UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2008.2

DESENVOLVIMENTO DO MÓDULO DE REFORMULAÇÃO DE CONSULTAS NO SISTEMA SPEED

TRABALHO DE GRADUAÇÃO

Aluno: Thiago Arruda Neves (tan@cin.ufpe.br)
Orientadora: Ana Carolina Salgado (acs@cin.ufpe.br)
Co-Orientadora: Damires Yluska de Souza Fernandes (dysf@cin.ufpe.br)

Recife, Novembro de 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2008.2

THIAGO ARRUDA NEVES

DESENVOLVIMENTO DO MÓDULO DE REFORMULAÇÃO DE CONSULTAS NO SISTEMA SPEED

Trabalho apresentado à graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora – Ana Carolina Salgado (acs@cin.ufpe.br)
Co-Orientadora – Damires Yluska de Souza Fernandes (dysf@cin.ufpe.br)

Recife, Novembro de 2008

“A persistência é o caminho do êxito.” (Charles Chaplin)

A Deus, que sempre me guiou
nessa jornada.

À minha família, em especial aos meus pais.
Minha gratidão pelo incentivo, apoio,
paciência, compreensão, e por me
proporcionarem esse curso.

Agradecimentos

Agradeço primeiramente a Deus por minha vida e por ter me dado a oportunidade de estudar em uma universidade de qualidade, num curso com o qual me identifiquei muito.

Agradeço a toda minha família, principalmente à minha mãe e ao meu pai, por me apoiarem e me incentivarem em todas as etapas da minha vida, por me proporcionarem esse curso e me ajudarem nos momentos de dificuldade. Eles são os grandes responsáveis pela pessoa que sou hoje.

Agradeço a minha orientadora, a professora Doutora Ana Carolina Salgado, pela oportunidade e confiança para trabalhar neste projeto. Aos meus companheiros de projeto, em especial a Damires Souza, pelo enorme apoio e contribuição durante o desenvolvimento deste trabalho.

Agradeço aos Professores do CIn, em especial ao Professor Fernando Fonseca, por despertar meu interesse pela área de Banco de Dados, como também pelos conhecimentos passados e a amizade.

Agradeço aos meus amigos do CIn pelo apoio e companheirismo durante o curso, além dos momentos de descontração proporcionados em meio a um curso tão exigente. Com certeza irei manter muitas amizades por toda minha vida.

Agradeço, enfim, a todos que acreditaram, incentivaram e participaram de alguma maneira desta jornada.

Muito Obrigado!

Resumo

Os sistemas PDMS (*Peer Data Management System*) são aplicações *peer-to-peer* (P2P) nas quais cada um de seus membros constituintes são *peers* autônomos que podem compartilhar seus esquemas de dados entre si de forma total ou parcial.

O sistema SPEED (*Semantic PEEr-to-Peer Data Management System*) é um PDMS baseado em semântica cuja arquitetura é composta de uma topologia de super pontos (*super-peers*) e uso de DHT (*Distributed Hash Table*). Nesta estrutura de *super-peers*, os pontos de dados são agrupados de acordo com seus domínios em *clusters* semânticos. Cada *cluster* possui um tipo especial de ponto, chamado ponto de integração, que é responsável por tarefas mais complexas, como o processamento de consultas. Os pontos de dados e um ponto de integração associado estão ligados a outro tipo de ponto, chamado ponto semântico, que atua como ponto de acesso ao *cluster*.

Um dos principais serviços oferecidos pelo SPEED é o processamento de consultas, provido pelos pontos de integração. Dentro desse contexto, esse trabalho tem como objetivo o desenvolvimento do módulo de consultas do SPEED em um dado ponto de integração. Tal módulo deve receber a consulta, analisá-la, identificar a sua semântica (conceitos e propriedades), e a partir de correspondências semânticas entre os *peers*, reformular a consulta (com ou sem enriquecimento) e enviá-la a outro ponto de integração que possa respondê-la satisfatoriamente.

Abstract

PDMS (Peer Data Management System) are peer-to-peer applications (P2P) in which each of its constituent members are autonomous peers that can share their schemas among themselves totally or partially.

SPEED (Semantic PEEr-to-peer Data Management System) is a PDMS based on semantics whose architecture is composed by a topology of super peers and DHT (Distributed Hash Table) usage. In this super peers setting, the data peers are grouped according to their domains in semantic clusters. Each cluster has a special kind of peer, the so-called integration peer, which is responsible for more complex tasks such as query processing. The data peers and an associated integration peer are linked to another kind of peer, called semantic peer, which acts as an access point to the cluster.

One of the main services offered by SPEED is query processing, provided by the integration peers. In this light, this work aims to develop the SPEED query module in a given integration peer. Such module is able to receive the query, analyze it, identify its semantics (concepts and properties), and considering the set of semantic correspondences among peers, reformulate the query (with or without enrichment) and send it to another integration peer that can answer it accordingly.

Índice

Lista de Figuras

Lista de Quadros

Lista de Tabelas

1. Introdução

Neste capítulo será feita uma introdução do contexto no qual foi desenvolvido o trabalho. O capítulo abordará aspectos da motivação ao desenvolvimento do projeto e também seus objetivos, assim como a estrutura desta monografia.

1.1 Motivação

Os PDMS (*Peer Data Management System*) são sistemas de integração de dados distribuídos que possibilitam o acesso a diversas bases de dados heterogêneas, sem a necessidade de representação de um esquema único global e centralizado. Ao invés disso, cada *peer* dentro do sistema representa uma fonte de dados autônoma com sua própria representação de esquema de dados. O compartilhamento desses esquemas permite aos *peers* a troca de informações entre si e a possibilidade de realização de outras tarefas, como o processamento de consultas.

Recentemente, os PDMS se tornaram grande fonte de pesquisa no sentido de serem uma extensão às bases de dados distribuídas no contexto P2P (*peer-to-peer*) [1]. Devido à grande variedade na representação de dados e na semântica dos *peers*, um modelo de PDMS baseado em ontologias foi sugerido [8]. Xiao [5] introduziu uma nova definição para esse novo modelo, que possui o intuito de oferecer uma representação de dados em uma notação mais uniforme. Esse tipo de sistema é conhecido como OPDMS (*Ontology-based Peer Data Management System*).

O sistema SPEED é um PDMS (mais especificamente um OPDMS) baseado em semântica que tem como objetivo prover facilidades para o compartilhamento e a integração dos dados distribuídos ao longo de pontos conectados (*peers*). Um dos principais serviços oferecidos pelo SPEED (e por diversos tipos de PDMS) é o processamento de consultas. Uma consulta submetida em um *peer* pode obter uma resposta relevante de qualquer outro *peer* no PDMS, desde que estes estejam ligados por um caminho semântico através de mapeamentos [2].

No SPEED, os pontos de integração são responsáveis por processar consultas de seu *cluster* (grupo de *peers* de dados). Nesse sentido, quando um ponto de integração recebe uma consulta, ele a analisa, identifica sua semântica, e a partir de correspondências semânticas entre os esquemas dos *peers* de dados de seu *cluster*, reformula a consulta original e a envia aos *peers* que podem respondê-la [33]. A consulta será também reformulada quando encaminhada aos *peers* de integração vizinhos àquele que recebeu a consulta original para processamento.

1.2 Objetivos

Dentro do contexto exibido na seção anterior, este trabalho tem como objetivo o desenvolvimento de uma ferramenta de reformulação de consultas para o sistema SPEED que possibilite: a submissão de consultas em um dado ponto de integração; a identificação de sua semântica; seu enriquecimento (com base nas correspondências semânticas); sua reformulação de forma que os demais pontos de integração vizinhos deste ponto de submissão possam respondê-la; e a execução destas consultas.

1.3 Estrutura da Monografia

O restante deste trabalho será organizado do seguinte modo:

- Capítulo 2 – Conceitos Básicos: abordagem de conceitos introdutórios importantes para o melhor entendimento do trabalho, tais como banco de dados distribuídos, ontologias, RDF (*Resource Description Framework*) [7], OWL (*Ontology Web Language*) [9], Lógica Descritiva e a linguagem SPARQL [15];
- Capítulo 3 – *Peer-to-peer e* PDMS: maior ênfase à tecnologia *peer-to-peer* (P2P) e aos sistemas PDMS, bases deste trabalho, dando maior destaque às topologias de redes P2P, ao processamento de consultas em um PDMS, falando sobre enriquecimento e reformulação;
- Capítulo 4 – O Sistema SPEED: abordagem mais aprofundada do SPEED, detalhando suas características, tais como sua arquitetura, funcionamento e tratamento de consultas;

- Capítulo 5 – Reformulação de consultas no SPEED: apresenta detalhes do desenvolvimento e implementação do módulo de reformulação de consultas do SPEED, tema deste trabalho; e
- Capítulo 6 – Conclusão: uma conclusão com os conhecimentos adquiridos durante o trabalho, contribuições e sugestões para trabalhos futuros.

2. Conceitos Básicos

Neste capítulo serão apresentados e discutidos alguns conceitos básicos para o escopo deste trabalho. O entendimento de tais conceitos irá facilitar a compreensão das características e contexto no qual o projeto foi desenvolvido. Serão apresentados conceitos de Bancos de Dados Distribuídos e Centralizados, Ontologias, Lógica Descritiva e da linguagem SPARQL.

2.1 Banco de Dados Distribuídos

Para melhor entendimento do conceito e características de bancos de dados distribuídos, é conveniente iniciar a seção falando sobre os bancos de dados centralizados, que são mais difundidos entre os usuários da área.

2.1.1 Bancos de Dados Centralizados

Os sistemas de gerenciamento de bancos de dados (SGBD) centralizados são especificados em 3 níveis [3]:

- Nível externo: é a especificação da organização conceitual do banco de dados, vista por grupos de usuários. Os esquemas externos são mais úteis ao desenvolvimento de aplicações pelos usuários, pois oculta grande parte da complexidade do banco. Por meio dos esquemas externos, grupos de usuários podem também restringir acesso aos dados armazenados;
- Nível conceitual ou lógico: especificação do que é armazenado pelo banco de dados. Não se prende a todos os detalhes de armazenamento; é mais uma visão de alto nível, onde o que prevalece é a semântica do que é construído;
- Nível físico ou interno: especificação das estruturas de armazenamento do banco de dados. O esquema interno apresenta mais detalhes sobre como o banco de dados armazena seus dados. Essa distinção entre o esquema interno e o esquema conceitual é muito importante para os usuários, pois tais detalhes de armazenamento não devem ser relevantes às aplicações.

A especificação de um banco de dados centralizado em níveis facilita o entendimento das características e da organização destes bancos. Essa organização em níveis possui algumas vantagens, como por exemplo, a facilidade de manutenção dos bancos de dados, pois existe uma independência entre os dados, tanto lógica como física.

A ilustração dos níveis de um banco de dados centralizado é dada na Figura 01:

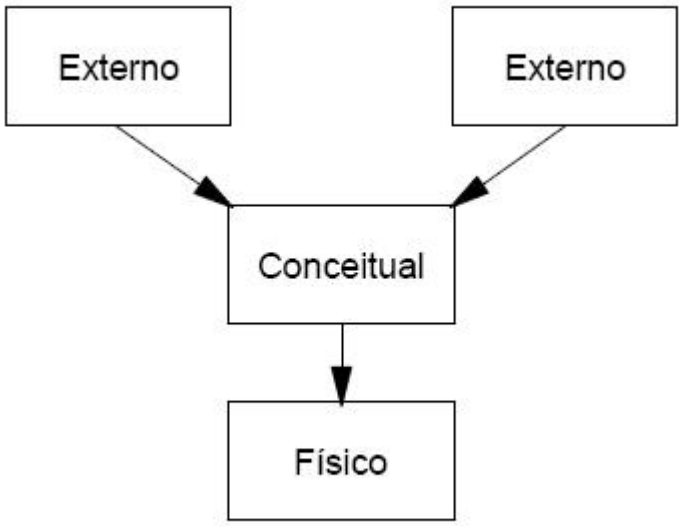


Figura 01 – Níveis de um banco de dados centralizado [3]

2.1.2 Bancos de Dados Distribuídos

A tecnologia dos bancos de dados distribuídos (BDD) surgiu como uma junção das tecnologias de bancos de dados com a de redes e comunicação de dados. Um BDD pode ser definido como uma coleção de vários bancos de dados logicamente inter-relacionados distribuídos por uma rede de computadores. O sistema de gerenciamento de banco de dados distribuídos (SGBDD) é o *software* que tem o papel de gerenciar o BDD enquanto torna a distribuição transparente para o usuário [4].

Os BDD trazem para o universo dos bancos de dados as vantagens da computação distribuída, visto que consistem de vários elementos de processamento, homogêneos ou não, interconectados, que cooperam entre si na execução de tarefas.

Como dito, a distribuição deve ser transparente ao usuário, então, num nível lógico, um BDD deve ser visto como um banco de dados centralizado, existindo então um esquema conceitual global e vários esquemas externos globais, para os grupos de usuários. A forma de relacionamento entre os esquemas do BDD é diferente em relação aos bancos de dados centralizados, como mostrado na Figura 02:

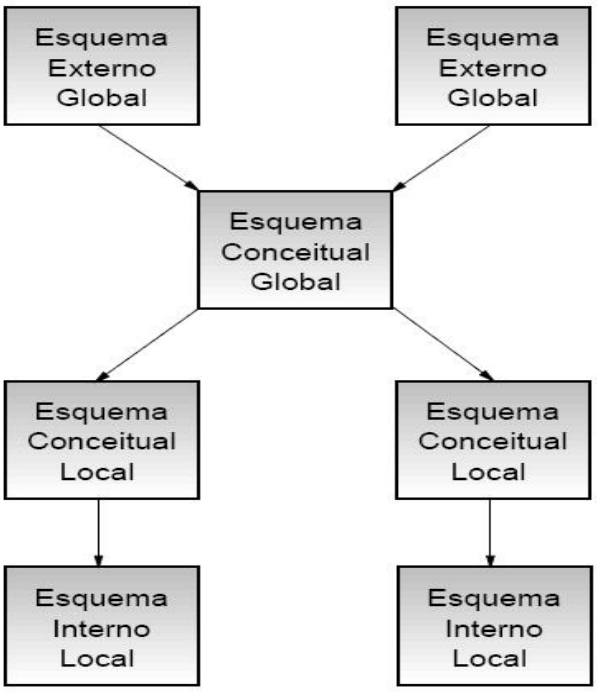


Figura 02 – Níveis de um banco de dados distribuído [3]

O uso de bancos de dados distribuídos tem sido proposto por várias razões, e algumas vantagens são destacadas a seguir [4]:

- Gerenciamento de dados distribuídos com vários níveis de transparência;

- Melhoria na confiabilidade e na disponibilidade: com os dados e o *software* do SGBD distribuídos em vários *sites*, um *site* pode falhar enquanto outros continuam em operação;
- Melhoria de desempenho: um banco de dados fragmentado mantém os dados mais próximos de onde eles são necessários;
- Expansão mais fácil: num ambiente distribuído, a expansão do sistema no que diz respeito ao acréscimo de dados se torna muito mais fácil.

2.2 Ontologia

Muitas definições de ontologia são encontradas na literatura no âmbito de inteligência artificial. Uma das definições diz que ontologia é uma descrição formal explícita de conceitos em um domínio do discurso, de propriedades de cada conceito descrevendo atributos ou características, e de restrições dessas propriedades. É conhecido também que uma ontologia juntamente com definições de instâncias de suas classes formam uma base de conhecimento [6].

As classes são as entidades principais de uma ontologia. Uma ontologia pode representar um domínio de aplicação, e várias classes podem ser apresentadas, com seus atributos, limitações, características e relacionamentos com outras classes do domínio, como por exemplo, o relacionamento de subclasse e superclasse, muito comuns nas ontologias.

Ontologias são muito comuns na *web*, servindo para uma diversidade de propósitos, desde a descrição de *sites* como de um simples produto comercial. As ontologias definem um vocabulário comum e estruturado para a troca de informações de um domínio entre pesquisadores. Algumas das principais razões para o desenvolvimento de uma ontologia são [6]:

- Compartilhar um conhecimento comum entre pessoas ou agentes de *software*;
- Permitir o reuso do conhecimento de um domínio;

- Melhorar a análise do conhecimento de um domínio;
- Separar o conhecimento do domínio do conhecimento operacional.

A fim de oferecer uma padronização na representação desse conhecimento da *web*, duas linguagens (arquiteturas) surgiram nesse contexto: RDF e OWL, ambas recomendações da W3C (*World Wide Web Consortium*), que serão mais bem explicadas nas próximas subseções.

2.2.1 RDF

A *web* provê acesso sem precedentes à informação. Metadados (campos descritivos de algum tipo de informação) aumentam o acesso a essa informação, e o RDF surge como um padrão da W3C proposto para definir a arquitetura necessária para suportar os metadados da *web* [10].

RDF é uma linguagem para representar informação na Internet. Seu maior objetivo é definir um mecanismo para descrever recursos que não faça suposições sobre qualquer domínio de aplicação em particular, nem defina (a *priori*) a semântica de nenhum domínio de aplicação [7].

O RDF possui um modelo padrão para descrever recursos da *web*. Um recurso, para RDF, é um objeto, que possui propriedades. Tais propriedades possuem tipos (numéricos, caracteres,...), e valores possíveis. Os valores de uma propriedade podem ser atômicos ou outros recursos. Um exemplo simples de modelo em RDF e sua representação sintática podem ser vistos na Figura 03 e Quadro 01, respectivamente:

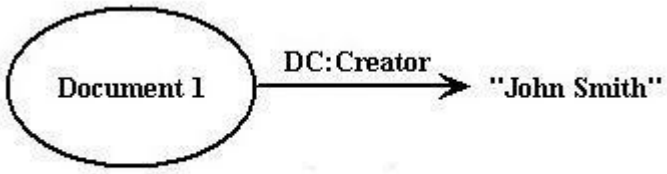


Figura 03 – Exemplo de modelo RDF – Documento e seu criador [10]

```
<?xml:namespace ns = "http://www.w3.org/RDF/RDF/" prefix ="RDF" ?>
<?xml:namespace ns = "http://purl.oclc.org/DC/" prefix = "DC" ?>

<RDF:RDF>
  <RDF:Description RDF:HREF = "http://uri-of-Document-1">
    <DC:Creator>John Smith</DC:Creator>
  </RDF:Description>
</RDF:RDF>
```

Quadro 01 – Sintaxe para o modelo RDF – Documento e seu criador [10]

2.2.2 OWL

OWL é uma linguagem para definição e instanciação de ontologias na *web*. É viável utilizar OWL quando a informação contida em documentos precisa ser processada por alguma aplicação, ao contrário do que ocorre quando a informação apenas necessita ser apresentada ao usuário [9].

A OWL é considerada uma extensão de RDF. Sendo assim, também pode incluir descrições de classes, suas respectivas propriedades e relacionamentos. Assim como o RDF, OWL é uma recomendação da W3C, mas possui um maior poder de interpretação e mais recursos para modelar o conteúdo da *web* do que o RDF ou XML, pois a OWL possui um vocabulário mais amplo e uma semântica mais bem definida.

A linguagem OWL é mencionada como uma tecnologia importante para a futura implementação da *web* semântica [11]. Possui um vocabulário mais amplo para expressar propriedades e relacionamentos, como o de cardinalidade, enumerações, entre outros. OWL possui 3 sublinguagens expressivas direcionadas a comunidades específicas [7]:

- OWL *Lite*: Suporta usuários que necessitam de uma classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. É mais simples fornecer ferramentas que suportam OWL *Lite*;

- OWL DL: Suporta usuários que querem a máxima expressividade, enquanto mantém a computabilidade e decidibilidade. OWL DL possui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições. Possui esse nome devido à sua correspondência com as lógicas de descrição; e
- OWL *Full*: Direcionada aos usuários que querem máxima expressividade e liberdade sintática sem nenhuma garantia computacional. OWL *Full* permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL.

2.3 Lógica Descritiva

A lógica descritiva (ou DL, do inglês *Description Logic*) é um conjunto de linguagens de representação de conhecimento de aplicações, que enfatizam o domínio de uma aplicação de uma maneira formal e estruturada, e que é bem compreendida [12].

A DL consegue explicitar e detalhar diversos domínios de aplicações graças ao seu poder de descrição de entidades através de operadores bem definidos e quantificadores, e também às regras de formação dessas descrições de algum objeto ou situação em particular. Além disso, a DL possui toda uma carga semântica apoiada na lógica matemática.

Na Figura 04 são ilustrados os construtores da lógica descritiva ALC (*Attribute Language with Complement*) [31] (na ordem, operador de conjunção / interseção, disjunção / união, negação, e os quantificadores “para todo” e “existe”). A Figura 05 apresenta um exemplo de descrição do conceito “Um homem que é casado com uma médica, e todos os seus filhos são ou médicos ou professores”.

Figura 04 – Construtores da Lógica Descritiva

$$\text{Human} \sqcap \neg\text{Female} \sqcap (\exists \text{married}.\text{Doctor}) \sqcap (\forall \text{hasChild}.(\text{Doctor} \sqcup \text{Professor})).$$

Figura 05 – Exemplo de uso da Lógica Descritiva [12]

A semântica da lógica descritiva é considerada uma característica importante, que diferencia a DL de predecessores como as redes semânticas [13,14]. É uma lógica que inclui operações mais expressivas que a lógica proposicional, e possui maior decidibilidade, ou seja, resoluções mais eficientes para problemas de decisão em comparação à lógica de predicados de primeira ordem [12].

2.4 SPARQL

SPARQL é uma linguagem de consulta para documentos RDF, mas que também faz buscas em estruturas de arquivos OWL, ou seja, busca em ontologias em geral. É padronizada pela DAWG (RDF *Data Access Working Group*) da W3C [15].

As consultas em SPARQL consistem de um padrão triplo (*triple patterns*): conjunções, disjunções e padrões opcionais de complementação. Essa idéia de tripla se baseia nas estruturas de grafo do RDF. SPARQL foi desenvolvida a partir de linguagens de consulta em RDF anteriores, como RDQL (RDF *Data Query Language*) [16] e SeRQL (*Sesame* RDF *Query Language*) [17], e sua implementação é compatível com uma série de plataformas.

Além da possibilidade de consulta de dados, SPARQL também oferece a capacidade de extração de informações de repositórios a partir de regras de elaboração do usuário, com o auxílio dos construtores reservados de SPARQL: *Construct*, *Ask*, e *Describe*. Um ambiente comum para a prática de consultas SPARQL é o Jena [18] com o auxílio do ARQ [19], um motor de consulta que trabalha com o Jena.

O Quadro 02 apresenta um exemplo de consulta em SPARQL, que deseja recuperar de uma ontologia todos os identificadores (IDs) de professores.

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX prf: <http://www.owl-ontologies.com/Ontology1220896776.owl#>
SELECT
        ?ID
FROM
        <http://www.owl-ontologies.com/Ontology1220896776.owl>
WHERE
{
        ?x rdf:type prf:Teacher .
        ?x prf:ID ?ID
}
```

Quadro 02 – Exemplo de Consulta SPARQL

Detalhando mais essa consulta a respeito da sintaxe da linguagem, temos os seguintes itens:

- Prefixos: no início há a declaração de prefixos com a palavra reservada *prefix*, que associa uma sigla com um URI (*Uniform Resource Identifier*) específico. Uma consulta pode incluir qualquer quantidade de prefixos desejada;
- *Select*: palavra reservada que define os itens que serão retornados pela consulta (as variáveis em SPARQL são precedidas de "?" ou "$");
- *From:* identifica a ontologia na qual a consulta irá executar;
- *Where:* indica as condições para o retorno do resultado. Neste caso, como explicado, podemos observar o padrão de triplas do SPARQL. A tripla pode ser entendida como uma construção de sujeito, predicado e objeto;
- ?x: a variável "x" é ligada a conceitos do tipo "*Teacher*" na ontologia;
- Operador "ponto": o operador "." indica uma concatenação de restrições. A segunda restrição indica que, dado que "x" contém um professor, seu ID deve ser atribuído a variável ID (?ID, que é a variável de retorno da consulta).

2.5 Considerações

Neste capítulo foram discutidos alguns conceitos básicos que fundamentam este trabalho, como os bancos de dados distribuídos, ontologias, e lógica descritiva. Também foi apresentada a linguagem de consultas em ontologias, a SPARQL. O entendimento desses conceitos é muito útil para a compreensão do contexto no qual o projeto foi desenvolvido.

3. *Peer-to-Peer* e PDMS

Este capítulo discutirá conceitos relacionados a sistemas *peer-to-peer*, com enfoque nas principais topologias envolvendo esses tipos de aplicações. Outra parte do capítulo será destinada a apresentar mais detalhes e características dos PDMS.

3.1 Sistemas e Aplicações Peer-to-Peer

O termo *peer-to-peer* (par-a-par) se refere a uma gama de sistemas e aplicações que empregam recursos de maneira distribuída para realizarem suas atividades de uma maneira descentralizada [20]. Numa rede P2P, as unidades de processamento não possuem um papel fixo para comunicação com outras máquinas, podendo agir tanto como um cliente como um servidor.

Como dito, a organização de uma rede P2P é descentralizada, ou seja, não existe um gerenciamento central. Por isso, quando há a transmissão de dados entre dois pontos da rede, a informação costuma passar por vários pontos desde a origem até alcançar o destino pretendido.

Com o intuito de compartilhar dados não estruturados, o Napster [22] foi o primeiro grande sistema enquadrado no paradigma P2P a surgir na mídia, principalmente graças à possibilidade de compartilhamento de músicas, e por isso, foi alvo de ataques jurídicos de companhias fonográficas. Após o Napster, várias outras aplicações desse tipo surgiram.

As aplicações *peer-to-peer* possuem características importantes como escalabilidade, volatilidade, autonomia dos pontos, roteamento, entre outras.

A Figura 06 ilustra um gráfico relativo às tendências de protocolos na Internet, do ano de 1993 a 2006.

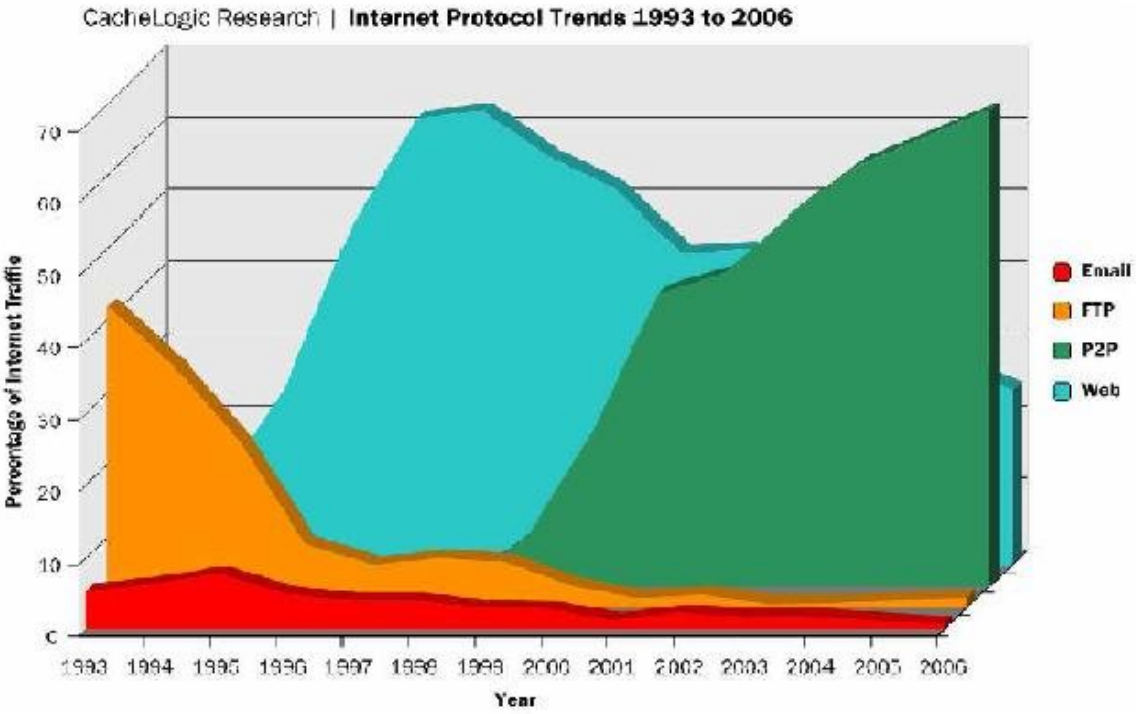


Figura 06 – Tendências de protocolos da Internet [23]

3.2 Topologias P2P

De acordo com as diferentes organizações e papéis de cada ponto numa rede P2P, estas redes foram categorizadas em diferentes topologias. As principais serão descritas nesta seção.

3.2.1 Topologia P2P Pura

Como comentado anteriormente neste capítulo, na topologia P2P pura, ilustrada na Figura 07, não existe centralização no processamento, todos os *peers* exercem as mesmas funções na rede, não existindo hierarquia, ao contrário, por exemplo, de um sistema centralizado de cliente e servidor.

Como dito anteriormente, a informação, ao trafegar por uma rede P2P pura, passa por outros pontos até chegar ao destino pretendido. Os outros pontos apenas repassam a informação, quando esta não é direcionada a eles.

Uma virtude da topologia P2P pura é a escalabilidade, já que qualquer ponto pode se juntar a uma rede P2P e começar a trocar informações com outros

pontos. Essas redes também são tolerantes a falhas, pois se algum *peer* em particular da rede falha, não impacta no restante do sistema [21].

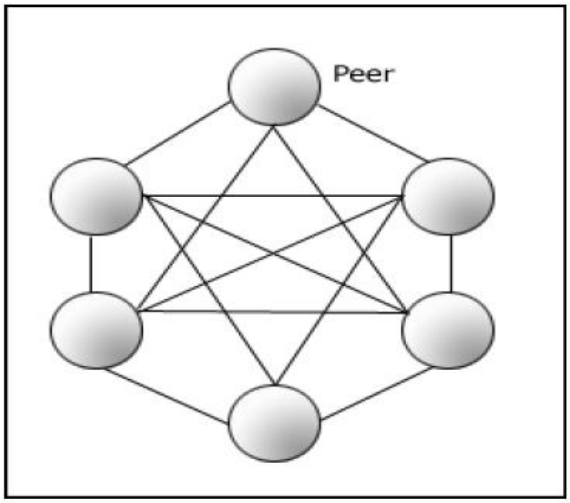


Figura 07 – Topologia P2P Pura [21]

3.2.2 Topologia P2P Híbrida

Na topologia P2P híbrida, ilustrada na Figura 08, há a presença de um ou mais servidores centrais para controle da troca de informações, enquanto o fluxo de dados ocorre da mesma maneira que na topologia P2P pura. A presença do servidor ameniza o problema de falta de gerenciamento, presente na topologia P2P pura. O papel principal do servidor é o monitoramento dos outros pontos da rede, garantindo a coerência na informação trocada por eles [21].

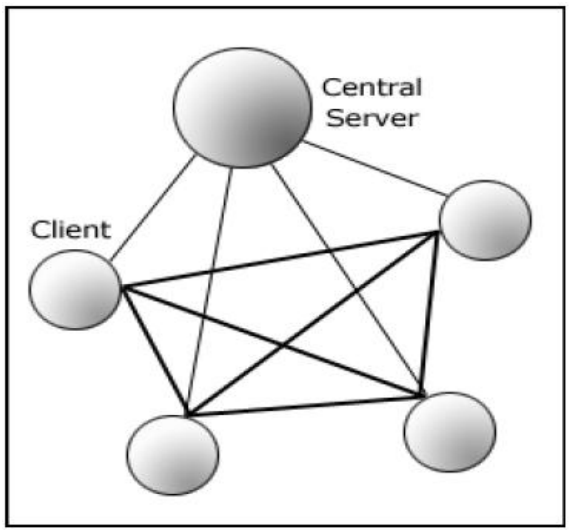


Figura 08 – Topologia P2P Híbrida [21]

Neste tipo de topologia, há o mesmo problema que pode ocorrer num sistema centralizado; caso o sistema central sofra algum tipo de falha, a rede perde a habilidade de gerenciamento e controle da troca de informações. Com o fluxo de informações um pouco mais centralizado em comparação com uma rede P2P pura, este tipo de topologia não é indicado para sistemas que necessitam de uma alta escalabilidade.

3.2.3 Topologia P2P *Super-Peer*

Na topologia *super-peer*, como mostra a Figura 09, a arquitetura é composta de uma mistura entre a abordagem centralizada, com existência de gerenciamento, e a abordagem descentralizada, na busca e troca de dados entre os pontos.

Alguns pontos podem assumir papéis distintos, sendo eleitos os *peers* de maior capacidade computacional para coordenar um subconjunto de pontos da rede. Esse ponto é chamado de *super-peer* [21].

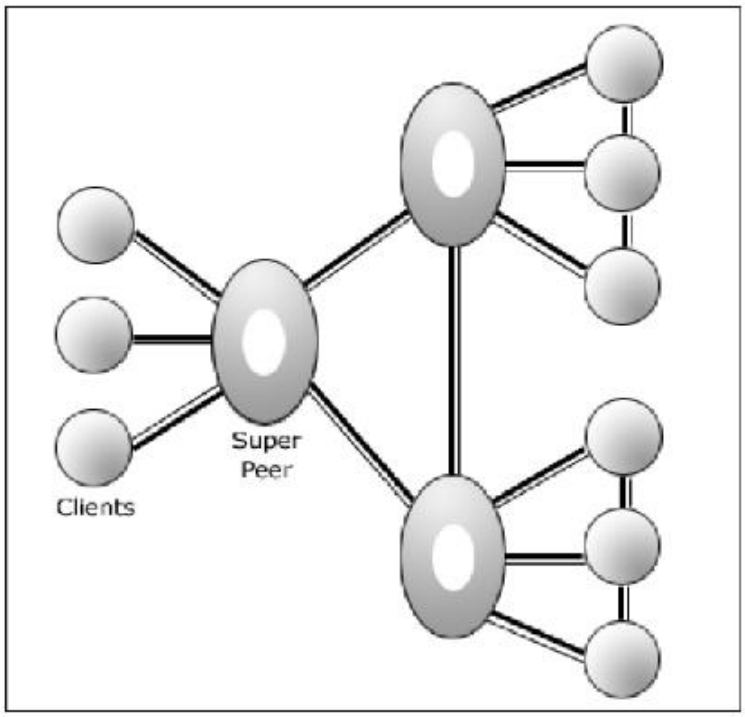


Figura 09 – Topologia P2P *Super-peer* [21]

Em [21], uma série de vantagens são listadas sobre a topologia P2P *super-peer*, que serão destacadas a seguir.

A busca nesse tipo de topologia é muito rápida em comparação a outras arquiteturas, já que o sistema possui seu espaço de busca particionado num conjunto menor de *peers* coordenados por *super-peers*, os quais possuem a informação de seus *peers* indexada.

Essa arquitetura define várias unidades autônomas colaborando entre si. Cada conjunto de pontos sob o mesmo *super-peer* (também chamado de *cluster* do *super-peer*) corresponde a uma unidade autônoma, no sentido de não haver dependência de um servidor central para a troca de informações.

Os *super-peers*, que são mais confiáveis e possuem melhor poder de processamento, monitoram a atividade de seus *clusters*. Isso garante um controle contra atividades maliciosas na rede. Nesta topologia, há também um melhor balanceamento das responsabilidades, dependendo da capacidade do *peer*, o que previne uma queda de performance graças a uma fragmentação da rede, no caso de todos os *peers* possuírem mesma responsabilidade e muita carga ficar sobre um *peer* com pouca capacidade.

A Tabela 01 compara as topologias discutidas neste capítulo, incluindo também a comparação com sistemas centralizados (estilo cliente–servidor), com relação a gerenciamento, coerência da informação, escalabilidade e confiabilidade.

Topology	Manageable	Coherent	Scalable	Reliability
Centralized	Yes	Yes	No	No
Decentralized	No	No	Yes	Yes
Hybrid P2P	Yes	Yes	Yes	No
Super-Peer	Yes	Yes	Yes	Yes

Tabela 01 – Comparação entre as topologias P2P e Sistema Centralizado [21]

Com o aumento da utilização das aplicações P2P, surgiram os sistemas PDMS, cujas características serão descritas na próxima seção.

3.3 PDMS

Como já foi citado anteriormente, um PDMS é uma aplicação *peer-to-peer* na qual cada um de seus elementos constituintes são *peers* de dados autônomos que podem compartilhar seus esquemas de dados entre si de forma total ou parcial.

Nesta seção, os PDMS serão mais bem caracterizados, tendo como enfoque o processamento de consultas em tais sistemas, com uma discussão sobre reformulação e enriquecimento de consultas.

3.3.1 Conceitos de PDMS

Os sistemas PDMS foram conceituados no Capítulo 1, mas podemos defini-los em outras palavras como sistemas de gerenciamento de dados *peer-to-peer* que provêem um ambiente de integração de dados armazenados em fontes autônomas, heterogêneas e dinâmicas.

A forma de compartilhamento de dados entre os *peers* em um PDMS é chamada de exportação de esquemas. Um PDMS possui as seguintes características:

- Compartilhamento descentralizado dos dados;
- Escalabilidade;
- Processamento e armazenamento de dados distribuídos é feito a partir de *peers* autônomos, que também armazenam os mapeamentos semânticos dos dados.

A organização descentralizada é importante, como já foi discutido na Seção 3.2 deste capítulo, para o PDMS não ficar inviabilizado com sua rede de *peers* caso algum ponto em particular venha a falhar. Os pontos em um PDMS são considerados autônomos devido à sua liberdade de entrada ou saída de uma rede, ou até mesmo sobre possíveis alterações em seus esquemas de dados.

Um dos serviços mais importantes em um PDMS é o processamento de consultas nos *peers*. Esse assunto será discutido na próxima seção.

3.3.2 Consultas em PDMS

Quando uma consulta é submetida em um ponto, o sistema que possui a rede consegue obter dados relevantes não só do ponto que recebeu a consulta, mas de qualquer outro conectado a este através de algum caminho semântico. Ao contrário de uma hierarquia estrita de integração de dados, um PDMS suporta qualquer rede de relacionamentos entre os *peers* [2].

Uma grande vantagem dos sistemas PDMS é a de que qualquer ponto pode submeter uma consulta na rede utilizando apenas seu próprio esquema de dados, sem precisar aprender esquemas de outros pontos. Com isso, através dos caminhos semânticos existentes de que o *peer* da consulta participa, a consulta original pode ser reformulada (adequada) aos outros pontos do caminho semântico, de acordo com seus respectivos esquemas de dados.

Por exemplo, se uma consulta Q é submetida em um *peer* A, o PDMS geralmente retorna primeiro um resultado baseando-se nos dados de A (também pode ocorrer de o PDMS integrar todos os resultados e retorná-los de uma vez). Então, a consulta Q é reformulada para os vizinhos semânticos de A e outros resultados são retornados baseados em outros esquemas de dados. Isso auxilia muito para uma resposta mais completa a respeito de um certo domínio.

Na Figura 10, uma consulta submetida no ponto Roma pode obter dados do ponto Paris, seguindo o caminho através de *DB-Projects* e Stanford.

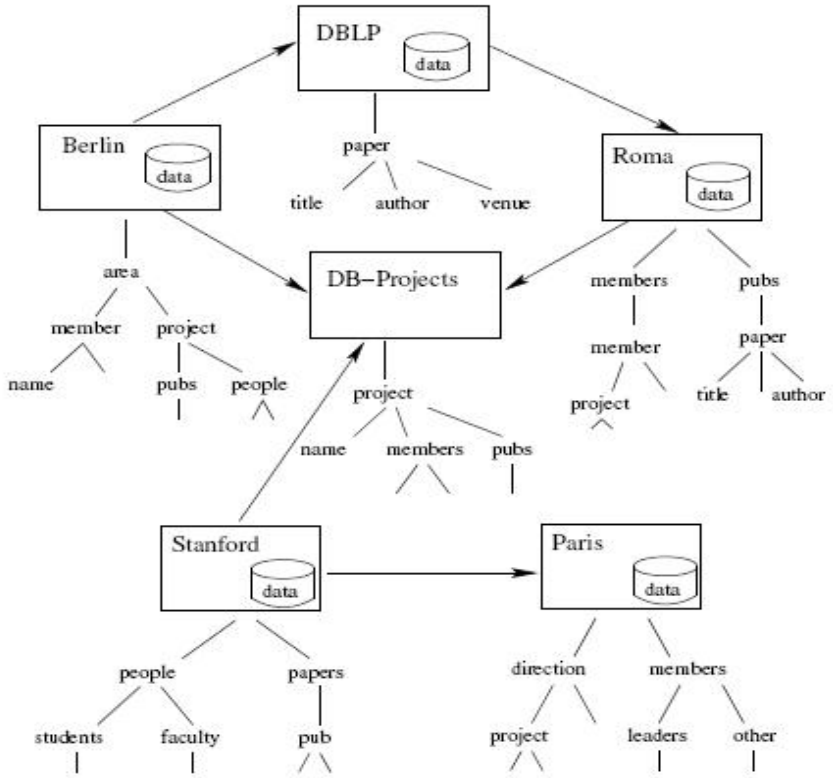


Figura 10 – Exemplo de PDMS e caminhos semânticos entre os *peers* [2]

Dependendo do caminho semântico seguido pela consulta, novas respostas podem ser obtidas. Isso pode resultar em alguns problemas; a escolha do caminho semântico a ser seguido é muito importante. Alguns exemplos de problemas ligados à escolha de um caminho incorreto citados em [2] são:

- Surgimento de respostas redundantes. Cada reformulação desnecessária degrada a performance do sistema;
- Um caminho que encontre um final muito cedo, passando por poucos pontos (ou, em outras palavras, podado precocemente);
- Reformulações ineficientes, ou seja, consultas em pontos que poderiam ser mais bem aproveitadas (otimizadas, ou enriquecidas) antes de serem executadas.

Esses problemas constituem um grande impedimento para reformulações de consultas cada vez mais eficientes e completas em PDMS.

Os pontos de uma rede em um PDMS não possuem a informação completa de um domínio, por isso há uma grande importância em se escolher um bom caminho semântico quando uma consulta é submetida, a fim de se extrair mais informações.

A importância do enriquecimento de uma consulta está no fato de se conseguir uma maior exploração de um *peer* que contenha, por exemplo, mais informações relevantes em seu esquema de dados. Com isso, um resultado mais rico (com mais dados) é alcançado.

3.4 Considerações

Neste capítulo foram abordadas as principais características referentes à tecnologia *peer-to-peer*, dando ênfase às topologias P2P. Também foi feita uma discussão a respeito dos PDMS, sistemas nos quais seus elementos constituintes são *peers* de dados autônomos. A discussão do funcionamento de consultas em PDMS é particularmente interessante para o escopo deste projeto.

4. O Sistema SPEED

Este capítulo tem como objetivo introduzir e apresentar maiores detalhes do sistema SPEED (*Semantic PEEr-to-Peer Data Management System*). As próximas seções descreverão as características do SPEED, sua arquitetura, e como funciona o processamento de consultas no sistema.

4.1 Arquitetura do Sistema

No SPEED, os pontos são organizados de acordo com o conteúdo que compartilham. Existem três tipos de pontos e relacionamentos entre eles no sistema: os pontos semânticos, os pontos de integração e os pontos de dados. A Figura 11 ilustra a arquitetura do SPEED.

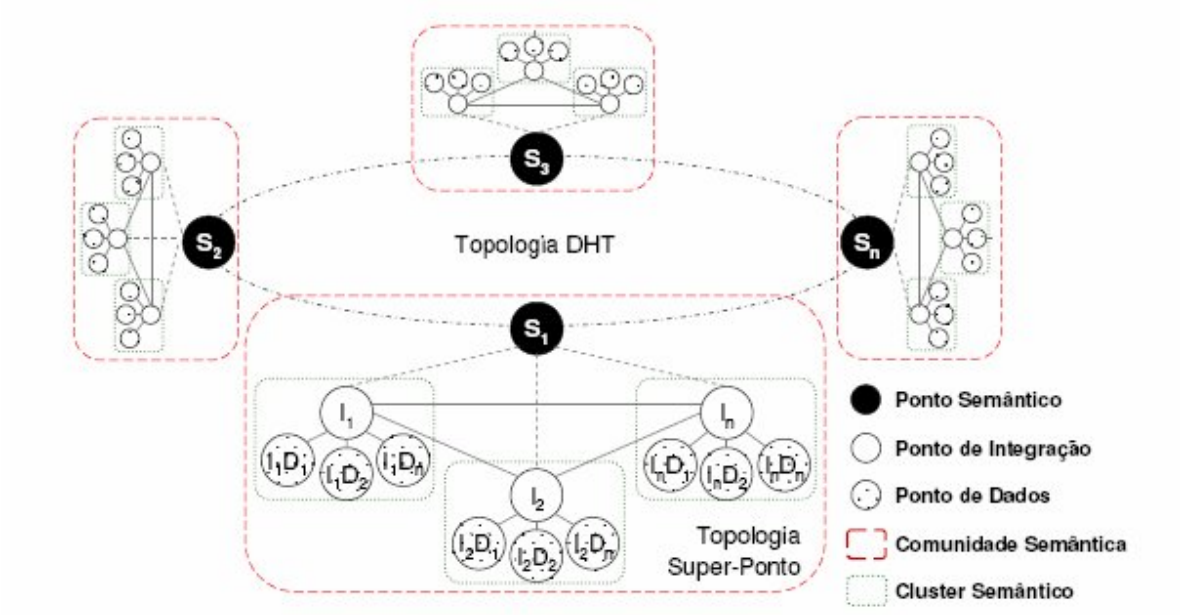


Figura 11 – Arquitetura do SPEED [1]

O sistema foi projetado com duas topologias de redes distintas: a DHT (*Distributed Hash Table*), como um anel superior e mais abstrato, que organiza os *peers* semânticos, e a *super-peer* para o gerenciamento de pontos de integração e pontos de dados.

A rede DHT é utilizada para auxiliar um *peer* a encontrar outros *peers* com características comuns através de uma função *hash*, e posteriormente formarem

comunidades semânticas. Dentro das comunidades semânticas, os *peers* são organizados de acordo com a topologia *super-peer* [1].

Um ponto de dados representa uma fonte de dados compartilhando dados estruturados ou semi-estruturados com outros pontos de dados no sistema, através de mapeamentos semânticos. Os *peers* de dados são os locais onde são realizadas as consultas no sistema. Na Figura 11, I_1D_1 e I_1D_2 são exemplos de pontos de dados.

Os pontos de dados são agrupados em *clusters* semânticos de acordo com seus interesses semânticos, ou seja, tema de interesse abordado e estrutura de sua ontologia local. O tema de interesse é uma descrição do domínio semântico do ponto, enquanto a ontologia local descreve o esquema exportado [1].

Cada *cluster* semântico possui um tipo diferenciado de ponto de dados com outras características e mais capacidade computacional. Esses pontos são os chamados pontos de integração, e possuem um conhecimento detalhado sobre os pontos de dados de seu *cluster*. Os *peers* de integração se comunicam com outros *peers* de integração e também com os *peers* de dados de seu *cluster* semântico. Sua maior capacidade computacional é refletida em suas atribuições, como, por exemplo, controlar e processar as consultas sobre os *peers* de dados. Exemplos de *peers* de integração na Figura 11 são I_1, I_2 e I_n.

Um ponto de integração mantém uma ontologia que representa seu *cluster*, obtida através de uma operação de *merge* entre as ontologias locais dos pontos de dados. Pontos de integração se comunicam também com o outro tipo de ponto do SPEED, o ponto semântico, que é responsável por armazenar e disponibilizar uma ontologia representativa de uma comunidade contendo elementos que caracterizem o conhecimento de um domínio.

Na Figura 11, S_1 é um exemplo de ponto semântico. Outra responsabilidade desse tipo de ponto é gerenciar os metadados dos pontos de integração que se conectem a ele. Um conjunto de *clusters* semânticos que

compartilhem informações a respeito de um mesmo domínio forma, junto com um ponto semântico, comunidades semânticas.

4.2 Principais Componentes

Esta seção abordará como os *peers* podem assumir diferentes papéis no SPEED. Suas arquiteturas são compostas de diferentes componentes, como mostra a Figura 12.

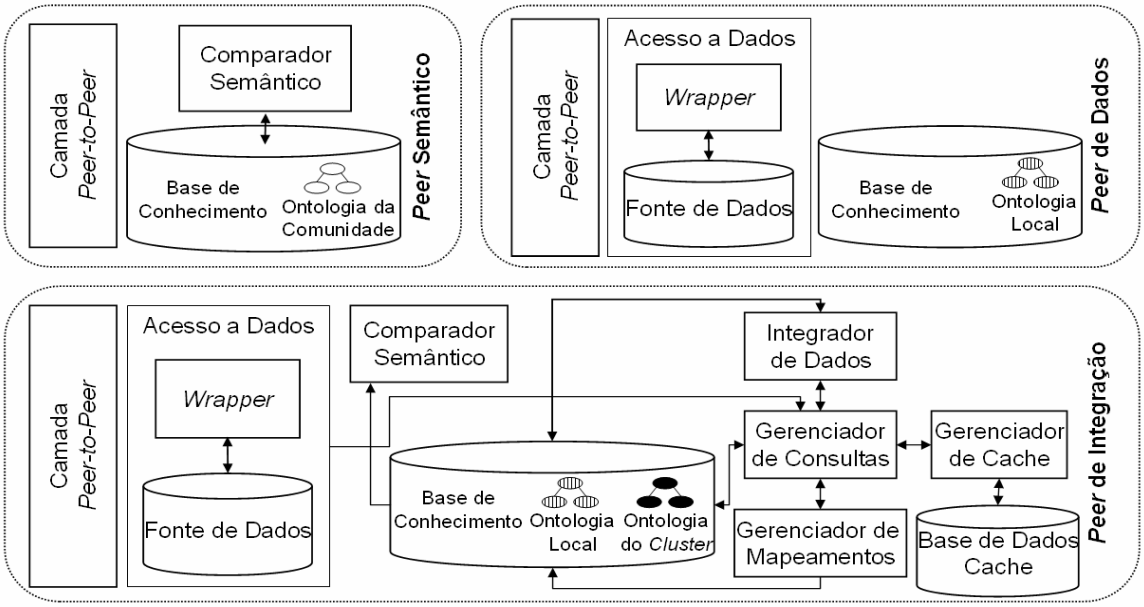


Figura 12 – Componentes da arquitetura dos *peers* do sistema SPEED [1]

Os pontos de dados são qualquer tipo de ponto (um computador simples ou um servidor, por exemplo), que pode freqüentemente conectar ou desconectar do sistema. Esses pontos compartilham dados com outros pontos da rede e seus principais componentes da arquitetura incluem: uma camada P2P, que faz a comunicação com o ponto de integração; uma base de conhecimento, que armazena os metadados do ponto; uma camada de acesso aos dados, composta de um *wrapper*, responsável por traduzir dados da fonte para o modelo de dados comum [1].

Um ponto de integração é um tipo de ponto especial em um *cluster* semântico que oferece maiores recursos computacionais. Os componentes internos dos pontos de integração e dos pontos de dados são praticamente os mesmos. Os

pontos de integração guardam as informações detalhadas dos pontos de dados ligados a ele em uma base de conhecimento. Essa base armazena informações para fazer a associação de esquemas exportados pelos pontos de dados.

Outro componente de um ponto de integração é o gerenciador de consultas, que processa as consultas que são enviadas para ele, e a partir daí reenvia essas consultas aos pontos de dados que são capazes de respondê-las satisfatoriamente, além de encaminhá-las também a outros pontos de integração. O integrador de resultados é o componente dos pontos de integração que junta os resultados dessas consultas que são recebidos dos pontos de dados, e então envia o resultado integrado ao ponto que submeteu a consulta originalmente.

O terceiro tipo de ponto, o ponto semântico, representa o ponto de entrada das comunidades semânticas do SPEED, e atua como um servidor de ontologias. Essas ontologias são utilizadas para distribuir pontos de dados e de integração em comunidades e *clusters* semânticos. Através de sua camada P2P, o ponto semântico se comunica com os pontos de integração assim como com outros pontos semânticos. Sua base de conhecimento armazena a ontologia de sua comunidade semântica assim como outras informações relevantes para gerenciamento e manutenção da comunidade.

4.3 Processamento de Consultas

Quando um ponto de dados se conecta ao SPEED, este primeiramente identifica a qual comunidade semântica deverá pertencer, através de palavras-chave dos domínios existentes, utilizando a estrutura DHT do sistema. Após essa etapa, a busca pelo *cluster* semântico a que ele pertence é realizada através de uma operação de *matching* entre as ontologias. O comparador semântico (Figura 12) realiza um *matching* entre a ontologia do ponto e a ontologia do *cluster*.

A operação de *matching* resulta em uma medida de similaridade; a maior medida de similaridade encontrada identifica o *cluster* do qual o ponto fará parte. Como resultado também dessa operação de *matching*, um conjunto de

correspondências semânticas é gerado entre a ontologia do ponto e a ontologia do *cluster* (todos os pontos de um *cluster* possuem essas correspondências).

As correspondências semânticas são os relacionamentos identificados entre os conceitos / propriedades dos pontos.

Os pontos de integração do SPEED também possuem correspondências semânticas entre si. Os grupos de pontos de integração que se relacionam de tal maneira são chamados de vizinhos semânticos. Para se determinar esses vizinhos, uma função de similaridade é calculada entre pares de pontos de integração, e aquelas com maior valor (do que um limiar pré-determinado) determinam os vizinhos.

Essas correspondências são importantes para o processamento de consultas no SPEED. Quando um usuário submete uma consulta num ponto de dados, a consulta vai até o ponto de integração daquele *cluster*. O ponto de integração identifica os pontos de dados capazes de responder àquela consulta, ao mesmo tempo em que a consulta é propagada para outros pontos de integração da mesma comunidade.

Os pontos capazes de responder à consulta são chamados de pontos relevantes (*relevant peers*). Através das correspondências semânticas existentes entre os pontos da comunidade (entre ponto de integração e pontos de dados e entre pontos de integração vizinhos) a consulta original é reformulada a fim de 'se adaptar' de acordo com os esquemas dos pontos relevantes, mas sem modificar sua semântica (essa reformulação considerada no caso da propagação da consulta entre os pontos de integração). No final, os resultados recebidos dos pontos que executaram a consulta são integrados no ponto de integração que propagou a consulta, e o resultado final é enviado ao ponto que a submeteu.

Essa fase de reformulação da consulta é muito interessante, pois permite ao usuário obter resultados mais completos para suas necessidades e explorar melhor os pontos relevantes de uma comunidade semântica. Lembrando que uma mesma comunidade é formada por *peers* do mesmo domínio de aplicação (mesma

semântica) e, portanto, o usuário não receberá resultados indesejados, mas sim resultados mais ricos semanticamente.

Um exemplo simples da importância das correspondências semânticas é a existência, em uma comunidade qualquer, de uma relação de equivalência entre o conceito *Teacher* e o conceito *Professor* (em *peers* distintos na mesma comunidade). O usuário, ao submeter uma consulta procurando por *Teacher*, receberá resultados referentes também aos outros conceitos semanticamente equivalentes, graças ao relacionamento de equivalência existente entre os pontos.

O usuário terá essa opção de enriquecer semanticamente a sua consulta quando for submetê-la. Isso tudo será realizado com o módulo de reformulação de consultas do sistema. Este módulo (sua especificação e implementação) será detalhado no próximo capítulo.

4.4 Considerações

Este capítulo abordou o SPEED, suas características, arquitetura e como funciona o processamento de consultas no sistema. A arquitetura do SPEED é formada por duas topologias: uma rede DHT e uma rede *super-peer*. Os principais componentes são os *peers* de dados, os *peers* de integração e os *peers* semânticos. Os diferenciais do sistema são o uso da semântica e a topologia mista. O uso da semântica facilita os mapeamentos entre esquemas e o processamento de consultas.

5. O Módulo de Reformulação de Consultas

Este capítulo descreve o módulo de reformulação de consultas (ou, mais abreviadamente, módulo de consultas) do sistema SPEED, tema central deste trabalho. Primeiramente, serão descritos os objetivos e características do módulo, através de sua especificação. Após isso, será detalhado o processo de implementação do módulo.

5.1 Especificação

Este trabalho tem como objetivo o desenvolvimento do módulo de reformulação de consultas no sistema SPEED, cujo propósito é ser um sistema de gerenciamento de dados para ambientes P2P. O sistema SPEED provê facilidades para o compartilhamento e integração dos dados distribuídos ao longo de pontos conectados.

Um dos principais serviços oferecidos pelo SPEED é o processamento de consultas. No SPEED, pontos de integração são responsáveis por processar consultas. Como mencionado anteriormente, os pontos de integração recebem a consulta, analisam, reformulam, enviam a outros pontos e por fim integram os resultados gerando uma resposta final ao usuário no ponto que submeteu a consulta.

O ponto crítico nesse processo é a reformulação da consulta entre pontos através de caminhos de correspondências semânticas disponíveis. Considerando a semântica implícita existente nas correspondências entre os pontos, é possível, no momento da reformulação, permitir o enriquecimento da consulta, provendo os usuários com resultados mais significativos [33].

O módulo de consultas do SPEED deve permitir a submissão de consultas por parte do usuário em um dado ponto de integração (ponto de submissão). O módulo deve ainda identificar a semântica das consultas, e a partir de correspondências semânticas entre os *peers*, reformular a consulta (com ou sem enriquecimento) e enviá-la a outros pontos de integração que possam respondê-la satisfatoriamente, gerando resultados mais completos.

A consulta é processada no SPEED em dois níveis: dentro do *cluster*, através das fontes de dados acopladas aos pontos de dados, e entre os pontos de integração vizinhos ao ponto que originalmente recebeu a consulta. Nesse sentido, o escopo tratado neste trabalho se resume à reformulação de consultas entre pontos de integração. A reformulação considerando pontos de dados será objeto de estudo posterior e será realizado com base na implementação atual de um sistema de integração chamado Integra [32].

A idéia é extrair semântica das correspondências entre pontos e utilizá-la no enriquecimento e reformulação da consulta. Para tal, o usuário tem a opção de permitir ou não que esta consulta seja enriquecida, por intermédio de variáveis que organizarão o escopo da reformulação da consulta. Caso o usuário opte pelo enriquecimento, mais resultados significativos serão disponibilizados. Caso a opção seja por um tempo de resposta mais curto, a reformulação seguirá caminhos mais rápidos, através da geração de reformulações sem a opção de enriquecimento.

Uma visão geral da arquitetura da ferramenta de consultas do SPEED é ilustrada na Figura 13. Considerando a referida figura, este trabalho tem como foco os módulos *Query Handler* e *Query Reformulator*, ressaltados no retângulo tracejado.

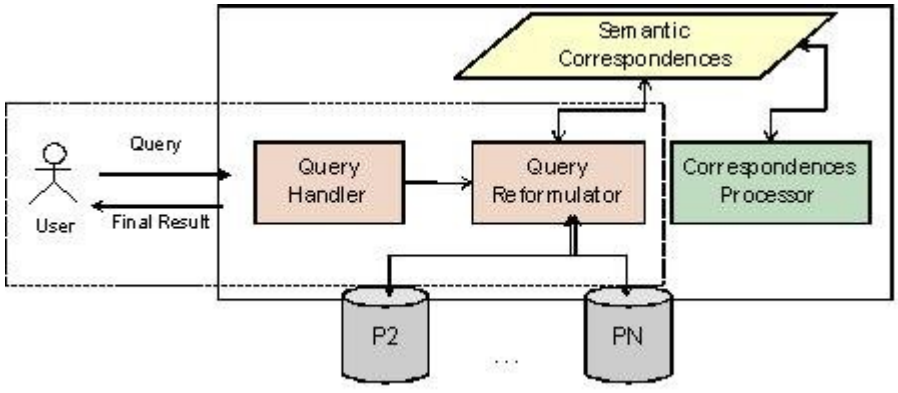


Figura 13 – Visão geral da arquitetura do módulo de Consultas do SPEED

No SPEED, cada ponto de dados (e de integração) representa uma fonte que tem seu esquema exportado para um formato padrão – OWL [9]. Diante disto, a interface de consultas permite a formulação da consulta de três maneiras: em lógica descritiva (ALC-DL) [31], SPARQL [15], ou usando conceitos da ontologia do ponto. Na versão atual do módulo, apenas a opção de formulação através da ALC-DL está implementada, mas, cada consulta em DL será traduzida para uma linguagem meio, a SPARQL. Depois de reformulada e traduzida, será executada sobre ontologias representando as fontes de dados dos pontos de integração vizinhos.

Para isso, algumas funcionalidades foram identificadas:

- Uma interface para submissão da consulta, configuração das variáveis de reformulação e visualização dos resultados;
- Identificação da semântica da consulta (entidades, operadores);
- Verificação das correspondências semânticas que poderão ser utilizadas;
- Enriquecimento da consulta, de acordo com as correspondências;
- Reformulação da consulta;
- Envio da consulta ao ponto vizinho;
- Execução da consulta pelo ponto vizinho;
- Retorno da consulta;
- Integração dos resultados;
- Apresentação dos resultados.

5.1.1 Descrição Técnica

As funcionalidades esperadas do módulo de consultas também foram especificadas numa forma mais técnica:

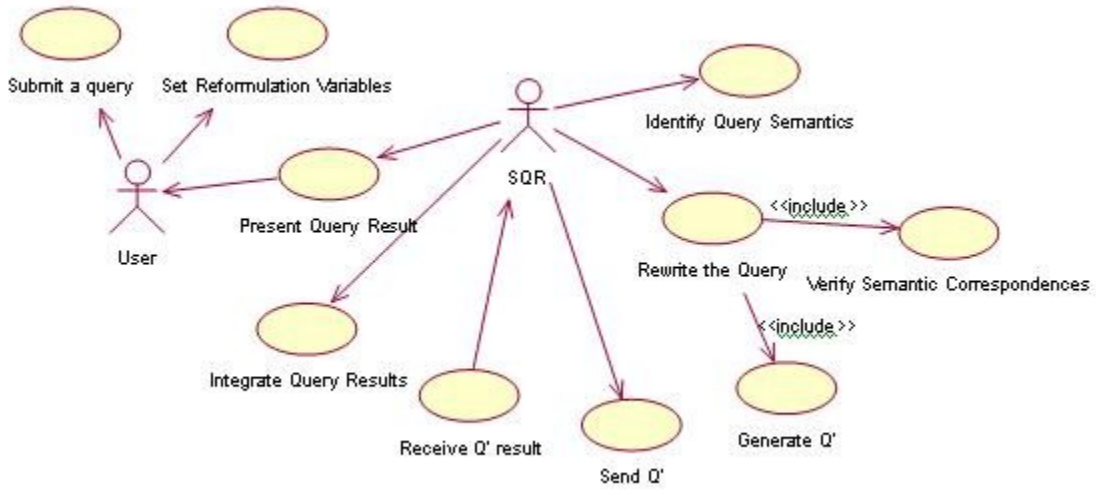
- Entrada: Consulta Q
- Saída 1: Q’ reformulada (possivelmente enriquecida), a ser enviada ao ponto vizinho

- Saída 2: Resultado da consulta Q, depois de reformulada e executada pelos pontos vizinhos
- Requisitos Funcionais:
    - Prover interface de submissão da consulta
    - Prover ao usuário opções de configuração da execução da consulta
    - Identificar a semântica da consulta: entidades e operadores
    - Verificar correspondências semânticas, de acordo com a semântica da consulta (entidades e relacionamentos entre elas)
    - Executar a reformulação da consulta com ou sem enriquecimento
    - Gerar Q'
    - Enviar Q' ao ponto vizinho
    - Executar Q'
    - Retornar o resultado de Q' ao ponto de submissão
    - Integrar resultados no ponto de submissão
    - Apresentar resultado final ao usuário
- Requisitos Não-Funcionais:
    - Ser independente de plataforma
    - Executar no ambiente de pontos de integração do SPEED
    - Interface amigável
- Usuário do módulo: qualquer usuário que necessite formular uma consulta através do sistema SPEED

5.1.2 Casos de Uso

No Quadro 03 está especificado o diagrama de casos de uso (em inglês) para o módulo de reformulação de consultas do SPEED, e logo a seguir estão as descrições de tais casos de uso.

Casos de Uso (em inglês):



Quadro 03 – Casos de Uso referentes ao módulo de consultas do SPEED

Descrição dos Casos de Uso:

Caso de Uso 1:	*Submit a Query*
Entrada:	Conjunto de elementos da ontologia do *cluster*, disponibilizados através da interface.
Saída:	Consulta Q formulada
Descrição:	O usuário irá formular uma consulta com base nas entidades existentes na ontologia disponibilizada, através da interface de consulta.
Observação:	A ontologia a ser disponibilizada será a ontologia do *cluster* (do ponto de integração).

Caso de Uso 2:	*Set Reformulation Variables*
Entrada:	Conjunto de variáveis a serem customizadas
Saída:	Conjunto de variáveis customizadas
Descrição:	O usuário irá configurar um conjunto de opções relacionadas à reformulação da consulta.
Observação:	As opções (variáveis) são: *generalize, compose, approximate.*

Caso de Uso 3:	*Identify Query Semantics*
Entrada:	Q formulada
Saída:	Q estruturada semanticamente: entidades + operadores
Descrição:	Depois de formulada, o *query handler* vai identificar a semântica da consulta Q, isto é, identificar quais as entidades solicitadas, e quais operadores estão sendo empregados.

Caso de Uso 4:	*Verify Semantic Correspondences*
Entrada:	Entidades identificadas na consulta Q
Saída:	Correspondências semânticas associadas às entidades solicitadas na consulta Q
Descrição:	Serão identificadas e recuperadas, pelo *Query Reformulator*, as correspondências semânticas de cada entidade da consulta.
Observação:	Este procedimento, na verdade, é parte do algoritmo geral de reformulação semântica.

Caso de Uso 5:	*Rewrite the Query*
Entrada:	Q + Semântica da Consulta
Saída:	Q' reescrita e possivelmente enriquecida
Descrição:	Este caso de uso reflete o algoritmo de reformulação semântica da consulta. Assim, ele recebe a consulta Q, sua semântica, verifica operadores, identifica as correspondências que podem ser utilizadas e enriquece a consulta com as mesmas, de acordo com o que o usuário definiu na configuração inicial.

Caso de Uso 6:	*Receive Q' result*
Entrada:	Resultado de Q' de um ponto de integração vizinho
Saída:	Resultado armazenado
Descrição:	Cada ponto de integração retorna seu resultado ao ponto de integração que deu origem à mesma. Este deve salvá-lo a fim de poder integrá-lo aos demais resultados.

Caso de Uso 7:	*Integrate Q' results*
Entrada:	Resultados de Q'
Saída:	Resultado integrado

Descrição:	Todos os resultados das execuções de Q' são integrados.

Caso de Uso 9:	*Present Q' Final Result*
Entrada:	Resultado final integrado de Q'
Saída:	Resultado apresentado na interface do usuário
Descrição:	O resultado vai ser apresentado na interface do usuário que formulou a consulta, de acordo com as características da consulta, da interface e do resultado.

5.2 Implementação

Para a implementação do módulo de consultas do SPEED, foi escolhida a linguagem Java [30], com o auxílio da ferramenta Eclipse [28]. Java foi escolhida por ser uma linguagem independente de plataforma, além de permitir (através da ferramenta Eclipse) a criação de interfaces amigáveis necessárias ao sistema, como foi especificado nos requisitos não-funcionais.

Além disso, Java possui uma API (*Application Programming Interface*) [6] que dá suporte à programação com ontologias, facilitando sua manipulação e inclusive permitindo o desenvolvimento de métodos de consultas sobre as ontologias, com o uso da linguagem SPARQL [26, 27]. A implementação dos casos de uso, explicitados no Quadro 03, será detalhada a seguir.

A interface com o usuário permite a submissão de consultas bem como a escolha das variáveis de reformulação, que influenciam no tipo e na qualidade da resposta. Além disso, através da interface, o usuário tem a opção de visualizar um *log* de reformulações e um *log* de resultados para cada consulta submetida durante sua sessão.

A Figura 14 ilustra a tela inicial do módulo de consultas.

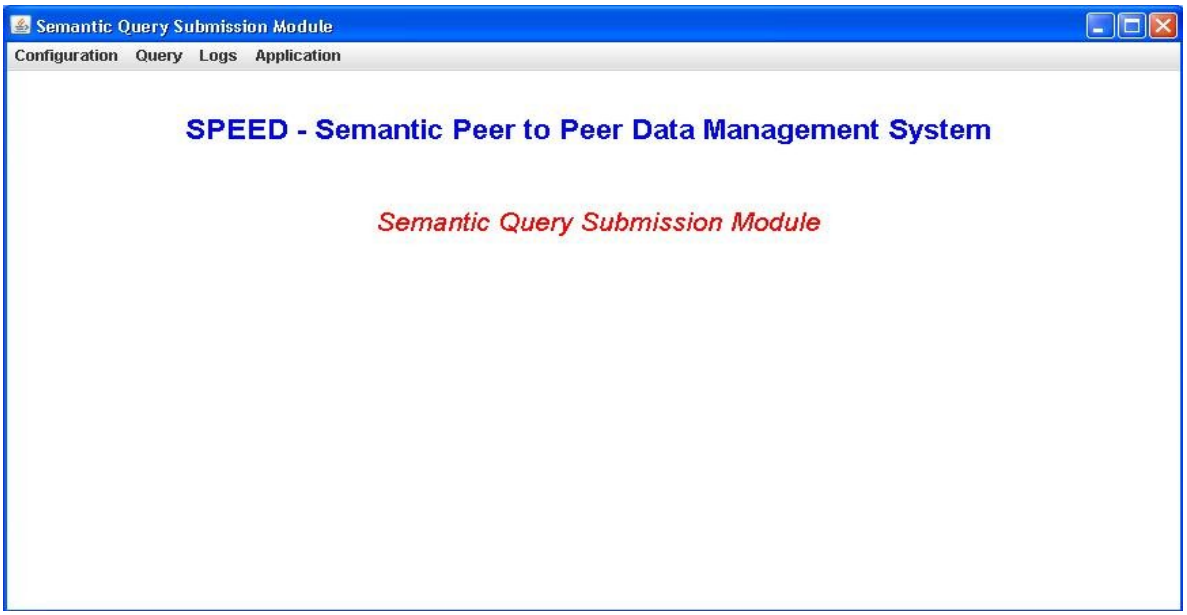


Figura 14 – Tela principal do módulo de consultas

Através do menu *Configuration -> Set Reformulation Variables*, o usuário acessa a tela para escolher as variáveis a serem consideradas na reformulação de suas consultas. A Figura 15 ilustra essa tela.

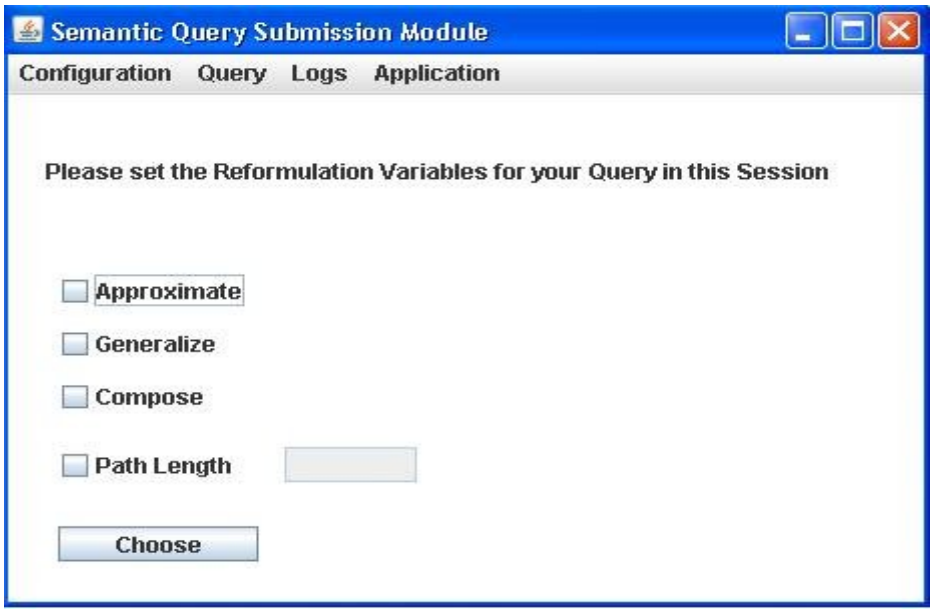


Figura 15 – Tela de escolha das variáveis de reformulação

A opção de *Path Length* não está habilitada para essa versão do módulo de consultas, visto que a reformulação está sendo realizada apenas entre dois *peers*. Estas variáveis influenciam na reformulação da consulta submetida, quando esta

está sendo reformulada. Relembrando que, para se reescrever uma consulta, é necessário verificar as correspondências semânticas entre os *peers* a fim de se gerar Q' (a consulta reformulada a ser executada no *peer* destino).

O módulo de consulta permite ao usuário executar uma consulta num ponto (*source peer*), enquanto esta mesma consulta é reformulada a partir de correspondências semânticas entre o *peer* de integração fonte e um *peer* de integração destino (*target peer*) (vizinho ao *peer* de integração fonte). Depois de reescrita, a consulta reformulada é executada no *peer* destino e, posteriormente, os resultados são integrados e exibidos na interface.

Para melhor entendimento do significado de cada variável de reformulação, o Quadro 04 ilustra um trecho de um arquivo de correspondências semânticas entre dois *peers* sobre o domínio Educação.

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1220896776.owl#Teacher">
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#Faculty</j.0:isCloseTo>
  <j.0:isSubConceptOf>http://www.owl-ontologies.com/Ontology1220896776.owl#Education</j.0:isSubConceptOf>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#AssistantProfessor</j.0:isCloseTo>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#Student</j.0:isCloseTo>
  <j.0:isSubConceptOf>http://www.owl-ontologies.com/Ontology1220896776.owl#Person</j.0:isSubConceptOf>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#Work</j.0:isCloseTo>
  <j.0:isSubConceptOf>http://www.owl-ontologies.com/Ontology1220896776.owl#Faculty</j.0:isSubConceptOf>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#Monitor</j.0:isCloseTo>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#FullProfessor</j.0:isCloseTo>
  <j.0:isCloseTo>http://www.owl-ontologies.com/Ontology1220896776.owl#Professor</j.0:isCloseTo>
</rdf:Description>
```

Quadro 04 – Trecho de um arquivo de correspondências semânticas

Neste trecho, podemos observar que o conceito *Teacher* tem certa semelhança (*close to*) com o conceito *Faculty* do outro *peer*, e também é subconceito (*subconcept*) de *Person*. Os relacionamentos possíveis entre dois conceitos são:

- Equivalência (*isEquivalentTo)*;
- Subconceito (*isSubConceptOf*);
- Superconceito (*isSuperConceptOf*);
- Semelhança (*isCloseTo*);
- "Conjunto de" (*isWholeOf*);

- "Parte de" (*isPartOf*).
- Disjunção (*Disjointness*)

A consulta original é formulada pelo usuário através da linguagem ALC-DL, explicitando os conceitos desejados e os construtores que deseja explorar (união, interseção ou negação). Nesta versão do módulo, as consultas formuladas não aceitam o uso de propriedades, mas apenas os conceitos das ontologias.

Por padrão, a reformulação da consulta é feita realizando-se uma busca no arquivo de correspondências dos conceitos equivalentes, no *peer* destino, aos conceitos desejados pelo usuário (lembrando que o usuário formula sua consulta para os conceitos da ontologia do *peer* fonte, escolhidos através da interface). Também é padrão na reformulação que sejam buscados os subconceitos dos conceitos especificados na consulta. Ao negar-se um conceito, a busca no arquivo de correspondências é algumas vezes realizada através das disjunções deste conceito (*disjointness*).

A escolha das variáveis de reformulação objetiva enriquecer a reformulação da consulta, visto que mais conceitos podem ser buscados no arquivo de correspondências, por meio dos outros tipos de relacionamentos, ocasionando assim uma exploração maior dos conceitos da ontologia do *peer* destino.

Cada variável identifica que tipo de relacionamento extra deve ser levado em consideração na reformulação da consulta:

- *Approximate*: Conceitos semelhantes (*isCloseTo*);
- *Generalize*: Superconceitos (*isSubConceptOf*);
- *Compose*: "Conjunto de" e "Parte de" (*isWholeOf* e *isPartOf*).

A reescrita da consulta original pode resultar em duas consultas para serem executadas no *peer* destino: uma consulta 1 (chamada de *Qexact*, ou consulta exata) derivada dos subconceitos e conceitos equivalentes (possivelmente de negação), e outra consulta 2 (chamada de *Qenriched*, ou consulta enriquecida)

derivada dos outros tipos de correspondências. A consulta 1 é sempre criada, já a consulta 2 depende da escolha das variáveis de reformulação pelo usuário.

Escolhidas as variáveis, o usuário pode partir para a submissão da consulta. Através do menu Q*uery -> Query Submission*, o usuário escolhe a ontologia que representa o *peer* fonte (Figura 16), e então é redirecionado para a tela de consultas, ilustrada na Figura 17.

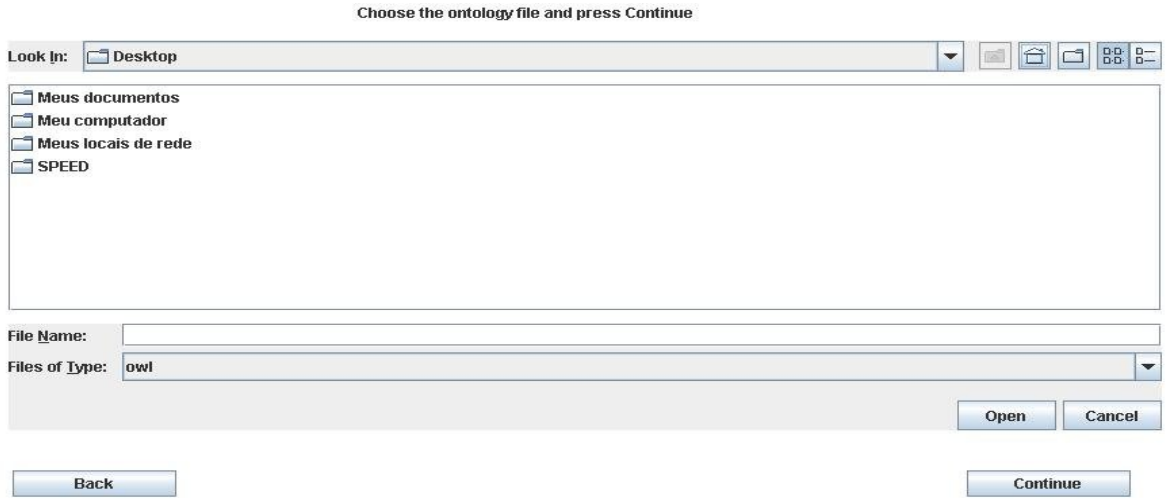


Figura 16 – Tela de escolha da ontologia

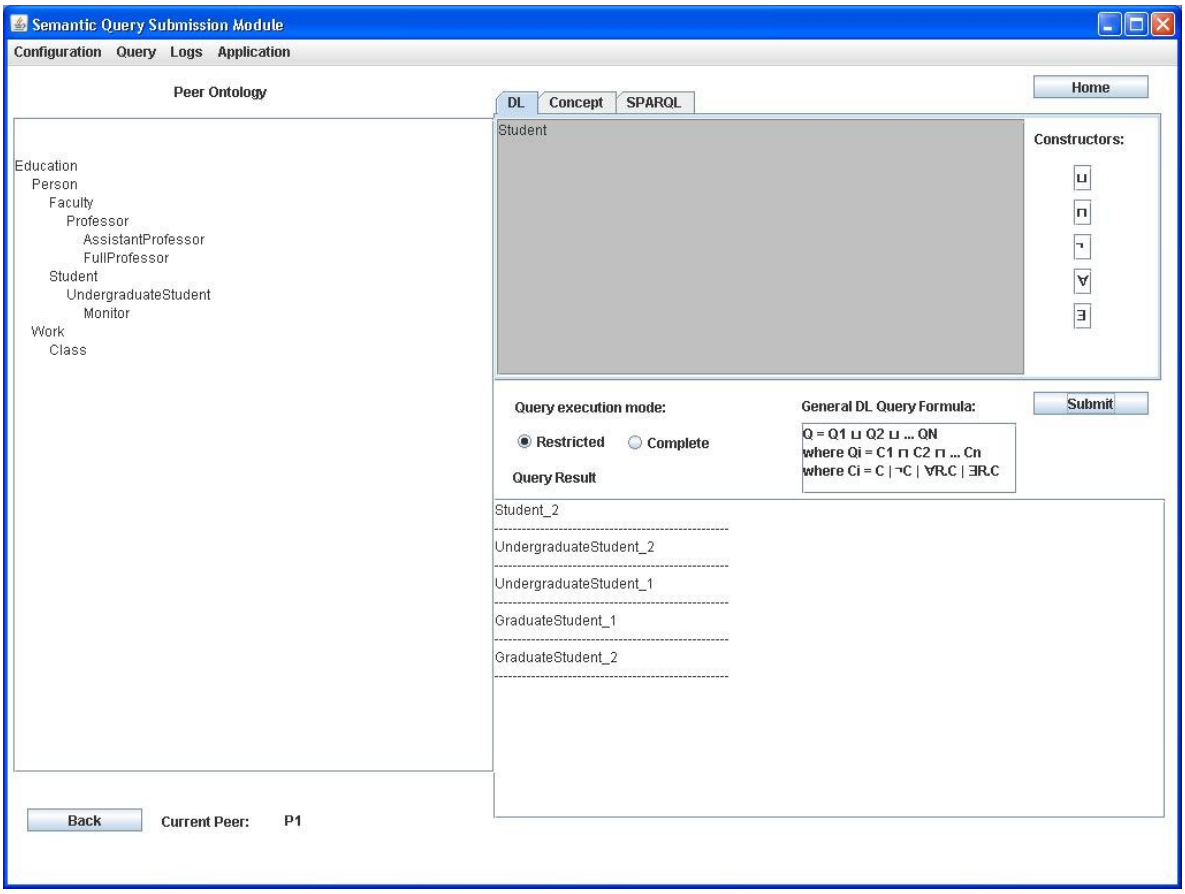


Figura 17 – Tela de consultas

Do lado esquerdo da tela é exibida a hierarquia dos conceitos da ontologia escolhida pelo usuário, especificando os conceitos que podem ser buscados neste *peer*. Do lado superior direito está a área de formulação das consultas em ALC-DL. Os construtores da DL estão especificados ao lado da área de consultas para utilização por parte do usuário. O usuário, ao clicar sobre um construtor, o estará utilizando na consulta.

Abaixo da área de formulação das consultas, do lado esquerdo, o usuário escolhe o modo de execução da consulta, que pode ser:

- Restrita (*Restricted*): a reformulação da consulta considerará apenas os conceitos do *peer* destino equivalentes e subconceitos dos conceitos inseridos na consulta;

- Completa (*Complete*): a reformulação da consulta considera todas as outras correspondências, dependendo das variáveis de reformulação escolhidas pelo usuário.

A combinação da escolha das variáveis de reformulação e do modo de execução da consulta influencia nas consultas reformuladas produzidas, como mostra a Tabela 02.

Variáveis de Reformulação *Approximate Compose Generalize*	**Modo da Consulta**		**Consultas Reformuladas Produzidas**
	Completa	Restrita	
Pelo menos uma é Verdadeira	Verdadeira	Falsa	*Qexact Qenriched*
Todas são Falsas	Verdadeira	Falsa	*Qexact*
Pelo menos uma é Verdadeira	Falsa	Verdadeira	*Qexact Qenriched*, se *Qexact* for vazia
Todas são Falsas	Falsa	Verdadeira	*Qexact*

Tabela 02 – Mapa de consultas reformuladas produzidas

Do lado direito da escolha do modo de execução da consulta, encontra-se a fórmula geral em DL para as consultas a serem submetidas, que devem ser uma disjunção de conjunções, onde cada conjunção pode envolver um ou mais conceitos.

Do lado direito da fórmula está o botão para submissão da consulta. O primeiro passo da submissão é a reformulação da consulta original a partir do arquivo de correspondências com o *peer* destino. O segundo passo é a tradução da consulta original e da reformulada para a linguagem SPARQL e sua execução nas ontologias do *peer* fonte e do *peer* destino.

Do lado inferior direito está a área de resultados da consulta, onde são mostrados os resultados já integrados do *peer* fonte e do *peer* destino (para a

consulta exata e a consulta enriquecida). A consulta pode ser executada com o *peer* fonte e o *peer* destino estando na mesma máquina ou em máquinas distintas, utilizando RMI (*Remote Method Invocation*) [29] para comunicação.

A tela de consulta ainda possui botões para voltar à escolha da ontologia do *peer* fonte (lado inferior esquerdo) e para retornar à tela principal do módulo (acima da área de formulação da consulta).

Como dito, o módulo de consultas do SPEED também oferece ao usuário dois *logs*, um para as reformulações e outro para os resultados das consultas submetidas durante sua sessão no sistema. As Figuras 18 e 19 ilustram os *logs* de reformulações e o de resultados, respectivamente.

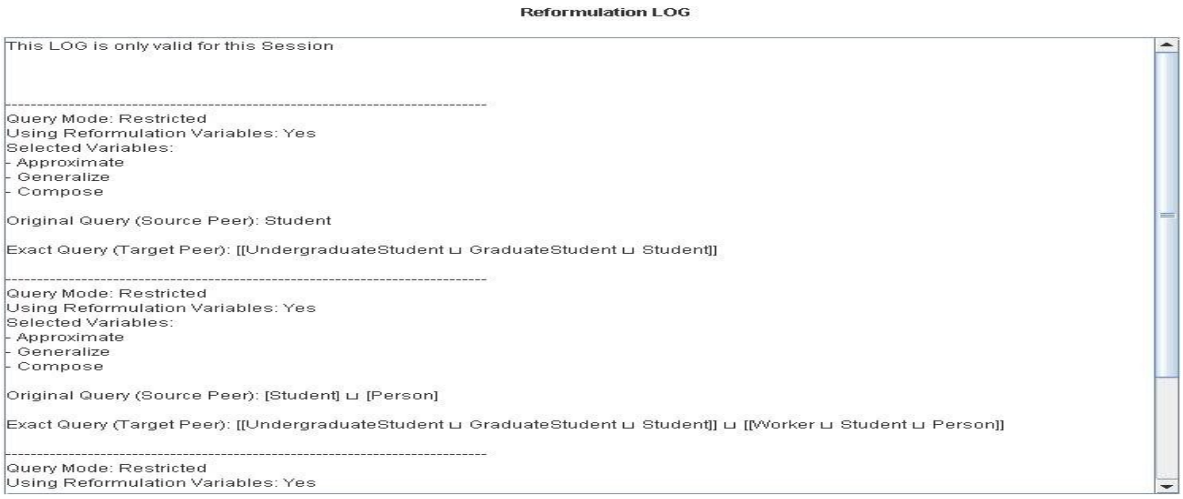


Figura 18 – *Log* de Reformulação

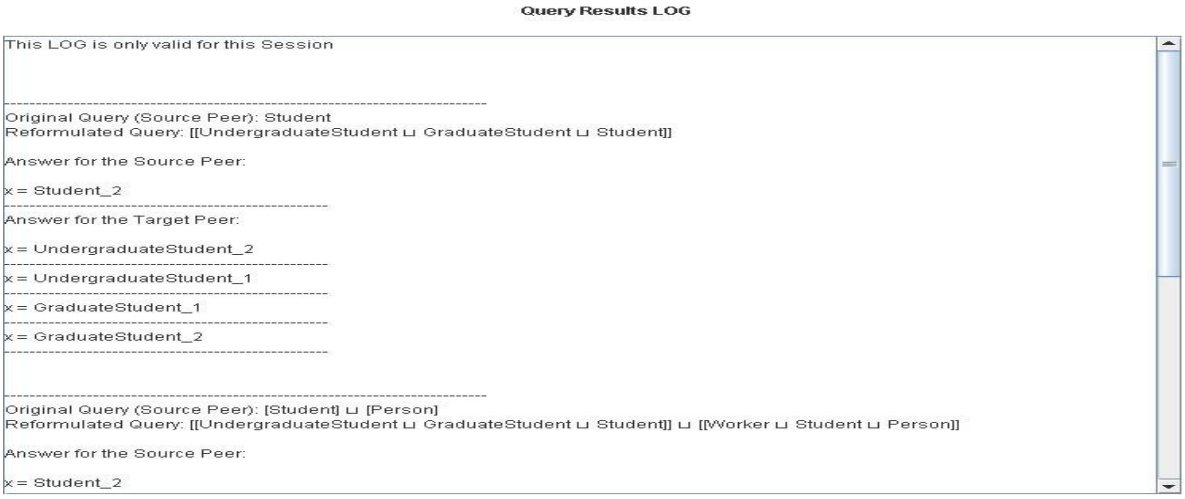


Figura 19 – *Log* de Resultados

O *log* de reformulação traz, para cada consulta submetida, o respectivo modo de execução, as variáveis de reformulação utilizadas, a consulta original, e as consultas reformuladas (exata e enriquecida, caso seja produzida). Já o *log* de resultados ilustra, para cada consulta, a respectiva fórmula original e a reformulação final (que nada mais é do que uma união entre a consulta exata e a consulta enriquecida). Essa consulta reformulada final é a consulta executada no *peer* destino. Além disso, os resultados são exibidos em separado, ou seja, os resultados que vieram do *peer* fonte, e os resultados que vieram do *peer* destino.

Na tela de consultas onde o usuário submeteu a consulta, os resultados são exibidos como um único, sem essa diferenciação. Por fim, o menu *Application -> Exit* encerra a sessão do usuário no módulo de consultas do SPEED.

5.3 Considerações

Este capítulo descreveu o módulo de consultas do SPEED, tema central deste trabalho. Foram detalhadas sua especificação e implementação, descrevendo todas as funcionalidades disponíveis aos usuários, assim como os conceitos de reformulação de consultas utilizados no módulo. O uso da semântica no SPEED facilitou o enriquecimento de consultas submetidas no sistema.

6. Conclusão

Este capítulo apresenta algumas considerações a respeito do presente trabalho, especificando seus objetivos principais. A seguir, são apresentadas as contribuições dadas com os resultados alcançados, e por fim algumas sugestões de melhorias no módulo, servindo como trabalhos futuros.

6.1 Considerações Finais

Este trabalho possuiu como contexto os PDMS, mais especificamente o SPEED, um PDMS baseado em semântica que adota uma arquitetura de *super-peers*. Neste universo, uma das funcionalidades do SPEED foi explorada (e de PDMS em geral), que é o processamento de consultas. O módulo do SPEED desenvolvido não é apenas um módulo de reformulação de consultas, mas também um módulo de submissão e execução destas consultas.

A reformulação foi o objetivo principal, pois assim surgiu o diferencial para as consultas feitas em PDMS normalmente. Com as consultas reformuladas entre *peers* de integração, a semântica foi enriquecida, mais conceitos puderam ser explorados e os resultados se tornaram mais abrangentes.

Nos três primeiros capítulos foram feitos levantamentos dos elementos envolvidos e que serviram de base teórica para o estudo e desenvolvimento deste trabalho. O Capítulo 4 apresentou o SPEED, sua arquitetura e componentes, assim como a lógica das consultas no sistema.

O capítulo seguinte apresentou e detalhou a especificação e implementação do módulo de consultas do SPEED, que foi o tema central do trabalho.

6.2 Contribuições

Os resultados deste trabalho comprovaram a idéia de enriquecer o processamento de consultas em PDMS. A idéia de se obter um mapeamento semântico entre os conceitos dos esquemas dos *peers* de um PDMS serviu como fonte para uma consulta realizada em um *peer* conseguir ser reformulada a partir dos conceitos de outro *peer*. Assim, a consulta respeita a semântica do *peer* cujos

dados estão sendo acessados. Neste sentido, citamos como principais contribuições:

- Estudo de PDMS;
- Estudo de processamento de consultas em PDMS;
- Estudo sobre o enriquecimento de consultas em PDMS;
- Especificação do módulo de consultas do SPEED;
- Implementação do módulo de consultas do SPEED.

O objetivo central, de reformulação de consultas no SPEED, a fim de obter resultados mais detalhados e ricos semanticamente, foi alcançado.

6.3 Trabalhos Futuros

O módulo de consulta desenvolvido para este trabalho possui apenas algumas características necessárias para um processamento de consultas no SPEED de forma robusta. Um ponto a ser estendido é a possibilidade de consultas, com ALC-DL, utilizando as propriedades (*roles*) das instâncias das ontologias dos *peers*.

Outro ponto é que a reformulação das consultas possa ocorrer não apenas entre pontos de integração, mas considerando também os pontos de dados dos *clusters*. A interface também é um ponto que pode ser aprimorado, habilitando-se as consultas diretas através da linguagem SPARQL e também dos conceitos da ontologia do ponto.

Novas funcionalidades também podem ser criadas a fim de aumentar a interação com os usuários.

Bibliografia

[1] Pires, C.E. (2007). "Um Sistema P2P de Gerenciamento de Dados com Conectividade Baseada em Semântica". Monografia de Qualificação e Proposta de Doutorado, CIn, UFPE. Abril, 2007

[2] Tatarinov and Halevy, 2004. Igor Tatarinov and Alon Halevy. Efficient query reformulation in peer data management systems. In Proceedings of the SIGMOD International Conference on Management of Data (SIGMOD'04), 2004

[3] Banco de Dados - Módulo 1 – Introdução, Prof. Marco Antonio Casanova, (2001). Pontifícia Universidade Católica do Rio de Janeiro, http://www.inf.puc-rio.br/~casanova/INF1731-BD/modulo01.pdf

[4] Elmasri and Navathe. "Sistemas de Banco de Dados" (4ª Edição), Pearson - Addison Wesley, 2005.

[5] Xiao, H. (2006). Query Processing for Heterogeneous Data Integration using Ontologies. Ph.D. Thesis. University of Illinois, Chicago, USA

[6] Noy, N. F. and McGuiness D. L. (2002). "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford University.

[7] Resource Description Framework (RDF) (2004), http://www.w3.org/RDF/, Outubro de 2008

[8] Guarino, N. 1998. Formal Ontology and Information Systems. In Proc. of the 1st International Conference on Formal Ontologies in Information Systems. pp. 3-15

[9] OWL Web Ontology Language (2004), http://www.w3.org/TR/owl-features/, Novembro de 2008

[10] An Introduction to the Resource Description Framework (1998), Eric Miller, http://www.dlib.org/dlib/may98/miller/05miller.html, Outubro de 2008

[11] Shadbolt, N., Berners-Lee T., and Hall W. (2006). The Semantic Web Revisited, IEEE Intelligent Systems, v.21 n.3, May 2006

[12] Baader F., Calvanese D., McGuiness D., Nardi D., and Patel-Schneider P. editors. 2003. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press

[13] Deliyanni A. and Kowalski R. A. (1979). "Logic and Semantic Networks". Communications of the ACM, Volume 22, Issue 3, Pages: 184 – 192, March 1979

[14] Alani H., Hoser B., Schmitz C. and Stumme G. (2006). "Semantic Network Analysis". In Proceedings of the 2nd Workshop on Semantic Network Analysis, Collocated with the 3rd European Semantic Web Conference, Budva, Montenegro. June 12, 2006.

[15] SPARQL Query Language for RDF (2008), http://www.w3.org/TR/rdf-sparql-query/, Novembro de 2008

[16] RDQL - A Query Language for RDF (2005), http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, Setembro de 2008

[17] Broekstra J. and Kampman A. (2003). "SeRQL: A Second Generation RDF Query Language". SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 13—14 November 2003, Vrije Universiteit, Amsterdam, Netherlands

[18] Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/, Outubro de 2008

[19] ARQ - A SPARQL Processor for Jena, http://jena.sourceforge.net/ARQ/, Outubro de 2008

[20] Milojicic S. D., Kalogeraki V., Lukose R., Nagaraja K., Pruyne J., Richard B., Rollins S. and Xu Z. (2002). "Peer-to-Peer Computing". HP Laboratories Palo Alto. HPL-2002-57 (R.1), July 3rd – 2003* (International Accession Date Only)

[21] Fiorano Software (2003). "Super-peer Architectures for Distributed Computing". White Paper, Fiorano Software, Inc

[22] Napster (2007), http://free.napster.com/, Outubro de 2008

[23] Incorporação de Segurança em Aplicações Peer-to-Peer: Habilitando Novas Oportunidades além de Compartilhamento de Arquivos, Luciano Paschoal Gaspary, (2006). Universidade Federal do Rio Grande do Sul, http://si3.inf.ufrgs.br/HomePage/seminariosII/arquivos/seminarioII-2006-LUCIANO_PASCHOAL_GASPARY.pdf

[24] Halevy, A., Rajaraman A. and Ordille J. (2006). "Data Integration: The Teenage Years". In Proceedings of the 32nd International Conference on Very Large Databases – Volume 32, Pages: 9 – 16, 2006.

[25] Faye D., Nachouki G. and Valduriez P. (2007). "Semantic Query Routing in SenPeer, a P2P Data Management System". In International Conference on Network-Based Information Systems (NBiS), Regensburg, Germany, 2007, p. 365 – 374

[26] Jena Framework, http://jena.sourceforge.net/javadoc/index.html, Outubro de 2008

[27] Protégé-OWL 3.3.1, http://protege.stanford.edu/download/release-javadoc-owl/, Novembro de 2008

[28] Eclipse IDE, http://www.eclipse.org/, Agosto de 2008

[29] Java Platform SE 6, http://java.sun.com/javase/6/docs/api/, Novembro de 2008

[30] Java, http://www.java.com, Outubro de 2008

[31] Krdzavac N. and Gasevic D.(2005). "A Method for Implementation Description Logic Reasoner", Univ. Beograd. Publ. Elektrotehn. Fak, Ser. Mat. 16 (2005), 119 – 130. Available electronically at http://pefmath.etf.bg.ac.yu

[32] Afonso, R.A., Novaes, M. A., Carneiro, E. A. L., Fidalgo, R. N., Campos, R. S., Oliveira, H. P. L., Barbosa, A. K. P., Silva, V. M., Araújo, K. S., Salgado, A. C. (2006). "GridVida: Um Ambiente de Grade Computacional para Recuperação de Dados Heterogêneos e Comparação de Imagens Médicas". In: Congresso Brasileiro de Informática em Saúde, 2006, Florianópolis. Anais do CBIS 2006, 2006. p. 1417-1422

[33] Souza, D. Y., 2007. “Reformulação de Consulta Baseada em Semântica para PDMS”. Monografia de Qualificação e Proposta de Doutorado. CIn, UFPE. Abril, 2007.

Assinaturas

_____

Ana Carolina Salgado

Orientadora

_____

Thiago Arruda Neves

Aluno

Recife, Novembro de 2008