# Processamento de Consultas Baseado em Ontologias para Sistemas de Biodiversidade

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Bruno Siqueira Campos Mendonça Vilar e aprovada pela Banca Examinadora.

Campinas, 21 de setembro de 2009.

Profª. Drª. Claudia M. Bauzer Medeiros
Instituto de Computação – UNICAMP
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

**FICHA CATALOGRÁFICA GLABORADA PGLA**
**BIBLIOTGCA DO IMGCC DA UNICAMP**
Bibliotecáriaá Maria Fabiana Bezerra Müller – CRB8 / 6162

Vilar, Bruno Siqueira Campos Mendonça

V71p Processamento de consultas baseado em ontologias para sistemas de biodiversidade/ Bruno Siqueira Campos Mendonça Vilar -- Campinas, [S.P. á s.n.], 2009.

Orientador á Claudia M. Bauzer Medeiros.

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Processamento de consulta. 2.Ontologia. 3.Diversidade biológica. 4.Serviços na Web. I. Medeiros, Claudia Maria Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglêsá Ontology based query processing for biodiversity systems

Palavras-chave em inglês (Keywords)á 1.Query processing. 2.Biodiversity. 3.Ontology. 4.Web services.

Área de concentraçãoá Banco de Dados

Titulaçãoá Mestre em Ciência da Computação

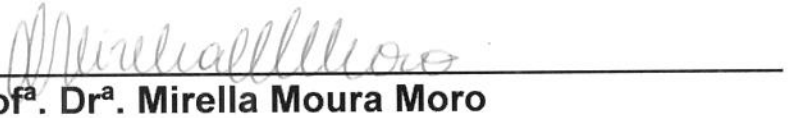
Banca examinadoraá Profa. Dra. Claudia M. Bauzer Medeiros (IC-UNICAMP)
Profa. Dra. Mirella M. Moro (DCC-UFMC)
Prof. Dr. Rodolfo J. de Azevedo (IC-UNICAMP)

Data da defesaá 21/09/2009

Programa de Pós-Graduaçãoá Mestrado em Ciência da Computação

# TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 21 de setembro de 2009, pela Banca examinadora composta pelos Professores Doutores:



**Profª. Drª. Mirella Moura Moro**
**DCC / Universidade Federal de Minas Gerais**

**Prof. Dr. Rodolfo Jardim de Azevedo**
**IC / UNICAMP.**

**Profª. Drª. Claudia Maria Bauzer Medeiros**
**IC / UNICAMP.**

Instituto de Computação
Universidade Estadual de Campinas

# Processamento de Consultas Baseado em Ontologias para Sistemas de Biodiversidade

**Bruno Siqueira Campos Mendonça Vilar**[1]

Outubro de 2009

**Banca Examinadora:**

- Profª. Drª. Claudia M. Bauzer Medeiros
  Instituto de Computação – UNICAMP (Orientadora)
- Profª. Drª. Mirella M. Moro
  Departamento de Ciência da Computação – UFMG
- Prof. Dr. Rodolfo J. de Azevedo
  Instituto de Computação – UNICAMP
- Profª. Drª. Eliane Martins (Suplente)
  Instituto de Computação – UNICAMP
- Drª. Michela Borges (Suplente)
  Museu de Zoologia – UNICAMP

[1]Suporte financeiro de: Bolsa do CNPq (135173/2007–8 e 133251/2008–0) 2007–2009

# Resumo

Sistemas de informação de biodiversidade lidam com um conjunto heterogêneo de informações providas por diferentes grupos de pesquisa. A diversificação pode ocorrer com relação às espécies estudadas, à estruturação das informações coletadas, ao local de estudo, metodologias de trabalho ou objetivos dos pesquisadores, dentre outros fatores. Esta heterogeneidade de dados, usuários e procedimentos dificulta o reuso e o compartilhamento de informações.

Este trabalho contribui para diminuir tal obstáculo, melhorando o processo de consulta às informações em sistemas de biodiversidade. Para tanto, propõe um mecanismo de expansão de consultas que pré-processa uma consulta de usuário (cientista) agregando informações adicionais, provenientes de ontologias, para aproximar o resultado da intenção do usuário. Este mecanismo é baseado em serviços Web e foi implementado e testado usados dados e casos de uso reais.

# Abstract

Biodiversity information systems need and manage heterogeneous information provided by different research groups. Heterogeneity occur with respect to the species studied, the structure of the information gathered, the region of study, the work methodologies, or the vocabularies and objectives of the researchers, among other factors. This heterogeneity of data, users and procedures hampers information sharing and reuse.

This work contributes to reduce this obstacle, improving the query processing mechanisms in biodiversity systems. Its main interpretation is a query expansion mechanism that pre-processes a user (scientist) query aggregating additional information from ontologies, thereby approximating query results to what is intended by the user. This mechanism is based on Web services and was implemented and tested using real case studies.

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução e Motivação

Estudos em biodiversidade se baseiam em diversos tipos de modelos para definir fatores como riqueza de espécies, abundância, endemismo, distribuição e diferentes outras variáveis [20], correlacionadas a dados geográficos. Conforme explica Bisby [4], três fatores contribuem para que a ciência da biodiversidade exija estudos globais. O primeiro consiste na distribuição geográfica dos fenômenos e espécies envolvidos. O segundo se refere à interdependência global de eventos, que faz com que uma região seja afetada por fenômenos que ocorrem em regiões vizinhas ou mesmo em regiões distantes, que possuem características climáticas similares. O último fator ocorre pela necessidade dos estudos da ciência da biodiversidade global de sintetizar numerosas observações e estudos feitos por observadores, equipes e instituições.

Os Sistemas de Informação de Biodiversidade estão inseridos neste contexto. Estes sistemas dão apoio à condução dos processos de avaliação, predição, planejamento e tomada de decisão feita sobre biodiversidade [58]. Para isso, baseiam-se em informações derivadas de coleções heterogêneas de dados providos por grupos de pesquisa, os quais atuam com metodologias e visões diferentes das áreas de pesquisa. Um tipo especial de dados usado por esses sistemas são os repositórios de coleta/observação de espécies, que detalham, para cada espécie, onde foi coletado, quando, por quem e como.

Um dos desafios encontrados no desenvolvimento desses sistemas é fornecer aos usuários (cientistas) a capacidade de consultar as diferentes fontes de informação sem obrigá-los a lidar com os aspectos de heterogeneidade de estrutura, representação, vocabulário e dados incompletos. O objetivo deste trabalho é solucionar este desafio, de forma a facilitar o processo de consulta em sistemas de biodiversidade. Para tanto, propõe algoritmos que pré-processam uma consulta de usuário, desambiguando termos e agregando informações.

A base para tal pré-processamento reside em utilizar ontologias para aumentar a semântica. Uma ontologia define um conjunto de primitivas de representação com as

quais se pode modelar um domínio do conhecimento ou de discurso [22]. Fornece, por exemplo, a possibilidade de sinônimos, especialização e generalização de termos, seus relacionamentos, entre outros. Com isto, uma consulta pode ser estendida. Além disso, informações não armazenadas podem ser deduzidas.

O trabalho pressupõe que os dados a serem consultados estão distribuídos em repositórios na *Web*. Cada repositório é mantido por um grupo de cientistas e seu conteúdo é disponibilizado por serviços *Web*, que também disponibilizam o esquema dos dados armazenados. As ontologias usadas são fornecidas pelos especialistas de domínio. Desta forma, não cabe a este trabalho questões como atualização ou curadoria de dados, ou manutenção de ontologias. O foco é especificar e validar mecanismos de pré-processamento de consultas em sistemas de biodiversidade. As principais contribuições são assim:

- Levantamento bibliográfico de trabalhos de expansão e reescrita semântica de consultas;
- Especificação da arquitetura para o pré-processamento e expansão de consultas, incluindo as etapas de representação do esquema do banco de dados em ontologia, reparo de consultas utilizando as informações do esquema do banco de dados e expansão de consultas usando ontologias de domínio;
- Definição de um padrão de entrada para a personalização das expansões a serem realizadas nas consultas e criação de algoritmos para a utilização do padrão em conjunto com a arquitetura para expansão;
- Implementação do protótipo da arquitetura especificada com acesso por serviços *Web*.

Parte da pesquisa foi publicada nos seguintes artigos:

- B. S. C. M. Vilar e C. B. Medeiros. Processamento Semântico de Consultas para Sistemas de Biodiversidade, VII WTDBD - Workshop de Teses e Dissertações em Bancos de Dados, 2008 [53];
- J. E. G. Malaverri, B. S. C. M. Vilar e C. B. Medeiros. A Tool Based on Web Services to Query Biodiversity Information, 5th International Conference on Web Information Systems and Technologies (Webist 2009), [29].

Esta dissertação está estruturada da seguinte forma: o Capítulo 2 apresenta os conceitos que fundamentam o trabalho e aborda trabalhos correlacionados ao tema de pesquisa. O Capítulo 3 apresenta a proposta, com os algoritmos de expansão de consultas, enquanto o Capítulo 4 descreve os aspectos de implementação e validação com dados reais. Por fim, o Capítulo 5 apresenta as conclusões desta dissertação e trabalhos futuros que poderão ser realizados.

# Capítulo 2

# Revisão Bibliográfica

Este capítulo aborda os temas que embasam o trabalho realizado, em especial, dentro da área de Sistemas de Biodiversidade, apresentada na Seção 2.1. A Seção 2.2 apresenta a definição de ontologia e descreve seus elementos e aplicações. A Seção 2.3 discorre sobre Serviços Web e os elementos que os compõem. A Seção 2.4 apresenta trabalhos sobre processamento de consulta correlacionados ao tema desta dissertação.

## 2.1 Sistemas de Biodiversidade

Sistemas de Informação de Biodiversidade (SIB) são sistemas de informação ambientais que gerenciam grandes conjuntos de dados geográficos, envolvendo bancos de dados volumosos a respeito de espécies – coleções de história natural, registros de observações de campo, dados experimentais, entre outros [14]. Tais sistemas têm como objetivo facilitar a pesquisa sobre biodiversidade, correlacionando múltiplas informações para permitir identificar espécies, relacionar efeito e causa de mudanças em ecossistemas, realizar o planejamento de ações, promover a conservação e o uso sustentável da biodiversidade, dentre outros.

O estudo, a conservação e o uso sustentável da biodiversidade requerem um tratamento multi e interdisciplinar em um ambiente de colaboração global [8]. A utilização de um SIB para essa finalidade requer gerenciar e correlacionar dados de ocorrência de espécies com diferentes outros tipos de informação, como dados geográficos, dicionários de termos geográficos e topônimos, catálogos de nomes científicos, registros históricos e vários outros [16].

No meio acadêmico podem ser encontrados diferentes sistemas que atuam na área de biodiversidade, com variações de escopo e finalidade. O OBIS (*Ocean Biogeographic Information System*) [12] trata de informações referentes à biodiversidade marinha, enquanto o GBIF (*Global Biodiversity Information Facility*) cobre aspectos de biodiversidade em um

contexto global, com múltiplas e variadas fontes de informação. Além destes, é possível citar o SEEK (*Science Environment for Ecological Knowledge*) [32] e o CNBIQS (*China National Biodiversity Information Query System*) [24], os quais oferecem formas de assistir o usuário no processo de consulta a diferentes fontes de dados sobre biodiversidade.

O BioCORE (BIOdiversity and COmputing REsearch) [49] é um projeto desenvolvido no Laboratório de Sistemas de Informação (LIS - http://www.lis.ic.unicamp.br) do Instituto de Computação da UNICAMP em conjunto com o Instituto de Biologia e com o IME-USP. Seu objetivo é fornecer apoio a cientistas e pesquisadores da área de Biologia para que estes possam realizar consultas exploratórias multimodais sobre fontes de dados heterogêneas a respeito de biodiversidade. O projeto é uma evolução do projeto WeBIOS [48].

A arquitetura do BioCORE é proposta sobre um conjunto de serviços *Web*, com isto, visa permitir um acesso facilitado aos recursos do projeto e promover a colaboração entre grupos de pesquisa, como consequência do compartilhamento e troca de informações. A Figura 2.1 apresenta a arquitetura do BioCORE.

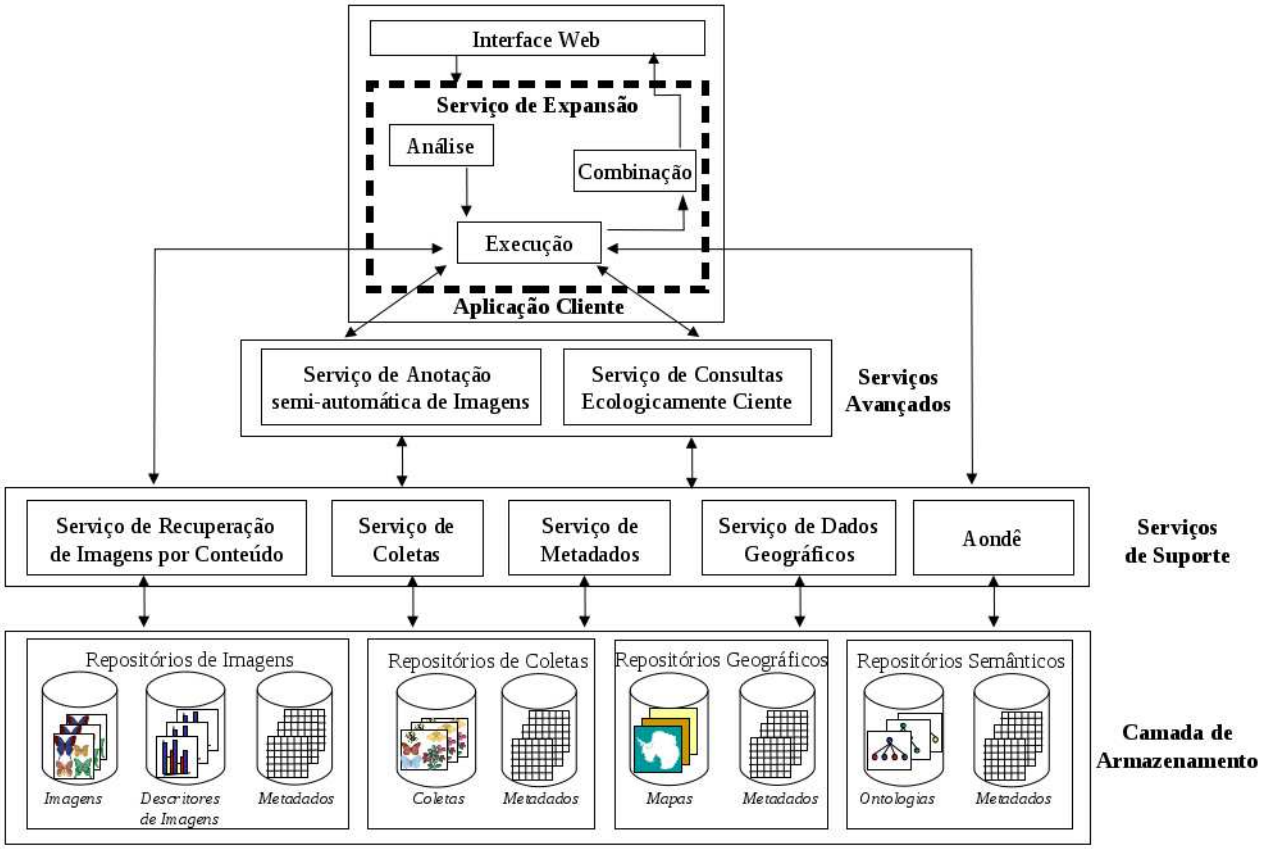



Figura 2.1: Arquitetura do BioCORE

A arquitetura abrange a *aplicação cliente*, que fornece uma interface entre o usuário e os serviços disponíveis. Os serviços são categorizados como de *suporte* e *avançados*. O primeiro grupo, de *serviços de suporte*, fornece consultas de acesso aos dados dos repositórios localizados na *camada de armazenamento* e compreendem os serviços de: recuperação de imagens por conteúdo, coletas, metadados, dados geográficos e ontologias. O segundo conjunto, de *serviços avançados*, é composto pelos serviços de anotação semi-automática de imagens e de consultas ecologicamente cientes.

A *camada de armazenamento* possui um conjunto de repositórios responsável pelo gerenciamento de informações sobre imagens, mapas, repositórios de coletas e ontologias. Estes recursos permitem associar informações coletadas por pesquisadores, gerenciar imagens para posteriores análise e processamento. As ontologias fornecem a possibilidade de guardar informações conceituais sobre as áreas trabalhadas e empregá-las na criação de instâncias, que podem ser, por exemplo, dados coletados sobre espécies. Além disso, cada repositório mantém um conjunto de metadados para facilitar o gerenciamento e a recuperação das informações.

Esta dissertação está inserida na *aplicação cliente*, destacado na Figura 2.1. Atua como um módulo de pré-processamento de consultas, por meio do qual as consultas dos usuários são analisadas e aperfeiçoadas, com obtenção de informações das ontologias, para a seguir acessarem as informações dos repositórios. As consultas deverão utilizar os serviços *Web* disponíveis, conforme a arquitetura do BioCORE.

Em particular, o trabalho irá utilizar o Aondê [15], um serviço *Web* que provê diversas funcionalidades para permitir realizar operações sobre ontologias, como armazenamento, gerenciamento, comparação, integração e *ranking*. Este serviço, localizado na camada de *serviços de suporte* do BioCORE, será o meio pelo qual as ontologias serão obtidas e os termos serão desambiguados.

## 2.2 Ontologias

Uma ontologia é uma especificação de uma conceitualização [21]. Por conceitualização, entende-se a visão e o entendimento que se tem da realidade. Dessa forma, ontologias procuram capturar a semântica de um domínio pelo desenvolvimento de primitivas de representação do conhecimento, habilitando que máquinas possam (parcialmente) entender o significado dos relacionamentos entre os conceitos em um domínio [46].

A característica de processamento em máquinas também é evidenciada na definição de Quiou [42], para o qual uma ontologia "... descreve entidades em um mundo e seus relacionamentos, combinando o entendimento humano de símbolos com a capacidade de processamento por máquinas". Para Noy [39], é a descrição formal de um domínio, projetada para o compartilhamento entre diferentes aplicações e expressa em uma linguagem

que pode ser usada para o raciocínio.

A capacidade de descrever formalmente um conhecimento e de raciocinar sobre ele torna possível adotar uma ontologia como forma de representar um domínio, padrão ou consenso de um grupo e, posteriormente, utilizar a mesma representação por meio de mecanismos e ferramentas que a processem. Assim, nota-se uma disposição para o uso de ontologias como uma forma de representar concepções, regras e diferentes recursos em tarefas que podem comportar serviços automatizados.

Ontologias restringem o conjunto de possíveis mapeamentos entre símbolos e seus significados [45]. Isso ocorre através da utilização de conceitos, propriedades, relacionamentos, restrições, axiomas e instâncias:

- Conceitos são abstrações de objetos ou partes enumeráveis do universo representado;
- Propriedades permitem traçar as características dos conceitos, sob a forma de atributos que os caracterizem;
- Restrições tornam possível aproximar cada conceito da realidade representada, impondo condições a serem satisfeitas;
- Relacionamentos representam as diferentes relações existentes entre os conceitos, como as relações de herança e de parte/todo, bem como quaisquer alternativas que possam especificar a interação entre dois objetos;
- Axiomas permitem tornar explícito o conhecimento que está implícito em um domínio;
- Instâncias representam casos, ou exemplos de conceitos, que utilizam os componentes supracitados com a atribuição de valores que as tornem únicas.

Entre os motivos pelos quais se empregam ontologias podem ser citados [39]:

- Compartilhar um entendimento comum da estrutura da informação entre pessoas ou agentes de *software*: se um conjunto de aplicativos está contextualizado em um domínio comum ou em domínios relacionados, é possível utilizar agentes para consultar as informações abrangidas por diferentes ontologias para responder a consultas de forma eficiente, ainda que o conhecimento não esteja diretamente armazenado;
- Habilitar o reuso do domínio do conhecimento: se um grupo de pesquisa trabalha com uma área do conhecimento, é possível que outros grupos, envolvidos em áreas relacionadas, aproveitem o conhecimento produzido pelo primeiro grupo e reutilizem as informações produzidas;

- Analisar o domínio do conhecimento: ter uma especificação de um domínio é útil à medida em que é possível realizar uma análise formal dos termos e assim poder reutilizar ou estender tal domínio.

O trabalho de McGuinness [30], apresenta um conjunto de aplicações que empregam desde ontologias estruturalmente mais simples, como taxonomias, até ontologias com estruturas mais complexas, que extrapolam as características de uma hierarquia. Dentre as aplicações estão: verificar a consistência das informações, suportar a interoperabilidade, inferir e explorar a generalização/especialização de informações, organizar e navegar entre conceitos, eliminar ambigüidade, dentre outras.

No contexto da Biologia, ontologias são um fator de auxílio porque incorporam o conhecimento abstrato exigido para a integração e a análise de dados [43]. Isso ocorre, segundo os autores, porque o problema da análise e recuperação eficiente de dados se torna um gargalo científico, à medida em que mais dados são armazenados em meios digitais, especialmente pela forma complexa e semântica característica dos dados desse contexto. Assim, são criados sitemas como o TAMBIS (*Transparent Access to Multiple Bioinformatics Information Sources*) [41], que utiliza ontologias em um projeto cujo objetivo é tornar transparente, para usuários que realizam buscas sobre bioinformática, a diversidade de estruturas de dados, interfaces e localização das fontes de dados.

Particularmente entre Sistemas de Informação de Biodiversidade, é possível encontrar diferentes finalidades conferidas às ontologias. Trabalhos como OBSERVER [31], SEEK [32] e WeBIOS [48] incluem o uso de ontologias para permitir a consulta e a análise de dados em fontes de informação múltiplas e heterogêneas. Além disso, é possível encontrar aplicações específicas como descrever recursos e serviços para automatizar processos, controlar um vocabulário de termos, contextualizar e inferir informações para o processamento de consultas, dentre outros.

## 2.3 Serviços *Web*

Serviços *Web* são aplicações auto-contidas, auto-descritas e modulares que podem ser publicadas, localizadas e invocadas a partir da *Web* [54]. A base para o funcionamento de tais serviços é o conjunto de especificações XML (*Extensible Markup Language*), formado por: WSDL, UDDI e SOAP.

O WSDL (*Web Service Description Language*) é um formato XML para descrever serviços de rede como um conjunto de terminais operando sobre mensagens e que contém informações orientadas a documentos ou a procedimentos [9]. A partir da descrição de um serviço, que atua como sua interface pública, é possível determinar os métodos disponíveis para utilização, bem como os parâmetros recebidos e o tipo de retorno dos mesmos. Além

disso, são especificados os tipos abstratos de dados envolvidos nas operações e o endereço de rede no qual o serviço está disponível.

O UDDI (*Universal Description, Discovery and Integration*) tem como finalidade auxiliar na descoberta de serviços. O formato consiste numa especificação para um registro online que habilita a publicação e descoberta dinâmica de serviços *Web* [56]. Por meio dele, são fornecidas informações a respeito da entidade que fornece o serviço, como nome e dados para contato e sobre os serviços prestados.

O SOAP (*Simple Object Access Protocol*) é um protocolo criado para a troca de informações estruturadas em um ambiente descentralizado e distribuído [23]. Ele trabalha em protocolos de transporte, como HTTP[1] e o SMTP[2] [13]. Sua principal característica é a simplicidade, sendo composto de três partes [23]:

- Envelope: mantém informações referentes ao destinatário, ao conteúdo da mensagem e ao seu caráter obrigatório ou opcional;
- Regras de codificação: definem o mecanismo de serialização que pode ser usado para transmitir dados entre aplicações;
- Representação RPC (*Remote Procedure Call*): determina as convenções adotadas para representar as chamadas remotas a procedimentos.

Por suas especificações representadas em XML [7] e definidas em XML Schema [44], os WSDL e o SOAP apresentam características como auto-descrição e portabilidade. Isso ocorre porque o XML possui natureza auto-descritiva e nomes de elementos compreensíveis aos humanos. A habilidade de referenciar especificações externas de documentos permite às aplicações trocar e interpretar dinamicamente dados sem um conjunto de pressuposições [55].

A principal vantagem do uso de serviços *Web* é a habilidade de criar aplicações dinamicamente por meio do uso de componentes de *software* reusáveis e com acoplamento fraco [54]. Com isso, é possível que diferentes grupos e organizações possam trocar dados e criar uma rede de serviços, sem que tais serviços dependam de características específicas da implementação dos demais.

À medida em que o número de serviços *Web* disponíveis cresce, cria-se a expectativa de automatizar serviços e agregá-los a fim de compor a solução de problemas. Neste contexto, a composição de serviços *Web* se torna uma área de interesse, criando novas soluções a partir de serviços já existentes.

A composição automatizada de serviços *Web* é a tarefa de gerar automaticamente um novo serviço *Web* que atinja um objetivo pela interação de alguns serviços *Web* disponíveis

[1] http://www.w3.org/Protocols/
[2] http://www.ietf.org/rfc/rfc0821.txt

[52]. Sua utilização acelera o desenvolvimento rápido de aplicações, o reuso de serviços e o consumo de serviços complexos [33].

Dadas as dificuldades encontradas para realizar a composição manual dos serviços, são estudadas técnica para a composição automática. Para atingir tal objetivo é preciso atender a um conjunto de requisitos [33], como conectividade, propriedades não funcionais de qualidade de serviço, corretude e escalabilidade.

Para atender a tais requisitos, diferentes linguagens e modelos vêm sendo criados e comparados para promover a composição entre serviços *Web*. Entre as propostas estão a OWL-S (*OWL-based Web Service Ontology*) e o BPEL4WS (*Business Process Execution Language for Web Services*). A OWL-S é uma ontologia, baseada em OWL (*Web Ontology Language*), que fornece um conjunto de construções, por meio de marcações, para descrever propriedades e capacidades de serviços *Web*, de modo não ambíguo e capaz de ser processado por computadores [47]. O BPEL4WS fornece uma linguagem para a especificação formal de processos de negócio e protocolos de interação entre negócios, os quais descrevem o comportamento de processos de negócio baseados em serviços *Web* [3].

## 2.4 Técnicas de Processamento de Consultas para Expansão Semântica

Técnicas de processamento de consulta são utilizadas como forma de ajustar uma consulta às fontes de informação ou de aperfeiçoar algumas de suas características, seja por uma semântica melhor definida ou por uma sintaxe que beneficie sua execução. Outro objetivo é acelerar a execução de consultas. Como características podem ser citadas a capacidade de retornar resultados mais ou menos expressivos, a utilização de recursos para a sua execução, a forma pela qual seu significado é escrito, entre outras.

Há diversas técnicas de processamento, dentre as quais: reformulação [6, 18], expansão [2, 26, 35], substituição [25], enriquecimento [50] e relaxamento [27, 5]. Entre os objetivos visados por essas técnicas estão:

- O aumento de desempenho na execução, com a diminuição do tempo de execução ou de recursos utilizados;
- A capacidade de consultar informações em bases de dados heterogêneas, com a finalidade de promover sua integração;
- A possibilidade de obter resultados diferentes a partir da criação de consultas com o mesmo significado, mas escritas em formas variadas;

- A obtenção de resultados mais precisos a partir da modificação de uma consulta originalmente falha, que não produziria retorno.

A Tabela 2.1 apresenta um quadro comparativo entre as diferentes publicações estudadas no que tange às formas de processamento de consulta. Apesar de tais formas terem um conjunto diversificado de denominações, é possível perceber grupos de técnicas que apresentam semelhanças quanto aos objetivos e definições. Ressalta-se que nem sempre há consenso na terminologia - vide os trabalhos de Necib e Freytag [36, 37, 38], os quais denominam "Processamento de Consulta" a tarefa de reformular uma consulta a fim de obter resultados significativos adicionais. Em contrapartida, Florescu et al. [18] definem por "Reformulação de Consulta" a técnica cujo objetivo é transformar uma consulta de entrada em consultas equivalentes, cada qual correspondendo a uma forma alternativa de computar o resultado. A tabela também referencia termos específicos a técnicas de processamento (*caching* e *folding*), explicados a seguir.

Como observado na última linha, o trabalho desenvolvido na dissertação se caracteriza por usar expansão de consultas com ontologias de domínio e de banco de dados, utilizando relações hierárquicas, equivalências e instâncias. O trabalho se diferencia dos demais pela possibilidade de personalizar as expansões de consulta, fornecendo ao usuário o controle sobre modificações que poderão ser feitas.

| **Autor** | **Método** | **Ontologias** | **Técnica** | **Aplicação** | **Tipo de Execução** |
|---|---|---|---|---|---|
| [36, 37, 38] | Reescrita de Consulta | Ontologia para descrever o contexto do BD e as relações entre ambos. | Regras de dedução, regras de generalização e hierarquias conceituais | Base de dados relacional única | Automático |
| [57] | Reescrita de Consulta (global e local) | Mapeamento e integração de bases de dados heterogêneas - Ontologias em RDFS[6] - Linguagem resultante: RDQL(*RDF Data Query Language*) | Integração dos dados | Bases de dados heterogêneas (RDF e XML) | Automático |

**Tabela 2.1 – Continuação da página anterior**

| Autor | Método | Ontologias | Técnica | Aplicação | Tipo de Execução |
|---|---|---|---|---|---|
| [25] | Substituição de Consulta | - | Utiliza o *log* das sessões dos usuários para identificar reformulações feitas por eles e encontrar palavras correlacionadas nas buscas | *Web* - Buscas patrocinadas e propagandas | Automático, por histórico de ações dos usuários |
| [34] | Reformulação de Consulta | Ontologias de domínio | Recebe consultas especificadas por termos de ontologias e as reescreve gerando consultas relacionais | Base de Dados relacionais | Automático |
| [6] | Reformulação de Consulta | - | Baseado em *Query Clarity* - *ranking* de similaridade ou coerência de um termo com relação ao espaço de busca. | *Web* | Semi-automático / interativo |
| [18] | Reformulação e decomposição de consulta | - | Reescrita semântica (regras de reescrita e conhecimento semântico) e reescrita sintática (forma canônica). | CIS e Multidatabase Systems | Automático |
| [2] | Expansão de Consulta | WordNET - Ontologias léxica para avaliar a similaridade semântica e desambiguização | Híbrido - Ontologias e método probabilístico (*Pseudo-Relevance Feedback: Local Context Analysis*) | *Web* | Automático |
| [26] | Expansão de Consulta | | *Thesaurus* - Utilizado para a representação das consultas em nível conceitual. Há outros 2 níveis: linguístico e de *string*. | Recuperação de Informação - Documentos | Automático |

**Tabela 2.1 – Continuação da página anterior**

| Autor | Método | Ontologias | Técnica | Aplicação | Tipo de Execução |
|---|---|---|---|---|---|
| [59] | Expansão de Consultas | Ontologias fuzzy | Baseada em ontologias fuzzy, que permitem realizar expansão considerando relacionamentos difusos. | Base de Dados relacionais | Automático |
| [35] | Expansão de Consulta | WordNET | Uso de ontologias para capturar o domínio semântico das palavras. | *Web* | Automático |
| [5] | Relaxamento de Consultas | - | Baseada nos conceitos de *fuzzy query* e proposições falsas | Base de Dados | Automático |
| [27] | Relaxamento de Consulta | Ontologia - descrição para a descoberta de serviços *Web*. | Divide a ontologia em visões e classifica conceitos por sua densidade de informações. Considera-se também o comportamento do usuário. | Descoberta de serviços *Web* (Baseada em estudos em IR e BD) | Automático |
| [50] | Enriquecimento de Consulta | Ontologias de domínio construídas a partir do trabalho com mineração de texto | Vetor de características (sinônimos, conjugações e termos relacionados contextualmente) para cada conceito da ontologia presente na consulta | RI - Documentos | Automático |
| [19] | *Intensional Query Optimization* (aplicado a *Views*) | - | *Semantic Query Caching* e *Query Folding* | - | Automático |
| [40] | Otimização Semântica de Consulta | - | - | OODB e restrições representadas em Cláusulas Horn | Automático |

**Tabela 2.1 – Continuação da página anterior**

| Autor | Método | Ontologias | Técnica | Aplicação | Tipo de Execução |
|---|---|---|---|---|---|
| [53] | Expansão de Consulta | Ontologias de Domínio e de BD | Relações hierárquicas, equivalências e instâncias | Base de Dados relacionais | Automático |

Tabela 2.1: Trabalhos com Processamento de Consulta.

O **processamento de consulta baseado em ontologias** consiste em reformular uma consulta de usuário de tal modo que a consulta resultante forneça resultados significativos adicionais que correspondam à intenção do usuário [36]. Entre as formas utilizadas para aperfeiçoar as consultas está a utilização de conhecimento semanticamente representado através de ontologias. Denominada Otimização Semântica de Consultas, essa técnica objetiva reformular uma consulta em outra mais eficiente, a qual é semanticamente equivalente, ou seja, fornece a mesma resposta [37].

A **reescrita de consulta** é uma das técnicas que se beneficiam do conhecimento semanticamente representado. As técnicas de reescrita que têm sido propostas exploram *caches* semânticos de consulta, visões materializadas e conhecimento semântico sobre o domínio da base de dados para otimizar a avaliação da consulta [19]. A reescrita da consulta é frequentemente um recurso chave para sistemas de integração baseados em mediadores e sistemas baseados em *peer-to-peer* [57]. O *caching* semântico armazena descrições semânticas sobre os dados de consultas executadas anterioremente.

Dentro do conceito de mediadores, Xiao [57] emprega essa técnica, especificamente, para que uma consulta global a múltiplas fontes de dados seja reescrita por mapeamento em várias consultas locais. A resposta da consulta global consiste na integração das respostas retornadas por cada consulta local. Desta forma, obtém-se a transparência no processo de consulta a bases de dados múltiplas.

Semelhante à reescrita, a técnica de **substituição de consulta** tem por objetivo substituir a consulta original do usuário por uma outra similar. A nova consulta se mantém fortemente relacionada à busca original, contendo termos intimamente relacionados aos termos originais [25]. O trabalho de Jones et al. [25] deriva as consultas a partir das sessões dos usuários e adota recursos como a mudança ortográfica, substituição por sinônimo, generalização, especialização e termos relacionados para que uma nova consulta dê lugar à do usuário.

A **reformulação de consulta** consiste em outra técnica com conceito próximo ao das técnicas citadas anteriormente. Seu objetivo é transformar uma consulta, fornecida por um usuário, em consultas diferentes, cada qual correspondendo a uma forma alternativa de computar o resultado [18]. Entre os artifícios adotados por Florescu et al. [18] para a

reformulação de uma consulta estão a reescrita semântica – baseada em regras de reescrita e conhecimento semântico – e a reescrita sintática. No trabalho de [34] as consultas são criadas em uma interface, utilizando conceitos de ontologias, e são transformadas em consultas relacionais pelo sistema que emprega mapeamentos entre as ontologias e os bancos de dados.

A **expansão de consulta** é o processo de aumentar a consulta do usuário com termos adicionais, tendo como propósito a melhora dos resultados obtidos [2]. As técnicas e recursos utilizados para expandir as consultas incluem ontologias e métodos probabilísticos, no trabalho de Andreou [2], e extração de termos a partir de um conjunto de documentos obtidos ou *query logs*, conforme apresentam Navigli e Velardi [35], e a utilização de ontologias *fuzzy*, no trabalho de [59]. O último trabalho, apesar de ser caracterizado como expansão de consulta, baseia-se nas técnicas de [38].

A adoção desta técnica pode levar a problemas como *query drift* [2, 25], *outweighting* [2] e custo alto de processamento [25]. O problema relacionado ao *query drift* consiste tornar uma consulta distante do interesse original do usuário. Considerado um tipo de *query drift*, o *outweighting* é caracterizado por ter os termos de expansão fortemente relacionados a termos específicos de uma consulta e não da consulta como um todo [2]. Por exemplo, em uma consulta "retornar insetos endófagos", podem ser adicionados termos como "borboleta", "grilo", "gafanhoto" e "besouro", ainda que nem todos os termos representem, necessariamente, insetos endófagos, mas apenas insetos.

O **relaxamento de consultas**, diferente das técnicas citadas anteriormente, não procura manter o significado exato de uma consulta. Seu objetivo consiste em generalizar uma consulta falha em uma bem sucedida, por meio da remoção de algumas sub-consultas da consulta original [27]. Em um sentido mais amplo de relaxamento, o objetivo pode ser a expansão do escopo de uma consulta pelo relaxamento de restrições envolvidas na consulta [5]. O problema desta técnica está na redução da especificidade da consulta, que resulta no atendimento incompleto do objetivo do usuário [25].

Apesar da aparente simplicidade de se remover um critério de uma consulta, a técnica baseada no relaxamento das consultas pode envolver diferentes abordagens. Como exemplo, pode-se aumentar ou diminuir o valor utilizado em uma comparação com operador "menor que" ou ampliar uma localização visada, como a de uma cidade, para uma mais abrangente, como a do estado no qual a cidade está inserida. Assim, formas de promover o relaxamento incluem: remover de partes da consulta, generalizar critérios, encontrar e eliminar falsas proposições e utilizar uma "relação de tolerância" para satisfazer consultas caracterizadas como *fuzzy* [5].

A relação de tolerância permite relaxar uma consulta que não tem um critério cujo valor é estritamente estabelecido. Por exemplo, uma consulta "retornar insetos grandes", pode ter como resultado *Phobaeticus kirbyi* e *Phobaeticus serratipes* [51] – conhecidas

como bichos-pau, cujas fêmeas podem medir, respectivamente, a 32,8cm e 27,8cm (54,8cm e 55,5cm, com as pernas esticadas) – mas também poderia incluir o *Titanus giganteus* [11], com tamanho de 20cm. Apesar de possuir cerca de 60% do tamanho da *Phobaeticus kirbyi*, este inseto é o maior besouro conhecido e um dos maiores insetos do mundo.

## 2.5 Conclusões

Este capítulo apresentou aspectos relacionados a Sistemas de Biodiversidade e outros conceitos que fundamentam esta dissertação: ontologias e serviços *Web*. Por fim, apresentou trabalhos correlatos ao tema central da dissertação, técnicas de processamento semântico de consulta.

Entre as técnicas estudadas foram encontradas diferentes formas de melhorar os resultados obtidos: a reescrita de uma consulta para um equivalente semântico, o enfraquecimento das restrições, a fim de obter um número maior de resultados, a substituição de termos por outros contextualmente relacionados, ainda que sem uma semântica equivalente, e a adição de novos termos a fim de restringir os resultados e aproximá-los do contexto pretendido pelo usuário.

Percebe-se que o papel de um termo dentro do seu domínio – seja pela compreensão do seu significado ou por seu contexto – é um recurso pertinente para melhorar os resultados de uma consulta. Dentro dessa concepção, uma das formas utilizadas para aperfeiçoar as consultas é a utilização de ontologias para representar semanticamente o conhecimento. O Capítulo 3 apresenta a proposta deste trabalho, inserida no conceito de Otimização Semântica de Consultas.

Ressalta-se que a revisão realizada não encontrou trabalhos em expansã ou pré-processamento de consulta, em sistemas de biodiversidade, que usam ontologias. As consultas, em tais sistemas, são baseadas em execução de expansões em SQL a bancos de dados relacionais. Além disso, não foram encontrados trabalhos que permitissem escolher e personalizar as operações que podem ser realizadas nos valores das consultas.

# Capítulo 3

# Especificação do Serviço de Expansão de Consultas

Este capítulo descreve a arquitetura do Serviço de Expansão de Consultas, cujo objetivo é obter resultados mais expressivos a partir da reformulação de consultas aplicadas a bases de dados envolvendo biodiversidade.

Ainda que a otimização de desempenho seja uma necessidade, este não é o objetivo do trabalho. Uma das prioridades é ajudar o usuário a obter informações de coleções cujos dados são escritos e estruturados de formas variadas e por pessoas diferentes, com lacunas frequentes.

A Seção 3.1 apresenta a visão geral da arquitetura proposta. As demais seções detalham os diferentes aspectos desta arquitetura.

## 3.1 Arquitetura do Serviço de Expansão de Consultas

O serviço de expansão de consultas visa permitir aos biólogos consultar informações de repositórios variados a fim de encontrar os registros procurados, detectar relações que não estão diretamente representadas e correlacionar os registros. Nesse processo, é preciso facilitar a especificação da consulta para que as características de cada repositório não limitem a abrangência do resultado e os dados coletados não sejam subutilizados.

Esta seção apresenta a arquitetura proposta e introduz, por meio de exemplos, o funcionamento do módulo de processamento de consultas. Este módulo utiliza as informações do domínio, representadas em ontologias, para contextualizar e desambigüizar os termos das consultas. O trabalho se integra ao BIO-CORE e aproveita os serviços do Aondê para dar suporte às consultas dos pesquisadores.

A Figura 3.1 apresenta a arquitetura para a expansão de consultas. As consultas recebidas pela aplicação cliente são passadas ao serviço de consultas, onde são verificadas e formatadas de acordo com um padrão definido para este trabalho. Após a padronização, as consultas passam por um processo de decomposição sintática da consulta e desambigüização dos termos por meio das ontologias acessadas usando o Aondê. As consultas assim estendidas são retornadas à aplicação cliente, que solicita sua execução por meio dos serviços *Web* que dão acesso às informações dos repositórios.

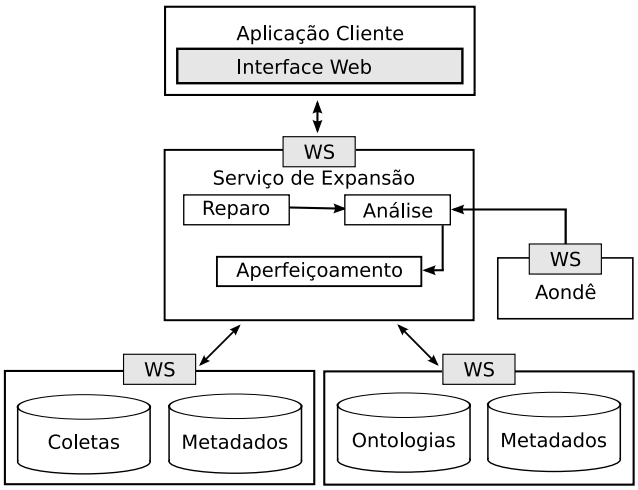



Figura 3.1: Arquitetura do Módulo de Processamento de Consulta

Os dados são armazenados nos repositórios de coletas e de ontologias. O primeiro contém registros de coletas realizadas por biólogos, enquanto o segundo mantém ontologias criadas por pesquisadores ou encontradas na Internet e que representam o consenso de grupos a respeito de informações de biodiversidade. Ambos repositórios possuem um conjunto de metadados associados, os quais têm como objetivo facilitar o gerenciamento e a recuperação de informações.

A expansão de uma consulta exige conhecimento prévio dos bancos de dados utilizados e das ontologias de domínio para a realização da expansão. Há, no mínimo, duas opções para isto:

- a cada consulta recebida, analisar os campos usando acesso ao SGBD e expandi-la por meio de acesso a ontologias;
- transformar o esquema das bases de dados em uma ontologia, que passa a fazer parte do repositório de ontologias, e expandir a consulta baseando-se apenas em manipulações de ontologias.

A dissertação optou pela segunda opção por três razões principais: I) Elimina a necessidade de consultar o esquema do banco de dados a cada expansão, evitando mais

acessos a ele; II) Facilita a consulta às informações do banco de dados e das ontologias de domínio, por utilizar um formato comum a ambos; III) Permite estabelecer relações entre os recursos do banco de dados e os das ontologias. Desta forma, a expansão de consultas passa a se basear apenas em ontologias. Para isto, a cada nova base de dados incorporada ao sistema, é necessário criar uma nova ontologia que descreve seu esquema - trata-se de uma etapa de configuração para permitir a expansão posterior. O trabalho distingue assim entre dois tipos de ontologia:

- ontologia de esquema – representa o esquema de um banco de dados conteúdo registros de coletas para consulta;
- ontologia de domínio – representa o conhecimento do domínio (por exemplo, informações taxonômicas ou ecológicas).

A Figura 3.2 ilustra a criação de uma ontologia de esquema. Nesta etapa, o módulo principal do *Serviço de Expansão* invoca o módulo de *Representação* (1), por meio do qual o esquema do banco de dados é acessado (2) transformado em uma ontologia de esquema (3), que é enviada para o Aondê (4) e armazenada no repositório semântico (5).

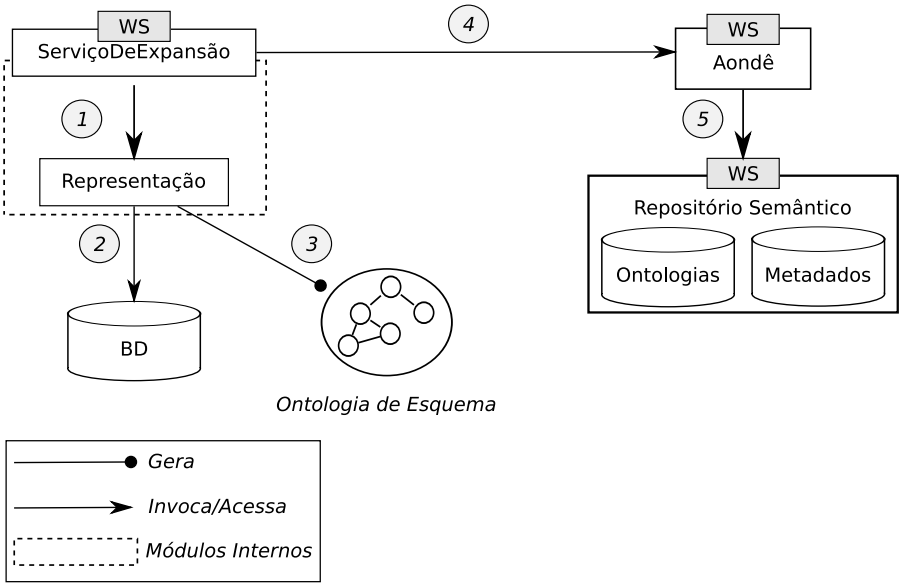



Figura 3.2: Representação do esquema do banco de dados para o formato de uma ontologia.

O módulo de representação transforma o esquema (tabelas, atributos e chaves estrangeiras) em uma ontologia. Isto elimina a necessidade de se fazer uma conexão com o banco de dados a cada consulta processada e permite consultar os recursos da base de

dados utilizando as mesmas formas de acesso já utilizadas nas ontologias, apresentadas no Capítulo 4.

Criada a representação do esquema do banco por uma ontologia, pode-se processar a reescrita da consulta. O processo pode ser acompanhado na Figura 3.3.

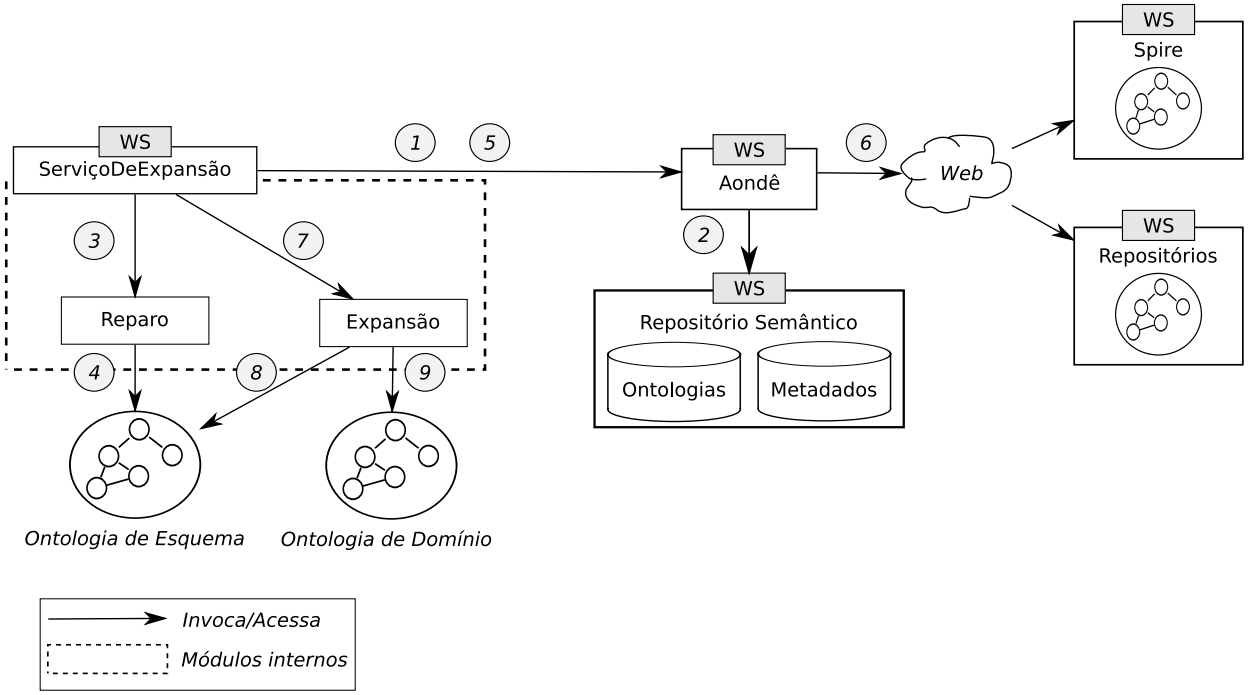



Figura 3.3: Fases do processamento de uma consulta.

O *Serviço de Expansão* solicita ao Aondê a ontologia do esquema do banco de dados (1) que é recuperada do repositório semântico (2). A ontologia é enviada para o módulo de *Reparo* (3), o qual identifica atributos e tabelas envolvidas para verificar se não ocorrem erros de digitação e de sintaxe. A sintaxe esperada é:

> *SELECT* {atributos}
> *FROM* {lista de tabelas separadas por vírgula}
> *WHERE* {predicado composto},
> onde predicado composto é formado por conjunções e/ou disjunções de predicados formados por <*operando operador valor*>ou <*operando operador operando*>.

Além disso, padroniza o formato da consulta SQL para facilitar sua expansão. Para tanto, a consulta é decomposta e cada termo envolvido é verificado na ontologia de esquema (4).

Terminado o processo de reparo, o *Serviço de Expansão* utiliza o Aondê para encontrar ontologias de domínio que contenham termos relacionados à consulta (5). Tais ontologias

podem estar registradas no repositório semântico ou em repositórios disponíveis na Web (6). As ontologias de esquema e de domínio são passadas para o módulo de *Expansão* (7). Este módulo decompõe a consulta e, para cada elemento, encontra o conceito ao qual ele se refere em ontologias do esquema (8) e procura uma equivalência em ontologias do domínio (9). Ao final do processo, a consulta expandida e retornada pelo *Serviço de Expansão*.

Em resumo, o pré-processamento de consultas tem duas fases: preparatória (criação da ontologia de esquema) e expansão. Esta última tem novamente duas fases: reparo e expansão semântica propriamente dita. As próximas seções apresentam os algoritmos implementados, conforme organização a seguir:

- Seção 3.2 – criação de uma ontologia de esquema - Algoritmo 3.1;
- Seção 3.3 – reparo de uma consulta – padronizar todos os nomes dos atributos em uma consulta, precedendo-os do nome da tabela associado;
  - Algoritmo 3.2 – concatena nome da tabela a um atributo, invocando o algoritmo 3.3 para correção de digitação;
  - Algoritmo 3.3 – busca termos na ontologia de esquema com nomes semelhantes a um operando de predicado (corrige erros de digitação);
- Seção 3.4 – expansão de consulta a partir de relacionamentos de instanciação, equivalência, especialização ou generalização – Algoritmos 3.4 a 3.10.

## 3.2 Representação de Esquemas de Banco de Dados em Ontologias

Este módulo cria uma representação do esquema do banco de dados sob forma de uma ontologia. Esta ontologia é empregada por outros módulos para consultar informações do banco de dados e para relacioná-las a ontologias do domínio.

Há vários exemplos de pesquisa (por exemplo [36, 37, 38] e [59]) que tratam do mapeamento entre ontologias de domínio e o banco de dados. Diferente desta dissertação, esquema do banco de dados não é representado como uma ontologia. Os trabalhos apresentam as relações por meio de um arquivo com regras de mapeamentos, que especificam qual conceito ou propriedade da ontologia corresponde a determinada tabela ou atributo do banco de dados. Já a dissertação cria uma ontologia específica para cada esquema de banco de dados a ser manipulado.

O módulo de criação de ontologia de esquema trata os seguintes aspectos (Algoritmo 3.1):

- Tabela: representada como classe (Linhas 14 e 15);
- Atributo: resulta em propriedades das ontologias (Linhas 17 a 26);
- Tipo de atributo: representado como o tipo equivalente pertencente à ontologia (Linha 26);
- Chave primária: tratado como atributo (Linhas 17 a 26);
- Chave estrangeira: resulta em relacionamentos (Linhas 22 e 28 a 32);
- Chave primária e estrangeira: representada como herança (Linhas 20 e 34 a 37);
- Chave composta: tem o resultado equivalente a atributo e chave estrangeira, conforme o caso tratado.

O objetivo é manter as informações referentes ao esquema do banco de dados; Os dados do banco de dados não são representados na ontologia. Com isso, o tratamento realizado para chaves compostas é simplificado.

Algoritmo 3.1: Representar Esquema do Banco de Dados em Ontologia

```
1  GerarRepresentacao(BD)
2
3  Entrada:
4    BD: BancoDeDados
5
6  Saída
7    OE: Ontologia do Esquema do Banco de Dados
8
9  seja OE uma nova ontologia
10
11 seja LEsp uma lista vazia
12 seja LRel uma lista vazia
13
14 para cada tabela T de BD
15   criar classe C com nome de T e adicionar a OE
16
17   para cada atributo A de T
18     se A é chave estrangeira
19       se A é chave primária
20         LEsp = LEsp U <C, nome da tabela da chave estrangeira de T>
21       senão
```

```
22          LRel = LRel U <C, nome da tabela da chave estrangeira de T,
                A>
23      senão
24        criar propriedade P com tipo de dado
25        dominio(P) = classe de OE que representa T
26        imagem(P) = tipo de dado XMLSchema equivalente ao tipo de A
27
28    para cada elemento E de LRel
29      criar propriedade P com o tipo relacionamento
30      nome(P) = A
31      dominio(P) = C
32      imagem(P) = classe de OE que representa a tabela da chave
            estrangeira de T
33
34    para cada elemento E de LEsp
35      criar H relacionamento de herança
36      classe(H) = classe de OE que representa a tabela da chave
            estrangeira de T
37      subclasse(H) = C
38
39  retornar OE
```

Caso o esquema do banco de dados seja atualizado, é possível fazer uma nova invocação ao módulo e atualizar o arquivo. Assim, a ontologia refletirá o estado mais recente do esquema.

## 3.3 Reparo de Consultas

O objetivo deste módulo é padronizar e pré-processar consultas em SQL simplificando o algoritmo de expansão. Este considera que as consultas estejam de acordo com a regra:

> os nomes dos atributos são precedidos do nome da tabela correspondente,

ainda que exista apenas uma tabela na consulta ou que não exista ambiguidade. Por exemplo, a consulta sintaticamente correta em SQL:

```
1   SELECT genero FROM Colecao WHERE ordem = 'lepidoptera'
```

é transformada em:

```
1   SELECT Colecao.genero FROM Colecao WHERE Colecao.ordem =
        'lepidoptera'
```

A finalidade deste módulo é eliminar restrições para a entrada do módulo de expansão e mantê-lo robusto às possibilidades de uso. Assim, além de padronizar as consultas para que atendam à regra especificada, o módulo utiliza as informações das ontologias de esquema para corrigir nomes com erros de digitação, mas que possam ser identificados por terem um alto grau de similaridade com alguma tabela ou atributo existente no banco.

Algoritmo 3.2: Reparar Consulta

```
1  RepararConsulta(Q, OE)
2
3  Entrada:
4    Q: Consulta
5    OE: Ontologia do Esquema do Banco de Dados
6
7  Saída:
8    Q': Consulta Q reparada
9
10 Q' = Q
11
12 obter lista LC de classes da ontologia OE
13
14 seja LF uma lista vazia
15
16 para cada tabela T da clausula FROM de Q
17   obter classe C de OE com label idêntico ao nome de T
18   se C não existir
19     C' = ObterElementoSimilar(LC, label de C)
20     se C' for diferente de nulo
21       Substituir referências a C por C' em Q'
22     senão
23       abortar
24   LF = LF U {C}
25
26 obter lista L de propriedades da OE
27
28 para cada predicado Pr da consulta do tipo: <operando, operador,
     operando>
29   se operando O de Pr é um atributo e não possui tabela definida
30     se não há propriedade P com label idêntico a O em OE
31       P' = ObterElementoSimilar(L, label de P)
32       Se P' for diferente de nulo
33         Substituir referências a P por P' em Q'
```

```
34        senão
35          abortar
36      LCP a lista de classes de OE que possuem P
37
38      se LCP possui apenas uma classe
39          associar a tabela correspondente a O e à clausula FROM de Q'
40      senão
41        se há mais de uma classe em LCP e LCP interseção LF possuir
              possuir apenas um elemento
42          associar tabela ao operando em Q'
43        senão
44          abortar
45
46  retornar Q'
```

O Algoritmo 3.2 analisa os predicados (Linhas 28 a 44) obtidos pela decomposição da consulta para verificar sua validade de acordo com a ontologia que representa o esquema do banco de dados. Cada predicado é formado por um conjunto de operandos, recuperados na linha 5, os quais podem ser um atributo, um valor ou um operador.

O método para reparo de consultas, Algoritmo 3.2, realiza as ações:

- Verifica inicialmente a existência das tabelas fonte (clausula FROM) (Linhas 16 a 24);
- Verifica e corrige, se necessário, a definição da tabela que contém os atributos utilizados na consulta (Linhas de 28 a 43);
- Elimina parte dos casos de ambiguidade em que há duas tabelas que podem ser associadas a um atributo (Linhas de 41 e 42);
- Verifica a existência das colunas e das tabelas, referenciadas na consulta, na ontologia do esquema banco de dados (Linha 16 a 24 e de 31 a 35);
- Corrige os nomes de atributos e tabelas que não existam no banco de dados mas que tenham grau de similaridade próximo ao de campos e tabelas existentes (Linhas 19 a 21 e 30 a 33).

O Algoritmo 3.3 verifica se há um termo da ontologia que atenda a um grau mínimo de similaridade com um atributo ou tabela da consulta. Este algoritmo recebe uma lista de nomes de elementos e obtém o número que indica a proximidade de cada valor para o nome do atributo ou da tabela, obtido pelo resultado da execução algoritmo que implementa a técnica de comparação de distância de *strings* de Jaro [10], que leva em consideração o

número de caracteres em comum e a ordem deles. A partir dos valores obtidos, verifica qual elemento da lista possui maior similaridade com o elemento buscado (Linhas 8 a 10). Se este elemento tiver o grau mínimo exigido é retornado como sendo o de valor correto (Linhas 11 e 12). Caso contrário, retorna-se um elemento nulo (Linhas 13 e 14), indicando que não há elemento similar na ontologia de esquema.

Algoritmo 3.3: Obter Elemento Similar

```
1  ObterElementoSimilar(L, Lr)
2  Entrada:
3    L: Lista de strings
4    Lr: Literal
5  Saída:
6    E': Elemento com maior grau de similaridade de string com Lr
7
8  para cada elemento E em L
9    calcular similaridade de string entre E e Lr
10 obter elemento E' com maior grau de similaridade
11 se E' tiver grau de similaridade superior a um valor mínimo
12   retornar E'
13 senão
14   retornar nulo
```

## 3.4 Expansão de Consultas

O módulo de expansão de consultas deve trabalhar com três recursos: i) consulta a ser expandida, ii) ontologia de esquema e iii) ontologias de domínio. Neste trabalho foram adotadas duas abordagens que têm como intuito tornar mais flexível a utilização do Serviço de Expansão de Consulta. A primeira consiste em permitir que o Aondê encontre ontologias de domínio que podem ser usadas para realizar as expansões de consulta. A segunda consiste em permitir que o usuário escolha quais expansões serão realizadas na consulta. O Algoritmo 3.4 é responsável por invocar as operações para selecionar uma ontologia (Linha 11) e distribuir operações que devem ser realizadas na consulta (Linha 12).

Algoritmo 3.4: Expandir

```
1  Expandir(R, Q, LOp) : Consulta
2
3  Entrada:
4    R: Repositorio
5    Q: Consulta Reparada
```

```
6    LOp: Lista de Operacoes das Consultas (indica o tipo de expansão
         ontológica para cada cláusula do predicado)
7
8  Saída:
9    Q': Consulta Expandida
10
11 OE = SelecionarOntologia(Q) /* Algoritmo 3.5 */
12 D = MontarTabelaDeOperacoes(Q, LOp) /* Algoritmo 3.6 */
13
14 retornar ExpandirConsulta(OE, OD, Q, D) /* Algoritmos 3.7 a 3.10 */
```

### 3.4.1 Formato de Entrada

As expansões que podem ser realizadas nas consultas consistem em:

- Especializações: utilizar subclasses de uma classe, preservando a semântica da consulta;
- Generalizações: adicionar superclasses de uma classe, como forma de obter resultados próximos ao original;
- Instanciação: incluir instâncias de uma classe, mantendo o significado da consulta;
- Equivalência: utilizar classes relacionadas por axiomas de equivalência, sem alterar a semântica da consulta.

Cada cláusula do tipo <operando, operador, operando>do predicado de uma consulta é expandida de acordo com a operação de expansão especificada para aqueles operandos. A Tabela 3.1 mostra as operações que podem ser realizadas.

Em mais detalhes, cada consulta é acompanhada pela especificação dos tipos de expansão por cláusula, conforme a seguinte expressão regular:

(((CAMPO=)?VALOR:((OPERAÇÃO(=NÍVEL)?),)+);)*

onde *CAMPO* é um nome de atributo, *VALOR* é um valor, *OPERAÇÃO* $\in$ {esp, gen, ins, equ, sem, todas} e NÍVEL um inteiro.

Se operação for hierárquica, como especialização ou generalização, *NÍVEL* indica a profundidade da busca. Caso deseje, o usuário pode atribuir o valor "*" para especificar as operações para todos os valores da consulta. Porém, caso ele utilize o valor "*" e especifique um ou mais termos com operações específicas, os termos especificamente tratados não incluirão as operações definidas para todos os demais.

Por exemplo, para a consulta:

| Operação | Descrição |
|---|---|
| esp | Subclasses do termo expandido |
| gen | Superclasses do termo expandido |
| ins | Instâncias do termo expandido |
| equ | Equivalências do termo expandido |
| sem | Todas as operações que não resultam em perda de semântica |
| todas | Todas as operações disponíveis |

Tabela 3.1: Tabela de Operações

```
1  SELECT *
2  FROM Colecao
3  WHERE Colecao.classe = 'insecta'
4  AND Colecao.estagioDeVida = 'larval'
5  OR Colecao.order = 'lepidoptera'
```

é possível definir as operações:

lepidoptera:sem=2;Colecao.classe=insecta:gen=2;Colecao.classe=*:todas=3;

A expressão "lepidoptera:sem=2" indica que o valor *lepidoptera* na consulta pode ser expandido para englobar todas as possibilidades relacionadas a lepidoptera, que não impliquem na perda de semântica (especialização, equivalência e instanciação), em até 2 níveis. A expressão "Coleção.classe=insecta:gen=2" indica que o predicado "Coleção.classe=insecta" deve ser generalizado para o nível superior. Finalmente, a expressão "Colecao.classe=*:todas=3" indica que todos os predicados do tipo "Colecao.classe=operando" devem ser expandidas de todas as formas possíveis (operação todas), à exceção de insecta.

Assim, por exemplo, "lepidoptera:sem=2" corresponde a substituir "Colecao.ordem=lepidoptera" por "Colecao.ordem='lepidoptera' OR Colecao.superfamilia IN ('Acanthopteroctetoidea', 'Alucitoidea', 'Bombycoidea', 'Papilionoidea', ...) OR Colecao.familia IN ('Acanthopteroctetidae', 'Lycaenidae', 'Nymphalidae', 'Papilionidae', 'Pieridae', 'Riodinidae', ...)", dentre outras expansões, onde o conjunto de valores associado a *Colecao.superfamília* é especialização de nível 1 de *lepidoptera*, o conjunto de valores associado a *Colecao.familia* é especialização de nível 2 de *lepidoptera* e assim por diante.

### 3.4.2 Selecionar Ontologia

A seleção da ontologia do domínio a ser utilizada é realizada pelo Algoritmo 3.5. Este algoritmo cria uma lista contendo os termos presentes na consulta e os submete à operação de Busca com *Ranking* do Aondê (Linha 12). Além dos termos da consulta, a operação recebe os parâmetros relacionados à busca. Ao final é retornada a ontologia de domínio que possui maior número de elementos presentes na ontologia a ser utilizada na expansão da consulta. Alternativamente, é possível usar a operação de busca para cada termo da consulta e realizar a integração entre as ontologias. Entre os trabalhos futuros, apresentados no Capítulo 5 está o estudo e aperfeiçoamento da técnica de seleção de ontologias de domínio para a expansão.

Algoritmo 3.5: Selecionar Ontologia de Dominio

```
1  SelecionarOntologia(Q): OD
2
3  Entrada:
4    Q: Consulta
5  Saída:
6    OD: Ontologia do Domínio
7
8  Seja L uma lista vazia
9  para cada Te termo de Q
10   L = L U Te
11
12 Ranking = Aonde.SearchRank(L, match, density, betweenness,
       semanticSimilarity, {"Aonde", "Swoogle"})
13
14 retornar Ranking[1] //ontologia que contém maior correspondência com
       elementos
```

### 3.4.3 Montar Tabela de Operações

Para facilitar o reconhecimento das operações que devem ser aplicadas à consulta foi criado o Algoritmo 3.6. O algoritmo armazena as operações sobre cada valor da consulta em uma estrutura de dados que usa tabelas *hash*. A estrutura básica da tabela (Figura 3.4) é formada por tabelas aninhadas que contêm, respectivamente, informações sobre campo, valor, operação e nível.

Para o conjunto de parâmetros:

lepidoptera:sem=2;Colecao.classe=insecta:gen=1;Colecao.classe=*:todas=4;

o resultado do algoritmo é a tabela da Figura 3.5. Nela, existem duas referências a campos: *Colecao.classe* e ***. Dentro de cada um deles há uma referência para uma tabela *hash* que especifica quais valores e operações poderão ser realizados para ele, dentro da consulta.

A tabela referenciada por *Colecao.classe* possui como valores *insecta* e ***. O primeiro contém o valor *gen*, associado ao nível *1*, indicando que a consulta poderá expandir *Colecao.classe='insecta'* utilizando 1 nível de generalização. O segundo valor, que representa qualquer valor que possa ocorrer, indica que é possível realizar generalizações, especializações, instanciações e equivalências em até 4 níveis para qualquer valor que esteja associado a *Colecao.classe*.

Pelo fato de existir uma regra específica para o valor 'insecta' e outra que pode ser considerada para qualquer valor, o algoritmo de expansão (3.7) dá maior prioridade à regra específica. Dessa forma, é possível especificar regras gerais para valores variados e atribuir regras para campos, como forma de restringir as operações para casos específicos.

A definição de um valor * para o campo possui a mesma interpretação que para o valor. É possível definir quais regras serão aplicadas a um determinado campo e atribuir um conjunto de operações para os demais campos. No exemplo dado, se a consulta apresentar algum predicado que não esteja associado a *Colecao.classe* e contenha o valor *lepidoptera*, serão realizadas as operações de especialização, instanciação e equivalência com até 2 níveis.

Algoritmo 3.6: Montar Tabela de Operações

```
1  MontarTabelaDeOperacoes(Q, LOp): D
2
3  Entrada:
4    Q: Consulta
5    LOp: String com operacoes da forma:
         (((CAMPO=)?VALOR:((OPERACAO(=NIVEL)?),)+);)*
6  Saída:
7    D: Tabela de Operações (tabela hash indicando operações a serem
         executadas sobre cada coluna)
8
9  todasOperacoes = {"esp", "gen", "ins", "equ"}
10 semOperacoes = {"esp", "equ", "ins"}
11
12 HashTable D
13
14 para cada elemento E em LOp do tipo <campo=valor:operacao=nivel>
15   E1 = E.<campo=valor>
16   HashTable D1
17   para cada elemento E2 em E1 do tipo <operacao=nivel>
```

```
18      se nivel for vazio
19        D1[E2.operacao] = 0
20      se não
21        D1[E2.operacao] = nivel
22
23    se D1 tem chave "sem"
24      para cada elemento E2 em sem
25        se D1 não tem chave E2
26          D1[E2] = D1["sem"]
27
28    se D1 tem chave "todas"
29      para cada elemento E2 em todas
30        se D1 não tem chave E2
31          D1[E2] = D1["todas"]
32
33    se E1.campo é vazio
34      E1.campo = "*"
35
36    D[E1.campo] = D1
37
38  retorne D
```

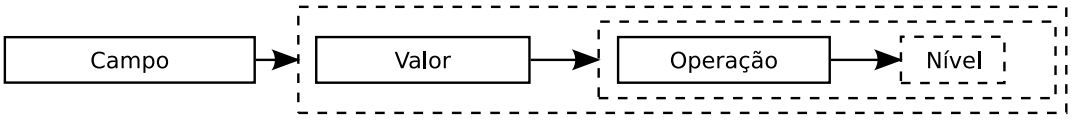



Figura 3.4: Estrutura criada para armazenar operações da consulta usando Tabelas Hash

O algoritmo recebe as operações que podem ser realizadas na consulta de forma serializada, em uma *string*, e organiza os valores criando uma chave para cada termo da consulta e associando uma outra tabela *hash* que possui como chave a operação que pode ser realizada e os níveis que podem ser incluídos na expansão. Ao final, a tabela de distribuição das chaves é retornada pelo algoritmo.

### 3.4.4 Expandir Consulta

Após os processos de seleção da ontologia do domínio e distribuição das operações de expansão o Algoritmo 3.4 invoca o método de ExpandirConsulta (Algoritmo 3.7). Este algoritmo é o responsável por analisar a consulta e realizar as expansões possíveis.

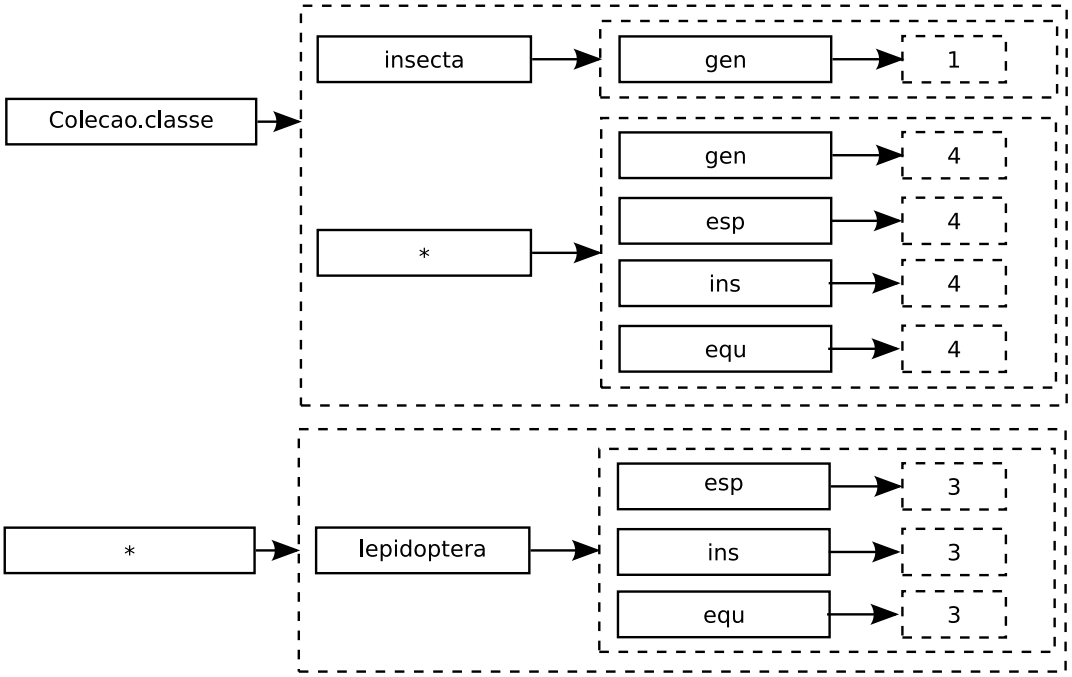



Figura 3.5: Exemplo de organização das operações relacionadas a uma consulta.

Algoritmo 3.7: Expandir Consulta

```
1  ExpandirConsulta(OE, OD, Q, D)
2
3  Entrada:
4    OE: Ontologia do Esquema do Banco de Dados
5    OD: Ontologia de Domínio
6    Q: Consulta
7    D: Tabela de Operações
8
9  Saída:
10   Q': Consulta Q Expandida
11
12 seja Pr um predicado formado pela tupla: <operando, operador, valor>
13
14 Q' = Q
15
16 para cada predicado Pr de Q
17   para cada valor V comparado em Pr
18     se existe classe C com label idêntico em OD
19       ExpansãoPorClasse(C, OD, D)
```

```
20        atualizar Q' com os termos retornados
21
22      se existe propriedade P com label idêntico em OD
23        encontrar termos para expansão por propriedade para P
24        atualizar Q' com termos
25
26  retornar Q'
```

O Algoritmo 3.7 realiza a decomposição da consulta e a análise dos elementos encontrados, expandindo os termos conforme indicado na tabela de operações. O processo de expansão por classe (Algoritmo 3.8) encontra o conjunto de elementos da ontologia que podem ser usados em substituição ou adicionalmente às classes. Este algoritmo chama repetidas vezes o algoritmo *ContemChaves* (3.9) cujo objetivo é navegar na tabela de operações.

Algoritmo 3.8: Expansão por Classe

```
1   ExpansaoPorClasse(C, OD, D)
2
3   Entrada
4     C: C
5     OD: Ontologia de Domínio
6     D: Tabela de Operações
7
8   Saída
9     L: Lista de classes com a mesma semântica de C
10
11  dado um termo Te identificado como uma classe C em OD
12
13  criar lista L de classes com a mesma semântica de C
14
15  N = ContemChave(label de L, "equ", D)
16  se N >= 0
17    para cada classe C associada a Te com axioma de equivalência
18      adicione à lista seu label
19      L = L U (label de C)
20
21  N = ContemChave(label de L, "esp", D)
22  se N >= 0
23    se N = 0
24      nivel = OD.numeroDeClasses
25    senão
```

```
26        nivel = N
27    L = ObterSubclasses(C, OD, L, nivel)
28
29  N = ContemChave(label de L, "ins", D)
30  se N >= 0
31    para cada classe C em L
32      L = L U (indivíduo de C)
33
34  N = ContemChave(label de L, "gen", D)
35  se N >= 0
36    se N = 0
37      nivel = OD.numeroDeClasses
38    senão
39      nivel = N
40    L = ObterSuperclasses(C, OD, L, nivel)
41
42  retornar lista
```

Inicialmente, calculam-se as equivalências (Linhas 16 a 19). A seguir, no momento em que subclasses (Linhas 21 a 27) e superclasses (Linhas 34 a 40) são recuperadas, pode-se realizar invocações a métodos recursivos de obter subclasses (Algoritmo 3.10) e superclasses (omitido, idêntico ao Algoritmo 3.10, para generalização). Dessa forma, é ampliado o espaço de valores que pode ser utilizado para expandir a consulta.

Para cada elemento, verifica-se a existência de uma chave *hash* para a operação (Linhas 15, 21, 29 e 34), a qual identifica se o termo deve ser expandido usando o tipo correspondente de elemento (subclasse, equivalência, etc.). A chave é verificada utilizando o Algoritmo 3.9, responsável por simplificar o reconhecimento de que há uma chave específica para o termo ou se a operação a ser realizada é aplicável a todos os termos da consulta.

Algoritmo 3.9: Contém chave

```
1  ContemChave(campo, valor, operacao, D)
2  Entrada:
3    campo: campo associado ao valor
4    valor: valor a expandir
5    operacao: operacao a ser realizada
6    D: Tabela de Operações
7  Saída
8    N: Nível, extraído da tabela de operações
9
10 se existe D[campo]
```

```
11    D = D[campo]
12  senão
13    se existe D["*"]
14      D = D["*"]
15    senão
16      retornar -1
17
18  se existe D[valor]
19    D1 = D[valor]
20    se existe D1[operacao]
21      retornar D1[operacao]
22    senão
23      D1 = D["*"]
24      se existe D1[operacao]
25        retornar D1[operacao]
26      senão
27        retornar -1
```

A recuperação dos níveis de subclasses é feita pelo Algoritmo 3.10, e os níveis de generalizações são realizados de forma análoga. Assim, para uma determinada classe são recuperados seus primeiros filhos e destes os próximos, até que os níveis de expansão solicitados pelo usuário sejam atendidos. O processo ocorre como uma busca em largura.

Algoritmo 3.10: Obter Subclasses

```
1  ObterSubclasses(C, OD, L, nivel)
2
3  Entrada
4    C: Classe
5    OD: Ontologia do Dominio
6    nivel: nivel de especializacao a ser recuperado
7    L: lista de classes com mesma semantica de C
8  Saída
9    L
10
11 nivel = nivel - 1
12 para cada subclasse Sc de C em OD
13   L = L U {Sc}
14   se nivel >= 0
15     L = ObterSubclasses(Sc, OD, L, nivel)
16 retornar L
```

## 3.5 Cenários de Expansão

A capacidade de transformar consultas depende de um conjunto de fatores, dentre os quais a natureza e a complexidade das informações do banco de dados, a abrangência e o grau de detalhes das ontologias relacionadas ao domínio e o conhecimento que se tem da correspondência entre os elementos do banco de dados e das ontologias. Dado esse conjunto de fatores, as próximas seções descrevem dois cenários de atuação do processador de consultas. Nestes cenários, ontologia do esquema e ontologia do domínio:

- Não apresentam qualquer ligação;
- São mapeados de forma interligada, com ligações especificadas por especialistas do domínio, através de um editor de ontologias ou interface criada para este propósito. Tais ligações não poderiam ser facilmente inferidas.

Para facilitar o entendimento, as consultas usadas como exemplo não usarão o padrão adotado (descrito na Seção 3.3).

### 3.5.1 Cenário 1: Ausência de mapeamento entre banco de dados e domínio

No processo de expansão de consulta, se não há correspondência entre os elementos da ontologia do esquema usado e as ontologias de domínio, as expansões possíveis podem ser limitadas. O reconhecimento de que um elemento do esquema faz parte de uma ontologia do domínio é restrita às formas sintáticas. Dessa forma, se há casos de elementos com mesma semântica mas escritos de formas diferentes, o algoritmo de expansão não poderá reconhecê-la sem auxílio de um recurso externo, como um especialista ou um dicionário. A Figura 3.6 apresenta um exemplo de cenário descrito. Na figura, classes estão em branco e propriedades estão em cinza. A ontologia do esquema tem uma classe chamada *Coleção*, com propriedades *id*, *ordem*, etc. A ontologia de domínio é uma ontologia taxonômica, com classes *lepidoptera*, *papilionoidea*, etc. A classe *lepidoptera*, tem propriedades *antena*, *coloração*, e assim por diante. No entanto, as ontologias não estão alinhadas. Não há nenhuma ligação entre as ontologias que permita identificar que *lepidoptera* é uma *ordem*. Assim, não é possível identificar qual a relação entre cada elemento do esquema do banco de dados e os elementos da ontologia, situação que impede que sejam feitas modificações envolvendo campos das consultas. Por exemplo, considere a seguinte consulta aplicada à Tabela 3.2:

```
1 SELECT * FROM Colecao WHERE superfamilia = 'Papilionoidea'
```

| id | ordem | superfamilia | familia | antena | coloracao |
|---|---|---|---|---|---|
| 1 | Lepidoptera | | Papilionidae | | |
| 2 | Lepidoptera | | Nymphalidae | clavada | colorida |
| 3 | Lepidoptera | | Micropterigidae | | ”retilíneo” |

Tabela 3.2: Tabela de banco de dados com registros de insetos.

A Tabela 3.2 ilustra ser comum que algumas informações não sejam registradas no banco de dados. Apesar de as demais informações serem suficientes para identificar registros que tenham como *superfamilia* o valor *Papilionoidea*, a consulta SQL do usuário não será capaz de fazer este reconhecimento. Para tanto, pode-se usar as informações de uma ontologia, como a da Figura 3.6.

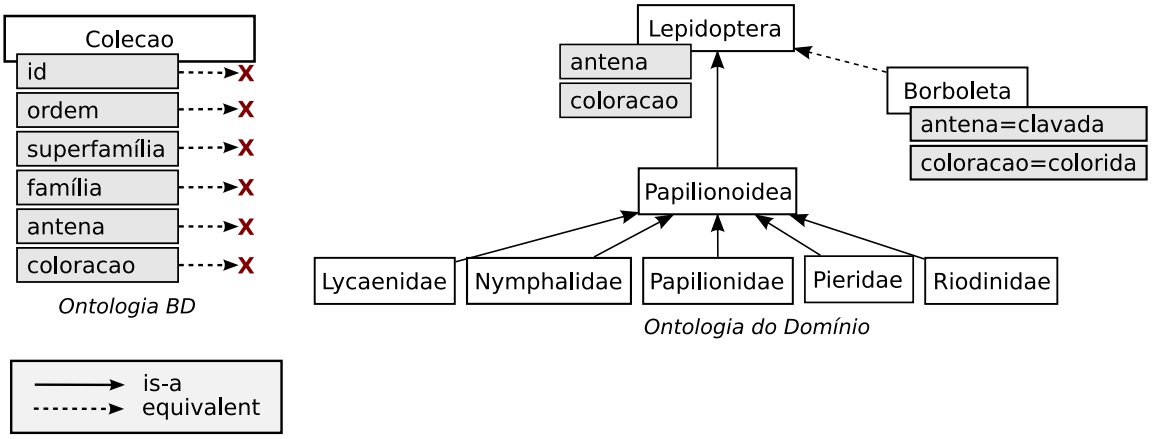



Figura 3.6: Relação entre as ontologias do banco de dados e do domínio.

Não há um mapeamento claro entre o significado do atributo *superfamilia* do banco de dados e o conceito *Papilionoidea* da ontologia de esquema. Assim, não é possível determinar que o atributo *familia* da mesma tabela consiste numa especialização de classe, de acordo com o conceito associado na ontologia e que esse significado é mantido no banco.

A inferência que seria possível a partir desse cenário consiste em identificar que o valor *Papilionoidea*, no predicado da consulta, está presente em uma ontologia referente a insetos. A partir do conceito de nome idêntico, é possível buscar outros conceitos associados na ontologia de domínio, para expandir a consulta.

Assim, ao encontrar um operando que seja sintaticamente idêntico a um elemento de uma ontologia, é possível aplicar o algoritmo de expansão apropriado ao tipo de elemento encontrado. Por exemplo, caso o elemento encontrado na ontologia seja uma classe, aplica-se o Algoritmo 3.8 de expansão por classe.

O processamento da consulta dentro das restrições de processamento do primeiro cenário resultaria na seguinte consulta:

```
1 SELECT *
2 FROM Colecao
3 WHERE superfamilia IN ('papilionoidea', 'lycaenidae', 'nymphalidae',
    'papilionidae', 'pieridae', 'riodinidae')
```

Com o resultado apresentado, o espaço de valores atribuído a *superfamilia* seria expandido para as diferentes possibilidades de especializações e equivalências da classe *papilionoidea*, como nomes de família e suas subclasses ou equivalências, caso estejam representadas na ontologia. Neste caso, os valores adicionados são aqueles que preservam a semântica do valor originalmente atribuído. A Tabela 3.3 apresenta o resultado da consulta expandida.

| id | ordem | superfamilia | familia | antena | coloracao |
|---|---|---|---|---|---|
| 1 | Lepidoptera | | Papilionidae | | |
| 2 | Lepidoptera | | Nymphalidae | clavada | colorida |

Tabela 3.3: Registros retornados com a execução da consulta expandida.

Apesar de ser possível identificar os casos em que o nome de um campo é idêntico ao nome existente na ontologia e, a partir disso, observar a relação desse campo com os demais, de acordo com a ontologia, nem sempre os nomes são escritos de forma igual. Por exemplo, o campo *superfamilia* poderia estar escrito como 'super_familia' ou 'superfamily', além da possibilidade de existir uma palavra homônima. Com isso, a confiabilidade de uma transformação seria baixa e, possivelmente, inapropriada para um serviço de transformação de consulta, quando o usuário não informa os alinhamentos desejados.

### 3.5.2 Cenário 2: Expansão com alinhamento manual

Se especialistas do domínio e do banco de dados especificam a relação entre cada elemento do banco de dados e da ontologia, é possível reescrever as consultas sem comprometer o significado da consulta e seus predicados. A Figura 3.7 apresenta a Figura 3.6 após os especialistas realizarem alinhamento manual das ontologias.

Como mostra a figura, é possível determinar, por exemplo, que *lepidoptera* é uma *ordem*, pois o usuário especificou um mapeamento entre *ordem* e *lepidoptera*.

Cada associação explicitamente definida entre os elementos das ontologias permite que os algoritmos de expansão de classe e de propriedade identifiquem os elementos que

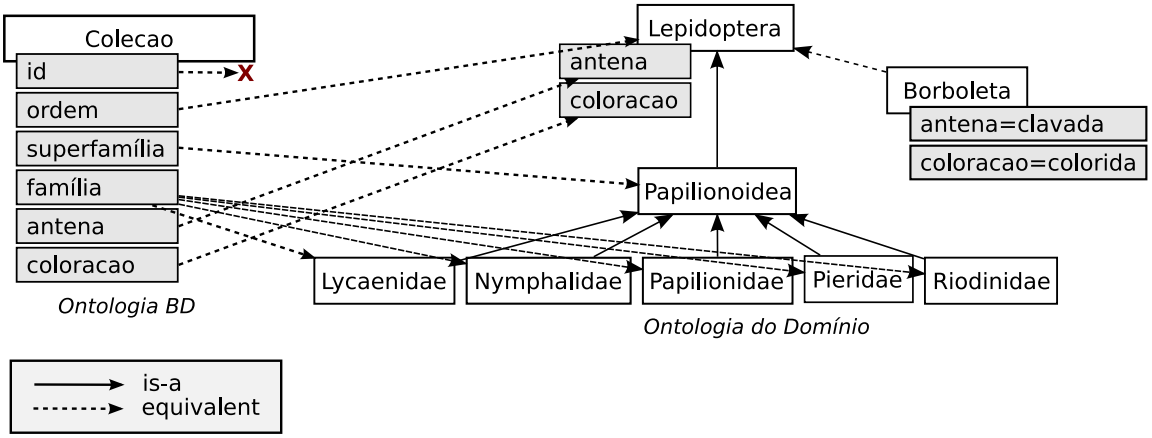



Figura 3.7: Relação entre as ontologias do banco de dados e do domínio determinada por especialistas.

possuem o mesmo significado. Assim, as expansões de consulta podem ser realizadas de modo independente da existência de uma similaridade sintática entre os elementos.

Quanto mais relações forem especificadas, maiores são as possibilidades de expansão. Ao analisar que *papilionoidea* é uma especialização de *lepidoptera*, infere-se que o uso do segundo no lugar do primeiro não resulta na perda de semântica.

Considere, por exemplo, a consulta:

```
1 SELECT * FROM Colecao WHERE ordem = 'lepidoptera'
```

Como resultado da transformação, utilizando o mapeamento entre as ontologias especificado na Figura 3.7, pode-se obter:

```
1 SELECT *
2 FROM Colecao
3 WHERE ordem = 'lepidoptera'
4   OR superfamilia = 'papilionoidea'
5   OR familia in ('lycaenidae', 'nymphalidae', 'papilionidae ',
      'pieridae', 'riodinidae')
```

### 3.5.3 Mapeamento entre ontologias com alinhamento do Aondê

Conforme apresentado na seção 3.5.1, as expansões que podem ser feitas quando banco de dados e ontologias estão dissociados são limitadas. Uma das formas previstas para contornar o problema consiste na utilização do Aondê para realizar um mapeamento automático entre os elementos das ontologias.

O Aondê realiza a integração de ontologias utilizando alinhamento. As ontologias originais são preservadas e uma nova ontologia é criada, especificando a correspondência que há entre os elementos das ontologias originais. São realizados dois tipos de comparação para determinar se dois elementos das ontologias representam o mesmo conceito: de elemento (parâmetro $\alpha$) e de estrutura (parâmetro $\beta$).

A comparação de elemento considera: a similaridade sintática dos nomes dos elementos e o fato de eles serem considerados sinônimos ou não. A análise de similaridade sintática usa algoritmos de comparação de *string*. Para verificar se os nomes dos elementos são sinônimos, o Aondê utiliza um dicionário de termos. Ao final da análise é criado um valor $\alpha$ que indica a similaridade, ou grau de confiabilidade, entre os elementos comparados.

A similaridade de estrutura do Aondê analisa as propriedades, os axiomas, as superclasses e as subclasses dos elementos comparados. O resultado de cada uma dessas análises possui valores entre 0 e 1 e recebe o peso 0.25 no total da similaridade de estrutura. A soma dos valores das análises resulta num valor entre 0 e 1, e é atribuído a $\beta$.

A partir de $\alpha$ e $\beta$, define-se a similaridade entre dois conceitos de ontologias. Por exemplo, pode-se definir que a similaridade de elemento tem peso de 40% na comparação enquanto a de estrutura tem 60%. Dessa forma, o grau de confiabilidade de semelhança entre dois elementos seria obtido por: $0.4 * \alpha + 0.6 * \beta$.

Com o Aondê é possível realizar o mapeamento entre as ontologias de esquema do banco de dados e as de domínio. Posteriormente, pode-se refinar os mapeamentos manualmente com o auxílio de um especialista. Neste caso, podem ser definidos os relacionamentos que não puderam ser reconhecidos com as técnicas de comparação de elemento e de estrutura.

## 3.6 Integração com o Serviço de Coletas

Um cenário previsto para a utilização do processador de consulta consiste na sua atuação como Serviço de Expansão de Consultas e no seu uso em conjunto com o Serviço de Coletas. Nesse cenário, as consultas sobre a base de dados passam por uma etapa adicional de processamento. A Figura 3.8 apresenta o fluxo de execução de uma consulta envolvendo a interação entre os serviços de Coleta e de Expansão.

Inicialmente uma aplicação Cliente envia a solicitação de consulta de um usuário ao Serviço de Coletas (1). Esta consulta pode ser ou não expandida, segundo indicação do usuário. Caso não seja expandida, o serviço disponibiliza uma conexão com o Repositório de Coletas, executa a consulta (2) e recebe um resultado (3), os dados são retornados à aplicação cliente (12). Caso contrário, o serviço de Coletas encaminha uma solicitação para o Serviço de Expansão de Consultas (4) que utiliza ontologias para reescrever a consulta (5). Essas ontologias são gerenciadas Aondê [15]. A consulta reescrita é enviada

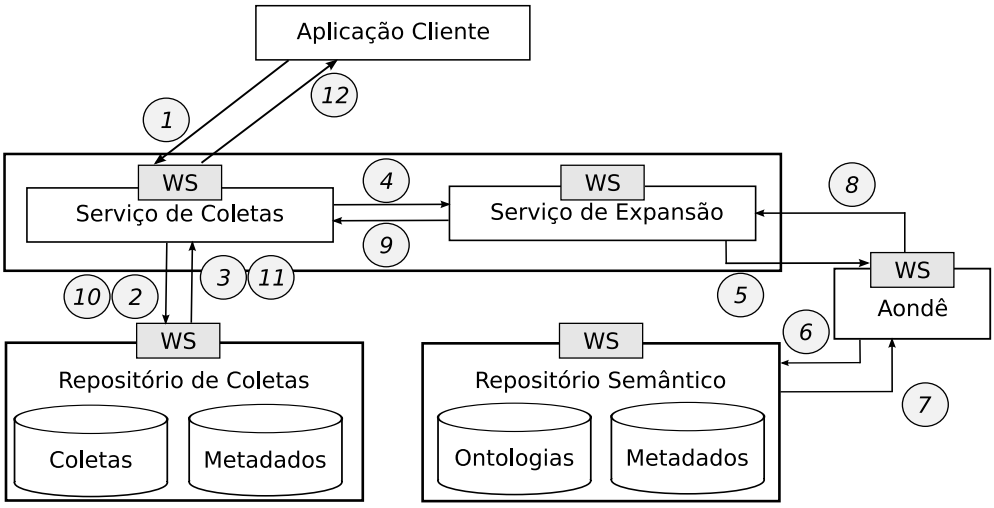



Figura 3.8: Arquitetura do Módulo de Processamento de Consulta

de volta ao serviço de Coletas (9), que finalmente executa a consulta (10) e retorna o resultado ao cliente (11,12). Mais detalhes sobre esta integração estão na dissertação de Malaverri [28].

## 3.7 Conclusões

Este capítulo apresentou a especificação da arquitetura e dos módulos propostos para a expansão de consultas. O Capítulo 4 apresentará os aspectos de implementação dos módulos propostos e exemplos de utilização do serviço.

# Capítulo 4

# Aspectos de Implementação

Este capítulo apresenta detalhes de implementação do protótipo do Serviço de Expansão de Consultas, baseada na linguagem Java. A Seção 4.1 detalha as tecnologias adotadas. As seções subsequentes detalham os aspectos de implementação realizados em cada etapa da transformação das consultas. A Seção 4.2 se refere à representação do banco de dados em uma ontologia e detalha a implementação do reparo das consultas. Este capítulo explica a invocação do serviço com uma consulta que mostra diferentes facetas do sistema implementado. Ressalta-se que o serviço foi testado inclusive com outros tipos de domínos, inclusive gerenciamento de produtos.

## 4.1 Tecnologias Adotadas

As ontologias utilizadas no trabalho foram recuperadas utilizando o Aondê, o qual possui ontologias inseridas diretamente pelos usuários ou buscadas em repositórios na Web, dentre os quais o Spire[1] [17] e o Swoogle[2] [1]. A leitura das ontologias, bem como a navegação por seus elementos, foi realizada utilizando a API Jena.

As ontologias utilizadas no trabalho estão representadas em OWL. Embora o Aondê forneça um conjunto de operações que permite buscar ontologias com termos associados é necessário o uso da API Jena para a manipulação de ontologias na expansão. Neste trabalho, o Aondê é utilizado para encontrar, armazenar, recuperar e integrar ontologias. A partir do momento em que uma ontologia é recuperada, o processamento da expansão usa a API Jena para carregá-la em memória e navegar por seus elementos na forma de um grafo.

A base de dados empregada para a realização das consultas e testes utilizou dados do Museu de Zoologia da UNICAMP. Essa base inclui informações detalhadas sobre espécies

[1] http://spire.umbc.edu/
[2] http://swoogle.umbc.edu/

estudadas, incluindo informações sobre coleta e características de espécimes. Os dados estão armazenados no PostgreSQL.

Foi utilizada a *engine* de serviços *Web* Axis para prover o acesso às operações do Serviço de Expansão de Consultas por meio de serviços *Web*. Essa engine fornece meios para consumir ou prover serviços Web utilizando uma implementação em Java. Para tanto, é possível enviar ou receber mensagens SOAP, criar classes Java que implementam cliente e servidor a partir de WSDL, dentre outros. O Axis também foi empregado para o acesso às operações do Aondê, fornecidas como serviços *Web*.

## 4.2 Ontologias de esquema e reparo de consultas

O processo é realizado com a leitura do esquema do banco de dados usando a API do Java. Conforme o Algoritmo 3.1, cada tabela do banco de dados resulta em uma classe de ontologia. Cada atributo da tabela corresponde a uma propriedade do conceito, mantendo um tipo de dados equivalente. As chaves estrangeiras são representadas por relacionamentos entre os conceitos.

Isto permite consultar o esquema diretamente, sem realizar conexão com o banco de dados. Permite, também, utilizar os mesmos recursos necessários à leitura de ontologias, já utilizada pelo processador de consultas por meio da API Jena.

O processo de reparo de uma consulta envolve a correção de erros de digitação de nomes de tabelas ou atributos, usando técnicas de *string matching*, e a padronização do formato de entrada esperado para as consultas no formato SQL. O reparo de consulta foi implementado em um módulo invocado da forma:

Reparar(strSQL)

onde o valor para *strSQL* consiste na consulta especificada pelo usuário. A invocação do método é realizada como um processo interno ao módulo de expansão de consulta. Como resultado da invocação do módulo é obtido uma consulta SQL padronizada.

O reparo inclui, ainda: (I) adição de tabelas à consulta; e (II) eliminação de ambiguidade na especificação de atributos, precedendo seus nomes pelo nome da tabelas correspondente.

Os reparos realizados são de baixa complexidade. A utilização de uma interface para a construção automática das consultas poderia evitar erros de digitação ou de sintaxe de consultas em SQL. Além disso, com a utilização de uma interface com o usuário, pode-se eliminar casos de ambiguidade, como, por exemplo, a definição de um atributo que está presente em duas tabelas, mas que não especifica a qual delas está associado.

Outro exemplo de consulta que pode ser reparada é:

```
1 SELECT * FROM Colecao WHERE classe='insecta' and
    Colecao.familis='Lycaenidae' and Colecao.antena = 'clavada';
```

A consulta apresenta um caso de falta de especificação de tabela associada ao atributo *classe* e a digitação incorreta do atributo *familia*, incorretamente escrito como *familis*.

O resultado da correção da consulta consiste em:

```
1 SELECT * FROM Colecao WHERE ((Colecao.classe = 'insecta') AND
    (Colecao.familia = 'Lycaenidae') AND (Colecao.antena =
    'clavada'));
```

## 4.3 Aplicação do Serviço

Esta seção apresenta um exemplo de utilização dos módulos desenvolvidos nesta dissertação. Os exemplos compreendem as etapas de configuração do banco de dados, na qual o esquema do banco de dados é representado em uma ontologia e o pré-processamento que envolve as fases de análise, reparo e expansão.

Os dados utilizados fazem parte do Museu de Zoologia Natural da UNICAMP[3] e foram desenvolvidos no trabalho de mestrado do Instituto de Computação da UNICAMP [28]. As tabelas *Catalog* e *Taxonomy*, utilizadas nas consultas de exemplo, contêm, respectivamente, 2592 e 783 registros, até o momento da entrega da dissertação.

### 4.3.1 Configuração do Banco de Dados

Conforme a Seção 3.2, antes que uma consulta possa ser expandida, é preciso realizar a etapa de configuração do banco de dados, criando a ontologia de esquema. Para tanto, invoca-se o módulo de Representação da seguinte forma:

Representacao(repositorio:"museu")

A Figura 4.1, extraída de [29], apresenta o modelo entidade relacionamento da base de dados utilizada e submetida como exemplo. Como resultado tem-se uma ontologia de esquema que representa o modelo (Figura 4.2). A ontologia de esquema reflete as alterações que são realizadas entre o modelo conceitual do banco de dados e sua implementação. Assim, tabelas como *Bio* e *N_Bio*, especificadas como especializações de *Substrate*, foram tratadas como uma única tabela, integrante de *Substrate*. O atributo booleano *Bio* permite distinguir entre registros que são considerados de *Bio* ou de *N_Bio*.

[3] http://proj.lis.ic.unicamp.br/biocore

### 4.3.2 Reparo de Consultas

Seja a consulta de usuário

```
1 SELECT *
2 FROM CATALOG, TAXINOMY
3 WHERE fk_taxa = idTaxa
4   AND SubOrder = 'Gnathophiurina'
5   AND lifeStagi= 'Adultos'
```

Após a invocação do módulo de reparo, esta consulta é transformada em

```
1 SELECT *
2 FROM CATALOG, TAXONOMY
3 WHERE CATALOG.fk_taxa = TAXONOMY.idTaxa
4   AND TAXONOMY.SubOrder = 'Gnathophiurina'
5   AND CATALOG.lifeStage= 'Adultos'
```

Os reparos realizados permitem corrigir os *strings* "Taxinomy" e "lifeStagi" para, respectivamente, "Taxonomy" e "lifeStage". Além disso, as tabelas *Taxonomy* e *Catalog* foram adicionadas aos atributos *fk_taxa*, *SubOrder* e *lifeStage*.

Observa-se que caso *Catalog* ou *Taxonomy* não estivessem especificados na cláusula *FROM*, a tabela seria adicionada. Contudo, se o usuário não tivesse especificado na cláusula *WHERE* a restrição "CATALOG.fk_taxa = TAXONOMY.idTaxa", esta correção geraria um produto cartesiano entre *Catalog* e *Taxonomy*, o que provavelmente não era a intenção do usuário. Para melhorar esta consulta, seria necessário acrescentar no exemplo cláusulas de junção natural ao predicado. Este tipo de expansão não foi considerado na dissertação e fica para trabalhos futuros.

### 4.3.3 Expansão de Consultas

Seja de novo a consulta da Seção 4.3.2, já reparada. Ao aplicar a consulta ao banco de dados não são obtidos resultados, porque as informações referentes ao atributo *SubOrder* não foram preenchidas.

Supondo, inicialmente, que o usuário não especifique os critérios de expansão, a consulta correspondente gerada será

```
1   SELECT *
2   FROM CATALOG, TAXONOMY
3   WHERE CATALOG.fk_taxa = TAXONOMY.idTaxa
4   AND (TAXONOMY.SubOrder = 'Gnathophiurina'
5     OR TAXONOMY.Family = 'Amphilepididae'
6     OR TAXONOMY.Family = 'Amphiuridae'
```

```
7       OR TAXONOMY.Family = 'Ophiactidae'
8       OR TAXONOMY.Family = 'Ophiothricidae' )
9     AND CATALOG.lifeStage= 'Adultos'
```

Pelo fato de não ser especificado algum critério de expansão, o Serviço de Expansão automaticamente associa expansões que não alteram o significado da consulta. A especificação padrão para estes casos é:

*=*:sem;

Em detalhes, a ontologia de domínio utilizada especifica apenas 1 nível de especialização de *SubOrder* que corresponde a *Family*. Dessa forma, são incluídos na consulta os termos *Amphilepididae*, *Amphiuridae*, *Ophiactidae* e *Ophiothricidae*. Como resultado foram obtidos 560 registros.

Se, por outro lado, o usuário definir como especificação de expansão:

TAXONOMY.SubOrder=*:gen=1;sem;

A consulta expandida será:

```
1   SELECT *
2   FROM CATALOG, TAXONOMY
3   WHERE CATALOG.fk_taxa = TAXONOMY.idTaxa
4   AND (TAXONOMY.SubOrder = 'Gnathophiurina'
5       OR TAXONOMY.Family = 'Amphilepididae'
6       OR TAXONOMY.Family = 'Amphiuridae'
7       OR TAXONOMY.Family = 'Ophiactidae'
8       OR TAXONOMY.Family = 'Ophiothricidae'
9       OR TAXONOMY.Order = 'Ophiurida')
10  AND CATALOG.lifeStage= 'Adultos'
```

Em detalhes, *Ophiurida* é o ancestral de classe na ontologia taxonômica, que equivale a ordem, e*Amphilepididae*, *Amphiuridae*, *Ophiactidae* e *Ophiothricidae* são as especializações de *Gnathophiurina* e constituem a família. Como resultado da consulta expandida, foram obtidos 630 registros.

## 4.4 Conclusões

Este capítulo apresentou exemplos de utilização do Serviço de Expansão de Consultas cobrindo a utilização de seus diferentes módulos. Os exemplos criados empregaram dados reais coletados e utilizados por biólogos em suas atividades. O Capítulo 5 apresenta as conclusões deste trabalho e algumas das possibilidades de trabalhos futuros.

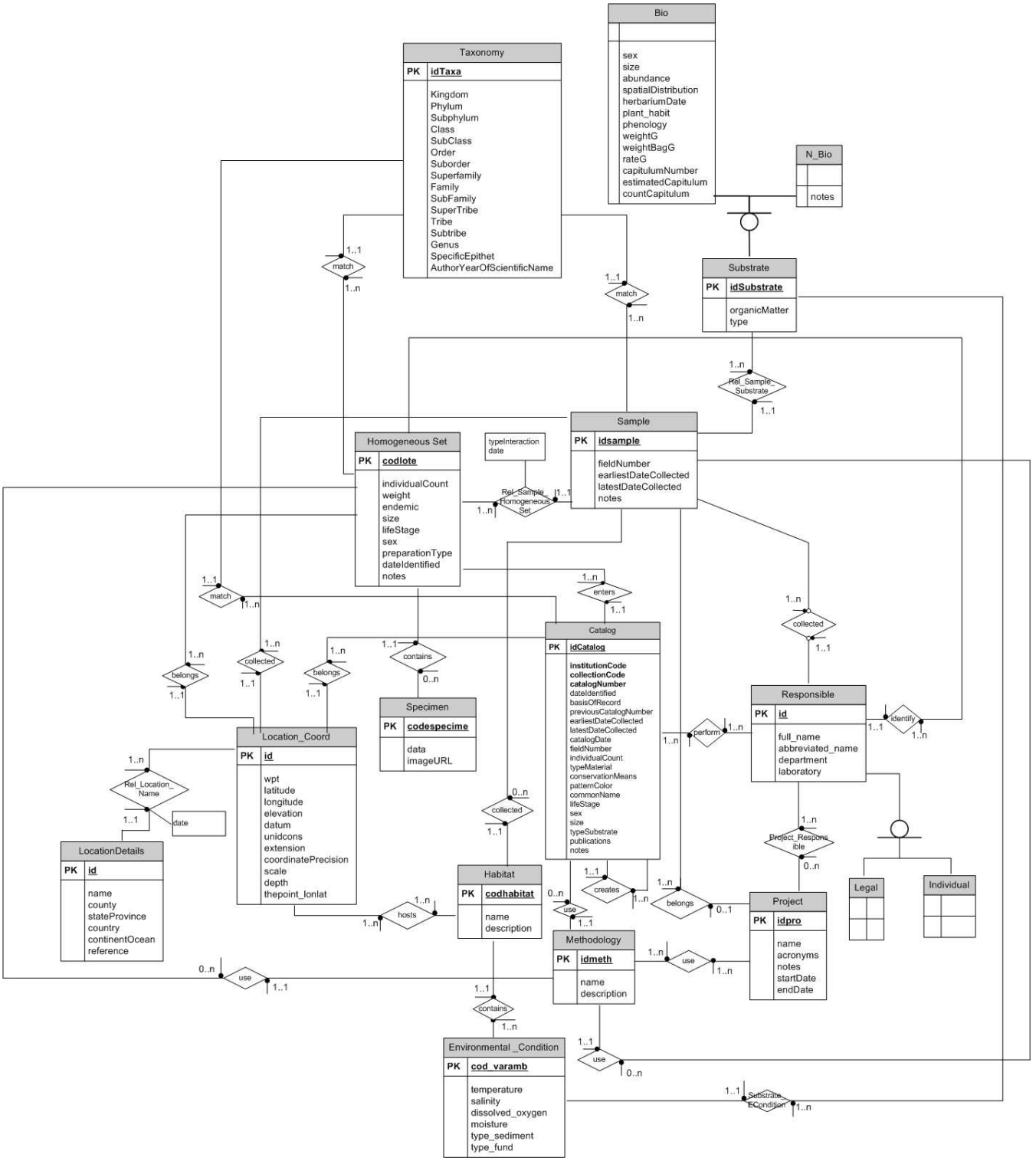



Figura 4.1: Diagrama entidade-relacionamento do banco de dados do Museu de Zoologia da UNICAMP [28].

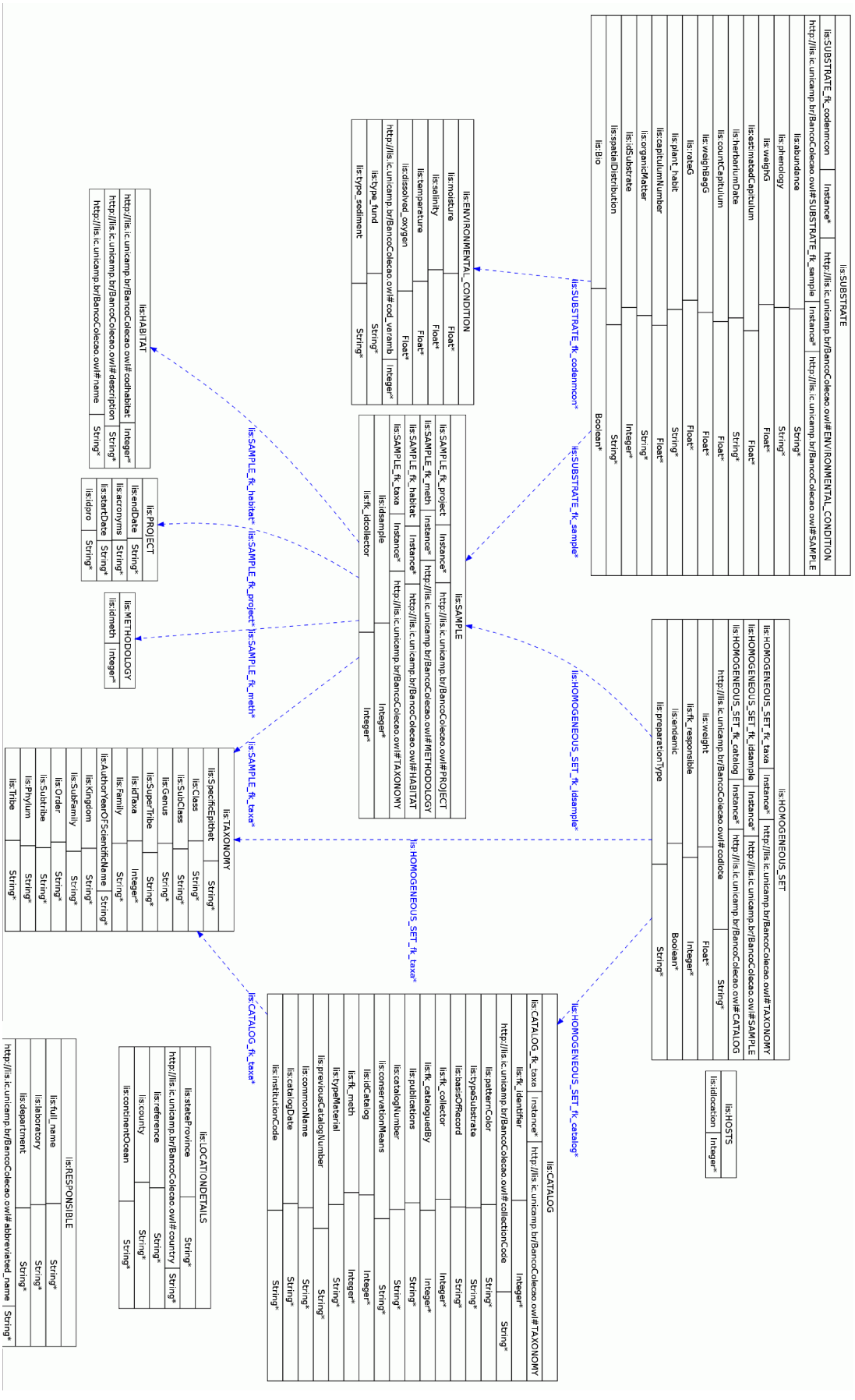



Figura 4.2: Ontologia do esquema do banco de dados criada pelo Algoritmo 3.1.

# Capítulo 5

# Conclusões e Extensões

## 5.1 Conclusões

O trabalho desenvolvido nesta dissertação consiste na especificação e implementação do Serviço de Expansão de Consultas, que tem como finalidade dar suporte à busca em bancos de dados de biodiversidade e especificar melhor a intenção do usuário. O serviço é baseado em utilizar o conhecimento compartilhado por pesquisadores ou grupos de pesquisa, representado por ontologias.

A proposta desta dissertação inclui a utilização de ontologias já criadas e disponíveis por pesquisadores da Biodiversidade. O serviço de expansão pode utilizar ontologias criadas para um banco de dados específico, mas também utilizar repositórios de ontologias sobre biodiversidade na *Web*.

Umas das soluções criadas no trabalho foi a representação do esquema do banco de dados a partir de ontologias. Isto apresenta duas vantagens: mantém as informações do esquema do banco de dados disponível para consulta, sem aumentar o número de acessos ao esquema a cada expansão realizada; e permite trabalhar com a interação entre as ontologias de domínio e a de esquema, além de explorar a própria ontologia de esquema. O serviço envolve vários módulos. Um deles é o módulo de reparo de consulta, que padroniza a consulta de entrada sem exigir do usuário a adaptação da consulta. Além disso, permite relacionar termos das ontologias de domínio à ontologia de esquema, usando a operação de alinhamento do Aondê.

Outra contribuição do trabalho consistiu na definição de um padrão de entrada para controlar as expansões feitas na consulta, a partir das colunas ou dos termos. Com isso, pode-se ampliar ou restringir a proximidade semântica entre a consulta original e a consulta expandida, de acordo com a pretensão do usuário.

Assim, as principais contribuições deste trabalho são:

- Estudo, revisão bibliográfica e comparação de características de técnicas de proces-

samento de consulta para melhor definir a consulta e ampliar os resultados;

- Especificação e implementação do serviço de expansão de consultas, definindo etapas de configuração do banco de dados, padronização de consulta e expansão;

- Especificação de um padrão para personalizar as operações a serem realizadas nas consultas dos usuários, com a implementação e adaptação do serviço de expansão para reconhecer, armazenar e aplicar as operações personalizadas.

As expansões de consulta permitem recuperar registros utilizando equivalência ou redundância semântica que pode existir entre campos e valores dos bancos de dados. Quando uma consulta é especificada, o usuário pode se ater a uma forma de expressar a consulta a fim de obter o resultado. Além disso, usando outras formas de representar a consulta, pode ampliar os resultados desejados preservando a semântica da consulta e a intenção do usuário. Esse cenário é pertinente na área de Biodiversidade em função das informações taxonômicas das espécies, nas quais os níveis taxonômicos inferiores agregam informações dos níveis superiores que especializam. Desse modo, é possível expandir as consultas dos usuários utlizando os diferentes níveis de especialização, tornando possível dar suporte a consultas em bases de dados que podem possuir campos incompletos ou digitados incorretamente.

O trabalho utilizou recursos e características que beneficiam consultas em Biodiversidade. No entanto, com a especificação de ontologias ou repositórios semânticos relacionados a outros domínios, é possível aplicar o serviço de expansão de consultas a outros domínios. O ajuste dos parâmetros e características do Aondê para obter melhores resultados com domínios que não sejam da Biodiversidade também pode ser necessário. Esta última necessidade pode ser amenizada dentro das propostas de extensão sugeridas na Seção 5.2.

## 5.2 Extensões

O estudo realizado neste trabalho permite, dentre outras, as seguintes extensões:

- Criação de axiomas em ontologias a partir de observações nos registros de coletas: a identificação de novas espécies ou a observação de novos comportamentos ou características de espécimes podem ser registradas nos repositórios de coletas. Como as ontologias utilizadas no trabalho consistem na visão conceitual que biólogos têm sobre sua área de atuação, essa visão deve ser aperfeiçoada e atualizada à medida em que seus estudos avançam. Dessa forma, um mecanismo que note novas informações que podem ser inferidas pelas coletas realizadas pode facilitar a observação de novos padrões e informações.

- Validação de dados inseridos por meio de ontologias: as ontologias contêm um conhecimento sobre o domínio. Assim, é possível usar essas informações para validar as informações que são inseridas por biólogos, como registros de observação de campo e busca por informações.

- Construção de uma interface para a manipulação de mapeamentos entre ontologias (BD e Domínio): O manuseio e a especificação das relações existentes entre o banco de dados e a ontologia permitem realizar transformações mais precisas e sofisticadas nas consultas. Desse modo, pode-se construir uma interface que permita que especialistas de um domínio trabalhem com o mapeamento entre o banco de dados e o domínio, permitindo que as consultas realizadas nas bases de dados sejam melhor aproveitadas.

- Aperfeiçoamento do mecanismo de seleção de ontologias de domínio para a expansão de consultas: O Aondê permite encontrar ontologias de diferentes formas, como a busca em um repositório específico e a busca baseada em vários termos e combinada com *ranking*. Adicionalmente, existe a possibilidade de realizar o alinhamento das ontologias recuperadas, como modo de obter uma ontologia de domínio com mais recursos. Contudo, o processo de busca, recuperação de ontologias, ranking e alinhamento demandam uma quantidade de tempo que o usuário pode não estar diposto a aceitar. Dessa forma, é preciso realizar estudo de formas de balancear o tempo de espera para se obter uma ontologia de domínio e a qualidade da ontologia. Uma das possibilidades é construir a base de ontologias de domínio à medida em que as consultas são realizadas, obtendo resultados incipientes no início, mas melhorando as expansões após realizar alguns alinhamentos em segundo plano, sem a necessidade de retornar os resultados para o usuário.

- Uso de Programação Genética e Realimentação de Relevância para ajustar os parâmetros atribuídos ao Aondê durante o processo de alinhamento entre ontologias: O Aondê permite ao usuário especificar parâmetros que influenciam o resultado de suas operações. Por exemplo, para realizar a integração entre duas ontologias tem-se o método com a seguinte assinatura:

```
1 public String integration(String ontNameA, String URLRepA,
2         String ontNameB, String URLRepB,
3         String[] dictionaries,
4         double trustworthiness,
5         double alpha,
6         double beta,
7         String targetRep)
```

Entre os parâmetros que devem ser passados estão a relevância da similaridade de estrutura e linguística (*alpha*, *beta*), além do grau mínimo de confiabilidade que deve existir para que dois elementos sejam considerados equivalentes (*trustworthiness*). A atribuição dos valores dos campos influencia diretamente os resultados dos alinhamentos. Para melhorar tal atribuição, pode-se utilizar a combinação de Programação Genética e Realimentação de Relevância para, respectivamente, fazer uma busca ampla pelo espaço de valores possível e obter o *feedback* do usuário quanto aos resultados obtidos. A partir do *feedback* do usuário, o algoritmo de Programação Genética consegue analisar melhor os resultados e convergir para um conjunto de parâmetros que resulte em alinhamentos melhores, sob a perspectiva do usuário.

# Referências Bibliográficas


[1] Swoogle: A semantic web search and metadata engine. 2004. Swoogle is a crawler-based indexing and retrieval system for the Semantic Web. URL: http://ebiquity.umbc.edu/get/a/publication/116.pdf.

[2] A. Andreou. Ontologies and query expansion. Master's thesis, University of Edinburgh, 2005.

[3] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, Inernational Business Machines Corporation, Microsoft Corporation Inc., 5 May 2003.

[4] F. A. Bisby. The Quiet Revolution: Biodiversity Informatics and the Internet. *Science*, 289(5488):2309–2312, 2000.

[5] P. Bosc, A. HadjAli, and O. Pivert. Relaxation paradigm in a flexible querying context. In H. L. Larsen, G. Pasi, D. O. Arroyo, T. Andreasen, and H. Christiansen, editors, *FQAS*, volume 4027 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2006.

[6] A. Bozzon, P.-A. Chirita, C. S. Firan, and W. Nejdl. Lexical analysis for modeling web query reformulation. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *SIGIR*, pages 739–740. ACM, 2007.

[7] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible Markup Language (XML) 1.1. Technical report, World Wide Web Consortium (W3C), 2004. URL: http://www.w3.org/TR/2004/REC-xml11-20040204.

[8] D. A. L. Canhos, S. de Souza, and V. P. Canhos. Coleções biológicas e sistemas de informação. Technical report, Centro de Referência em Informação Ambiental -(Cria), 2005.

[9] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1.* W3C, 1.1 edition, March 2001. URL: http://www.w3c.org/TR/wsdl.

[10] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78, August 2003.

[11] C. COSTA. Coleoptera Linnaeus, 1758. *Brandão C. R. F. Cancello (eds) Invertebrados Terrestres*, V(5488):115–122, 1999.

[12] M. J. Costello and E. V. Berghe. 'ocean biodiversity informatics': a new era in marine biology research and management. In *Marine Ecology Progress Series*, volume 316, pages 203–214, May 2006.

[13] F. Curbera, M. J. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: An introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, 6(2):86–93, 2002.

[14] R. da Silva Torres, C. B. Medeiros, M. A. Gonçalves, and E. A. Fox. A digital library framework for biodiversity information systems. *Int. J. on Digital Libraries*, 6(1):3–17, 2006.

[15] J. Daltio. Aondê: Um serviço web de ontologias para interoperabilidade em sistemas de biodiversidade. Master's thesis, Instituto de Computação - Unicamp, August 2007.

[16] J. Daltio, C. B. Medeiros, L. C. G. Jr, and T. M. Lewinsohn. A framework to process complex biodiversity queries. In *Proc. ACM Symposium on Applied Computing (ACM SAC)*, March 2008.

[17] C. P. et al. Integrating Ecoinformatics Resources on the Semantic Web. May 2006.

[18] D. Florescu, L. Raschid, and P. Valduriez. A methodology for query reformulation in cis using semantic knowledge. *Int. J. Cooperative Inf. Syst.*, 5(4):431–468, 1996.

[19] P. Godfrey and J. Gryz. A framework for intensional query optimization. In *DDLP*, pages 57–68, 1996.

[20] L. C. Gomes. An architecture for querying biodiversity repositories on the web. Master's thesis, Instituto de Computação - Unicamp, May 2007.

[21] T. Gruber. Toward principles for the design for ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.

[22] T. R. Gruber. Ontology. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*. Springer, Berlin, Heidelberg, 2009.

[23] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. Soap version 1.2 part 1: Messaging framework. Technical report, W3C, MIT, ERCIM, Keio, 2006.

[24] X. Haigen, G. Zhenning, X. Dayuan, and W. Xiaoming. China national biodiversity information query system. *Journal of Environmental Management (United Kingdom)*, 56, 1999.

[25] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 387–396. ACM, 2006.

[26] K. Järvelin, J. Kekäläinen, and T. Niemi. Expansiontool: Concept-based query expansion and construction. *Inf. Retr.*, 4(3-4):231–255, 2001.

[27] L. Lian, J. Ma, J. Lei, L. Song, and D. Zhang. Query relaxing based on ontology and users' behavior in service discovery. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 2007.

[28] J. E. G. Malaverri. Um serviço de gerenciamento de coletas para sistemas de informação de biodiversidade. Master's thesis, Instituto de Computação - Unicamp, April 2009.

[29] J. E. G. Malaverri, B. S. C. M. Vilar, and C. M. B. Medeiros. A tool based on web services to query biodiversity information. In *5th International Conference on Web Information Systems and Technologies (Webist 2009)*, March 2009.

[30] D. L. McGuinness. Ontologies come of age. In D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 171–194. MIT Press, 2003.

[31] E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[32] W. Michener, J. Beach, M. Jones, B. Ludäscher, D. Pennington, R. S. Pereira, A. Rajasekar, and M. Schildhauer. A knowledge environment for the biodiversity and ecological sciences. *J. Intell. Inf. Syst.*, 29(1):111–126, 2007.

[33] N. Milanovic and M. Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51–59, 2004.

[34] K. Munir, M. Odeh, and R. McClatchey. Ontology assisted query reformulation using the semantic and assertion capabilities of owl-dl ontologies. In B. C. Desai, editor, *IDEAS*, volume 299 of *ACM International Conference Proceeding Series*, pages 81–90. ACM, 2008.

[35] R. Navigli and P. Velardi. An analysis of ontology-based query expansion strategies. In *Proc. of Workshop on Adaptive Text Extraction and Mining (ATEM 2003)*, pages 22–26, Cavtat-Dubrovnik, Croatia, September 2003. 14th European Conference on Machine Learning (ECML 2003).

[36] C. B. Necib and J. C. Freytag. Ontology based query processing in database management systems. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *CoopIS/-DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 839–857. Springer, 2003.

[37] C. B. Necib and J. C. Freytag. Using ontologies for database query reformulation. In *ADBIS (Local Proceedings)*, 2004.

[38] C. B. Necib and J.-C. Freytag. Query processing using ontologies. In *Proc. of the 17th Conference on Advanced Information Systems Engineering (CAISE), Porto, Portugal*, June 2005.

[39] N. F. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford University School of Medicine, 2001.

[40] H. Pang, H. Lu, and B. C. Ooi. An efficient semantic query optimization algorithm. In *ICDE*, pages 326–335. IEEE Computer Society, 1991.

[41] N. W. Paton, R. Stevens, P. G. Baker, C. A. Goble, S. Bechhofer, and A. Brass. Query processing in the tambis bioinformatics source integration system. In *SSDBM*, pages 138–147, 1999.

[42] P.-D. Qiou. Design configuration of composite services on semantic web. Master's thesis, Tatung University, 2005.

[43] L. N. Soldatova and R. D. King. Are the current ontologies in biology good ontologies? *Nature Biotechnology*, 23(9):1095–1098, 2005.

[44] C. M. Sperberg-McQueen and H. Thompson. Xml schema specification. URL: http://www.w3.org/XML/Schema, 2004.

[45] L. Stojanovic, S. Staab, and R. Studer. e-learning based on the semantic web. In *World Conference on the WWW and the Internet (WebNet 01), Orlando, Florida*, 2001.

[46] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid - the grid meets the semantic web. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 706–721. Springer, 2003.

[47] The OWL-S Coalition. OWL-S 1.1 Release, 2004. URL: http://www.daml.org/services/owl-s/1.1/.

[48] WeBIOS: Web Service Multimodal Tools for Strategic Biodiversity Research, Assessment and Monitoring. http://www.lis.ic.unicamp.br/projects/webios, started 2005.

[49] BIO-CORE: Bio-CORE - Tools, models and techniques to support research in biodiversity. http://www.lis.ic.unicamp.br/projects/bio-core/, started 2008.

[50] S. L. Tomassen, J. A. Gulla, and D. Strasunskas. Document space adapted ontology: Application in query enrichment. In C. Kop, G. Fliedl, H. C. Mayr, and E. Métais, editors, *NLDB*, volume 3999 of *Lecture Notes in Computer Science*, pages 46–57. Springer, 2006.

[51] R. Toms. Meet SA's longest stick insect - almost the length of a 30cm ruler. *Science in Africa*, (36), 2004.

[52] P. Traverso and M. Pistore. Automated composition of semantic web services into executable processes. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 380–394. Springer, 2004.

[53] B. S. C. M. Vilar and C. M. B. Medeiros. Processamento semântico de consultas para sistemas de biodiversidade. In *VII WTDBD - Workshop de Teses e Dissertações em Bancos de Dados*, pages 37–42, Campinas, SP, October 2008.

[54] H. Wang, J. Z. Huang, Y. Qu, and J. Xie. Web services: problems and future directions. *J. Web Sem.*, 1(3):309–320, 2004.

[55] J. White, B. Kolpackov, B. Natarajan, and D. C. Schmidt. Reducing application code complexity with vocabulary-specific xml language bindings. In M. Guimarães, editor, *ACM Southeast Regional Conference (2)*, pages 281–287. ACM, 2005.

[56] D. Woelk. E-learning, semantic web services and competency ontologies. In *World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA)*, 2002.

[57] H. Xiao. *Query Processing for Heterogeneous Data Integration Using Ontologies*. PhD thesis, University of Illinois, August 2006.

[58] H. Xu, D. Wang, and X. Sun. Biodiversity clearing-house mechanism in china: present status and future needs. *Biodiversity and Conservation*, 9, March 2000.

[59] C. A. Yaguinuma, M. Biajiz, and M. T. P. Santos. Sistema foque para expansão semântica de consultas baseada em ontologias difusas. In *XXII Simpósio Brasileiro de Banco de Dados*, pages 208–222, João Pessoa, PB, 2007.