }
      }
}




private ResultSet getResultSet() {
      return resultSet;
}



private void setResultSet(ResultSet resultSet) {
      this.resultSet = resultSet;
```

```
	}


	/**
	 * @return the isClosed
	 */
	public boolean isClosed() {
		return isClosed;
	}


	/**
	 * @param isClosed the isClosed to set
	 */
	public void setClosed(boolean isClosed) {
		this.isClosed = isClosed;
	}


	public BIOntologyInstance getMappingInstance() {
		return mappingInstance;
	}


	public void setMappingInstance(BIOntologyInstance mappingInstance) {
		this.mappingInstance = mappingInstance;
	}


}
package info.stela.sbi.ontology.collection;


public interface SBICollection<S> {

	/**
	 * Método que indica se há novas instâncias na coleção
	 * @return
	 * @throws Exception
	 */
	public abstract boolean hasNext() throws Exception;

	/**
	 * Método que retorna a próxima instância da coleção
	 * @return Próxima instância da coleção
	 * @throws Exception
	 */
	public abstract S next() throws Exception ;


}
package info.stela.sbi.ontology.collection;

import info.stela.sbi.ontology.SBIOntologyInstance;

import java.util.Iterator;


public class SBIIterator<S> implements SBICollection{
```

```
	private Iterator<S> iterator;

	public SBIIterator(Iterator<S> iterator){
		this.setIterator(iterator);
	}

	public boolean hasNext() throws Exception {
		return this.getIterator().hasNext();
	}

	public SBIOntologyInstance next() throws Exception {
		return (SBIOntologyInstance) this.getIterator().next();
	}

	private Iterator<S> getIterator() {
		return iterator;
	}

	private void setIterator(Iterator<S> iterator) {
		this.iterator = iterator;
	}

	public void add(Iterator<S> iterator){

	}




}

package info.stela.sbi.ontology.bi;

/**
 * DBElement representa um elemento relacionado ao Banco de Dados. Dessa forma, um
	DBElement pode representar
 * uma tabela (@see info.stela.sbi.ontology.bi.DBCollection) ou atributos de uma
	table (@see info.stela.sbi.ontology.bi.DBAttribute)
 *
 * @since 2006
 * @author Dhiogo
 *
 */
public class DBElement {

	/**
	 * URI - identificador universal do elemento
	 */
	private String uri;

	/**
	 * Representa o nome do elemento
	 */
	private String name;

	/**
	 * Representa o alias, atalho ou apelido para o elemento
```

```
 */
private String alias;

/**
 * Representa o conceito da ontologia de domínio
 */
private String domainConcept;

/**
 * Referência para um elemento pai
 */
private DBElement parentElement;


public DBElement(String uri, String name) {
      super();
      // TODO Auto-generated constructor stub
      this.uri = uri;
      this.name = name;
}

public DBElement(String uri, String name, String alias) {
      super();
      // TODO Auto-generated constructor stub
      this.uri = uri;
      this.name = name;
      this.alias = alias;
}

public DBElement(String uri, String name, String alias, String concept) {
      super();
      // TODO Auto-generated constructor stub
      this.uri = uri;
      this.name = name;
      this.alias = alias;
      domainConcept = concept;
}

public DBElement(String uri, String name, String alias, String concept,
DBElement element) {
      super();
      // TODO Auto-generated constructor stub
      this.uri = uri;
      this.name = name;
      this.alias = alias;
      domainConcept = concept;
      parentElement = element;
}

public String getName() {
      return name;
}

public void setName(String name) {
      this.name = name;
}

public String getDomainConcept() {
      return domainConcept;
```

```
	}

	public void setDomainConcept(String domainConcept) {
		this.domainConcept = domainConcept;
	}

	public String getURI(){
		return this.uri;
	}

	public void setURI(String uri){
		this.uri = uri;
	}

	/**
	 * @return the parentElement
	 */
	public DBElement getParentElement() {
		return parentElement;
	}

	/**
	 * @param parentElement the parentElement to set
	 */
	public void setParentElement(DBElement parentElement) {
		this.parentElement = parentElement;
	}

	public String getAlias() {
		return alias;
	}

	public void setAlias(String alias) {
		this.alias = alias;
	}

}
package info.stela.sbi.ontology.bi;

public class DBAttribute extends DBElement{

	private boolean isPrimaryKey;
	private String type;
	private DBAttribute referenceAttribute;
	private Object value;


	public DBAttribute(String uri, String name) {
		super(uri, name);
	}

	public DBAttribute(String uri, String name, String alias) {
		super(uri, name, alias);
	}

	public DBAttribute(String uri, String name, String alias, String concept)
	{
		super(uri, name, alias, concept);
	}
```

```
public DBAttribute(String uri, String name, String alias, String concept,
DBElement element) {
	super(uri, name, alias, concept, element);
}

public DBAttribute(String uri, String name, String alias, String concept,
DBElement element, boolean isPrimaryKey) {
	super(uri, name, alias, concept, element);
	this.setPrimaryKey(isPrimaryKey);
}

public DBAttribute(String uri, String name, String alias, String concept,
DBElement element, boolean isPrimaryKey, DBAttribute referenceAttribute)
{
	this(uri, name, alias, concept, element, isPrimaryKey);
	this.setReferenceAttribute(referenceAttribute);
}

public DBAttribute(String uri, String name, String alias, String concept,
DBElement element, boolean isPrimaryKey, DBAttribute referenceAttribute,
String type) {
	this(uri, name, alias, concept, element, isPrimaryKey,
referenceAttribute);
	this.setType(type);
}

public DBAttribute(String uri, String name, String alias, String concept,
DBElement element, boolean isPrimaryKey, DBAttribute referenceAttribute,
String type, Object value) {
	this(uri, name, alias, concept, element, isPrimaryKey,
referenceAttribute, type);
	this.setValue(value);
}



public boolean hasReference() {
	return this.getReferenceAttribute() != null;
}

public boolean isPrimaryKey() {
	return isPrimaryKey;
}
public void setPrimaryKey(boolean isPrimaryKey) {
	this.isPrimaryKey = isPrimaryKey;
}
public String getType() {
	return type;
}
public void setType(String type) {
	this.type = type;
}

/**
 * @return the referenceAttribute
 */
public DBAttribute getReferenceAttribute() {
	return referenceAttribute;
```

```
	}

	/**
	 * @param referenceAttribute the referenceAttribute to set
	 */
	public void setReferenceAttribute(DBAttribute foreignAttribute) {
		this.referenceAttribute = foreignAttribute;
	}

	public Object getValue() {
		return value;
	}
	public void setValue(Object value) {
		this.value = value;
	}


}
package info.stela.sbi.ontology.bi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class DBCollection extends DBElement {

	private String schema;
	private List<DBAttribute> dbAttributes;
	private HashMap<Integer, List<DBAttribute>> referenceKeyAttributes;
	private List<DBAttribute> primaryKeyAttributes;


	public DBCollection(String uri, String name) {
		super(uri, name);
		this.initializeDBAttributes();
	}

	public DBCollection(String uri, String name, String alias) {
		super(uri, name, alias);
		this.initializeDBAttributes();
	}

	public DBCollection(String uri, String name, String alias, String
	concept) {
		super(uri, name, alias, concept);
		this.initializeDBAttributes();
	}

	public DBCollection(String uri, String name, String alias, String
	concept, DBElement element) {
		super(uri, name, alias, concept, element);
		this.initializeDBAttributes();
	}

	public DBCollection(String uri, String name, String alias, String
	concept, DBElement element, String schema) {
		super(uri, name, alias, concept, element);
		this.setSchema(schema);
		this.initializeDBAttributes();
```

```
}

public DBCollection(String uri, String name, String alias, String
concept, DBElement element, String schema, List<DBAttribute>
dbAttributes) {
      super(uri, name, alias, concept, element);
      this.setSchema(schema);
      this.initializeDBAttributes(dbAttributes);
}

private void initializeDBAttributes(){
      this.setDBAttributes(new ArrayList<DBAttribute>());
      this.setPrimaryKeyAttributes(new ArrayList<DBAttribute>());
      this.setReferenceKeyAttributes( new HashMap<Integer,
List<DBAttribute>>() );
}

private void initializeDBAttributes(List<DBAttribute> dbAttributeList){
      this.initializeDBAttributes();
      for (DBAttribute attribute: dbAttributeList)
            this.add(attribute);
}



public List<DBAttribute> getAttributes() {
      return dbAttributes;
}



public DBAttribute getAttribute(int position){
      return this.getAttributes().get(position);
}


public void setDBAttributes(List<DBAttribute> dbAttributes) {
      this.dbAttributes = dbAttributes;
}

public String getSchema() {
      return schema;
}


public void setSchema(String schema) {
      this.schema = schema;
}

public void add(DBAttribute dbAttribute){
      dbAttribute.setParentElement(this);
      this.getAttributes().add(dbAttribute);
      if ( dbAttribute.isPrimaryKey() )
            this.getPrimaryKeyAttributes().add(dbAttribute);
      if ( dbAttribute.hasReference() )
            this.addReferenceDBAttribute(dbAttribute);
}

private void addReferenceDBAttribute(DBAttribute referenceDBAttribute){
```

```
            int hasCode =
        referenceDBAttribute.getReferenceAttribute().getParentElement().getName()
        .hashCode();
            List<DBAttribute> list =
        this.getReferenceKeyAttributes().get(hasCode);
            if ( list == null)
                  list = new ArrayList<DBAttribute>();
            list.add(referenceDBAttribute);
            this.getReferenceKeyAttributes().put(hasCode, list);
        }



        public void remove(DBAttribute dbAttribute){
            this.getAttributes().remove(dbAttribute);
            this.getPrimaryKeyAttributes().remove(dbAttribute);
            this.getReferenceKeyAttributes().remove(dbAttribute);
            dbAttribute.setParentElement(null);
        }

        public void setReferenceKeyAttributes(HashMap<Integer, List<DBAttribute>>
        referenceKeyAttributes ) {
            this.referenceKeyAttributes = referenceKeyAttributes;
        }

        public HashMap<Integer, List<DBAttribute>> getReferenceKeyAttributes() {
            return referenceKeyAttributes;
        }

        public void setPrimaryKeyAttributes(List<DBAttribute>
        primaryKeyAttributes) {
            this.primaryKeyAttributes = primaryKeyAttributes;
        }


        public List<DBAttribute> getPrimaryKeyAttributes(){
            return this.primaryKeyAttributes;
        }

}

package info.stela.sbi.manager;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

public abstract class SBIMetadata {

        public final static String APPLICATION_NAME    = "SBI";

        public final static String DOMAIN_CONCEPT_URI  =
        "http://www.stela.info#domainConcept";
        public final static String NAME_URI                 =
        "http://www.stela.info#name";
        public final static String SCHEMA_URI               =
        "http://www.stela.info#schema";
```

```
        public final static String ATTRIBUTE_URI       =
        "http://www.stela.info#hasAttribute";
        public final static String PRIMARY_KEY_URI           =
        "http://www.stela.info#isPrimaryKey";
        public final static String REFERENCE_URI       =
        "http://www.stela.info#hasReference";
        public final static String PARENT_URI                =
        "http://www.stela.info#hasParent";
        public final static String DBCOLLECTION_URI    =
        "http://www.stela.info#DBCollection";
        public final static String DBATTRIBUTE_URI           =
        "http://www.stela.info#DBAttribute";

        public static final Object EMPTY_URI                 = "rdf:nil";

        public static final String ATTRIBUTES_SEPARATOR = ",";

        public static Properties getProperties() throws FileNotFoundException,
        IOException{
              Properties properties = new Properties();
              properties.load(new FileInputStream(System.getProperty("user.dir")
        + File.separatorChar + "config" + File.separatorChar +
        "SBIConfig.properties"));
              return properties;
        }



}
package info.stela.sbi.manager;

import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIInference;
import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.repository.BIOntologyRepository;
import info.stela.sbi.repository.DomainOntologyRepository;
import info.stela.sbi.repository.SBIRepository;
import info.stela.sbi.utils.ScreenPrinter;

import java.util.Calendar;

/**
 * O Gerenciador de Instâncias (InstanceManager) é responsável pela realização de
        inferências de Ontologia de Domínio (DomainOntology) sobre uma base de
        dados.
 * As instâncias da Ontologia de Domínio são criadas através do mapeamento entre
        essa ontologia e a fonte de dados através da Ontologia de BI
        (BIOntologyInstance).
 *
 *
 * Data: 15/09/2006
 * @author Dhiogo
 *
 */
public class InstanceManager extends OntologyManager{

        private ScreenPrinter printer;
```

```
public InstanceManager(DomainOntologyRepository domainRepository,
BIOntologyRepository biRepository, SBIRepository instanceRepository,
boolean printable) throws Exception {
      super(domainRepository, biRepository, instanceRepository);
      this.setPrinter(new ScreenPrinter(printable));
}

public void doInferences(String domainURI) throws Exception{
      this.getPrinter().println("----------------------------------------
-----");
      this.getPrinter().println(SBIMetadata.APPLICATION_NAME,
Calendar.getInstance().getTime().toString() +" started");

      SBICollection biInstances;
      BIOntologyInstance biOntologyInstance;

      SBICollection<SBIOntologyInstance> domainOntologyInstances;
      SBIOntologyInstance domainOntologyInstance;

      SBICollection<SBIInference> inferences;
      SBIInference inference;

      this.getPrinter().print(SBIMetadata.APPLICATION_NAME, "Getting BI
instances of " + domainURI + "...");

      biInstances = this.getBIRepository().get(domainURI); // Monta os
aninhamentos a partir do conceito do domínio

      this.getPrinter().println("OK");

      while ( biInstances.hasNext() ){ //Caso tenha instâncias de BI para
domainURI
            biOntologyInstance = (BIOntologyInstance) biInstances.next();
//Pega a instância de BI

            this.getPrinter().print(SBIMetadata.APPLICATION_NAME,
"Getting instances of Data Sources...");
            domainOntologyInstances =
this.getInstanceRepository().get(biOntologyInstance); // Retorna o
ResultSet com os dados do BD
            this.getPrinter().println("OK");

            while ( domainOntologyInstances.hasNext() ){ // Caso tenha
dados relacionados ao domainURI no BD
                  this.getPrinter().print(SBIMetadata.APPLICATION_NAME,
"Creating instance of Domain Ontology...");
                  domainOntologyInstance =
domainOntologyInstances.next(); // retorna a instância de BI com os
valores da ontologia de domínio definidos
                  this.getPrinter().println("OK");

                  this.getPrinter().print(SBIMetadata.APPLICATION_NAME,
"Applying business rules on ontology instance...");
                  inferences  =
this.getDomainRepository().getInferences(biOntologyInstance); // Aplica e
pega as inferências definidas no repositório de Regras
                  this.getPrinter().println("OK");
```

```
                        this.getPrinter().print(SBIMetadata.APPLICATION_NAME,
        "Saving semantic inferences in Data Sources...");
                        while ( inferences.hasNext() ) {
                              inference = inferences.next();
                              this.getInstanceRepository().add(inference);
                              this.getInstanceRepository().save();
                        }
                        this.getPrinter().println("OK");
                  }
            }
            this.getInstanceRepository().save();
            this.getPrinter().println(SBIMetadata.APPLICATION_NAME,
        Calendar.getInstance().getTime().toString() + " Finished");
            this.getPrinter().println("-----------------------------------------
        -----");
        }



        public ScreenPrinter getPrinter() {
            return printer;
        }

        public void setPrinter(ScreenPrinter printer) {
            this.printer = printer;
        }



}

package info.stela.sbi.manager;

import info.stela.sbi.repository.BIOntologyRepository;
import info.stela.sbi.repository.DomainOntologyRepository;
import info.stela.sbi.repository.SBIRepository;
import info.stela.sbi.repository.SBIRepositoryException;

public abstract class OntologyManager {


        //Repositório da ontologia de domínio
        private DomainOntologyRepository domainRepository;

        //Repositório da ontologia de BI (mapeamentos entre domainOntoloy e
        ontologyRepository)
        private BIOntologyRepository biRepository;

        //Repositório de Instâncias
        private SBIRepository instanceRepository;

        /**
         * Método construtor que define as três ontologias necessárias para guiar
        o processo de inferência
         * @param domainRepository Repositório de Ontologia de domínio que
        representa o contexto e as terminologias de negócio
         * @param biRepository Ontologia de BI utilizada como metadados para
        guiar o processo de mapeamento entre a Ontologia de Domínio e a Fonte de
        Dados
```

```
 * @param inferenceRepository Repositório de dados onde as instâncias dos
conceitos da ontologia de domínio estão representados e armazenados.
 *                                          Esse repositório também são
armazenadas as conclusões sobre a ontologia de domínio
 * @throws Exception
 */
public OntologyManager(DomainOntologyRepository domainRepository ,
BIOntologyRepository biRepository, SBIRepository instanceRepository)
throws Exception{
      this.setDomainRepository(domainRepository);
      this.setBIRepository(biRepository);
      this.setInstanceRepository(instanceRepository);
}

/**
 * Método responsável por realizar as inferências semânticas sobre a
Ontologia do Domínio
 * a partir das regras de negócio definidas
 * @param domainURI Conceito do domínio inicial para a navegação sobre a
ontologia
 * @throws Exception Caso algum problema na realização das inferências
ocorrer
 */
public abstract void doInferences(String domainURI) throws Exception;

/**
 * Método que fecha todas as conexões com os repositórios
 * @throws SBIRepositoryException Caso ocorra um problema no fechamento
das conexões
 */
public void close() throws SBIRepositoryException{
      this.getDomainRepository().disconnect();
      this.getBIRepository().disconnect();
      this.getInstanceRepository().disconnect();
}



////////////////////////////////////////////////////////////////
//GET's and SET's

public BIOntologyRepository getBIRepository() {
      return biRepository;
}

public void setBIRepository(BIOntologyRepository biRepository) {
      this.biRepository = biRepository;
}

public DomainOntologyRepository getDomainRepository() {
      return domainRepository;
}

public void setDomainRepository(DomainOntologyRepository
domainRepository) {
      this.domainRepository = domainRepository;
}

public SBIRepository getInstanceRepository() {
```

```
		return instanceRepository;
	}

	public void setInstanceRepository(SBIRepository instanceRepository) {
		this.instanceRepository = instanceRepository;
	}



}
package info.stela.sbi.manager;

import info.stela.sbi.repository.SBIRepositoryException;

public class OntologyValidationException extends SBIRepositoryException {

	private static final long serialVersionUID = 1L;

	public OntologyValidationException() {
		super();
	}

	public OntologyValidationException(String message) {
		super(message);
	}

	public OntologyValidationException(String message, Throwable cause) {
		super(message, cause);
	}

	public OntologyValidationException(Throwable cause) {
		super(cause);
	}

}
package info.stela.sbi.manager;

import info.stela.sbi.repository.OntologyRepository;
import info.stela.sbi.repository.SBIRepositoryException;

public abstract class OntologyValidator {

	public OntologyValidator() {
		super();
	}

	// FIXME Validar repositorio:
	// 1) Verificar se todos as propriedades estão corretamente definidas
	// 2) Verificar se as tabelas e atributos do Banco de Dados são válidos
	// 3) Verificar se há referências cíclicas
	public abstract void validateRepository(OntologyRepository
	ontologyRepository) throws SBIRepositoryException;



}

package info.stela.sbi.utils;
```

```
public class ScreenPrinter {

	private InnerScreenPrinter screenPrinter;

	public ScreenPrinter(){
		this.setScreenPrinter(new InnerEnableScreenPrinter());
	}

	public ScreenPrinter(boolean enablePrint){
		this.setEnable(enablePrint);
	}

	public void setEnable(boolean enablePrint){
		if ( enablePrint )
			this.setScreenPrinter(new InnerEnableScreenPrinter());
		else
			this.setScreenPrinter(new InnerDisableScreenPrinter());
	}

	public void println(String msg){
		this.getScreenPrinter().println(msg);
	}

	public void print(String identifier, String msg) {
		this.getScreenPrinter().print(identifier, msg);
	}

	public void println(String identifier, String msg) {
		this.getScreenPrinter().println(identifier, msg);
	}

	public void print(String msg){
		this.getScreenPrinter().print(msg);
	}


	public static void println(boolean enable , String msg){
		if ( enable )
			System.out.println(msg);
	}

	public static void print(boolean enable , String msg){
		if ( enable )
			System.out.print(msg);
	}

	public static void println(boolean enable , String identifier, String
	msg){
		if ( enable )
			System.out.println("[" + identifier + "] - " + msg);
	}

	public static void print(boolean enable , String identifier, String msg){
		if ( enable )
			System.out.print("[" + identifier + "] - " + msg);
	}

	//////////////////////////////////////////////////
```

```
//INNER CLASS
abstract class InnerScreenPrinter{

	public abstract void println(String msg);

	public abstract void print(String msg);

	public abstract void println(String identifier, String msg);

	public abstract void print(String identifier, String msg);

}


class InnerEnableScreenPrinter extends InnerScreenPrinter {


	public InnerEnableScreenPrinter(){
	}

	public void println(String msg){
		System.out.println(msg);
	}

	public void print(String msg){
		System.out.print(msg);
	}

	public void println(String identifier, String msg){
		System.out.println("[" + identifier + "] - " + msg);
	}

	public void print(String identifier, String msg){
		System.out.print("[" + identifier + "] - " + msg);
	}

}

class InnerDisableScreenPrinter extends InnerScreenPrinter {


	public InnerDisableScreenPrinter(){
	}

	public void println(String msg){
	}

	public void print(String msg){
	}

	public void printApplication(String msg) {
	}

	public void print(String identifier, String msg) {

	}
```

```
            public void println(String identifier, String msg) {

            }

        }

        public InnerScreenPrinter getScreenPrinter() {
            return screenPrinter;
        }

        public void setScreenPrinter(InnerScreenPrinter screenPrinter) {
            this.screenPrinter = screenPrinter;
        }



}
/**
 *
 */
package info.stela.sbi.utils;

import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;


/**
 * Data: 12/09/2006
 * @author Dhiogo
 *
 */
public class DBQueryWriter {

        private StringBuffer query;
        private ArrayList<String> columns;
        private ArrayList<String> tables;
        private ArrayList<String> filters;


        private String dbCollectionAlias;
        private int actualDBCollectionAlias;

        private String dbAttributeAlias;
        private int actualDBAttributeAlias;

        public static final int      INNER_JOIN_TYPE        = 1;
        public static final String   INNER_JOIN_TEXT        = " INNER JOIN ";

        public static final int      RIGHT_OUTER_JOIN_TYPE  = 2;
        public static final String   RIGHT_OUTER_JOIN_TEXT  = " RIGHT OUTER JOIN
        ";

        public static final int      LEFT_OUTER_JOIN_TYPE   = 3;
        public static final String   LEFT_OUTER_JOIN_TEXT   = " LEFT OUTER JOIN
        ";
```

```
public static final int      FULL_OUTER_JOIN_TYPE    = 4;
public static final String   FULL_OUTER_JOIN_TEXT    = " FULL OUTER JOIN
";

public static final int      NATURAL_JOIN_TYPE            = 5;
public static final String   NATURAL_JOIN_TEXT            = " NATURAL
JOIN ";


public DBQueryWriter(){
      this.setActualDBCollectionAlias(0);
      this.setActualDBAttributeAlias(0);
      this.setDBCollectionAlias("C");
      this.setDBAttributeAlias("A");
      this.setColumns(new ArrayList<String>());
      this.setFilters(new ArrayList<String>());
      this.setTables(new ArrayList<String>());
      this.setQuery(new StringBuffer());
}
/**
 * Adiciona o nome da coluna na lista a ser retornada na consulta
 * @param column Nome da coluna adicinada ao final da lista de colunas do
SELECT
 */
public void addColumnResult(String column){
      this.getColumns().add(column);
}
public void addTable(String table){
      this.getTables().add(table);
}



public static void main(String[] args) {

      String[] selectedColumns = new String[] { "CH.*" };
      String tables1[] = new String[] {"EN_PESSOA PE" ,
"RE_PESSOA_VINCULO_IES PV" , "EN_DOCENTE_AFASTAMENTO DA" ,
"EN_CARGA_HORARIA_IES CH"};
      String[][] filters1 = new String[][]{ { "PE.NRO_ID_PESSOA" },
                  { "PV.NRO_ID_PESSOA"  , "PV.SEQ_VINCULO_IES"},
                  { "DA.NRO_ID_PESSOA" , "DA.SEQ_VINCULO_IES"  },
                  { "CH.NRO_ID_PESSOA" , "CH.SEQ_VINCULO_IES"  }
      };

      String tables2[] = new String[] {"EN_PESSOA PE" ,
"RE_PESSOA_VINCULO_IES PV" , "EN_DOCENTE_AFASTAMENTO DA"};
      String[][] filters2 = new String[][]{ { "PE.NRO_ID_PESSOA" },
                  { "PV.NRO_ID_PESSOA"  , "PV.SEQ_VINCULO_IES"},
                  { "DA.NRO_ID_PESSOA" , "DA.SEQ_VINCULO_IES"  }
      };


}

public static String getSQLInnerJoin(String[] selectedColumns, String[]
tableNames, String[][]filters){
```

```
 return DBQueryWriter.getSQLJoin(selectedColumns, tableNames, filters,
DBQueryWriter.INNER_JOIN_TEXT);
 }

 public static String getSQLLeftOuterJoin(String[] selectedColumns,
String[] tableNames, String[][]filters){
 return DBQueryWriter.getSQLJoin(selectedColumns, tableNames, filters,
DBQueryWriter.LEFT_OUTER_JOIN_TEXT);
 }

 public static String getSQLRightOuterJoin(String[] selectedColumns,
String[] tableNames, String[][]filters){
 return DBQueryWriter.getSQLJoin(selectedColumns, tableNames, filters,
DBQueryWriter.RIGHT_OUTER_JOIN_TEXT);
 }

 public static String getSQLFullOuterJoin(String[] selectedColumns,
String[] tableNames, String[][]filters){
 return DBQueryWriter.getSQLJoin(selectedColumns, tableNames, filters,
DBQueryWriter.FULL_OUTER_JOIN_TEXT);
 }

public static String getSQLJoin(String[] selectedColumns, String[]
tableNames, String[][]filters, String joinType ){
      StringBuffer join = new StringBuffer(
DBQueryWriter.getSelectColumns(selectedColumns) );
      boolean hasJoin = false;
      int i , x ;
      for ( i = 0 ; i < tableNames.length ; i++){
            join.append( tableNames[i] );
            if ( hasJoin ){
                  join.append("\n ON ");
                  for ( x = 0 ; x < filters[i - 1].length ; x++){
                        join.append(filters[i - 1][x] + " = " +
filters[i][x] );
                        if ( x < filters[i - 1].length  - 1)
                              join.append(" AND \n ");
                  }
                  hasJoin = false;
            }
            if ( i < tableNames.length - 1){
                  join.append("\n " + joinType + " ");
                  hasJoin = true;
            }
      }
      return join.toString();
}

private static String getSelectColumns(String[] selectedColumns){
      StringBuffer columns = new StringBuffer("SELECT ");
      if ( selectedColumns == null  || selectedColumns.length == 0)
            columns.append("* \nFROM ");
      else{
            columns.append(selectedColumns[0] == null ? "* " :
selectedColumns[0]);
            for ( int i = 1; i < selectedColumns.length; i++){
                  columns.append(", " +selectedColumns[i]);
            }
            columns.append(" \nFROM ");
```

```
	}
	return columns.toString();

}

public String addSQLJoin(StringBuffer sqlJoin, DBCollection dbCollection
, String joinTypeTex){
	Iterator<List<DBAttribute>> iterator =
dbCollection.getReferenceKeyAttributes().values().iterator();
	List<DBAttribute> referenceAttributes;
	String alias;
	DBAttribute attribute;
	while ( iterator.hasNext() ){

		referenceAttributes = iterator.next();
		attribute = referenceAttributes.get(0);
		alias = this.newDBCollectionAlias();
		sqlJoin.append(joinTypeTex +
attribute.getReferenceAttribute().getParentElement().getName() + " "+
alias + " ON ");
		for ( int x = 0 ; x < referenceAttributes.size(); x++){
			attribute = referenceAttributes.get(x);

	attribute.getReferenceAttribute().getParentElement().setAlias(alias
);

			sqlJoin.append( dbCollection.getAlias() + "." +
attribute.getName() + " = " +
attribute.getReferenceAttribute().getParentElement().getAlias()+"."+
attribute.getReferenceAttribute().getName());
			if ( x + 1 < referenceAttributes.size() )
				sqlJoin.append(" AND ");
		}
		addSQLJoin( sqlJoin, (DBCollection)
attribute.getReferenceAttribute().getParentElement() , joinTypeTex);

	}
	return sqlJoin.toString();
}

public String getSQLJoin(DBCollection dbCollection, int joinType){
	StringBuffer sqlMain = new StringBuffer("SELECT ");
	dbCollection.setAlias( this.newDBCollectionAlias() );

	String sqlJoin = this.addSQLJoin( new StringBuffer(" "
+dbCollection.getName() + " " + dbCollection.getAlias() ), dbCollection,
DBQueryWriter.getJoinTypeText(joinType));


	this.addQueryColumns(sqlMain, dbCollection);
	sqlMain.append(" FROM ");
	sqlMain.append(sqlJoin);

	return sqlMain.toString();
}

private void addQueryColumns(StringBuffer sql, DBCollection dbCollection)
{
```

```
      this.addQueryColumns(sql, dbCollection.getAttributes());

      Iterator<List<DBAttribute>> iterator =
dbCollection.getReferenceKeyAttributes().values().iterator();
      List<DBAttribute> referenceAttributes;
      DBAttribute attribute;
      while ( iterator.hasNext() ){
            referenceAttributes = iterator.next();
            sql.append(", ");
            addQueryColumns(sql, (DBCollection)
referenceAttributes.get(0).getReferenceAttribute().getParentElement());

      }


}

private void addQueryColumns(StringBuffer sql, List<DBAttribute>
attributes) {

      for ( int x = 0 ; x < attributes.size(); x++){
            attributes.get(x).setAlias( this.newDBAttributeAlias() );
            sql.append(attributes.get(x).getParentElement().getAlias()
+"."+attributes.get(x).getName()+" "+ attributes.get(x).getAlias());
            if ( x + 1 < attributes.size() )
                  sql.append(", ");


      }
}




private static String getJoinTypeText(int joinType){
      String join;
      switch( joinType){
      case DBQueryWriter.INNER_JOIN_TYPE:
            join = DBQueryWriter.INNER_JOIN_TEXT;
            break;
      case DBQueryWriter.LEFT_OUTER_JOIN_TYPE:
            join = DBQueryWriter.LEFT_OUTER_JOIN_TEXT;
            break;
      case DBQueryWriter.RIGHT_OUTER_JOIN_TYPE:
            join = DBQueryWriter.RIGHT_OUTER_JOIN_TEXT;
            break;
      case DBQueryWriter.FULL_OUTER_JOIN_TYPE:
            join = DBQueryWriter.FULL_OUTER_JOIN_TEXT;
            break;
      case DBQueryWriter.NATURAL_JOIN_TYPE:
            join = DBQueryWriter.NATURAL_JOIN_TEXT;
            break;
      default: join = DBQueryWriter.INNER_JOIN_TEXT;

      }
      return join;
}
```

```
public String newDBCollectionAlias(){
      this.setActualDBCollectionAlias(this.actualDBCollectionAlias+1);
      return
this.getDBCollectionAlias()+this.getActualDBCollectionAlias();
}

public String newDBAttributeAlias(){
      this.setActualDBAttributeAlias(this.actualDBAttributeAlias+1);
      return this.getDBAttributeAlias()+this.getActualDBAttributeAlias();
}

private int getActualDBCollectionAlias() {
      return actualDBCollectionAlias;
}

private void setActualDBCollectionAlias(int actualAliasNumber) {
      this.actualDBCollectionAlias = actualAliasNumber;
}

private String getDBCollectionAlias() {
      return dbCollectionAlias;
}

private void setDBCollectionAlias(String dbCollectionAlias) {
      this.dbCollectionAlias = dbCollectionAlias;
}
private int getActualDBAttributeAlias() {
      return actualDBAttributeAlias;
}
private void setActualDBAttributeAlias(int actualDBAttributeAlias) {
      this.actualDBAttributeAlias = actualDBAttributeAlias;
}
private String getDBAttributeAlias() {
      return dbAttributeAlias;
}
private void setDBAttributeAlias(String dbAttributeAlias) {
      this.dbAttributeAlias = dbAttributeAlias;
}
private ArrayList<String> getColumns() {
      return columns;
}
private void setColumns(ArrayList<String> columns) {
      this.columns = columns;
}
private ArrayList<String> getFilters() {
      return filters;
}
private void setFilters(ArrayList<String> filters) {
      this.filters = filters;
}
private StringBuffer getQuery() {
      return query;
}
private void setQuery(StringBuffer query) {
      this.query = query;
}
private ArrayList<String> getTables() {
      return tables;
```

```
	}
	private void setTables(ArrayList<String> tables) {
		this.tables = tables;
	}


}

package info.stela.sbi.jena;

import info.stela.sbi.jena.repository.factory.DWInstanceRepositoryFactory;
import info.stela.sbi.jena.repository.factory.OWLBIOntologyRepositoryFactory;
import info.stela.sbi.jena.repository.factory.OWLDomainOntologyRepositoryFactory;
import info.stela.sbi.manager.InstanceManager;
import info.stela.sbi.manager.OntologyManager;
import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.repository.BIOntologyRepository;
import info.stela.sbi.repository.DomainOntologyRepository;
import info.stela.sbi.repository.SBIRepository;
import info.stela.sbi.repository.SBIRepositoryException;

import java.util.Properties;

public class MainSBI implements Runnable {


	public static void main(String[] args){
		MainSBI main = new MainSBI();
		main.run();
	}

	public void run(){

		OntologyManager manager = null;
		try {

			//Entrada de Parâmetros
			Properties properties		= SBIMetadata.getProperties();

			//URLs dos Repositórios de Ontologias
			String domainOntologyURL	= MainSBI.getString(properties,
	"sbi.repository.domain");
			String biOntologyURL		= MainSBI.getString(properties,
	"sbi.repository.bi");
			String domainRuleURL		= MainSBI.getString(properties,
	"sbi.repository.rules");

			//Configuração do Data Warehouse
			String dbDriver			= MainSBI.getString(properties,
	"sbi.db.jdbcDriver");
			String dbURL				=
	MainSBI.getString(properties, "sbi.db.url");
			String dbLogin				=
	MainSBI.getString(properties, "sbi.db.user");
			String dbPass				=
	MainSBI.getString(properties, "sbi.db.pass");

			//Habilita ou desabilita as impressões na tela
```

```
            boolean printable                   =
MainSBI.getBoolean(properties, "sbi.print.enable");

            //Conceito inicial para navegação sobre a ontologia
            String personURI              = MainSBI.getString(properties,
"sbi.ontology.start");

            //Definições do Fato Tripla
            String tripleFactName         = MainSBI.getString(properties,
"sbi.db.tripleFact.name");
            String subject                      =
MainSBI.getString(properties, "sbi.db.tripleFact.subject");
            String predicate              = MainSBI.getString(properties,
"sbi.db.tripleFact.predicate");
            String object                       =
MainSBI.getString(properties, "sbi.db.tripleFact.object");

            //Criação dos repositórios
            DomainOntologyRepository domainRepository =
OWLDomainOntologyRepositoryFactory.getInstance().createRepository(domainO
ntologyURL, domainRuleURL);
            BIOntologyRepository biRepository =
OWLBIOntologyRepositoryFactory.getInstance().createRepository(biOntologyU
RL);
            SBIRepository instanceRepository =
DWInstanceRepositoryFactory.getInstance().createRepository(dbDriver,
dbURL, dbLogin, dbPass, tripleFactName, subject,predicate ,object);

            //Execução
            manager = new InstanceManager(domainRepository, biRepository,
instanceRepository, printable);
            manager.doInferences(personURI);


      } catch (Exception e) {
            e.printStackTrace();
      } finally{
            try {
                  manager.close();
            } catch (SBIRepositoryException e) {
                  e.printStackTrace();
            }
      }
}




public static String getString(Properties properties, String key) throws
Exception{
      String value = properties.getProperty(key);
      if ( value == null)
            throw new Exception("Parameter \""+ key +"\" not defined in
SBIConfig.properties");
      return value.trim();
}
```

```
	public static boolean getBoolean(Properties properties, String key)
	throws Exception{
		return "true".equalsIgnoreCase(MainSBI.getString(properties, key));
	}


}

package info.stela.sbi.jena.repository;

import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
import info.stela.sbi.manager.OntologyValidationException;
import info.stela.sbi.manager.OntologyValidator;
import info.stela.sbi.repository.OntologyRepository;
import info.stela.sbi.repository.SBIRepositoryException;

public class OWLDomainOntologyRepositoryValidator extends OntologyValidator{

	public OWLDomainOntologyRepositoryValidator() {
		super();
	}


	public void validateRepository(OntologyRepository ontologyRepository)
	throws SBIRepositoryException {
		if ( ontologyRepository.isConnected() == false )
			ontologyRepository.connect();
		this.valid((OWLFileOntologyRepositoryImpl)
	ontologyRepository.getImplementator());
	}


	private void valid(OWLFileOntologyRepositoryImpl impl) throws
	SBIRepositoryException {
		if ( impl.isEmpty() )
			throw new OntologyValidationException("Domain Ontology
	Repository is invalid: The Ontology Model is Empty");

	}



}
package info.stela.sbi.jena.repository;

import info.stela.sbi.repository.InstanceRepository;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public class DWInstanceRepository extends InstanceRepository {

	public DWInstanceRepository(OntologyRepositoryImpl implementator) {
		super(implementator);
	}

}
package info.stela.sbi.jena.repository;

import info.stela.sbi.jena.ontology.OWLBIOntology;
import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
```

```
import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.ontology.collection.SBIIterator;
import info.stela.sbi.repository.BIOntologyRepository;
import info.stela.sbi.repository.SBIRepositoryException;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

import java.util.ArrayList;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.core.ResultBinding;
import com.hp.hpl.jena.rdf.model.Resource;

public class OWLBIOntologyRepository extends BIOntologyRepository{

	public OWLBIOntologyRepository(OntologyRepositoryImpl implementator) {
		super(implementator);
	}

	public OWLFileOntologyRepositoryImpl getImplementator(){
		return (OWLFileOntologyRepositoryImpl) super.getImplementator();
	}

	public SBICollection<BIOntologyInstance> getInstancesByDomain(String
	domainURI) throws SBIRepositoryException {
		return this.createIterator(domainURI);
	}

	public SBIIterator<SBIOntologyInstance> get(String uri) throws
	SBIRepositoryException {
		return this.createIterator(uri);
	}

	public SBIIterator<SBIOntologyInstance> createIterator(String domainURI)
	throws SBIRepositoryException {
		ArrayList<SBIOntologyInstance> list = new
	ArrayList<SBIOntologyInstance>(1);
		list.add(new OWLBIOntology( domainURI,  this.getResouce(domainURI)
	) );
		return new SBIIterator<SBIOntologyInstance>(list.iterator());
	}

	private Resource getResouce(String uri) throws SBIRepositoryException{
		Resource resource = null;
		Query query = null;
		QueryExecution queryExecution = null;
		try {
			String querySPARQL = "SELECT ?domainConcept WHERE {
	?domainConcept <" + SBIMetadata.DOMAIN_CONCEPT_URI + "> \"" + uri + "\"
	}";
			query = QueryFactory.create(querySPARQL);
			queryExecution = QueryExecutionFactory.create(query,
	this.getImplementator().getOntologyModel());
```

```
                  ResultSet result = queryExecution.execSelect();
                  ResultBinding resourceBinding;
                  if ( result.hasNext() ) {
                        resourceBinding = (ResultBinding) result.next();
                        resource =
            resourceBinding.getResource("domainConcept");
                  } else
                        throw new SBIRepositoryException("Resource not found: "
            + uri);

            } catch (Exception e) {
                  throw new SBIRepositoryException(e);
            } finally {
                  if ( queryExecution != null)
                        queryExecution.close();
            }
            return resource;
      }



}
package info.stela.sbi.jena.repository;

import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
import info.stela.sbi.manager.OntologyValidationException;
import info.stela.sbi.manager.OntologyValidator;
import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.repository.OntologyRepository;
import info.stela.sbi.repository.SBIRepositoryException;

import java.util.Iterator;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;

public class OWLBIOntologyRepositoryValidator extends OntologyValidator{


      public OWLBIOntologyRepositoryValidator() {
            super();
      }


      public void validateRepository(OntologyRepository ontologyRepository)
      throws SBIRepositoryException {
            if ( ontologyRepository.isConnected() == false )
                  ontologyRepository.connect();
            this.valid((OWLFileOntologyRepositoryImpl)
      ontologyRepository.getImplementator());
      }

      private void valid(OWLFileOntologyRepositoryImpl impl) throws
      OntologyValidationException{
            Resource dbCollectionResource =
      impl.getOntologyModel().getResource(SBIMetadata.DBCOLLECTION_URI);
            if ( dbCollectionResource == null ||
      dbCollectionResource.isResource() ==  false )
```

```
            throw new OntologyValidationException("BI Ontology Repository
      is invalid: DBCollection class not found");

      Iterator<Individual> iterator =
      impl.getOntologyModel().listIndividuals( dbCollectionResource );
      Individual individual;
      while ( iterator.hasNext() ){
            individual = iterator.next();
            if ( individual.getURI() == null )
                  throw new OntologyValidationException("BI Ontology
      Repository is invalid: DBCollection's URI is null");
            if ( this.hasProperty(individual, SBIMetadata.NAME_URI) ==
      false )
                  throw new OntologyValidationException("BI Ontology
      Repository is invalid: DBCollection " + individual.getURI() + " does not
      have name");
            this.validAttributes(individual);
      }

      }

      private boolean hasProperty(Resource resource, String propertyName){
      return resource.getProperty(
      resource.getModel().getProperty(propertyName)) != null;
      }


      private void validAttributes(Resource individual) throws
      OntologyValidationException{
      Iterator<Statement> statements = individual.listProperties(
      individual.getModel().getProperty(SBIMetadata.ATTRIBUTE_URI));
      Statement statement = null;
      Resource attribute;

      if ( statements.hasNext() ){
            while ( statements.hasNext() ){
                  statement = statements.next();
                  attribute = (Resource)statement.getObject();
                  if ( attribute.getURI() == null ){
                        throw new OntologyValidationException("BI Ontology
      Repository is invalid: DBAttribute's URI is null");
                  }
                  if ( this.hasProperty(attribute, SBIMetadata.NAME_URI)
      == false )
                        throw new OntologyValidationException("BI Ontology
      Repository is invalid: DBAttribute " + attribute.getURI() + " does not
      have name");
            }
      }else
            throw new OntologyValidationException("BI Ontology Repository
      is invalid: DBCollection " + individual.getURI() + " does not have
      attributes");

      }



}
package info.stela.sbi.jena.repository;
```

```
import info.stela.sbi.jena.ontology.OWLInference;
import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIInference;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.ontology.collection.SBIIterator;
import info.stela.sbi.repository.DomainOntologyRepository;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import com.hp.hpl.jena.reasoner.rulesys.RuleDerivation;

public class OWLDomainOntologyRepository extends DomainOntologyRepository {

	public OWLDomainOntologyRepository(OntologyRepositoryImpl implementator)
	{
		super(implementator);
	}

	public OWLFileOntologyRepositoryImpl getImplementator(){
		return (OWLFileOntologyRepositoryImpl) super.getImplementator();
	}

	private void clearModels(){
		this.getImplementator().getInferenceModel().removeAll();
		this.getImplementator().getOntologyModel().removeAll();
	}

	public Iterator<SBIInference> getInferencesList(DBCollection collection){
		ArrayList<SBIInference> inferences = new ArrayList<SBIInference>();
		this.doInferences(collection, inferences);
		Iterator<List<DBAttribute>> attributes =
	collection.getReferenceKeyAttributes().values().iterator();
		List<DBAttribute> list;
		while( attributes.hasNext() ){
			list = attributes.next();
			this.getInferencesList((DBCollection)
	list.get(0).getReferenceAttribute().getParentElement());
		}
		return inferences.iterator();
	}

	public void doInferences(DBCollection collection, ArrayList<SBIInference>
	inferences){
		String uri = collection.getURI();
		SimpleSelector simpleSelector = new
	SimpleSelector(this.getImplementator().getInferenceModel().getResource(ur
	i), null,(String) null);
```

```
		StmtIterator statements;
		Statement statement;
		Iterator derivations;
		RuleDerivation ruleDerivation;
		OWLInference inference;
		for (statements=
	this.getImplementator().getInferenceModel().listStatements(simpleSelector
	); statements.hasNext(); ) {
				statement = statements.nextStatement();
				for (derivations =
	this.getImplementator().getInferenceModel().getDerivation(statement);
	derivations.hasNext(); ) {
					ruleDerivation = (RuleDerivation) derivations.next();
					inference = new OWLInference(ruleDerivation,
	collection);
					inferences.add(inference);
					derivations.remove();
				}
		}
	}

	public SBICollection<SBIInference> getInferences(BIOntologyInstance
	instance) throws Exception {
		this.clearModels();
		this.add(instance);

		return new SBIIterator<SBIInference>(
	this.getInferencesList(instance.getDBCollection()) );
	}


	public Set<String> getAllDomainConcepts() {
		return this.getImplementator().getDomainConcepts();
	}


}

package info.stela.sbi.jena.repository.impl;

import info.stela.backend.general.file.FileHandler;
import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;
import info.stela.sbi.ontology.collection.SBICollection;
import info.stela.sbi.ontology.collection.SBIIterator;
import info.stela.sbi.repository.SBIRepositoryException;
import info.stela.sbi.repository.impl.FileOntologyRepositoryImpl;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.StringReader;
import java.util.HashSet;
```

```
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.InfModel;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.reasoner.rulesys.GenericRuleReasoner;
import com.hp.hpl.jena.reasoner.rulesys.Rule;
import com.hp.hpl.jena.reasoner.rulesys.Rule.ParserException;

public class OWLFileOntologyRepositoryImpl extends FileOntologyRepositoryImpl{

         private boolean isConnected;
         private OntModel ontologyModel;
         private InfModel inferenceModel;
         private String ruleFileName;
         private Set domainConcepts;

         public static final String DOMAIN_FILE_EXTENSION = "OWL";
         public static final String RULE_FILE_EXTENSION = "RULES";

         public OWLFileOntologyRepositoryImpl(String fileName, String
         ruleFileName,  OntModel model) throws FileNotFoundException {
               super(fileName);
               this.setRuleFileName(ruleFileName);
               this.setConnected(false);
               this.setOntologyModel( model );
               this.setDomainConcepts(new HashSet<String>());
         }

         public OWLFileOntologyRepositoryImpl(String fileName, String
         ruleFileName) throws FileNotFoundException {
               super(fileName);
               this.setRuleFileName(ruleFileName);
               this.setConnected(false);
               this.setDomainConcepts(new HashSet<String>());
               this.setOntologyModel( ModelFactory.createOntologyModel(
         OntModelSpec.OWL_MEM_MICRO_RULE_INF ));
         }

         protected void connect(File[] files) throws FileNotFoundException{
               for (File file: files)
                     this.connect(file);
         }

         protected void connect(File file) throws FileNotFoundException{
               this.getOntologyModel().read( new BufferedInputStream(new
         FileInputStream(file.getAbsolutePath())) , "" );
         }

         public void connect() throws SBIRepositoryException {
               try {
                     File file = new File(this.getFileName());
                     if ( file.isDirectory() ){
```

```
                File[] files =
file.listFiles(FileHandler.getFileFilter(OWLFileOntologyRepositoryImpl.DO
MAIN_FILE_EXTENSION));
                this.connect(files);
            }else
                this.connect(file);

            this.createInferenceModel();
            this.setConnected(true);
      } catch (FileNotFoundException e) {
            throw new SBIRepositoryException(e);
      } catch (ParserException e) {
            throw new SBIRepositoryException(e);
      } catch (IOException e) {
            throw new SBIRepositoryException(e);
      }
}

public void disconnect() throws SBIRepositoryException {
      this.getDomainConcepts().clear();
      this.getOntologyModel().close();
      if ( this.getInferenceModel() != null )
            this.getInferenceModel().close();
      this.setConnected(false);
}

public boolean isConnected() {
      return this.isConnected;
}

public OntModel getOntologyModel() {
      return ontologyModel;
}

private void createInferenceModel() throws ParserException, IOException,
SBIRepositoryException{
      if ( this.getRuleFileName() != null ){
            GenericRuleReasoner reasoner = new
GenericRuleReasoner(Rule.parseRules(Rule.rulesParserFromReader(this.getRe
ader())));
            reasoner.setMode(GenericRuleReasoner.HYBRID);
            reasoner.setTransitiveClosureCaching(false);
            reasoner.setOWLTranslation(false);
            this.setInferenceModel( ModelFactory.createInfModel(reasoner,
this.getOntologyModel() ));
      }
}

private BufferedReader getReader() throws IOException {
      StringBuffer stringBuffer = new StringBuffer();
      File ruleFile = new File ( this.getRuleFileName() );
      if ( ruleFile.isDirectory() ){
            File[] files =
ruleFile.listFiles(FileHandler.getFileFilter(OWLFileOntologyRepositoryImp
l.RULE_FILE_EXTENSION));
            for (File file : files ){
                  stringBuffer.append(FileHandler.read(file));
            }
      }else
```

```
	stringBuffer.append(FileHandler.read(this.getRuleFileName()));

	return new BufferedReader(new
StringReader(stringBuffer.toString()));
}

public void setOntologyModel(OntModel ontologyModel) {
	this.ontologyModel = ontologyModel;
	this.ontologyModel.setDerivationLogging(false);
}
private void setConnected(boolean isConnected) {
	this.isConnected = isConnected;
}


public Resource add(DBCollection collection){
	Individual domainResource =
this.getOntologyModel().createIndividual(collection.getURI(),
this.getOntologyModel().getResource(collection.getDomainConcept()));
	this.getDomainConcepts().add(collection.getURI());
	Iterator<DBAttribute> attributes =
collection.getAttributes().iterator();
	DBAttribute attribute;
	Property property;
	while ( attributes.hasNext()  ){
		attribute = attributes.next();

		if ( attribute.hasReference() ){
			if ( attribute.getValue() != null ){
				property =
this.getOntologyModel().createProperty(attribute.getDomainConcept());
				this.getOntologyModel().add(domainResource,
property, this.add( (DBCollection)
attribute.getReferenceAttribute().getParentElement() ));
			}

		}else{
			if ( attribute.getDomainConcept() != null ){
				property =
this.getOntologyModel().createProperty(attribute.getDomainConcept());
				if ( attribute.getValue() != null)
					this.getOntologyModel().add(domainResource,
property, attribute.getValue());
				else
					this.getOntologyModel().add(domainResource,
property , SBIMetadata.EMPTY_URI);
			}
		}
	}

	return domainResource;
}




public boolean add(SBIOntologyInstance instance) {
```

```
		BIOntologyInstance biInstance = (BIOntologyInstance) instance;
		this.add(biInstance.getDBCollection());
		return true;
	}

	public SBICollection<SBIOntologyInstance> get() throws
	SBIRepositoryException {
		return new SBIIterator(this.getOntologyModel().listIndividuals());
	}

	public SBICollection<SBIOntologyInstance> get(String uri) throws
	SBIRepositoryException {
		return new
	SBIIterator(this.getOntologyModel().listIndividuals(this.getOntologyModel
	().getResource(uri)));
	}

	public SBICollection<SBIOntologyInstance> get(SBIOntologyInstance
	instance) throws SBIRepositoryException {
		return new
	SBIIterator(this.getOntologyModel().listIndividuals(this.getOntologyModel
	().getResource(instance.getURI())));
	}

	public boolean remove(SBIOntologyInstance instance) throws
	SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}

	public boolean remove(String uri) throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}

	public boolean removeAll() throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}

	public boolean set(String uri, SBIOntologyInstance instance) throws
	SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}

	public boolean set(SBIOntologyInstance instance, SBIOntologyInstance
	newInstance) throws SBIRepositoryException {
		// TODO Auto-generated method stub
		return false;
	}

	/**
	 * @return the inferenceModel
	 */
	public InfModel getInferenceModel() {
		return inferenceModel;
	}
```

```
	/**
	 * @param inferenceModel the inferenceModel to set
	 */
	public void setInferenceModel(InfModel inferenceModel) {
		this.inferenceModel = inferenceModel;
		this.inferenceModel.setDerivationLogging(true);
	}

	public String getRuleFileName() {
		return ruleFileName;
	}

	public void setRuleFileName(String ruleFileName) {
		this.ruleFileName = ruleFileName;
	}

	public void cancel() throws SBIRepositoryException {

	}

	public void save() throws SBIRepositoryException {

	}

	public boolean isEmpty() throws SBIRepositoryException {
		return this.getOntologyModel().isEmpty();
	}

	public Set<String> getDomainConcepts() {
		return domainConcepts;
	}

	public void setDomainConcepts(Set<String> domainConcepts) {
		this.domainConcepts = domainConcepts;
	}

	public List<String> getAllConclusionsNames(){
		return null;
	}




}
package info.stela.sbi.jena.repository.impl;

import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.SBIInference;
import info.stela.sbi.ontology.SBIOntologyInstance;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.repository.SBIRepositoryException;
import info.stela.sbi.repository.impl.DBOntologyRepositoryImpl;
import info.stela.sbi.utils.DBQueryWriter;
import info.stela.writer.SQLWriter;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
```

```
public class DWOntologyRepositoryImpl extends DBOntologyRepositoryImpl {

	public static String SQL_INSERT_FATO_STATEMENT;
	public static String SQL_DELETE_FATO_STATEMENT;

	private DBQueryWriter joinWriter;

	private String[] sujects;
	private String[] predicates;
	private String[] objects;

	public DWOntologyRepositoryImpl(Connection connection, String
	tripleFactName, String sujectAttributes, String predicateAttributes,
	String objectAttributes) throws SQLException {
		super(connection);
		this.setJoinWriter(new DBQueryWriter());
		this.initDBStatements(tripleFactName, sujectAttributes,
	predicateAttributes, objectAttributes);
	}

	public DWOntologyRepositoryImpl(String driverJDBC, String url, String
	username, String password,String tripleFactName, String sujectAttributes,
	String predicateAttributes, String objectAttributes) {
		super(driverJDBC, url, username, password);
		this.setJoinWriter(new DBQueryWriter());
		this.initDBStatements(tripleFactName, sujectAttributes,
	predicateAttributes, objectAttributes);
	}

	private void initDBStatements(String tripleFactName, String
	sujectAttributes, String predicateAttributes, String objectAttributes){
		this.setSujects(
	sujectAttributes.split(SBIMetadata.ATTRIBUTES_SEPARATOR));
		this.setPredicates(
	predicateAttributes.split(SBIMetadata.ATTRIBUTES_SEPARATOR));

		this.setObjects(objectAttributes.split(SBIMetadata.ATTRIBUTES_SEPAR
	ATOR));
		SQL_DELETE_FATO_STATEMENT = this.getSQLDelete(tripleFactName,
	sujectAttributes, predicateAttributes, objectAttributes);
		SQL_INSERT_FATO_STATEMENT = this.getSQLInsert(tripleFactName,
	sujectAttributes, predicateAttributes, objectAttributes);
	}

	private String getSQLDelete(String tripleFactName, String
	sujectAttributes, String predicateAttributes, String objectAttributes){
		StringBuffer stringBuffer = new StringBuffer("DELETE FROM " +
	tripleFactName + " WHERE ");

		for ( int x = 0; x < sujects.length; x++){
			stringBuffer.append(sujects[x] + " = :" + sujects[x] );
			stringBuffer.append( " AND ");
		}
		for ( int x = 0; x < predicates.length; x++){
			stringBuffer.append(predicates[x] + " = :" + predicates[x] );
			stringBuffer.append( " AND ");
		}
		for ( int x = 0; x < objects.length; x++){
```

```
            stringBuffer.append(" " + objects[x] + " = :" + objects[x] );
            if ( x + 1 < objects.length )
                  stringBuffer.append( " AND ");
      }

      return stringBuffer.toString();
}

private String getSQLInsert(String tripleFactName, String
sujectAttributes, String predicateAttributes, String objectAttributes) {
      StringBuffer stringBuffer = new StringBuffer("INSERT INTO " +
tripleFactName + " ( ");
      StringBuffer bufferTmp = new StringBuffer();

      for ( int x = 0; x < sujects.length; x++){
            stringBuffer.append(sujects[x] + ", ");
            bufferTmp.append(":"+sujects[x] + ", ");
      }

      for ( int x = 0; x < predicates.length; x++){
            stringBuffer.append(predicates[x] + ", ");
            bufferTmp.append(":"+predicates[x] + ", ");
      }

      for ( int x = 0; x < objects.length; x++){
            stringBuffer.append(objects[x]);
            bufferTmp.append(":"+objects[x]);
            if ( x + 1 < objects.length ){
                  stringBuffer.append(", ");
                  bufferTmp.append(", ");
            }
      }
      //stringBuffer.append(", DATA");
      //bufferTmp.append(", CURRENT_TIMESTAMP");

      stringBuffer.append(") VALUES ("+ bufferTmp.toString() + ")" );
      return stringBuffer.toString();
}

private SQLWriter getStatemetSQLWriter(SBIOntologyInstance instance,
String sqlStatement){
      SQLWriter writer = null;
      SBIInference inference = (SBIInference) instance;
      List<DBAttribute> objects = inference.getObjectKeys();
      if ( objects.get(0).getValue() != null){
            writer = new SQLWriter(sqlStatement);
            List<DBAttribute> sujects = inference.getSubjectKeys();
            for ( int x = 0 ; x < sujects.size(); x++){
                  writer.setParam(this.getSujects()[x],
sujects.get(x).getValue());
            }
            writer.setParam(this.getPredicates()[0],
inference.getPredicate());


            for ( int x = 0 ; x < sujects.size(); x++){
                  writer.setParam(this.getObjects()[x],
objects.get(x).getValue());
            }
```

```
		}
		return writer;
	}

	public SQLWriter getSQLDelete(SBIOntologyInstance instance) {
		return this.getStatemetSQLWriter(instance,
	DWOntologyRepositoryImpl.SQL_DELETE_FATO_STATEMENT);
	}

	public SQLWriter getSQLInsert(SBIOntologyInstance instance) {
		return this.getStatemetSQLWriter(instance,
	DWOntologyRepositoryImpl.SQL_INSERT_FATO_STATEMENT);
	}


	public SQLWriter getSQLSelect(SBIOntologyInstance instance) {
		return new
	SQLWriter(this.getJoinWriter().getSQLJoin(((BIOntologyInstance)instance).
	getDBCollection() , DBQueryWriter.LEFT_OUTER_JOIN_TYPE ));
	}

	public SQLWriter getSQLSelect(String uri) {
		return null;
	}

	public SQLWriter getSQLSelect() {
		return null;
	}

	public SQLWriter getSQLUpdate(SBIOntologyInstance instance) {
		return null;
	}

	public void cancel() throws SBIRepositoryException {
		try {
			this.getConnection().rollback();
		} catch (SQLException e) {
			throw new SBIRepositoryException(e);
		}
	}

	public void save() throws SBIRepositoryException {
		try {
			this.getConnection().commit();
		} catch (SQLException e) {
			throw new SBIRepositoryException(e);
		}
	}

	public boolean isEmpty() throws SBIRepositoryException {
		return false;
	}

	public DBQueryWriter getJoinWriter() {
		return joinWriter;
	}

	public void setJoinWriter(DBQueryWriter joinWriter) {
		this.joinWriter = joinWriter;
```

```
	}

////////////////////////////////////////////////////////////////


	public String[] getObjects() {
		return objects;
	}

	public void setObjects(String[] objects) {
		this.objects = objects;
	}

	public String[] getPredicates() {
		return predicates;
	}

	public void setPredicates(String[] predicates) {
		this.predicates = predicates;
	}

	public String[] getSujects() {
		return sujects;
	}

	public void setSujects(String[] sujects) {
		this.sujects = sujects;
	}



}

package info.stela.sbi.jena.repository.factory;

import info.stela.sbi.jena.repository.OWLDomainOntologyRepository;
import info.stela.sbi.jena.repository.OWLDomainOntologyRepositoryValidator;
import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
import info.stela.sbi.repository.SBIRepositoryFactory;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

public class OWLDomainOntologyRepositoryFactory implements SBIRepositoryFactory {

	private static OWLDomainOntologyRepositoryFactory factory;

	private OWLDomainOntologyRepositoryValidator validator;


	static {
		factory = new OWLDomainOntologyRepositoryFactory();
	}

	public static OWLDomainOntologyRepositoryFactory getInstance(){
		return factory;
	}

	private OWLDomainOntologyRepositoryFactory(){
		this.setValidator(new OWLDomainOntologyRepositoryValidator());
```

```
	}

	public OWLDomainOntologyRepository
	createRepository(OntologyRepositoryImpl implementator) throws Exception {
		if ( implementator instanceof OWLFileOntologyRepositoryImpl ==
	false)
			throw new
	IllegalArgumentException("OWLFileOntologyRepositoryImpl class is
	required");
		OWLDomainOntologyRepository repository = new
	OWLDomainOntologyRepository(implementator);
		repository.connect();
		this.getValidator().validateRepository(repository);
		return repository;
	}


	public OWLDomainOntologyRepository createRepository(String
	ontologyFileName) throws Exception {
		return this.createRepository(new
	OWLFileOntologyRepositoryImpl(ontologyFileName, null));
	}

	public OWLDomainOntologyRepository createRepository(String
	ontologyFileName, String rulesFileName) throws Exception {
		return this.createRepository(new
	OWLFileOntologyRepositoryImpl(ontologyFileName, rulesFileName));
	}


	protected OWLDomainOntologyRepositoryValidator getValidator() {
		return validator;
	}


	protected void setValidator(OWLDomainOntologyRepositoryValidator
	validator) {
		this.validator = validator;
	}


}
package info.stela.sbi.jena.repository.factory;

import info.stela.sbi.jena.repository.DWInstanceRepository;
import info.stela.sbi.jena.repository.impl.DWOntologyRepositoryImpl;
import info.stela.sbi.repository.SBIRepositoryFactory;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;

import java.sql.Connection;

public class DWInstanceRepositoryFactory implements SBIRepositoryFactory{


	private static DWInstanceRepositoryFactory factory;

	static {
		factory = new DWInstanceRepositoryFactory();
	}
```

```
	public static DWInstanceRepositoryFactory getInstance(){
		return factory;
	}


	private DWInstanceRepositoryFactory(){

	}

	public  DWInstanceRepository createRepository(OntologyRepositoryImpl
	implementator) throws Exception {
		if ( implementator instanceof DWOntologyRepositoryImpl == false)
			throw new IllegalArgumentException("DWOntologyRepositoryImpl
	implementator is required");
		DWInstanceRepository repository = new
	DWInstanceRepository(implementator);
		repository.connect();
		return repository;
	}


	public DWInstanceRepository createRepository(String driverJDBC, String
	url, String username, String password, String tripleFactName, String
	sujectAttributes, String predicateAttributes, String objectAttributes)
	throws Exception {
		DWInstanceRepository repository = new DWInstanceRepository(new
	DWOntologyRepositoryImpl(driverJDBC, url, username,
	password,tripleFactName, sujectAttributes, predicateAttributes,
	objectAttributes));
		repository.connect();
		return repository;
	}

	public DWInstanceRepository createRepository(Connection connection,
	String tripleFactName, String sujectAttributes, String
	predicateAttributes, String objectAttributes) throws Exception {
		DWInstanceRepository repository = new DWInstanceRepository(new
	DWOntologyRepositoryImpl(connection, tripleFactName, sujectAttributes,
	predicateAttributes, objectAttributes));
		repository.connect();
		return repository;
	}



}
package info.stela.sbi.jena.repository.factory;

import info.stela.sbi.jena.repository.OWLBIOntologyRepository;
import info.stela.sbi.jena.repository.OWLBIOntologyRepositoryValidator;
import info.stela.sbi.jena.repository.impl.OWLFileOntologyRepositoryImpl;
import info.stela.sbi.manager.OntologyValidator;
import info.stela.sbi.repository.SBIRepositoryFactory;
import info.stela.sbi.repository.impl.OntologyRepositoryImpl;


public class OWLBIOntologyRepositoryFactory implements SBIRepositoryFactory {
```

```
private OntologyValidator validator;

private static OWLBIOntologyRepositoryFactory factory;


static {
      factory = new OWLBIOntologyRepositoryFactory();
}

public static OWLBIOntologyRepositoryFactory getInstance(){
      return factory;
}


private OWLBIOntologyRepositoryFactory(){
      this.setValidator( new OWLBIOntologyRepositoryValidator());
}

public OWLBIOntologyRepository createRepository(OntologyRepositoryImpl
implementator) throws Exception {
      if ( implementator instanceof OWLFileOntologyRepositoryImpl ==
false)
            throw new
IllegalArgumentException("OWLFileOntologyRepositoryImpl class is
required");
      OWLBIOntologyRepository repository = new
OWLBIOntologyRepository(implementator);
      repository.connect();
      this.getValidator().validateRepository(repository);
      return repository;
}


public OWLBIOntologyRepository createRepository(String ontologyFileName)
throws Exception {
      return this.createRepository(new
OWLFileOntologyRepositoryImpl(ontologyFileName, null));
}

public OWLBIOntologyRepository createRepository(String ontologyFileName,
String rulesFileName) throws Exception {
      return this.createRepository(new
OWLFileOntologyRepositoryImpl(ontologyFileName, rulesFileName));
}


protected OntologyValidator getValidator() {
      return validator;
}


protected void setValidator(OntologyValidator validator) {
      this.validator = validator;
}
```

```
}

package info.stela.sbi.jena.ontology;


import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.ontology.SBIInference;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;

import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

import com.hp.hpl.jena.reasoner.rulesys.RuleDerivation;

public class OWLInference extends SBIInference{

	private RuleDerivation ruleDerivation;

	public OWLInference(String uri, HashMap<String, Object > properties) {
		super(uri, properties);
	}

	public OWLInference(String uri, String id, HashMap<String, Object>
	properties) {
		super(uri, id, properties);
	}

	public OWLInference(String uri) {
		super(uri);
	}

	public OWLInference(RuleDerivation ruleDerivation, DBCollection
	collection) {
		this(collection.getURI(), ruleDerivation);
		this.setSubjectKeys( collection.getPrimaryKeyAttributes() );

		this.setPredicate(this.getRuleDerivation().getConclusion().getPredi
	cate().toString());
		if ( this.isAutoReference( ruleDerivation ))
			this.addObjectKey(collection.getPrimaryKeyAttributes());
		else
			this.addObjectKey(collection);
	}

	public boolean isAutoReference(RuleDerivation ruleDerivation){
		//FIXME Tratar quando o sujeito ou objeto for um valor
		return
	ruleDerivation.getConclusion().getSubject().getURI().equals(ruleDerivatio
	n.getConclusion().getObject().getURI());
	}

	public void addObjectKey(DBCollection collection){
		Iterator<List<DBAttribute>> iterator =
	collection.getReferenceKeyAttributes().values().iterator();
		String objectDomainConcept;
		String objectRule;
		List<DBAttribute> attributes;
```

```
            while ( iterator.hasNext() ){
                  attributes = iterator.next();
                  for ( DBAttribute attribute: attributes){
                        objectDomainConcept =
      attribute.getReferenceAttribute().getParentElement().getURI();

                        if (
      this.getRuleDerivation().getConclusion().getObject().isURI() ){
                              objectRule =
      this.getRuleDerivation().getConclusion().getObject().getURI();
                              if ( objectDomainConcept.equals(objectRule))
                                    this.addObjectKey(
      ((DBCollection)attribute.getReferenceAttribute().getParentElement()).getP
      rimaryKeyAttributes() );
                              else{

            this.addObjectKey(((DBCollection)attribute.getReferenceAttribute().
      getParentElement()));
                              }
                        }
                  }

            }
      }

      public void addObjectKey(List objectKey){
            this.getObjectKeys().addAll(objectKey);
      }

      public OWLInference(String uri,RuleDerivation ruleDerivation) {
            this(uri);
            this.setRuleDerivation(ruleDerivation);
      }

      public RuleDerivation getRuleDerivation() {
            return ruleDerivation;
      }


      public void setRuleDerivation(RuleDerivation ruleDerivation) {
            this.ruleDerivation = ruleDerivation;
      }

      public String getRule() {
            return this.getRuleDerivation().getRule().getName();
      }

}
package info.stela.sbi.jena.ontology;

import info.stela.sbi.manager.SBIMetadata;
import info.stela.sbi.ontology.BIOntologyInstance;
import info.stela.sbi.ontology.bi.DBAttribute;
import info.stela.sbi.ontology.bi.DBCollection;

import java.util.HashMap;
import java.util.Iterator;

import com.hp.hpl.jena.rdf.model.Property;
```

```
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.impl.StatementImpl;

public class OWLBIOntology extends BIOntologyInstance {

	private HashMap<Resource, DBCollection> collections;
	private HashMap<Resource, DBAttribute> attributes;


	public OWLBIOntology(String uri, Resource instance) {
		super(uri);
		this.setCollections(new HashMap<Resource, DBCollection>());
		this.setAttributes(new HashMap<Resource, DBAttribute>() );
		this.setDBCollection(this.createDBCollection(instance));
	}

	public DBCollection createDBCollection(Resource instance){
		DBCollection dbCollection = null;
		//FIXME Ocorre NullPointerException caso o Conceito não esteja
	mapeado na ontologia de BI
		String uri = instance.getURI();

		String name =
	(String)instance.getProperty(instance.getModel().getProperty(SBIMetadata.
	NAME_URI)).asTriple().getObject().getLiteralValue();
		String schema =
	(String)instance.getProperty(instance.getModel().getProperty(SBIMetadata.
	SCHEMA_URI)).asTriple().getObject().getLiteralValue();
		String domainConcept =
	(String)instance.getProperty(instance.getModel().getProperty(SBIMetadata.
	DOMAIN_CONCEPT_URI)).asTriple().getObject().getLiteralValue();
		dbCollection = new DBCollection(domainConcept, name, null,
	domainConcept, null, schema );

		//ATTRIBUTE
		Iterator attributes = instance.listProperties(
	instance.getModel().getProperty(SBIMetadata.ATTRIBUTE_URI)  );
		Resource attribute;
		DBAttribute dbAttribute;
		StatementImpl statement;
		while ( attributes.hasNext() ) {
			statement = (StatementImpl) attributes.next();
			attribute = (Resource) statement.getObject();
			dbAttribute = this.createDBAttribute(dbCollection,
	attribute);
			this.getAttributes().put(attribute, dbAttribute);
			dbCollection.add( dbAttribute );
		}

		return dbCollection;


	}



	private DBAttribute createDBAttribute(DBCollection dbCollection, Resource
	attribute) {
```

```
            String attributeName = null, attributeDomainConcept = null,
attributeType = null;
            boolean isPK = false;

            attributeName = attribute.getProperty(
attribute.getModel().getProperty(SBIMetadata.NAME_URI)
).getLiteral().getString();
            Property domainConceptProperty =
attribute.getModel().getProperty(SBIMetadata.DOMAIN_CONCEPT_URI) ;
            if ( attribute.getProperty( domainConceptProperty ) != null )
                  attributeDomainConcept = attribute.getProperty(
domainConceptProperty  ).getLiteral().getString();

            Property primaryKeyProperty =
attribute.getModel().getProperty(SBIMetadata.PRIMARY_KEY_URI);
            Statement stmPK = attribute.getProperty( primaryKeyProperty
);
            if ( stmPK.getObject() !=  null)
                  isPK = attribute.getProperty( primaryKeyProperty
).getBoolean();

            Property referenceKeyProperty =
attribute.getModel().getProperty(SBIMetadata.REFERENCE_URI);
            DBAttribute referenceAttribute = null;


            DBAttribute dbAttribute = new DBAttribute(attribute.getURI(),
attributeName, null, attributeDomainConcept, dbCollection, isPK,
referenceAttribute);


            if ( attribute.getProperty(referenceKeyProperty) != null ){
                  Statement fkStatement =
attribute.getProperty(referenceKeyProperty);

                  Resource foreignResource = (Resource)
fkStatement.getObject();

                  Resource collectionParent = (Resource)
foreignResource.getProperty(
attribute.getModel().getProperty(SBIMetadata.PARENT_URI)  ).getObject();

                  DBCollection parent =
this.getCollections().get(collectionParent);
                  if ( parent == null ){
                        parent = this.createDBCollection( collectionParent
);
                        this.getCollections().put(collectionParent,
parent);
                  }

                  DBAttribute att =
this.getAttributes().get(foreignResource);
                  if ( att != null ){
                        dbAttribute.setReferenceAttribute(att);
                  }
            }
```

```
            return dbAttribute;

      }

      private HashMap<Resource, DBCollection> getCollections() {
            return collections;
      }

      private void setCollections(HashMap<Resource, DBCollection>
      resourcesDomain) {
            this.collections = resourcesDomain;
      }

      public HashMap<Resource, DBAttribute> getAttributes() {
            return attributes;
      }

      public void setAttributes(HashMap<Resource, DBAttribute> attributes) {
            this.attributes = attributes;
      }

}
```