Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

Reformulação de Consulta Baseada em Semântica para PDMS

Por
Damires Yluska de Souza Fernandes

Exame de Qualificação e Proposta de Tese

Profa. Dra. Ana Carolina Salgado
Orientadora

Recife, Março de 2007

Eu amo tudo o que foi,
Tudo o que já não é,
A dor que já me não dói,
A antiga e errônea fé,
O ontem que dor deixou,
O que deixou alegria
Só porque foi, e voou
E hoje é já outro dia.

Fernando Pessoa.

Resumo

O gerenciamento de dados em ambientes Ponto-a-Ponto (P2P) representa uma tarefa complexa e desafiante devido ao número excessivo de pontos, sua natureza autônoma e à heterogeneidade de seus esquemas. Um PDMS (*Peer Data Management System*) é um sistema P2P que provê usuários com uma interface onde consultas são formuladas transparentemente sobre fontes de dados heterogêneas e autônomas, sendo considerados, diante disso, uma evolução dos sistemas de integração de dados.

O sistema SPEED (*Semantic PEEr-to-Peer Data Management System*) – base sobre a qual será desenvolvido este trabalho, é um PDMS baseado em semântica que adota uma arquitetura de super-pontos. Nesta arquitetura, pontos de dados (associados a fontes de dados) são agrupados de acordo com seu domínio em clusters semânticos. Cada cluster semântico possui um tipo especial de ponto (ponto de integração) que é responsável por tarefas mais complexas como processamento de consultas. Clusters semânticos são agrupados em comunidades semânticas, onde um ponto semântico atua como gerenciador da comunidade, oferecendo uma ontologia de domínio e organizando serviços.

Um dos principais serviços que um PDMS pode prover é o processamento de consultas. No SPEED, pontos de integração são responsáveis por processar consultas. Neste sentido, quando um ponto de integração recebe uma consulta, ele identifica os termos da consulta e os compara com os termos ontológicos da ontologia de domínio do cluster; daí, a partir de mapeamentos semânticos e de um índice de consultas, ele identifica os pontos que podem respondê-la. Em seguida, reformula a consulta de acordo com os mapeamentos, reenvia para cada um dos pontos, recebe os resultados e os integra. Ao término, envia o resultado final para o ponto que iniciou a consulta. O ponto crítico do processamento, no entanto, é a reformulação da consulta entre pontos através de caminhos de mapeamentos semânticos disponíveis.

Dentro deste escopo, o presente trabalho visa desenvolver um processo de reformulação de consulta baseado em semântica para o sistema SPEED. Para desenvolver este processo, duas questões chaves serão tratadas: a definição e utilização de mapeamentos semânticos e o emprego de fatores semânticos. Para a primeira questão, este trabalho irá formalizar uma Linguagem de Mapeamentos Semânticos (LMS) que será utilizada para definição dos mapeamentos e sua aplicação. A segunda questão, por sua vez, está associada a um importante diferencial deste trabalho que é a utilização de aspectos semânticos. Ou seja, metadados, ontologias e contexto (informação circunstancial que torna uma situação única e compreensível) serão utilizados como meio de enriquecer e otimizar todo o processo de reformulação de consulta.

Abstract

Data management in P2P systems is a challenging and difficult problem considering the excessive number of peers, their autonomous nature and the heterogeneity of their schemas. A Peer Data Management System (PDMS) is a P2P application which enables users to transparently query several heterogeneous and autonomous data sources. In this sense, PDMS are considered an evolution of Data Integration Systems.

Semantic PEEr-to-Peer Data Management System – SPEED, our current research system, is a semantic-based PDMS which adopts the super-peer architecture. In this architecture, data peers (associated to data sources) are grouped according to their domain forming one or more semantic clusters. Every semantic cluster has a special type of peer, called integration peer, which is responsible for complex processes like query processing. Semantic clusters are also grouped in semantic communities where a semantic peer acts as a manager. Semantic peers are intended to offer a domain ontology and special services.

Query processing is considered one of the most important services provided in a PDMS. In SPEED system, integration peers are responsible for query processing. So, when a query arrives at the integration peer, it identifies for each query element its corresponding ontological term, according to the cluster domain ontology. Then, the integration peer searches the query index and looks at semantic mappings to find out peers which are able to answer the query. Next, it reformulates the query into a set of queries which will be sent to the peers. The query reformulation is done according to semantic mappings. Finally, answers from every peer are returned to the integration peer, where they are integrated to produce the answer. The final result is sent back to the data peer from which the query was submitted. However, the key step in query processing is reformulating a peer's query over other peers on the available semantic mapping paths.

In this sense, this work aims to develop a semantic-based query reformulation process for SPEED system. In order to develop this process, two relevant issues will be dealt: semantic mapping definition and semantic knowledge employment. To the former, we will formalize a semantic mapping language which will be used to define mappings and their usage. A distinguishing of our work is the usage of semantic aspects. In this light, we use context information, i.e. the circumstantial information that makes a situation unique and comprehensible, as well as ontologies and metadata, as a way to enrich the query reformulation process.

Sumário

Lista de Figuras

Lista de Quadros

Capítulo 1

Introdução

Neste capítulo, introduzimos o contexto onde está inserida esta monografia de qualificação e proposta de tese. Para tal, apresentamos a motivação para este trabalho, definimos o problema, seus objetivos e a organização do documento.

1.1 Motivação

A sociedade da informação, que vem crescendo continuamente ao longo dos últimos anos, busca um completo acesso a todo tipo de informação disponível, esteja ela em um banco de dados local, distribuída em fontes diversas ou compartilhada através de projetos inter-institucionais. Entretanto, muito mais do que acessar tal informação, é desejável realizar consultas e análises sobre elas sem a preocupação em como localiza-las e combiná-las.

Neste sentido, usuários desejam se beneficiar de tecnologias que promovam o acesso a dados distribuídos, a partir de fontes heterogêneas ou não, onde serviços inter-relacionados permitam um nível de abstração total sobre como eles estão sendo gerenciados. Ao mesmo tempo, usuários podem ser beneficiados se sistemas levarem em consideração a semântica onde eles e seus objetivos estão envolvidos.

Tendo em vista esta necessidade, soluções vêm sendo pesquisadas com o intuito de prover usuários com mais facilidades de consulta, no nível de abstração desejado e com foco semântico. Duas soluções recentes merecem destaque: sistemas de integração de dados e PDMS (*Peer Data Management Systems*).

Sistemas de Integração de Dados têm como principal objetivo liberar o usuário de ter que, manualmente, localizar fontes, interagir com elas isoladamente e depois combinar os resultados obtidos. Um aspecto fundamental em sistemas desta categoria é que eles devem ser capazes de unir todos os esquemas das fontes associadas, de modo a criar uma visão única que será o elo de comunicação com o usuário. Esta visão única é chamada de esquema de mediação ou esquema global e é através dela que o usuário formula suas consultas.

PDMS são considerados uma extensão natural dos sistemas de integração de dados [Heese et al. 2005], que, por sua vez, já são uma extensão dos sistemas distribuídos de bancos de dados. A grande diferença, entretanto, é que PDMS não utilizam um esquema global, pois são, na verdade, sistemas de gerenciamento de dados construídos sobre redes *Peer-to-Peer* (P2P). Isto significa que eles são construídos sobre ambientes de computação distribuída sem controle centralizado onde o software que é executado em cada ponto (*peer*) é equivalente em

funcionalidade e tanto atua como cliente quanto servidor. Como não usam esquema global, PDMS normalmente adotam estratégias baseadas em índices de roteamento, metadados e mapeamentos entre pontos para conseguir executar as consultas.

Para que um ponto possa participar de um PDMS, ele deve publicar seu esquema de dados de modo que este possa ser visto pelos demais pontos da rede e, em contrapartida, o sistema deve prover mapeamentos deste ponto com outros. Através de mapeamentos ou da composição de mapeamentos, o sistema pode obter dados relevantes de qualquer ponto que esteja conectado [Tatarinov, Halevy 2004].

Como típicos sistemas P2P, PDMS herdam todas as suas características, dentre elas o fato de cada ponto poder entrar e sair da rede a qualquer momento e a autonomia que cada um tem. Do ponto de vista de gerenciamento de dados, PDMS devem lidar com questões como [Sung et al. 2005]:

- Localização dos Dados: pontos devem ser capazes de referenciar e localizar dados armazenados em outros pontos.
- Processamento de Consultas: dada uma consulta, o sistema deve ser capaz de descobrir os pontos que podem contribuir com dados relevantes à execução da consulta e processá-la eficientemente.
- Integração dos Dados: mesmo quando fontes de dados compartilhadas no sistema seguem diferentes esquemas ou representações, o sistema deve ainda ser capaz de acessar os dados, integrá-los e retornar os resultados.
- Consistência dos Dados: a consistência deve ser mantida em caso de replicação e uso de cache.

A título de ilustração, vamos supor que cientistas de várias instituições de pesquisa desejem compartilhar dados sobre “Bacias Hidrográficas”. De acordo com as propriedades de flexibilidade e descentralização, um PDMS é adequado ao compartilhamento e troca de dados de diversos domínios como, por exemplo, o projeto sobre “Bacias Hidrográficas”.

Para participar do PDMS, cada instituição deve disponibilizar o conjunto de dados local que será compartilhado com os demais pontos – o esquema exportado. Depois de exportar seu esquema, os cientistas da instituição podem utilizar o PDMS para realizar suas consultas. Mas, por que um cientista teria interesse em utilizar um PDMS?

Em linhas gerais, ele tem um objetivo fundamental: obter respostas às suas consultas, de forma transparente e rápida. Ao mesmo tempo, ele busca um sistema com fontes diversas, espalhadas e autônomas porque o que ele obtém utilizando apenas sua fonte local pode não ser suficiente. Em outras palavras, ele busca respostas mais completas e significativas para o que ele necessita naquele momento. Ele pode também desejar responder questões muito complexas, que envolvam critérios e condições diferentes e que, para isso, seja necessária a combinação de dados de várias fontes. Assim, no exemplo de Bacias Hidrográficas, cada instituição possui e gerencia dados sobre suas bacias, mas no momento em que compartilham esses dados, todos os envolvidos no sistema tendem a obter respostas mais completas e relevantes para suas pesquisas, sem intervir na autonomia de cada um.

Tanto sistemas de integração de dados quanto PDMS são soluções que envolvem a problemática da integração de dados, questão diretamente relacionada aos fatores de heterogeneidade, distribuição e autonomia das fontes de dados. Halevy e seu grupo realizaram uma retrospectiva sobre os dez anos de pesquisa na área de integração de dados, desde o projeto Manifold [Levy et al. 1996] até a busca por arquiteturas P2P [Halevy et al. 2006]. Nesta retrospectiva, eles apontam direções de pesquisa consideradas fundamentais para a solução de problemas de integração: geração de mapeamentos entre esquemas, processamento de consultas adaptativo, utilização de XML como padrão de representação de dados, formalização de álgebras para especificação de mapeamentos e manipulação de esquemas, utilização da Inteligência Artificial como meio de obter raciocínio e utilização de ambientes P2P.

Este trabalho está inserido dentro destas perspectivas de pesquisa. Ele busca desenvolver soluções para o problema da reformulação de consulta em PDMS, utilizando semântica como forma de enriquecer todo o processo e consequentemente otimizá-lo. O processo de reformulação de uma consulta pode ser dividido em duas etapas: a reescrita da consulta que gera uma expressão de consulta e a resolução da consulta cujo resultado é o conjunto de todas as respostas possíveis para aquela expressão de consulta [Halevy 2000]. Para viabilizar as duas etapas, é necessário fazer uso de mapeamentos semânticos entre os esquemas que estão sendo empregados na reformulação. Os mapeamentos semânticos descrevem relacionamentos entre os termos usados em dois ou mais esquemas. Isto significa que soluções para o problema de reformulação de consulta devem incluir linguagens para especificação de mapeamentos e algoritmos que usem estes mapeamentos para resolver ou responder adequadamente as consultas.

Desse modo, a reformulação de consultas está diretamente relacionada ao problema de definição e utilização de mapeamentos. Muita pesquisa vem sendo realizada sobre este tema [Friedman et al. 1999; Halevy 2001; Lenzerini 2002; Madhavan, Halevy 2003; Tatarinov, Halevy 2004; Karvounarakis 2005; McBrien, Poulovassilis 2006], mas pouco se tem realizado no sentido de aplicar semântica neste processo. O uso de conhecimento semântico em suas várias formas, incluindo meta-modelos, regras semânticas e restrições de integridade, pode otimizar as capacidades de transformação de consultas em outras semanticamente equivalentes que possam ser respondidas em menos tempo e/ou com menos recursos [Necib, Freytag 2005].

Este trabalho parte da premissa que o uso de semântica através de metadados, ontologias e contexto pode ser empregado satisfatoriamente como forma de otimizar todo o processo de reformulação de consultas. Metadados são utilizados para descrever os conteúdos das fontes de dados (em pontos) e podem também ser utilizados para descrever resultados parciais de consultas. Ontologias são utilizadas normalmente como um vocabulário compartilhado de termos, podendo ser empregadas para: (i) descrever os metadados das fontes ligadas aos pontos de dados, como meio de uniformização de representação; (ii) servir de referência de termos, na forma de ontologias de domínio, ajudando a resolver problemas de heterogeneidade semântica e (iii) representar informações contextuais, enquanto técnica de formalização.

Em especial, um diferencial deste trabalho em relação a outros é a utilização de contexto através de conceitos, regras, proposições e/ou suposições associados a uma situação específica como, por exemplo, uma entidade, uma consulta ou um ponto de dados. Neste sentido, o contexto ajuda a definir qual conhecimento deve ser utilizado, quais restrições devem ser consideradas nas diversas operações de manipulação, como na reformulação de consulta, quais as condições que irão ativar as ações e em que momento estas ações devem ser executadas.

No caso da reformulação da consulta, informações contextuais percebidas e/ou inferidas sobre o ambiente (altamente dinâmico), a submissão da consulta, o perfil do usuário, os pontos, os mapeamentos semânticos, os dados e entidades envolvidas e outras serão utilizadas para otimizar todo o processo de reformulação. Como resultado, espera-se evitar redundâncias na composição de caminhos de mapeamentos e reformulações que impliquem em resultados vazios, através da pré-computação de caminhos de mapeamentos [Tatarinov, Halevy 2004]. Além disso, resultados de consultas se tornarão mais significativos e completos para o usuário.

1.2 Definição do Problema

Como visto na seção anterior, um PDMS é um sistema de gerenciamento de dados que faz uso de uma arquitetura P2P. Seu objetivo principal é permitir a troca de dados, compartilhamento de informações e o processamento de consultas, sendo este último reconhecido como o principal serviço que um PDMS pode prover [Arenas et al. 2003]. Neste sentido, processar uma consulta em um sistema como esse significa prover capacidades de responder às consultas sobre uma rede arbitrária de pontos com esquemas locais e compartilhados e um conjunto de mapeamentos entre estes pontos.

O processamento de consultas em um PDMS envolve basicamente as seguintes etapas: submissão da consulta, análise da consulta, identificação dos pontos capazes de respondê-la, reformulação de acordo com mapeamentos, execução, integração e apresentação dos resultados. Dentro deste escopo, este trabalho apresenta um processo geral que envolve tais etapas, mas focaliza nos aspectos relacionados à reformulação da consulta através dos mapeamentos e de forma enriquecida semanticamente.

Assim, podemos definir o problema como segue:

Dado um PDMS, uma consulta $\mathbf{Q}$ é submetida em um ponto de dados e deve ser propagada por todos os demais pontos da rede que tenham dados para contribuir com a resposta. Mapeamentos entre pontos estão definidos e a semântica da consulta deve ser comum aos pontos que irão respondê-la. Então o problema se encontra em reformular a consulta $\mathbf{Q}$ através dos mapeamentos por todos os pontos com semântica comum à identificada na consulta.

A reformulação da consulta vai ser dividida em duas etapas: a reescrita da consulta que gera uma expressão de consulta ($\mathbf{Q}'$) e a resolução da consulta cujo resultado é o conjunto de todas as respostas possíveis para aquela expressão de consulta. Estes resultados serão retornados ao ponto onde foi submetida a consulta e sua integração será realizada.

O diferencial deste trabalho está em realizar todo este processo, considerando fatores semânticos, principalmente contexto, em cada etapa a ser cumprida.

1.3 Objetivos

Este trabalho visa desenvolver um processo de reformulação de consulta baseada em semântica para PDMS. Para tal, pretende:

- Estabelecer um formalismo para definição dos mapeamentos semânticos.
- Estabelecer técnicas para extrair a semântica dos mapeamentos.
- Construir um índice semântico de consultas, a partir de ontologias.
- Desenvolver uma ontologia de contexto como forma de representação e utilização deste tipo de informação.
- Aplicar contexto em todas as fases do processo de reformulação.
- Modelar e desenvolver o processo de reformulação de consultas para o sistema SPEED (**S**emantic **PEE**r-to-Peer **D**ata Management System), base de desenvolvimento deste trabalho.
- Validar o processo de reformulação de consultas, através de estudos de casos.

1.4 Estrutura do Documento

Esta monografia está organizada em duas partes - a qualificação e a proposta de tese, divididas em seis capítulos, sendo o capítulo seis especificamente dedicado à referida proposta. Desta maneira a monografia encontra-se organizada como segue:

O capítulo dois discute a problemática da integração de dados, através de conceitos fundamentais, problemas e desafios a serem trilhados.

O capítulo três aborda aspectos semânticos que podem contribuir para a otimização de processos de integração de dados, em especial o processamento de consultas. Os aspectos semânticos introduzidos são metadados, ontologias e contexto.

O capítulo quatro descreve os sistemas P2P e PDMS, mostrando suas características, topologias e conceitos básicos, além de analisar sistemas reais que vêm sendo pesquisados.

O capítulo cinco expõe o panorama do processamento de consultas em PDMS e, em especial, a problemática da reformulação de consulta e da sua relação com a definição de mapeamentos.

Finalmente, o capítulo seis apresenta a proposta de tese que norteia este trabalho. Para tal, introduz o sistema SPEED – base sobre o qual será desenvolvido este trabalho, traça metas relacionadas às possíveis soluções do problema (reformulação da consulta com base em aspectos semânticos), mostra as contribuições esperadas, a metodologia a ser empregada e o cronograma de trabalho.

Capítulo 2

A Problemática da Integração de Dados

A competitividade nos ambientes de negócios depende da qualidade do acesso às informações de que as empresas necessitam no dia-a-dia. Estas informações se encontram atualmente distribuídas pelas *intranets* das empresas e pela Internet, através de múltiplas fontes de dados que podem estar em diferentes formatos: sistemas legados, bancos de dados relacionais, bancos de dados objeto-relacionais, sistemas de arquivos, XML, páginas Web, planilhas eletrônicas, entre outros. Combinar todas estas informações não representa uma tarefa fácil, o que dificulta atividades de consulta a um domínio específico e impede que ações sejam tomadas em tempo hábil.

Entretanto, de que forma é possível obter acesso a estas informações sem intervir na autonomia dos sistemas que os mantêm? A busca pela interoperabilidade – compartilhamento e troca de informações - fez nascer a busca pela integração dos dados, ou seja, viabilizar a consulta a dados pertencentes a fontes heterogêneas, distribuídas e autônomas, através de uma visão integrada e única. Um sistema de integração de dados visa oferecer aos usuários uma interface uniforme de acesso a diferentes fontes de dados, de modo que os usuários possam definir "o que" eles desejam obter nas suas consultas, e o sistema determine "onde" as informações podem ser encontradas, realize a integração das mesmas e apresente o resultado desejado.

Com base neste contexto, este capítulo descreve a problemática da integração de dados, conceitua sistemas de integração de dados, mostra abordagens para sua construção, assim como um exemplo de sistema. Ao término, problemas e desafios são apontados com o intuito de fornecer o panorama de pesquisa na qual este trabalho está inserido.

2.1 A Evolução dos Bancos de Dados e a Era da Informação

Os bancos de dados surgiram aproximadamente em meados dos anos 60 com o objetivo de possibilitar o gerenciamento de grandes quantidades de dados, assim como de estruturar esses dados e prover rotinas padronizadas de acesso aos mesmos. O grande avanço da computação, a difusão das redes e a diversidade das aplicações que necessitam de persistência de dados propiciaram a evolução das tecnologias e arquiteturas de bancos de dados. Distribuição e heterogeneidade vêm se tornando premissas básicas para organizar a infra-estrutura dos sistemas computacionais recentes.

Como resultado, surgiram os bancos de dados distribuídos e, com eles, tipos diversos de SGBD e soluções de distribuição. Os SGBD distribuídos são considerados homogêneos quando utilizam o mesmo software em múltiplos sítios, caso contrário, são considerados heterogêneos [Navathe, Elmasri 2005]. A heterogeneidade pode acontecer quando as informações se encontram armazenadas em formatos diferentes (banco de dados relacional, banco de dados orientado a objetos, HTML, XML, etc), plataformas diferentes ou quando se apresentam diferenças estruturais e/ou semânticas entre as fontes das mesmas.

Alternativas para lidar com bancos de dados distribuídos e heterogêneos incluem organizá-los em federações, construir *data warehouses* ou ainda criar sistemas com múltiplos bancos de dados acoplados. Quando cada SGBD é um SGBD independente e autônomo que tem seus próprios usuários locais, transações locais e DBA (*Database Administrator*), forma-se uma federação de bancos de dados. Neste tipo de organização, os participantes de uma federação compartilham seus objetos sem perder o controle sobre os mesmos. Um sistema de banco de dados federado provê um esquema global da federação que é compartilhado por todas as aplicações envolvidas. Por outro lado, considera-se um sistema de múltiplos bancos de dados (*Multidatabase System*) aquele que não possui um esquema global, mas o constrói interativamente conforme a necessidade da aplicação [Navathe, Elmasri 2005].

A sociedade da informação, que vem crescendo ao longo dos últimos anos, busca um completo acesso a todo tipo de informação disponível, esteja ela em um banco de dados local, distribuída em sítios diversos, armazenada em formato relacional ou em XML ou ainda em sistemas de arquivos diversos. Halevy et al. [Halevy et al. 2005] identificam ambientes deste tipo como um "DataSpace" (*DataSpace Support Platforms* – DSSP) ou espaço de dados, uma nova abstração para gerenciamento de dados onde serviços inter-relacionados gerenciam grandes volumes de dados relacionados, mas distribuídos, de forma consistente e eficiente.

DSSP não constituem uma abordagem de integração de dados, mas de coexistência de dados [Halevy et al. 2005]. O objetivo de um *DataSpace* é prover a funcionalidade básica sobre todas as fontes de dados, independentemente de como elas estão integradas. Por exemplo, um DSSP pode prover uma busca por palavra-chave ou realizar operações mais sofisticadas como consultas *SQL-like*, mineração de dados, entre outras. Um DSSP deve estar habilitado a:

- Lidar com dados e aplicações numa grande variedade de formatos através de diferentes interfaces;
- Evitar controle completo sobre seus dados (diferentemente de um SGBD);
- Permitir a realização de consultas em diferentes níveis, retornando o melhor resultado possível (*best-effort*) ou respostas aproximadas. Isto significa que, quando determinadas fontes se encontrarem indisponíveis, o sistema deve ser capaz de produzir os melhores resultados, utilizando os dados disponíveis naquele momento;
- Oferecer ferramentas para prover integração dos dados, quando necessário.

Ao analisarmos o ambiente Web, tão propício a consultas e buscas, podemos considerá-lo como um exemplo de "*DataSpace*", visto que ele reflete um universo

heterogêneo, diversificado e altamente procurado por usuários para suas buscas. Ao mesmo tempo em que promove a usabilidade e favorece as buscas e pesquisas, a Web se torna um ambiente desafiador porque amplia consideravelmente a dificuldade de realizar consultas sobre um determinado domínio ou cruzar informações sobre diversas fontes de dados.

Tendo em vista tais dificuldades, soluções vêm sendo pesquisadas com o intuito de prover usuários com mais facilidades de consulta, ao mesmo tempo em que eles tenham visões simplificadas dos dados que se encontram, na verdade, localizados em fontes distribuídas, heterogêneas e autônomas. Duas soluções recentes merecem destaque: sistemas de integração de dados e PDMS (*Peer Data Management Systems*).

Sistemas de Integração de Dados têm como principal objetivo liberar o usuário de ter que, manualmente, localizar fontes, interagir com elas isoladamente e depois combinar os resultados obtidos. Um aspecto fundamental em sistemas desta categoria é que eles devem ser capazes de unir todos os esquemas das fontes associadas, de modo a criar uma visão única que será o elo de comunicação com o usuário. Esta visão única é chamada de esquema de mediação ou esquema global.

A manutenção de um esquema único pode se tornar um aspecto limitante se usuários desejam compartilhar dados sem qualquer autoridade central. *Peer Data Management Systems* (PDMS) têm sido propostos como uma alternativa de arquitetura para compartilhamento de dados descentralizados [Tatarinov, Halevy 2004]. Um PDMS é composto por um conjunto de pontos (*peers*), cada um com um esquema associado que representa seu domínio de interesse. Geralmente mapeamentos são providos entre pares ou um pequeno número de pontos que compartilham interesses comuns. Assim, quando um usuário formula uma consulta, o sistema pode obter dados relevantes de qualquer ponto conectado através destes mapeamentos.

Ambas as soluções envolvem a problemática da integração de dados. Usuários buscam compartilhar dados, realizar consultas num grau de abstração, onde o acesso às fontes e a integração dos resultados sejam efetuados da maneira mais transparente possível. Isso tudo aliado a um bom desempenho e a resultados mais completos. Estas premissas têm direcionado pesquisas em torno de sistemas de integração de dados em arquiteturas convencionais (com um esquema único) ou em ambientes *Peer-to-Peer* (P2P).

Em resumo, desde o surgimento dos bancos de dados, o problema de integração de dados tem sido reconhecido como amplo e de grande importância. Como conseqüência, a comunidade de Banco de Dados vem dedicando especial interesse e pesquisa nessa área com o objetivo de prover soluções de integração adequadas. Exemplos de sistemas convencionais de integração são o TSIMMIS [Chawathe et al. 1994], Integra [Lóscio 2003]. Exemplos de PDMS que vêm sendo desenvolvidos são PeerDB [Ng et al. 2003], Piazza [Kadiyska et al. 2003] e XPeer [Bellahsène et al. 2004].

2.2 Sistemas de Integração de Dados

O bem mais precioso de uma empresa é a informação que ela gerencia, entretanto a tarefa de prover acesso a informações combinadas e integradas não tem sido fácil. Ao mesmo tempo

em que cresce a importância da informação nas organizações, sejam elas públicas ou privadas, ocorre a crescente disseminação da mesma nas *intranets* e na Internet. Ou seja, as informações permanecem distribuídas em fontes diversas e autônomas.

Neste cenário, como é possível obter acesso aos milhares de dados espalhados na Internet e nas intranets, sem intervir na autonomia dos sistemas que os mantêm? A busca pela interoperabilidade – compartilhamento e troca de informações - fez nascer a busca pela integração dos dados. Um dos objetivos da integração num ambiente informacional é buscar o aumento do valor da informação no momento em que dados de várias fontes são acessados, relacionados e combinados [Almeida, Bax 2003].

A busca por soluções para integração de dados vem sendo pesquisada desde o início dos anos 90 e já é hoje uma realidade não somente no meio acadêmico, mas também no meio comercial [Halevy et al. 2006]. Nesta seção, apresentamos o conceito de sistemas de integração de dados, as abordagens existentes para o seu desenvolvimento e um exemplo de sistema.

2.2.1 Definição

A principal tarefa de um sistema de integração de dados é fornecer uma interface uniforme para responder consultas que normalmente requerem extração e combinação de dados originários de diversas fontes distintas, heterogêneas, distribuídas e autônomas [Levy99]. Exemplos de aplicações cuja integração de dados se faz necessária são sistemas legados, projetos científicos como aqueles oriundos de pesquisas biológicas e astronomia, sistemas de informações geográficas, comércio eletrônico, entre outras.

Sistemas de integração de dados vêm se tornando populares por serem capazes de, além de integrarem informações de fontes diversas, responderem consultas mais complexas ou fornecerem uma resposta mais completa, de acordo com as fontes relevantes disponíveis. Outro aspecto importante é que, apesar das bases serem distribuídas e heterogêneas, o usuário tem a impressão de estar utilizando um sistema centralizado e homogêneo, visto que a forma como a informação está sendo recuperada se torna transparente para ele.

Ambientes como a World Wide Web, dinâmicos e evolutivos, requerem sistemas de integração que permitam esse acesso da forma mais simples e transparente possível. Entretanto, nesse meio, alguns desafios como a pouca representação de metadados, as mudanças constantes nas fontes (evolução e autonomia), a diferença entre formatos de dados, alguns semi-estruturados ou não estruturados e a heterogeneidade das plataformas são fatores críticos que devem ser considerados [Batista 2003].

A problemática da integração de dados está diretamente relacionada com as questões de heterogeneidade. De maneira mais ampla, a heterogeneidade ocorre em dois níveis: sintático e semântico. O nível sintático se refere ao esquema próprio de cada sistema para armazenamento e documentação de seus dados. O nível semântico, por sua vez, refere-se à representação conceitual ou o significado da informação, presente em cada sistema.

De forma mais detalhada, a heterogeneidade das fontes pode ser encontrada em diversos níveis [Batista 2003]:

- Modelo de dados – fontes de dados podem apresentar modelos variados, como sistemas de arquivo, relacional, objeto-relacional, semi-estruturado;
- Esquema – determina objetos, atributos, relacionamentos e restrições, podendo ser estruturado, pouco estruturado ou não estruturado;
- Codificação de Valores – valores e itens de valores podem se encontrar formatados diferentemente, em cada fonte. Por exemplo, o valor de um produto pode estar em U$$ numa fonte e em R$ em outra;
- Linguagem de consulta – dependendo da fonte, pode haver ou não uma linguagem de consulta nativa, por exemplo, SQL, OQL, xQuery;
- Plataformas de hardware, de sistema operacional e de SGBD – podem variar de uma fonte para outra;

Percebe-se, no entanto, que além dos aspectos sintáticos ou estruturais, aspectos relacionados ao significado das informações têm sido considerados o grande "gargalo" das soluções de integração de dados existentes. Um entendimento da semântica dos conceitos e de seus relacionamentos pode ser a chave para um resultado de integração mais satisfatório.

Além de considerar questões de heterogeneidade, o sistema de integração de dados deve gerenciar aspectos relativos à autonomia e distribuição das fontes de dados. A autonomia está relacionada ao fato de que as fontes geralmente operam de forma independente e, por conseguinte, podem modificar seus dados e esquemas a qualquer momento sem notificar o sistema de integração. A principal questão da distribuição das fontes de dados diz respeito aos desafios impostos pela localização dessas fontes em ambientes heterogêneos como são as *intranets* e a própria Internet.

2.2.2 Abordagens para Integração de Dados

Três abordagens básicas têm sido utilizadas no desenvolvimento de sistemas de integração de dados: a abordagem virtual, a abordagem materializada e a híbrida que envolve a combinação das anteriores [Batista 2003].

Na abordagem virtual, as informações são extraídas diretamente das fontes no momento em que são solicitadas por intermédio de uma consulta. Isto significa que as informações se encontram sempre atualizadas (nas fontes). Por outro lado, caso as fontes estejam indisponíveis naquele momento, não será possível recuperar aquelas informações.

Na abordagem materializada, as informações relevantes são previamente recuperadas, integradas e armazenadas através de um repositório central. Assim, quando uma consulta é formulada, ela é executada sobre os dados que se encontram neste repositório, não havendo recuperação direta das fontes. Isto implica muitas vezes em inconsistência entre o repositório e as fontes (que são autônomas e evoluem rapidamente), podendo gerar dados desatualizados.

O Quadro 2.1 resume vantagens e desvantagens de ambas as abordagens. A escolha por uma ou outra vai depender da aplicação que irá utilizá-la. Vale salientar que, para aplicações complexas e de grande escala, a aplicação de recursos de ambas as abordagens pode ser necessária, o que implica em utilizar uma abordagem híbrida [Batista 2003, Costa

2003]. Em alguns casos, um sistema de integração de dados deve possuir algumas informações recuperadas, processadas e integradas previamente enquanto outras informações devem ser recuperadas apenas quando requisitadas nas consultas.

Quadro 2.1: Abordagem Materializada x Virtual

Abordagem	Vantagens	Desvantagens
Materializada	Interessante, se o tempo de resposta da consulta for mais importante do que o grau de atualização dela, ou quando consultas previsíveis são realizadas	Podem ocorrer inconsistências entre os dados do repositório e os das fontes Não recomendada quando as consultas devem fornecer informações sempre atualizadas
Virtual	Obtém-se uma contínua atualização dos dados, pois os mesmos são extraídos das fontes no momento em que são requisitados pelas consultas.	Ineficiente quando as fontes têm tempo de resposta alto ou quando estão inacessíveis

Uma das vantagens da abordagem virtual é que as informações recuperadas estão sempre atualizadas. Porém, ela passa a ser inadequada quando os passos de tradução e integração das informações são extensos e ficam muito caros ou quando há a possibilidade de as fontes de dados freqüentemente ficarem inacessíveis.

Uma das vantagens da abordagem materializada é o tempo de resposta de uma consulta que se torna bem pequeno, visto que não é necessário acessar as fontes. Diante disso, ela é indicada em algumas situações, como [Batista 2003]:

- Porções específicas das informações e consultas previsíveis são requisitadas;
- O tempo de resposta de uma consulta deve ser bastante curto, sem necessidade das informações estarem em seu estado mais atualizado;
- Informações que não constam diretamente nas fontes (como, informações históricas) são necessárias.

Para implementação destas abordagens, dois tipos de arquiteturas são comumente utilizadas: arquitetura de mediadores e arquitetura de *Data Warehouse*. As subseções seguintes detalham as características de cada uma.

2.2.2.1 Arquitetura de Mediadores

Sistemas baseados em mediadores são construídos através de uma arquitetura em camadas, onde os mediadores exercem um papel central de intermediação da integração dos dados. Segundo Wiederhold [Wiederhold92], mediadores são módulos de software que exploram o conhecimento representado em um conjunto ou subconjunto de dados para gerar informações

para aplicações residentes em uma camada superior. Ou seja, são um tipo de *middleware* que fornecem serviços a aplicações intermediando a integração de dados armazenados em diferentes fontes.

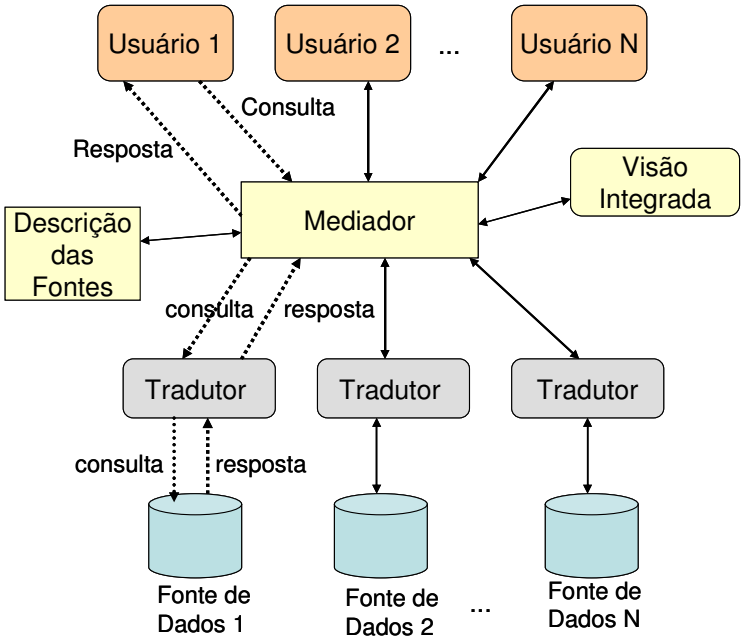

Fig. 2.1 Arquitetura de Mediadores

A arquitetura de mediadores [Wiederhold92], mostrada na Figura 2.1, é composta de dois elementos principais – o mediador e os *wrappers* ou tradutores. Os mediadores oferecem uma visão integrada sobre os dados distribuídos nas diversas fontes de dados e disponibilizam um esquema para essa visão, bem como descrições das fontes de dados. Desta maneira, o mediador recebe uma consulta e decompõe a mesma em várias sub-consultas menores direcionadas às fontes de dados. Essas sub-consultas deverão ser traduzidas para os formatos/linguagens de consultas suportados por cada fonte. As fontes de dados, por sua vez, recebem as sub-consultas traduzidas, processam e retornam o seu resultado para o mediador. O trabalho de tradução mediador/fonte/mediador é feito pelo módulo de software *wrapper* [Batista 2003]. Os tradutores provêem acesso às fontes heterogêneas de dados, convertendo consultas oriundas da aplicação em consultas direcionadas ao modelo e capacidade da fonte, assim como convertendo o resultado da consulta da fonte para o modelo de dados comum do sistema de integração [Lóscio 03].

2.2.2.2 Arquitetura de *Data Warehouse*

A arquitetura de integração de dados com *Data Warehouse* implementa a abordagem materializada, onde os dados são previamente recuperados, integrados e armazenados em um *data warehouse* [Gupta95] que funciona como um repositório central. Este *data warehouse* constitui a visão materializada dos dados e quaisquer consultas submetidas ao sistema de integração serão avaliadas no próprio *data warehouse*.

A abordagem de construção do repositório de dados, referenciada como *data warehousing*, é caracterizada pelas seguintes propriedades [Rundensteiner00, Batista 03]:

- Na instalação, as informações relevantes são extraídas das fontes de dados, transformadas e "limpas" (quando necessário), agrupadas com outras informações provenientes de outras fontes de dados e, em seguida, carregadas no *data warehouse*;
- No processamento de consultas, as consultas submetidas ao sistema são processadas diretamente no *data warehouse* sem haver interação com as fontes de dados;
- Em tempo de operação, modificações nas fontes de dados são filtradas e classificadas por relevância e de alguma forma são propagadas para atualizações no *data warehouse*.

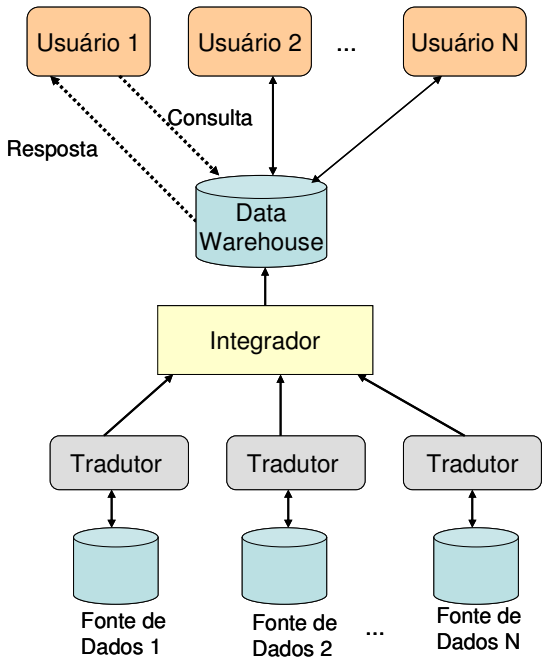

Fig. 2.2 Arquitetura de Datawarehouse

A Figura 2.2 mostra a arquitetura de integração de dados com *data warehouse*. As fontes de dados podem ser sistemas de arquivos, bancos de dados, documentos HTML, sistemas legados, bases de conhecimento, entre outras. Conectado com cada fonte está um módulo de software *wrapper* ou tradutor. O tradutor é responsável pela tradução da informação do formato da fonte de dados para o formato e modelo de dados utilizados pelo sistema de *data warehousing*. O integrador é responsável por "instalar" as informações no *warehouse*, através de procedimentos como filtragem, sumarização e integração das informações com outras provenientes de outras fontes.

2.2.2.3 Um *Framework* Formal para Integração de Dados

Considerando a abordagem virtual e definições comumente elaboradas na literatura [Ruzzi 2004], vamos definir formalmente um sistema de integração de dados:

Um sistema de integração de dados **I** é composto por uma tripla **< G, S, M >** onde:

- **G** é o esquema global, isto é, o conjunto de entidades globais, cada uma com uma aridade associada e um conjunto de restrições de integridade;
- **S** é o esquema da fonte de dados, ou seja, o conjunto de metadados da fonte;
- **M** é o mapeamento entre **G** e **S**, constituído por um conjunto de assertivas da forma {**qs**,**qg**}, na qual **qs** é uma consulta conjuntiva sobre o esquema da fonte enquanto que **qg** é a consulta conjuntiva sobre o esquema global.

A semântica de um sistema de integração é fornecida nos termos dos dados das fontes. Assim, a partir de uma instância de banco de dados **D** para um esquema de fonte, define-se a semântica do sistema **I** como o conjunto de todos os bancos de dados globais **B**, de acordo com as seguintes condições:

- **B** é uma instância de banco de dados que respeita a estrutura de **G** e satisfaz as restrições de integridade expressas nele;
- **B** satisfaz as assertivas de mapeamentos referentes à instancia da fonte **D**.

Em relação às assertivas de mapeamentos, diferentes definições podem ser realizadas com respeito à noção de satisfação do mapeamento [Ruzzi 2004]. Assume-se que se um mapeamento é reduzido (*sound)*, então os dados providos pelas fontes são um subconjunto dos dados globais, ou seja a extensão de **qs** está contida na extensão de **qg**. Se o mapeamento é considerado completo (*complete)*, os dados providos pelas fontes são um super conjunto dos dados globais, ou seja, a extensão de **qs** contém a extensão de **qg**. No entanto, se o mapeamento é exato (*exact*), então ele tanto é reduzido quanto completo. Como as fontes são normalmente distribuídas, autônomas e independentes, a definição de mapeamento reduzido é considerada razoável em ambientes de integração.

O projeto de mapeamentos é uma das tarefas mais complexas na especificação de um sistema de integração de dados. Existem basicamente quatro abordagens para definição de mapeamentos: Global-as-View (GAV) [Levy 2000], Local-as-View (LAV) [Lenzerini 2002], GLAV [Madhavan, Halevy 2003] e BAV [McBrien, Poulovassilis 2003a]. Estas abordagens serão descritas no capítulo 5.

De maneira geral, diversos problemas devem ser tratados na construção de sistemas para integração de dados: especificação do esquema global; linguagem de consulta a ser utilizada; consistência dos dados nas fontes; diferentes capacidades de processamento de consultas e modelos de dados nas diversas fontes de dados; e mecanismos para realizar a gerência, otimização e execução de consultas.

2.2.3 Estudo de Caso: O Sistema Integra

O Integra, desenvolvido no Centro de Informática da UFPE, é um sistema de integração para ambiente Web proposto por Lóscio [Lóscio 2003] baseado em mediadores que oferece uma visão integrada de dados distribuídos em diversas fontes autônomas e heterogêneas.

Ele faz uso da combinação das duas abordagens básicas de integração, dando suporte a consultas virtuais e materializadas. Diferenciais observados no Integra dizem respeito à utilização de uma cache para armazenar os resultados das consultas mais frequentemente submetidas e estratégias para geração e manutenção do esquema do mediador em ambientes dinâmicos [Costa 2005]. A arquitetura do sistema Integra está ilustrada na Figura 2.3 e pode ser dividida em quatro ambientes: Ambiente Comum, Ambiente de Integração de Dados, Ambiente de Geração e Manutenção do Mediador e Ambiente do Usuário. Estes ambientes são descritos a seguir.

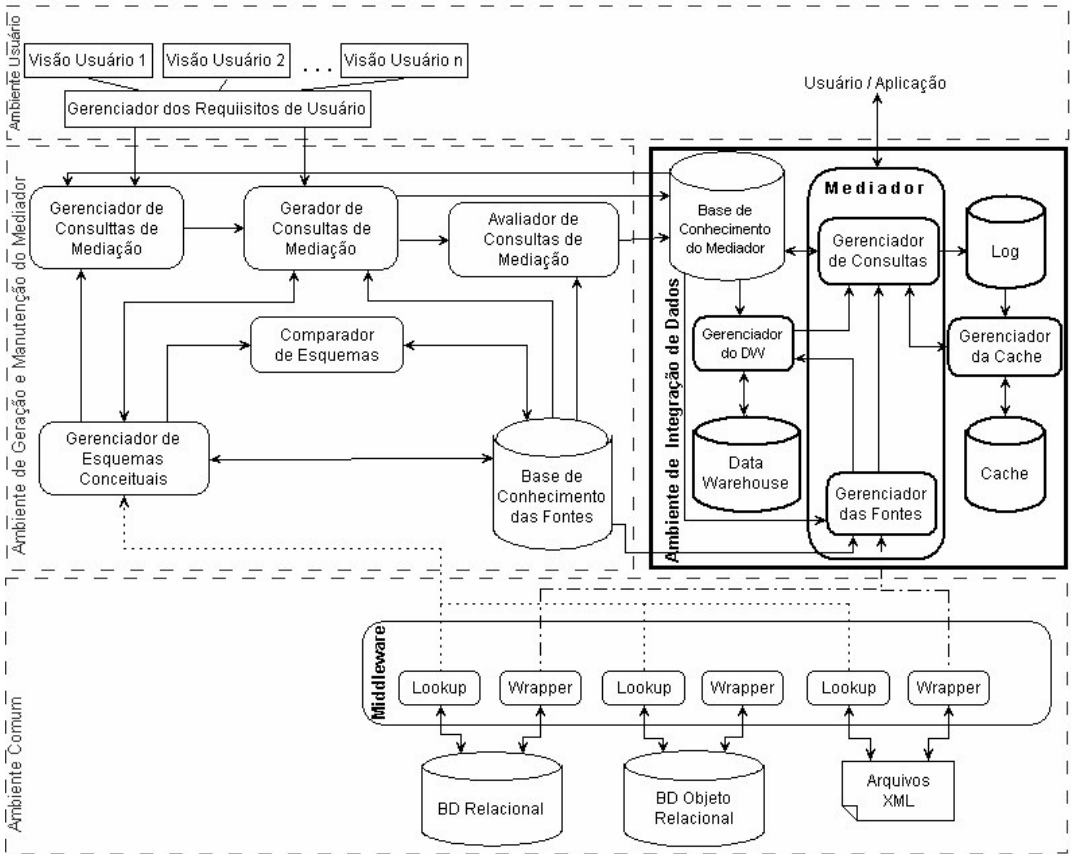

Fig. 2.3 Arquitetura do Sistema Integra [Costa 2005]

2.2.3.1 Ambiente Comum

Neste ambiente, são encontradas as fontes de dados que participam do sistema. Assim, o Ambiente Comum fornece ao sistema informações sobre os esquemas das fontes e tem por responsabilidade receber as consultas, executá-las e retornar os resultados. Ele é composto pelas próprias fontes de dados e pelo módulo de Middleware.

As fontes de dados são aquelas que disponibilizam dados para serem integrados pelo sistema. São geralmente heterogêneas (diversos tipos), autônomas (funcionam independentemente do sistema de integração) e dinâmicas (sofrem modificações freqüentes).

O Integra deve ser capaz de gerenciar entradas e saídas das fontes, reportando tais alterações às camadas superiores da arquitetura.

O **Middleware**, composto pelos serviços de **wrapper** e **lookup**, é o módulo que interage com os Ambiente de Integração de Dados e Ambiente de Geração e Manutenção do Mediador.

O Lookup extrai os esquemas das fontes mantendo-os atualizados na DSKB (Data Source Knowledge Base), além de traduzi-los para o formato comum do Integra. Para tal, ele interage com o Gerenciador de Esquemas Conceituais (Ambiente de Geração e Manutenção do Mediador).

A principal tarefa dos Wrappers (tradudores) é traduzir as consultas enviadas para cada fonte de acordo com sua linguagem de consulta, assim como traduzir os dados retornados pelas fontes para o formato comum do sistema (XML).

2.2.3.2 Ambiente de Geração e Manutenção do Mediador

O Ambiente de Geração e Manutenção do Mediador é composto pelo Gerenciador de Esquemas Conceituais, Comparador de Esquemas, Gerador de Consultas de Mediação, Gerenciador de Consultas de Mediação e Avaliador de Consultas de Mediação que serão descritos a seguir.

O **Gerenciador de Esquemas Conceituais** provê os eventos necessários à atualização do esquema de mediação. Ele converte o esquema exportado pelo *Lookup* para um esquema num formato próprio do Integra chamado X-Entity [Lóscio 2003].

O **Comparador de Esquemas** tem a função de identificar correspondências entre os elementos de esquemas. Em outras palavras, ele realiza uma análise sintática e semântica com o intuito de verificar quais elementos de um determinado esquema são logicamente correspondentes a elementos de outros esquemas.

O **Gerador de Consultas de Mediação** seleciona as fontes que são relevantes no processamento de uma consulta de mediação e identifica os possíveis operadores que serão utilizados entre os elementos das fontes.

Por outro lado, o Gerenciador de Consultas de Mediação propaga os eventos passados pelo Gerenciador de Esquemas Conceituais no esquema de mediação, isto é, a adição, remoção ou alteração das fontes de dados ligadas ao sistema.

O módulo **Avaliador de Consultas de Mediação** analisa o impacto das propagações das modificações do esquema na qualidade global do sistema. Métricas como disponibilidade da fonte e tempo de resposta são utilizadas para tal.

2.2.3.3 Ambiente de Integração de Dados

O Ambiente de Integração de Dados gerencia aspectos de reformulação de consultas, integração de dados e de desempenho, tendo como principal módulo o Mediador.

O **Mediador**, quando recebe uma consulta do usuário, faz a sua decomposição em subconsultas e, de acordo com as consultas de mediação, define a fonte de dados que será

capaz de responder a cada subconsulta. Uma vez respondida a consulta, este módulo deverá integrar os resultados. O Mediador é composto por um Gerenciador de Consultas e um Gerenciador de Fontes, apresentados abaixo:

O **Gerenciador de Consultas** usa uma base de conhecimento (MKB – Mediator Knowledge Base) para identificar onde se encontram as informações necessárias para responder a consulta. Ele interage com o Gerenciador de Cache e com o Gerenciador de Data Warehouse, além de tratar questões de otimização de consultas.

O **Gerenciador de Fontes**, por sua vez, interage com os Wrappers, enviando subconsultas e recebendo os resultados. Também monitora as fontes com o objetivo de identificar dados que possam ser materializados no *data warehouse*.

Além do Mediador, o Ambiente de Integração de Dados possui um Gerenciador de Cache, um Gerenciador de Data Warehouse e uma base de conhecimento do mediador. O primeiro é responsável por gerenciar o conteúdo que se encontra na cache, o que inclui atividades como análise do espaço disponível, políticas de substituição de consultas armazenadas e atualização do conteúdo. O Gerenciador de Data Warehouse analisa critérios de materialização de dados e realiza a manutenção do *data warehouse*. Finalmente, a base de conhecimento do mediador armazena o esquema de mediação e as assertivas de correspondência – usadas para representar semanticamente correspondências entre entidades de fontes distintas e seus atributos.

2.2.3.4 Ambiente do Usuário

Nesse ambiente são especificados e mantidos os requisitos dos usuários para o esquema de mediação. A partir daí, modificações nos requisitos são propagadas para o esquema do mediador e para os módulos responsáveis pela manutenção das consultas de mediação que compõem o esquema.

2.3 Problemas e Desafios na Integração de Dados

Halevy e seu grupo realizaram uma retrospectiva sobre os dez anos de pesquisa na área de integração de dados, desde o projeto Manifold [Levy et al. 1996] até a busca por arquiteturas P2P [Halevy et al. 2006]. Nesta retrospectiva, eles apontam direções de pesquisa consideradas fundamentais para a solução de problemas de integração:

- Geração de mapeamentos de esquema

  Definir mapeamentos, assim como realizar sua manutenção, requer um "*expertise*" em banco de dados (para expressá-los de modo formal) e conhecimento do negócio (para entender o significado dos esquemas sendo mapeados). Assim, muita pesquisa tem sido realizada na construção de ferramentas para geração semi-automática destes mapeamentos [Castano S. and Antonellis V. (1999); Do H. and Rahm E. (2002)], um problema considerado "*AI-Complete*" (ou seja, necessita de técnicas de Inteligência Artificial), visto que a idéia é reduzir a necessidade de intervenção humana, o máximo possível. Técnicas como similaridade lingüística entre elementos de

esquemas ou técnicas de aprendizado de máquina têm sido utilizadas com tal propósito.

- Processamento de consultas adaptativo

  Uma vez que uma consulta tenha sido formulada sobre um esquema global, ela será reformulada sobre o conjunto de fontes disponíveis e ela precisa ser executada o mais eficientemente possível. O contexto sobre o qual um sistema de integração opera é altamente dinâmico, ou seja, em ambientes como tais, um otimizador de consultas tem muito menos informação disponível do que num ambiente tradicional de banco de dados. Como resultado, duas situações podem acontecer: o otimizador pode não ter informação suficiente para decidir sobre um bom plano de execução ou um plano que, a princípio, pareça bom, pode não o ser se as fontes não responderem exatamente da forma esperada. Muita pesquisa tem sido realizada com o intuito de tornar a execução da consulta o mais adaptativa possível.

- XML, XML, XML

  É impossível ignorar o papel que a linguagem XML vem exercendo no escopo da integração de dados. XML é atualmente um padrão sintático para compartilhamento de dados entre diversas fontes, entretanto carece de opções que facilitem a integração sob o ponto de vista semântico. Desta maneira, pesquisas têm sido realizadas com o objetivo de prover semântica [Kashyap, Sheth 1996b] e linguagens de consulta para dados XML, como por exemplo a linguagem XQuery [Chamberlin et al. 2001].

- Gerenciamento de Modelo

  O objetivo do gerenciamento de modelo é prover uma álgebra para manipulação de esquemas e mapeamentos, de modo que as operações não precisem ser reinventadas em cada solução de implementação. Exemplos de operadores já formalizados são: criação de mapeamento, inversão e composição de mapeamentos [Fagin et al. 2004], *merging* e diferenciação de esquemas [Shvaiko, Euzenat 2005]. Muitos têm buscado o entendimento destes operadores, mas muito trabalho ainda necessita ser realizado.

- O Papel da Inteligência Artificial

  A comunidade de Inteligência Artificial tem contribuído bastante na pesquisa sobre soluções para integração de dados. Lógica descritiva, por exemplo, tem sido amplamente usada para descrever relacionamentos entre fontes de dados [Catarci, Lenzerini 1993], para representar um esquema global ou para otimização de consultas semânticas. Além disso, a utilização de agentes inteligentes na reformulação e processamento de consultas e o aprendizado de máquina na geração semi-automática de mapeamentos têm sido fundamentais.

- Gerenciamento de Dados *Peer-to-Peer*

  O surgimento de sistemas de compartilhamento de arquivos em ambientes P2P inspirou a pesquisa por gerenciamento de dados neste tipo de ambiente. A utilização deste tipo de topologia propicia dois benefícios básicos:

- Uma arquitetura P2P oferece um mecanismo verdadeiramente distribuído, sem a necessidade da empresa ter de obrigatoriamente criar um esquema global para poder possibilitar a integração de seus dados; em vez disso, a partir de um esquema de exportação, mapeamentos vão ser gerados entre um conjunto de pontos vizinhos e integrações complexas irão acontecer de acordo com caminhos semânticos obtidos a partir dos mapeamentos.
- A diversidade estrutural e semântica das fontes de dados determina a complexidade e inviabilidade de se ter um único esquema global; com ambientes P2P o compartilhamento dos dados ocorre entre vizinhos da rede.

2.4 Sistemas P2P e PDMS

Um sistema *Peer-to-Peer* (P2P) é caracterizado pelo compartilhamento de recursos computacionais e serviços através da comunicação direta e descentralizada entre os sistemas envolvidos [Ooi et al. 2003]. Caracteriza-se por ser um ambiente de computação distribuída sem controle centralizado onde o software que é executado em cada ponto (*peer*) é equivalente em funcionalidade e tanto atua como cliente quanto servidor.

Comparando a uma arquitetura cliente-servidor, onde o número de servidores é fixo, sistemas P2P são mais flexíveis e extensíveis, visto que, quando novos pontos entram na rede, a capacidade total do sistema aumenta, enquanto que nos ambientes clientes-servidores, a adição de novos clientes reduz o desempenho do sistema [Zhao 2006].

Sistemas P2P são atualmente uma realidade entre usuários que desejam compartilhar arquivos. Um dos exemplos mais difundidos é o sistema Napster [Napster 2007] – para compartilhamento de música, que logo no início de seu funcionamento conseguiu a marca de mais de 25 milhões de usuários. Desde então, vários outros sistemas têm sido desenvolvidos [Gnutella 2007, Kazaa 2007], principalmente com o objetivo de prover compartilhamento de arquivos, mensagens instantâneas e trabalho colaborativo [Rouse 2006].

Embora muito difundidos e úteis, sistemas P2P existentes funcionam geralmente para casos simples de compartilhamento de arquivos, suportando funções limitadas, como busca por palavra-chave. Entretanto, em outros cenários, como a troca de recursos específicos de áreas como educação, saúde, economia, entre outras, as pesquisas são bem mais complexas e necessitam de padrões para a descrição e busca dos metadados. Se pensarmos em um cenário onde cientistas de instituições diferentes desejam cooperar e compartilhar suas pesquisas e experimentos, percebemos que dados mais complexos serão manipulados, assim como metadados serão necessários para prover seu entendimento. Dentro deste contexto, inicialmente desenvolvidos para permitir compartilhamento de dados não estruturados, sistemas P2P vêm evoluindo para sistemas de gerenciamento de dados estruturados através dos chamados *Peer Data Management Systems* ou PDMS.

Um PDMS é um sistema de gerenciamento de dados que utiliza uma arquitetura P2P e busca aliar os benefícios de sistemas P2P, como a ausência de autoridade central, com o poder semântico dos bancos de dados [Zhao 2006]. Deste modo, a idéia é que cada ponto possa atuar como cliente realizando consultas e, ao mesmo tempo, como servidor, provendo

resultados às consultas formuladas. O Capítulo 4 apresentará detalhes conceituais e técnicos sobre PDMS, além de descrever exemplos de sistemas desta categoria.

2.5 Considerações

A atual dinamicidade das organizações, com constantes mudanças e evoluções nos modelos de negócios e a necessidade de disponibilização ágil das informações demanda a utilização de arquiteturas de integração de dados mais flexíveis e extensíveis, a um custo mais baixo, e ao mesmo tempo, que permitam o fácil acesso.

Esta proposta de tese está inserida dentro do escopo de integração de dados. Com o objetivo de fornecer um panorama acerca das tecnologias e soluções disponíveis, bem como dos desafios e problemas a serem enfrentados, este capítulo descreveu a problemática da integração. Conceituou sistemas de integração de dados, mostrou abordagens utilizadas para sua construção e um estudo de caso. Além disso, dentro das perspectivas apresentadas, introduziu o conceito de sistemas P2P e dos gerenciadores de dados para estes ambientes – foco desta pesquisa.

Capítulo 3

Enriquecimento Semântico dos Sistemas de Integração de Dados

O desenvolvimento de aplicações computacionais cada vez mais complexas e, ao mesmo tempo, adaptáveis e flexíveis, demanda a criação de mecanismos que aprimorem seus serviços, enriqueçam sua semântica e provejam funcionalidades que aproximem mais o "homem" da "máquina", de maneira transparente.

A integração de dados está relacionada com a unificação de dados que compartilham semânticas comuns mas que são originários de fontes não relacionadas. Como visto no capítulo 2, necessariamente quando se busca integrar dados, torna-se essencial tratar e gerenciar problemas de heterogeneidade. Muita pesquisa tem sido realizada com o propósito de lidar com dados heterogêneos que possuem diferentes conceitos e interpretações.

Neste capítulo, abordagens utilizadas para enriquecer semanticamente sistemas de integração de dados são exploradas. Estas abordagens dizem respeito ao uso de metadados, ontologias e contextos. Assim, inicialmente, apresentamos o conceito de metadados, seus tipos e padrões. Em seguida, o conceito de ontologia é explicado, juntamente com metodologias para sua construção, seus tipos e representações e seu papel na integração de dados. Depois, o conceito de contexto é abordado, o que implica em apresentar sistemas sensíveis ao contexto, técnicas de representação do contexto e mostrar o papel do contexto dentro do âmbito da integração de dados. E, por fim, algumas considerações sobre os três conceitos aplicados à integração de dados são tecidas.

3.1 Metadados

Metadados são normalmente definidos como dados sobre dados. A palavra originou-se do latim *metá* que significa "além", "através de" ou "sobre", ou seja, dados sobre outros dados. Metadados são considerados hoje uma tecnologia capaz de tratar as informações relativas aos dados, inclusive aquelas relacionadas ao contexto em que eles são utilizados. Definições gerais de metadados são ilustradas em [Mori, Carvalho 2004]:

- Metadados são dados que descrevem atributos de um recurso, geralmente relacionados à localização, descoberta, documentação, avaliação ou seleção;
- Metadados fornecem o contexto para entender os dados através do tempo;
- Metadados são o instrumental para transformar dados brutos em conhecimento.

Metadados são, na verdade, descrições de dados armazenados em bancos de dados ou, de maneira geral, dados sobre dados armazenados em um dicionário de dados. Este poderá conter uma seção descritiva sobre como os dados são subdivididos (por exemplo, em arquivos, registros e campos), bem como seu relacionamento. Outra seção poderá conter os próprios metadados, assim como as descrições de cada campo (por exemplo, nome do campo, descrição, formato). A finalidade principal dos metadados é, portanto, documentar e organizar de forma estruturada os dados das organizações, procurando minimizar esforços de recuperação e manutenção.

Os metadados tratam a interoperabilidade em nível de gerenciamento, facilitando a recuperação de uma informação contida num banco de dados, sendo atualmente ferramentas essenciais para a integração de dados. Sua importância pode ser compreendida quando analisamos o panorama da Web Semântica – uma extensão da Web onde o significado da informação é bem definido, de modo que dados possam ser compartilhados e reutilizados entre aplicações, empresas e comunidades [W3C Metadata]. Tomemos como exemplo uma pesquisa sobre marcas e produtos de chocolate, num mecanismo de busca como o Google através da palavra-chave "Chocolate". O resultado desta pesquisa é bastante amplo, mas, ao mesmo tempo, pobre devido à incapacidade de se identificar o real significado ou o contexto do dado sendo pesquisado. Utilizando-se metadados, os sites dos fabricantes poderiam ser rotulados da forma pela qual desejassem ser encontrados, como pelas categorias "marca" ou "produto". Assim, se na hora da pesquisa o contexto do dado fosse identificado e os metadados associados, o resultado seria mais relevante e completo.

Da mesma forma, em sistemas de integração de dados, metadados são utilizados para descrever os conteúdos das fontes e podem ser também utilizados para descrever resultados parciais de consultas.

3.1.1 XML e Padrões de Metadados

A linguagem HTML [Guimarães 2005] foi criada com o intuito de facilitar a publicação de informações pela Web. Assim, independente de plataforma, ela se tornou um padrão de exibição de informações. Nela, metadados são utilizados na forma de *tags*, mas estas não informam nada sobre o dado em si, apenas sobre sua formatação.

O conceito de *tags* personalizadas surgiu através da linguagem SGML [Guimarães 2005] que derivou a atual linguagem XML, criada como um padrão para a interoperabilidade e intercâmbio de informações. XML é considerada uma coleção de regras, diretivas e convenções para projetar formatos textuais para dados [Mori, Carvalho 2004]. Ela estende as opções de marcação da HTML, de modo que o desenvolvedor pode definir seus próprios metadados (através das *tags*). O resultado é a possibilidade de organizar os arquivos em formato XML, preenchê-los de metadados e publicá-los na Internet, com o propósito de facilitar a Web Semântica. Mas, diante disso, como organizar tais metadados? Como é possível documentá-los?

Com o propósito de facilitar o compartilhamento de dados entre diferentes instituições, surgiram os padrões de metadados. Estabelecer padrões de metadados possibilita às organizações melhor documentar os seus dados. O esforço neste sentido deve buscar uma

padronização mínima e necessária. Alguns tipos importantes são os padrões de conteúdo dos metadados, padrões de intercâmbio de dados por meio eletrônico e padrões para modelos de dados [Almeida 2001]. Existem muitos padrões de metadados, voltados a domínios diferentes. Exemplos destes padrões são:

- Federal Data Geographic Commitee (FGDC) – descrição de dados geo-espaciais [FGDC 2007];
- Machine Readable Cataloging Record (MARC) – catalogação bibliográfica [MARC 2007];
- Dublin Core (DC) – dados sobre páginas Web [DC 2007];
- Consortium for the Interchange of Museum Information (CIMI) – informações sobre museus [CIMI 2007].

A título de ilustração, vamos descrever aspectos do padrão Dublin Core.

Dublin Core

O padrão Dublin Core (DC) constitui um conjunto de elementos para descrever uma ampla quantidade de recursos eletrônicos [Mori, Carvalho 2004]. O DC compreende quinze elementos semânticos estabelecidos por grupos interdisciplinares de bibliotecários, cientistas da computação, profissionais de museus e outros estudiosos, mostrados no Quadro 3.1, de acordo com [Mori, Carvalho 2004].

Neste padrão, existem duas classes de termos: elementos (nomes) e qualificadores (adjetivos). Cada elemento é opcional e pode ser repetido. Um número limitado de qualificadores (refinam o significado da informação) é disponibilizado. O DC busca seguir algumas premissas:

- Simplicidade de criação e manutenção: o conjunto de elementos foi mantido pequeno a fim de permitir que um não-especialista crie registros descritivos de forma fácil.
- Semântica comumente compreendida: a descoberta de recursos pode ser facilitada através dos elementos comuns suportados pelo DC.
- Escopo internacional: originalmente desenvolvido em inglês, atualmente conta com versões em outras linguagens como japonês, português, alemão, grego, espanhol, finlandês, entre outras.
- Extensibilidade: outras comunidades podem criar e administrar conjuntos adicionais de elementos de metadados.

Um exemplo de um registro Dublin Core, em RDF (*Resource Description Framework)*, é apresentado na Figura 3.1. Neste exemplo, nas linhas 2 e 3, faz-se a declaração de quais fontes de metadados (*namespaces*) o registro está incorporando. Na linha 2 (xmlns:rdf) o registro incorpora a sintaxe referente ao endereço indicado (“http://www.w3.org/1999/02/22−rdf−syntax−ns#”). Da mesma maneira, a linha 3 (xmlns:dc) referencia o endereço “http://purl.org/dc/elements/1.1/” para utilizar as propriedades do padrão Dublin Core.

Quadro 3.1: Elementos do Padrão Dublin Core

Número	Elemento	Descrição
1	*Title*	Nome dado ao documento eletrônico
2	*Author or Creator*	Pessoas ou organizações responsáveis pelo conteúdo intelectual do objeto. Ex: fotógrafo – foto digital
3	*Subject and Keywords*	Assunto do documento, podendo ser definido a partir de sistemas de classificação (CDD – Classificação Decimal de Dewey, CDU – Classificação Decimal Universal, LCSH – *Library of Congress Subject Headings*) ou através de uma palavra ou conjunto de palavras.
4	*Description*	Resumo ou descrição do conteúdo
5	*Publisher*	Entidades responsáveis pela disponibilização do documento. Ex: editor, universidade ou entidade corporativa.
6	*Other Contributors*	Outras pessoas que contribuíram para realização da obra.
7	*Date*	Data de publicação do documento
8	*Resource Type*	Tipo do recurso, como home page, poema, dicionário, arquivo de dados, entre outros.
9	*Format*	Formato eletrônico do documento: postscript, PDF ou HTML.
10	*Resource Identifier*	Série ou número usado para identificar o documento (URL, ISBN)
11	*Source*	Fonte do documento
12	*Language*	Idioma do conteúdo do documento
13	*Relation*	Relacionamento com outros documentos impressos ou eletrônicos – por exemplo imagens em um documento, capítulos em um livro ou itens em uma coleção.
14	*Coverage*	Localização espacial ou duração temporal característica do documento.
15	*Rights management*	Informações sobre copyright.

3.1.2 Tipos de Metadados

Metadados podem ser classificados como estruturais ou semânticos [Mori, Carvalho 2004]. Estruturais são aqueles que descrevem a organização e estrutura dos dados, por exemplo, informações sobre o formato, os tipos de dados e os relacionamentos sintáticos entre eles.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">

   <rdf:Description
      rdf:about="http://media.example.com/audio/guide.ra">
      <dc:creator>Rose Bush</dc:creator>
      <dc:title>A Guide to Growing Roses</dc:title>
      <dc:description>
         Describes process for planting and nurturing
         different kinds of rose bushes.
      </dc:description>
      <dc:date>2001-01-20</dc:date>

   </rdf:Description>
</rdf:RDF>
```

Fig. 3.1 Registro Dublin Core Expresso em XML-RDF

Metadados semânticos fornecem informações sobre o significado dos dados e seus relacionamentos semânticos, por exemplo, informações sobre o valor do dado, como unidades de medida e escala ou sobre sua criação, fonte e qualidade (atualidade e precisão). Metadados constituem uma poderosa ferramenta semântica para descrever o contexto de uma informação de maneira não ambígua ou redundante.

Dados e metadados podem ser representados semanticamente por meio de ontologias. Cada dado vai possuir metadados associados a ele que se encontram enquadrados conceitualmente através de uma ontologia. Metadados podem especificar o relacionamento que este dado (ou objeto) possui com elementos do mundo real. Assim, o uso de ontologias visa facilitar a correta interpretação dos metadados associados a um objeto dentro de um contexto. O conceito e aplicação de ontologias e de contexto serão apresentados nas próximas seções.

3.2 Ontologias

Uma ontologia é uma especificação explícita de uma conceitualização compartilhada em que objetos, conceitos, entidades e relacionamentos do mundo real são definidos em uma determinada área de interesse ou domínio de conhecimento [Gruber 1995]. Historicamente o termo ontologia teve origem no grego *ontos*, ser e *logos*, palavra. É um termo introduzido na filosofia com o objetivo de distinguir o estudo do ser humano como tal, do estudo de outros seres vivos. A origem é a palavra aristotélica "categoria" que pode ser usada para classificar e caracterizar alguma coisa [Almeida, Bax 2003].

Borst [Borst 1997] define ontologia como uma especificação formal e explícita de uma conceitualização compartilhada. Nesta definição, "formal" significa legível para computadores, "especificação explícita" diz respeito a conceitos, propriedades, relações, funções, restrições e axiomas que são explicitamente definidos; "compartilhado" quer dizer

conhecimento consensual e "conceitualização" se refere a um modelo abstrato de algum fenômeno do mundo real.

Uma ontologia define então um vocabulário comum para o compartilhamento de informações em um domínio. Trata-se, na realidade, de uma estrutura de conhecimento que inclui definições de conceitos básicos de um universo e as relações entre esses conceitos. Todo modelo de conhecimento é, portanto, comprometido com alguma conceituação, implícita ou explicitamente [Gruber 1995]. Para representar um certo fenômeno ou parte do mundo – um universo ou domínio - é necessário concentrar a atenção em um número limitado de conceitos, suficientes e relevantes, com o objetivo de criar uma abstração do fenômeno em questão. Assim, de maneira geral, uma ontologia inclui [Gruber 1993]:

- Conceitos: representam entidades, idéias ou noções em um determinado domínio e incorporam dois aspectos importantes - intenção e extensão. A intenção do conceito é seu significado ou sua completa definição, enquanto que a extensão é o conjunto de elementos de um determinado universo para os quais o conceito se aplica. Por exemplo, um conceito "Lagoa" pode ter como elementos extensionais "Parque Sólon de Lucena" e "Lagoa Rodrigo de Freitas".
- Relações: representam um tipo de interação entre os conceitos no domínio, sendo a cardinalidade sempre n:n.
- Funções: são um caso especial de relações, sendo a cardinalidade n:1.
- Axiomas: são as sentenças que são sempre verdadeiras, independente da situação (são constantes).
- Instâncias: são utilizadas para representar os elementos do domínio.

Na computação, ontologias são tratadas como artefatos compostos de um vocabulário de conceitos, suas definições e suas possíveis propriedades, um modelo gráfico mostrando todas as possíveis relações entre os conceitos e um conjunto de axiomas formais que restringem a interpretação dos conceitos e relações, representando de maneira clara e não ambígua o conhecimento de um domínio.

Uma ontologia deve servir a um propósito específico. Num sentido mais amplo, ela pode ser de autoria neutra (enfatiza a reutilização de informações), pode ser utilizada para especificação (enfatiza documentação e manutenção) e pode servir de acesso comum à informação (enfatiza o acesso à informação) [Uschold, Jasper 1999]. Esta última categoria se aplica quando a informação desejada é expressa em um vocabulário inacessível e a ontologia possibilita o seu entendimento, proporcionando conhecimento compartilhado dos termos ou inter-relacionando grupos de termos [Almeida, Bax 2003].

Um dos grandes interesses na construção e uso de ontologias é tornar o conhecimento sobre o mundo real processável por máquinas. A Figura 3.2 ilustra as três principais aplicações de ontologias e o nível de formalismo na especificação do conhecimento exigido em cada uma [Mika e Akkermans 2005].

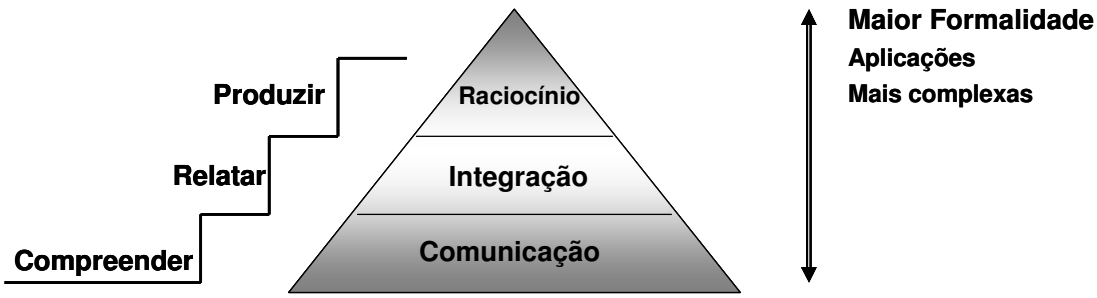

Fig. 3.2 Aplicações de Ontologia (Adaptada de [Mika e Akkermans 2005])

Na base da pirâmide, está o suporte à comunicação e compartilhamento do conhecimento entre agentes humanos e de software, uma vez que ontologias representam uma compreensão compartilhada sobre um conjunto de termos e descrições e descrevem de forma precisa propriedades do ambiente e os vários conceitos utilizados. O suporte à integração e reuso de conhecimento indica que diferentes ontologias, armazenadas em diferentes locais e criadas por diferentes autores, podem ser reutilizadas para compor uma ontologia em larga escala sem a necessidade de começar do zero. O suporte ao raciocínio exige um maior formalismo na representação do conhecimento, mas por outro lado permite construir sistemas mais complexos e produzir novos conhecimentos. Mecanismos de raciocínio são usados para inferir conhecimento complexo e para verificar e resolver inconsistências [Vieira et al. 2006].

De modo geral, os benefícios mais comuns obtidos através do uso de ontologias são:

- Definir semântica independentemente das representações dos dados;
- Compartilhar o conhecimento: agentes e serviços têm um conjunto comum de conceitos enquanto interagem;
- Permitir inferência lógica: mecanismos de raciocínio lógico são utilizados para deduzir informações (como, por exemplo, classificação de instâncias);
- Tornar as verdades absolutas do domínio explícitas;
- Separar o conhecimento do domínio do conhecimento operacional;
- Reutilizar o conhecimento: ontologias disponíveis na Web podem ser reutilizadas, evitando retrabalho de definição.

3.2.1 Tipos de Ontologias

Segundo Guarino [Guarino 1997], levando em consideração seu conteúdo, as ontologias podem ser classificadas nas seguintes categorias:

- **Ontologias genéricas (*upper* ontologies):** descrevem "meta-conceitos" gerais, limitando-se aos conceitos que são meta, genéricos, abstratos e filosóficos, tais como, espaço, tempo, matéria, objeto, evento, ação, entre outros, que são independentes de um problema ou domínio particular. As ontologias genéricas SUMO – *Suggested Upper Merged Ontology* – [SUMO, 2004] e *OpenCyc* [OpenCYC, 2004] são os exemplos mais conhecidos de ontologias genéricas. Tais ontologias são padrões do grupo de trabalho de ontologias genéricas da IEEE – *Institute of Electrical and Electronics Engineers* – [SUOWG, 2004].

- **Ontologias de domínio**: expressam conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, tal como Medicina ou Biologia.
- **Ontologias de tarefas**: expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como, diagnose ou vendas;
- **Ontologias de aplicação**: descrevem conceitos dependentes do domínio e da tarefa particulares. Tais conceitos normalmente correspondem a papéis desempenhados por entidades do domínio quando da realização de uma certa atividade;
- **Ontologias de representação**: explicam as conceituações que fundamentam os formalismos de representação de conhecimento.

Ontologias típicas do mundo real incluem taxonomias na Web (ex: Yahoo! Categories), catálogos para compras on-line (ex: Catálogo de produtos da Amazon) e terminologias padrões específicas de domínio (ex: UMLS e Gene Ontology) [Cruz, Xiao 2005]. Como um banco de dados léxico on-line, WordNet é bastante utilizada para descoberta de relacionamentos semânticos entre conceitos. As ontologias de domínio são os tipos mais comumente desenvolvidos de ontologias.

3.2.2 Construção de Ontologias

Segundo [Marietto et. al 2002], o desenvolvimento e o uso de metodologias para a construção de ontologias é fundamental, na medida que retiram o caráter subjetivo desta atividade. A área de Engenharia Ontológica estuda os aspectos relacionados a tal construção, bem como o desenvolvimento de sistemas que utilizem ontologias em sua estrutura. A título de ilustração, apresentamos uma metodologia para construção de ontologias [Cantele et al. 2004]:

Grüninger e Fox

Esta metodologia encontra-se inserida no projeto TOVE [Grüninger e Fox 1994] e envolve a construção de um modelo formal do conhecimento. Suas fases incluem:

a. Captura de cenários, considerando que as ontologias são motivadas por cenários específicos para a aplicação;
b. Formulação de questões em nível informal;
c. Especificação da terminologia da ontologia a partir de uma linguagem formal: criar um conjunto de termos, a partir da terminologia informal, para especificar uma terminologia formal;
d. Formulação de questões de competência formal utilizando a terminologia formal.
e. Especificação de axiomas e definições para os termos da ontologia com a linguagem formal;
f. Estabelecimento de condições para caracterizar a completude da ontologia.

3.2.3 Representação de Ontologias

A representação do conhecimento de ontologias deve ser realizada por linguagens específicas, normalmente variantes da lógica descritiva que provêem alta expressividade e raciocínio. Alguns exemplos são [Cruz, Xiao 2005], [Cantele et al.2004]:

- **RDF e RDFS**: RDF (*Resource Description Framework*) é um modelo de dados desenvolvido pela W3C para descrição de recursos da Web [W3C]. RDF permite a especificação da semântica do dado de uma maneira padrão e interoperável. Em RDF, um par de recursos (nós) conectados por uma propriedade forma uma sentença – (recurso, propriedade, valor). RDFS (*RDF Schema*) é uma linguagem para descrição de vocabulários de dados RDF em termos de primitivas tais como: rdfs:Class, rdf:Property, rdfs:domain e rdfs:range. Assim, RDFS é usada para definir o relacionamento semântico entre propriedades e recursos.
- **DAML+OIL**: DAML-OIL (*DARPA Agent Markup Language-Ontology Interface Language*) é uma linguagem baseada na Web construída no topo de RDFS, como uma gramática XML. Inclui primitivas normalmente usadas em abordagens de engenharia de ontologias baseadas em *frames*, além de prover semântica formal e raciocínio baseados em abordagens de lógica descritiva. Além disso, ela integra tipos de dados XML Schema para interoperabilidade semântica em XML.
- **OWL**: OWL (*Web Ontology Language*) é uma linguagem de marcação semântica para publicação de compartilhamento de ontologias na Web, sendo um dos padrões definidos para a Web Semântica. Foi desenvolvida como uma extensão de RDF e derivada de DAML+OIL. OWL é uma linguagem de descrição de vocabulário mais rica, permitindo outras relações entre classes (como disjunção), cardinalidade, igualdade, propriedades mais fortemente tipadas, características de propriedades (como simetria) e classes enumeradas. Do ponto de vista formal, OWL possui uma versão denominada OWL-DL que permite explorar os mecanismos de raciocínio em lógica descritiva existentes.
- **XOL:** linguagem baseada em XML, projetada para proporcionar um formato para troca de definições entre ontologias no domínio da biologia molecular.

3.2.4 O Uso de Ontologias na Integração de Dados

Como a ontologia pode ser utilizada para prover acesso comum a fontes de dados, seu papel na integração de dados vem crescendo bastante. Desta maneira, em se tratando de integração de dados, o papel da ontologia se encontra geralmente em explicitar o conteúdo semântico das fontes de dados [Wache 2001] e/ou do esquema de referência.

As ontologias têm sido muito utilizadas para prover visões semânticas. Como as ontologias possibilitam uma compreensão comum e compartilhada de um domínio de conhecimento, diferenças conceituais e semânticas são reduzidas, obtendo-se, assim, uma maior comunicação entre os agentes de software e os usuários envolvidos. Dessa maneira, ontologias são utilizadas como meio de estabelecer correspondências e relações entre as diversas fontes e seus objetos específicos.

A Figura 3.3 [Wache 2001] apresenta uma taxonomia proposta para as ontologias de acordo com o papel desempenhado por elas em arquiteturas de integração de dados. Na primeira arquitetura, todas as fontes de dados são relacionadas a uma única ontologia global que contém o conjunto de termos a ser compartilhado por todas as fontes. Na arquitetura multi-ontologias, cada fonte de dados é representada por sua própria ontologia, não sendo necessária uma ontologia global, mas algum mecanismo que proveja uma relação inter-ontologias. Na arquitetura de ontologias combinadas, a semântica de cada fonte é descrita por sua própria ontologia, entretanto, cada uma é construída com base em um vocabulário comum que pode inclusive ser outra ontologia. Exemplos de sistemas de integração que fazem uso dessas arquiteturas encontram-se no Quadro 3.2.

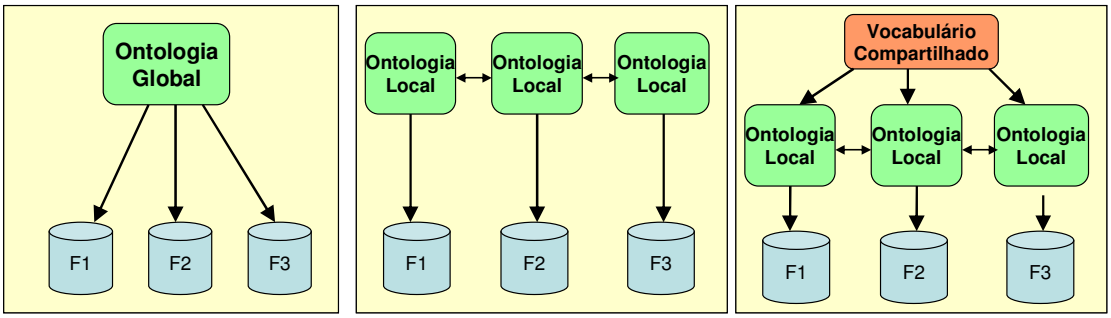

Fig. 3.3 Arquiteturas de Integração que Utilizam Ontologias

Quadro 3.2: Exemplos de Arquiteturas de Integração com Ontologias

Arquitetura	Exemplos
Ontologia Global	TSIMMIS [Chawathe 1994], SIMS [Arens, Hsu e Knoblock 1996]
Multi-Ontologias	OBSERVER [Mena et al. 2000], ONTOBROKER [Fensel et al. 1998]
Ontologias Combinadas	COIN [Bressan 1997], PICSEL [Goasdoué, Lattes, Rousset 2000]

As abordagens de ontologia global única e a híbrida são adequadas para construção de sistemas de integração de dados, sendo a primeira ainda mais adequada a sistemas que utilizam a abordagem GAV e a última para sistemas que fazem uso da abordagem LAV [Cruz, Xiao 2005]. Na abordagem GAV, para cada componente do esquema global, é escrita uma consulta sobre os esquemas locais. Na abordagem LAV, por outro lado, são definidas consultas que descrevem como obter a extensão das fontes de dados a partir do esquema global. Detalhes sobre estas duas abordagens serão vistos no Capítulo 5.

Em um sistema P2P, por outro lado, uma ontologia global pode existir em um super-ponto, ou seja, pode-se utilizar uma abordagem híbrida. Caso o sistema P2P possua uma topologia pura, a abordagem de multi-ontologias será a mais apropriada.

Cruz e Xiao [Cruz, Xiao 2005] identificam cinco papéis que uma ontologia pode desempenhar em sistemas que fazem integração de dados:

- Representação de Metadados: metadados (i.e. esquemas das fontes) em cada fonte de dados podem ser explicitamente representados por uma ontologia local, usando uma linguagem comum.
- Conceitualização Global: a ontologia global provê uma visão conceitual sobre esquemas de fontes sintaticamente heterogêneos.
- Suporte a Consultas de Alto-Nível: através de uma visão de alto nível, como uma provida por uma ontologia global, o usuário pode formular uma consulta sem conhecimento específico das diferentes fontes de dados. A consulta é então reescrita em consultas sobre as fontes, baseadas em mapeamentos semânticos entre as ontologias global e aquelas locais.
- Mediação Declarativa: o processamento de consultas num sistema ponto-a-ponto híbrido pode usar uma ontologia global como um mediador declarativo para reescrita de consultas entre pontos.
- Suporte a Mapeamentos: um tesauro, formalizado em termos de uma ontologia, pode ser usado para o processo de mapeamento com o objetivo de facilitar sua automatização. Um exemplo de tesauro que é utilizado para descobrir correspondências semânticas é o WordNet. Ele inclui um conjunto de termos e seus relacionamentos semânticos (como, hipernímia, sinonímia e hyponímia) e cada termo pode ter múltiplos significados.

3.2.5 Interoperabilidade de Ontologias

Muitos dos sistemas de integração de dados (como o OBSERVER [Mena et al. 1996]) utilizam mais do que uma ontologia para descrever a informação e prover visões semânticas dos dados. Em ambientes assim, onde as informações são heterogêneas e distribuídas, o uso de ontologias, além de prover uma teoria do domínio, provê também uma representação comum para a troca das informações. Deste modo, diferentes agentes de software podem compartilhar informações que serão úteis no processamento de consultas e integração dos dados. No entanto, quando se trabalha com diversas ontologias, são necessários mecanismos que garantam a interoperabilidade das mesmas.

Pesquisas vêm sendo efetuadas com o objetivo de prover mecanismos que garantam a compatibilidade de ontologias, principalmente no universo da Web Semântica. Alguns destes mecanismos são apresentados diagramaticamente através da Figura 3.4 e descritos resumidamente a seguir [Felicíssimo 2004]:

- **Combinação de Ontologias**: nesta estratégia, tem-se como resultado a versão das ontologias originais combinadas em uma ontologia única, com todos seus termos juntos e sem a definição clara de suas origens. Normalmente as ontologias originais descrevem domínios similares ou de sobreposição.
- **Alinhamento de Ontologias:** neste caso, tem-se como resultado as duas ontologias originais separadas, mas nestas são adicionadas às correspondências entre seus termos equivalentes. Estas correspondências permitem que as ontologias alinhadas reusem as

informações umas das outras. O alinhamento normalmente é realizado quando as ontologias são de domínios complementares.

- **Mapeamento de Ontologias:** no mapeamento de ontologias, tem-se como resultado uma estrutura formal com expressões que ligam os termos de uma ontologia nos termos de uma outra ontologia. Este mapeamento pode ser usado para transferir instâncias de dados, esquemas de integração e de combinação e para o processamento de consultas.
- **Integração de Ontologias:** neste mecanismo, tem-se como resultado uma ontologia única criada pela montagem, extensão, especialização ou adaptação de outras ontologias de assuntos diferentes. Na integração de ontologias, é possível identificar as regiões que foram criadas a partir das ontologias originais.

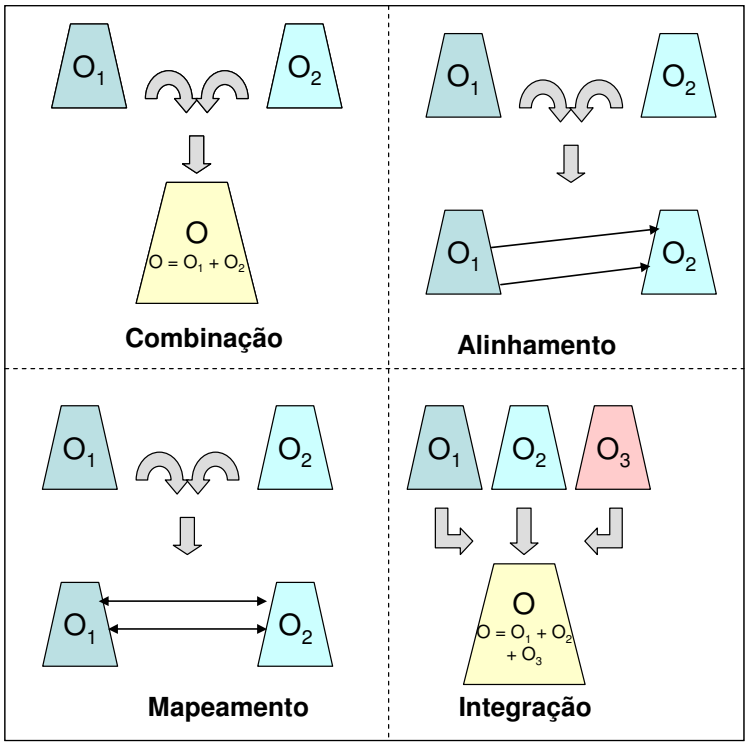

Fig 3.4 Estratégias de Interoperabilidade de Ontologias, adaptado de [Felicíssimo, 2004]

Ferramentas diversas estão sendo desenvolvidas com o propósito de automatizar inteira ou parcialmente estas estratégias de interoperabilidade de ontologias. Um exemplo é o conjunto de ferramentas PROMPT [Noy, Musen 2003], composta das seguintes ferramentas: *iPROMPT*, uma ferramenta interativa para combinação de ontologias; *AnchorPROMPT*, uma ferramenta automática baseada em grafos para alinhamento de ontologias; *PROMPTFactor*, uma ferramenta para extração de partes de ontologias e *PROMPTDiff*, uma ferramenta para identificação de diferenças entre duas versões da mesma ontologia.

3.3 Contexto

As pessoas utilizam, diariamente, informações contextuais para tomar decisões, fazer julgamentos ou interagir com outras pessoas. Entender o contexto em que ocorre uma determinada interação é fundamental para que os indivíduos possam responder de maneira apropriada à situação. Por exemplo, quando uma pessoa está em um estádio de futebol ou em um cinema ele sabe que o seu comportamento deve ser bem diferente em cada um desses locais. Enquanto no cinema ele deve desligar o celular, não fazer barulho e permanecer sentado, no estádio ele pode gritar, pular e vibrar com o seu time, e o celular pode ser utilizado sem ressalvas. Deste modo, contexto é o que está por trás da habilidade de discriminar o que é ou não importante em um dado momento e, com isso, permite ajudar indivíduos a melhorar a qualidade da conversação e a compreender certas situações, ações ou eventos [Vieira et al. 2006].

Em sistemas computacionais, o contexto é uma importante ferramenta de apoio à comunicação entre os sistemas e seus usuários. Compreendendo o contexto, o sistema pode se adaptar e mudar sua seqüência de ações, o estilo das interações e o tipo da informação fornecida aos usuários, em circunstâncias diversas. Além disso, o contexto auxilia os sistemas a, dinamicamente, habilitar ou desabilitar serviços e funcionalidades. Sistemas que utilizam o contexto para direcionar suas ações e comportamento recebem o nome de **Sistemas Sensíveis ao Contexto**.

Por outro ângulo, contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade [Dey et al. 2001]. Esta entidade pode ser um usuário de uma aplicação ou um objeto computacional como um dispositivo, uma fonte de dados ou uma tabela num banco de dados relacional. Como uma ilustração, suponhamos um banco de dados relacional que armazene a seguinte tupla: *Avenida ('Epitacio Pessoa', 3000, 'Bom').* Como estes valores podem ser interpretados? Qual a semântica associada a eles? Na busca de prover sentido a estes dados, algumas possibilidades de estrutura podem ser consideradas: (i) *Avenida (nome, comprimento, estado de conservação)*; ou *(ii) Avenida (nome, largura, nível de limpeza)*

A questão chave que pode ser observada é que cada banco de dados mantém suas próprias definições de forma independente. Assim, informações contextuais podem ser usadas para especificar as definições assumidas por cada projeto de banco de dados com o objetivo de entender sua "semântica". No domínio de banco de dados, contexto pode ser usado de diversas maneiras com o intuito de capturar a semântica relevante de um objeto e seus relacionamentos com outros objetos [Kashyap, Sheth 1996a]. Metadados, neste caso, são exemplos de informações que podem ser utilizadas como contexto [Souza et al. 2006].

O conceito de contexto tem sido objeto de estudo em diversas áreas (não somente computacionais) como a Psicologia Cognitiva e a Lingüística. Em Computação, as áreas da Computação Ubíqua e Inteligência Artificial foram as pioneiras nos estudos e utilização do conceito de contexto (por exemplo [Chen e Kotz 2000; McCarthy 1993]), e com isso foram as que demonstraram o potencial da aplicação desse conceito nos sistemas computacionais. Recentemente, outras áreas vêm buscando tirar proveito do conceito de contexto em prol de

melhorias em suas aplicações. É o caso dos Sistemas Colaborativos, sistemas de Hipermídia Adaptativa e Sistemas de Integração de Dados – foco deste trabalho.

Uma teoria bem aceita sobre o conceito de contexto foi apresentada por McCarthy (1993) e tem sido instrumento de inspiração para novas pesquisas. Sua idéia é que ao definir um axioma, ele somente é verdade dentro de um contexto, podendo ser falso em outro. Assim, ele denota uma "verdade relativizada dentro de um contexto" como um predicado especial ***ist(c,p)*** que significa que a proposição **p** é verdadeira (is true) dentro do contexto **c**. Um exemplo de um axioma associado a um contexto é:

C′: ist(contexto-de("Histórias de Sherlock Holmes"),("Holmes é detetive"))

Esse exemplo indica que o predicado "Holmes é detetive" é verdadeiro se o contexto for "Histórias de Sherlock Holmes".

Assim, o conceito de contexto pode ter diferentes visões, de acordo com o domínio corrente, mas existe consenso em relação a alguns aspectos [Zacarias et al. 2004]:

- A natureza relacional do contexto, ou seja, o contexto não é um conceito autônomo mas existe, apenas, quando relacionado a uma entidade;
- O contexto é abordado como uma coleção de itens (ex. conceitos, regras, recursos, proposições e suposições) associados a uma situação específica (ex. ambiente, domínio, entidade, tarefa ou agente).

Dessa maneira, o contexto pode ser visto como um conjunto de características situacionais, parâmetros ou dimensões. Isso significa que se uma determinada informação pode ser usada para caracterizar uma situação, então essa pode vir a ser uma informação contextual. Por exemplo, o fato de estar chovendo ou não. Em um sistema acoplado a um automóvel, essa informação é relevante visto que a chuva implica na visibilidade, na velocidade ou na escolha de local para estacionar; por outro lado, em um sistema voltado para um museu, essa informação deixa de ser relevante, pois o museu é um prédio coberto e, portanto, ela não deve ser considerada no contexto.

3.3.1 Sistemas Sensíveis ao Contexto

Sistemas considerados convencionais, que não fazem uso de informações contextuais, executam suas solicitações levando em consideração apenas as informações fornecidas pelos usuários explicitamente, como ilustrado na Figura 3.5.

Fig 3.5 Aplicação Tradicional

Os sistemas sensíveis ao contexto, por outro lado, consideram as informações explícitas fornecidas pelos usuários, aquelas armazenadas em uma base de conhecimento contextual, aquelas inferidas por meio de raciocínio e/ou ainda aquelas percebidas a partir do

ambiente, conforme mostra a Figura 3.6. Com base nessas informações contextuais, a aplicação tem condições de identificar e otimizar fluxos de interação, recuperar históricos, determinar ações e adaptações. Como resultado, serviços mais relevantes e direcionados ao que o usuário necessita naquele momento são providos.

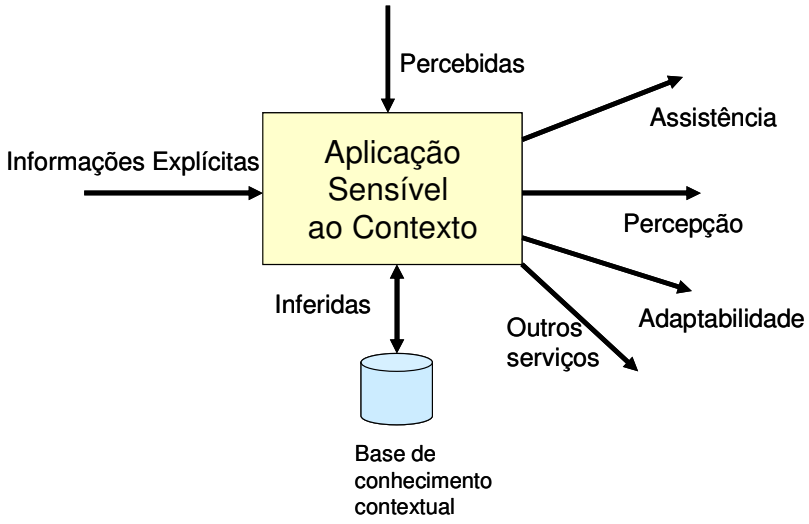

Fig. 3.6 Aplicação Sensível ao Contexto

Um dos serviços refere-se à assistência às tarefas que estão sendo realizadas, visando facilitar o uso do sistema dentro do ambiente onde o usuário está inserido. Inclui tutoria inteligente como aconselhar e/ou avisar sobre tarefas que ele pode realizar para obter seus objetivos, assim como recomendações que realizam filtragens de itens (ex. materiais, objetos e pessoas) com base no perfil e interesses dos usuários.

Outro serviço provido é a percepção do contexto, que se refere a notificar o usuário sobre o contexto associado a pessoas e interações do seu interesse. Esse serviço é muito relevante em sistemas colaborativos, em que membros de um grupo devem conhecer e compreender as ações realizadas pelos demais, como forma de coordenar suas próprias ações.

Um terceiro serviço resultante da utilização de contexto é a adaptação, ou seja, a capacidade do sistema modificar seu comportamento, respondendo de forma oportuna às mudanças ocorridas no ambiente e às ações e definições dos usuários. Um exemplo de adaptação é a personalização de sites *Web*, realizada a partir da análise do perfil do usuário e de seus padrões de navegação.

Existem também outros serviços, dependendo do tipo do sistema, onde o contexto pode ser empregado para enriquecer o conhecimento que já se tem sobre alguma coisa, antes mesmo de se produzir um resultado relevante para um usuário. Em outras palavras, o sistema sensível ao contexto pode realizar ações que, a princípio, não são direcionadas à interação com o usuário. Por exemplo, definições (metadados) sobre entidades armazenadas em diferentes bancos de dados podem ser enriquecidas semanticamente através do uso de contexto com o objetivo de integrar esquemas de dados e facilitar o processamento de consultas [Souza et al. 2006].

Assim, percebe-se que utilizar contexto em sistemas computacionais é uma tarefa não trivial. A lógica da aplicação sensível ao contexto é fortemente dependente da importância que o contexto vai desempenhar nas suas funcionalidades. Por outro lado, usar contexto atualmente é uma premissa quando se desenvolve sistemas mais inteligentes e semanticamente enriquecidos. Desta maneira, requisitos básicos que devem ser considerados na construção de sistemas sensíveis ao contexto incluem: aquisição, representação, processamento, armazenamento, uso (percepção, assistência e adaptação) e compartilhamento (com políticas de segurança e privacidade) [Vieira et al. 2006]. Para facilitar o desenvolvimento e promover a reusabilidade, gerenciadores de contexto (frameworks, middlewares, engines, e outros) podem ser utilizados [Vieira 2006]. Assim, a aplicação em si fica desacoplada dos serviços básicos de aquisição, processamento e disseminação do contexto.

3.3.2 Técnicas para Representação de Contexto

Um sistema sensível ao contexto requer que informações contextuais sejam trocadas e utilizadas por diferentes entidades, como agentes humanos e de software, dispositivos e serviços, com uma mesma compreensão semântica. Para isso, um modelo de contexto apropriado deve dar suporte à interoperabilidade semântica e deve permitir que esquemas comuns sejam compartilhados entre as diferentes entidades [Wang et al. 2004].

Alguns aspectos devem ser considerados quando se avalia técnicas para representar contexto [Souza et al. 2006]: o modelo deve ser portável; é desejável que tenha ferramentas para edição, checagem de tipos e conversão entre diferentes formatos; o modelo deve ser formal o suficiente pra facilitar sua definição, prover raciocínio e permitir reusabilidade, assim como o modelo deve prover mecanismos de raciocínio.

Fig. 3.7 Técnicas de Representação de Contexto

Diversas técnicas vêm sendo utilizadas na literatura por diferentes sistemas para representar o contexto, algumas mais simples, outras mais complexas de se implementar, conforme mostra a Figura 3.7. A seguir, apresentamos alguns exemplos destas técnicas:

- **Par Chave-Valor**: esta técnica de modelagem utiliza a estrutura de dados mais simples para representar o contexto. Nesta, o contexto é modelado por meio de

pares compostos por uma chave, que identifica o atributo de contexto, e por um valor associado a essa chave. Como exemplo de uso, encontram-se os sistemas operacionais que utilizam o modelo Chave-Valor para armazenar variáveis de ambiente. Exemplo de sistema sensível ao contexto que adota tal modelo é o Context Toolkit [Dey et al. 2001].

- **Linguagens de Marcação**: Esta abordagem utiliza o padrão XML (eXtensible Markup Language) para modelagem das informações contextuais, tendo sido utilizada para a modelagem de perfis de usuário e dispositivos. Um exemplo é o CSCP (*Comprehensive Structured Context Profiles*) [Held et al. 2002] que representa o contexto como perfis de sessão.

- **Orientação a Objetos:** visa empregar propriedades como encapsulamento, herança e reusabilidade, para lidar com os problemas de dinâmica do contexto [Strang e Linnhoff-Popien 2004]. Os detalhes do processamento do contexto são encapsulados no nível do objeto de contexto e o acesso à informação contextual é realizado por meio de interfaces. Um exemplo do uso dessa abordagem é o modelo proposto para uma versão sensível ao contexto do jogo Go [Bouzy e Cazenave 1997].

- **Mapa de Tópicos**: Um mapa de tópicos (centrado em três definições principais - Tópicos, Associações e Ocorrências) permite definir relações entre objetos físicos e lógicos, localizados dentro ou fora do mundo computacional, através da abordagem de redes semânticas [Power 2003]. Existem propostas de utilização desta técnica para integrar dados contextuais localizados em diferentes fontes [Power 2003].

- **Ontologias:** Como apresentado na seção 3.2, uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada [Borst 1997]. No caso de uma ontologia de contexto, contextos são modelados como conceitos e fatos. Uma ontologia de contexto viabiliza formalização, compreensão e compartilhamento por humanos e computadores. É bastante útil para validar a informação contextual e facilitar a especificação do comportamento de aplicações sensíveis ao contexto, uma vez que permite conhecer os tipos de contexto disponíveis e sua estrutura. Um exemplo de ontologia de contexto para integração de dados geográficos é apresentado por Souza, Salgado e Tedesco [Souza et al. 2006].

Cada técnica de representação possui vantagens e deficiências, como mostra o Quadro 3.3. Com isso, não há uma abordagem que possa ser considerada unanimemente a ideal para todos os sistemas sensíveis ao contexto, uma vez que diferentes sistemas impõem diferentes restrições. A abordagem de mapa de tópicos parece ser bastante promissora, porém ainda está em estágio imaturo em termos de definições de padrões. Strang e Linnhoff-Popien [Strang e Linnholff-Popien 2004] revisaram diversas abordagens para modelagem de contexto e concluíram que ontologias constituem a abordagem mais promissora, devido às suas características de permitir o compartilhamento de conhecimento entre humanos e entre

agentes de software, bem como a reutilização de conhecimento entre aplicações e de permitir o uso de motores de inferência existentes para inferir contextos complexos.

Quadro 3.3: Resumo das Técnicas de Representação de Contexto

Técnica	Vantagens	Desvantagens	Processamento e Recuperação
Par Chave Valor	Estrutura simples, de fácil implementação e uso.	Não considera hierarquia. Inadequado para aplicações com estruturas complexas.	Busca linear com casamento exato de nomes.
Linguagens de Marcação	Baseado em XML. Prevê hierarquia. Esquema de marcação implementa o próprio modelo. Utilização típica em *perfis*.	Incompletude e ambigüidade na informação devem ser tratadas pelo sistema. Inadequado para representar estruturas Complexas.	Linguagem de consulta baseada em marcação
Orientação a Objetos	Encapsulamento, herança, reusabilidade. Permite abstrair o tratamento do contexto	Invisibilidade no tratamento do contexto, pelo encapsulamento, dificulta formalismo do modelo.	Algoritmos e compiladores
Mapas de Tópicos	Facilita a navegação entre os contextos. Facilita modelagem por humanos.	Estágio inicial. Tecnologia imatura. Faltam exemplos reais.	Navegação por redes semânticas.
Ontologias	Contextos modelados como conceitos e fatos. Viabiliza formalização, compreensão e compartilhamento por humanos e computadores.	Tecnologia de manipulação imatura.	Motor de Inferência, Linguagem de consulta baseada em OWL

3.3.3 Contexto na Integração de Dados

De maneira geral, contexto vem sendo utilizado em uma vasta gama de aplicações, mas pouca atenção ainda tem sido dada à sua utilização na área de Banco de Dados [Stefanidis et al. 2005]. Para esta área, Brézillon [Brezillon 1999] comenta que o principal papel do contexto é prover um maior controle sobre o conhecimento presente no banco. Além disso, o contexto permite definir qual conhecimento deve ser utilizado, quais restrições devem ser consideradas nas diversas operações de manipulação, quais as condições que irão ativar as ações e em que momento estas ações devem ser executadas.

Entretanto, na medida em que o volume de informações e a variedade de estruturas de dados empregada crescem ao longo dos últimos anos, o foco da área de Banco de Dados tem

se tornado mais amplo. O contexto pode ser empregado como mecanismo de otimização e enriquecimento semântico em atividades como gerenciamento consistente de dados, processamento de consultas sensíveis ao contexto (*context-aware query processing*) e na integração de informações, sendo estas duas últimas atividades as que compõem nosso foco de pesquisa.

Como abordado no capítulo 2, sistemas de integração de dados buscam fornecer uma interface única para responder consultas que normalmente requerem extração e combinação de dados originários de diversas fontes heterogêneas, distribuídas e autônomas. A maior dificuldade para se obter a integração de dados de diferentes fontes é que suas visões de mundo diferem, ou seja, seus esquemas diferem assim como o significado de seus dados [Farquhar et al. 1995], sendo a integração semântica de dados considerada ainda o maior desafio da integração de dados.

Entretanto, para se obter reconciliação semântica entre dados de fontes heterogêneas, mais informações semânticas sobre os mesmos são necessárias. Metadados são geralmente utilizados para descrever tais dados e uma boa parte deles pode se tornar informação contextual (por exemplo, operadores disponíveis) [Souza et al. 2006]. Tais "metadados contextuais" serão utilizados para facilitar a identificação de correspondências entre esquemas e para a realização do processamento de consultas.

Além do contexto dos dados e das fontes, contextos dos usuários, da consulta e do ambiente de integração de dados devem ser gerenciados. Desta forma, podemos empregar o uso de contexto para facilitar os seguintes processos dentro da integração de dados: (i) na geração de mapeamentos entre esquemas, visto que o contexto ajuda a determinar o significado dos termos, juntamente com uma ontologia de domínio – usada como ontologia de referência; (ii) no processamento de uma consulta, onde resultados se tornam mais relevantes e (iii) na resolução de conflitos específicos, dependendo do domínio da aplicação. Para este último aspecto, podemos citar o caso de aplicações geográficas onde operadores espaciais específicos e/ou o nível de detalhe a ser trabalhado são fatores que somente poderão ser inferidos no momento da formulação da consulta, sendo caracterizados pelo contexto da formulação. A seguir, cada um destes aspectos será explicado mais detalhadamente.

3.3.3.1 Contexto na Identificação de Mapeamentos entre Esquemas

A maior dificuldade para a integração de dados está na identificação de objetos em diferentes fontes que estejam relacionados semanticamente. A proximidade semântica identifica o grau de similaridade semântica entre dois objetos utilizando a semântica do mundo real [Kashyap, Sheth 1996a]. Para serem considerados semanticamente similares, os objetos precisam possuir uma equivalência semântica, uma relação semântica, uma relevância semântica ou uma semelhança semântica. O grau de similaridade é máximo na equivalência semântica e mínimo na semelhança semântica. A associação de contexto representa um primeiro passo para a obtenção da reconcialiação entre as perspectivas semânticas e esquemáticas [Kashyap, Sheth 1996a].

O contexto pode ser usado de diversas maneiras para capturar a semântica relevante. Segundo [Kashyap, Sheth 1996a], o contexto pode ser identificado e representado: como o

relacionamento no qual uma entidade participa; pela associação entre um banco de dados ou um grupo de bancos de dados; como um esquema de exportação ou um esquema externo; ou como uma coleção de domínios de objetos.

Seguindo esse princípio, o contexto é indicado como o principal meio de obter a semântica do mundo real de um objeto (RWS – *Real World Semantic*). O contexto no qual dois objetos estão sendo comparados e suas associações ajudam a capturar a semântica do relacionamento entre eles. Dessa forma, considerando O1 e O2 dois objetos, a proximidade semântica entre eles pode ser vista como a seguinte 4-tupla:

ProxSem(O1,O2) = (contexto,abstração,(D1,D2),(E1,E2))

onde Dx é o domínio de Ox, e Ex é o estado de Ox.

O contexto, neste caso, refere-se ao contexto no qual uma determinada similaridade semântica ocorre. Cada objeto tem seu próprio contexto, e é possível que o grau de similaridade de dois objetos seja maior ou menor, dependendo do contexto. Basicamente, um contexto pode ser visto como uma coleção determinada de domínios dos objetos.

3.3.3.2 Contexto no Processamento de Consultas

Dentro do processamento de consultas, o contexto pode afetar: os resultados retornados por uma dada consulta; a otimização da consulta e a forma como os resultados serão apresentados aos usuários. Assim, de modo geral, o contexto pode ser empregado com o objetivo de prover usuários com resultados mais relevantes de acordo com o contexto da consulta. O trabalho de Bunningen [Bunningen 2004], por exemplo, apresenta uma abordagem onde o contexto é utilizado com o objetivo de otimizar o processamento de consultas de modo que estas retornem resultados mais relevantes e sejam executadas mais rapidamente.

Stefanidis, Pitoura e Zavlavsky [Stefanidis et al. 2005] apresentam um estudo de caso que contém informações sobre restaurantes implementado em um banco de dados relacional. Em seu esquema, tuplas são inseridas com base na seguinte estrutura: *Restaurant(id, type-of-food, address, outdoors, opening-hours, price)*. Parâmetros de contexto considerados são: tempo (*weather*), localização (*location*), hora (*time*) e o perfil do usuário (*user profile*). De acordo com este estudo de caso, algumas considerações são tecidas:

i. Uma mesma consulta pode produzir resultados diferentes dependendo do contexto sobre o qual ela é executada.

ii. O contexto pode também ajudar a recuperar dados com base em históricos de uso

iii. O contexto pode ajudar a construir planos de execução de consultas mais efetivos

iv. Apresentação de resultados de consultas pode se tornar mais completa através do uso de contexto.

Assim, o contexto pode ser inserido na consulta através da definição explícita de parâmetros de contexto, muitas vezes tratados como atributos, como mostra o exemplo abaixo (onde a hora (time) é a hora corrente):

Select Restaurant.id

From restaurants, Context

Where Context.time in Restaurant.opening-hours;

O contexto pode estar também associado a preferências definidas pelo usuário. Tais preferências podem ser definidas segundo duas abordagens [Stefanidis et al. 2005]: quantitativa ou qualitativa. Na primeira, preferências são expressas indiretamente através do uso de funções de *scoring* que associam um valor numérico a cada tupla da resposta da consulta. Por exemplo, a preferência do usuário por restaurantes pode ser expressa como *preference(type-of-food)* com valores *preference(chinese) = 0.2*, *preference(greek) = 0.8* e *preference(other) = 0.1*.

Na abordagem qualitativa, as preferências entre tuplas são definidas diretamente, tipicamente usando relações de preferência binárias. Por exemplo, a seguinte regra pode ser estabelecida:

Se Restaurant1.opening-hours = Restaurant2.opening-hours and Restaurant1.price < Restaurant2.price

Então Restaurante.preferido = Restaurant1.

3.3.3.3 Contexto na Resolução de Conflitos

Quando esquemas de fontes de dados heterogêneas são comparados, os esquemas locais apresentarão conflitos, o que naturalmente surge a partir de representações não compatíveis de entidades do mundo real. Assim, conflitos podem acontecer quando incompatibilidades sintáticas ou esquemáticas existem ou quando dois sistemas não interpretam a informação da mesma forma (conflitos semânticos). Exemplos de conflitos ocorrem quando: nomes diferentes são usados para identificar a mesma entidade ou diferentes entidades têm o mesmo nome; a mesma entidade é representada através de diferentes estruturas de atributos ou alguns atributos podem não ser representados; entidades são representadas por tabelas em um banco de dados e como atributos em outro; tipos diferentes de dados são atribuídos a atributos semanticamente equivalentes; ou ainda conflitos de escalas ocorrem quando unidades métricas diferentes são utilizadas em cada sistema, ou diferentes níveis de precisão são usados [Jakobovits 1997].

Conflitos tanto podem ocorrer durante a integração de esquemas ou identificação de mapeamentos entre esquemas quanto no processamento de consultas, principalmente na etapa de integração dos resultados. Para sua resolução, técnicas baseadas em transformações ou conversões são utilizadas. Por exemplo, renomear atributos e entidades, homogeneizar representações, converter tipos ou efetuar junções são técnicas que podem ser empregadas. Funções e serviços podem ser desenvolvidos para executarem tais operações.

Para otimizar a resolução de conflitos, podemos levar em consideração o contexto do dado. “Metadados contextuais” (relativos ao significado, às propriedades e à organização da informação) podem conter informações variadas, como especificações de medidas (moeda corrente, unidade métrica), de formato de dados (formato de datas, fator de escala), informações sobre qualidade (confiabilidade, fonte) e adoções utilizadas na obtenção ou cálculo de dados, em métodos ou na definição de fórmulas.

Como exemplo, podemos considerar o valor de um salário, que pode estar representado pelo número 600. O contexto do salário irá conter informações como moeda, precisão, escala, estado, entre outras [Amaral 2005]. Desta forma, quando o integrador de dados do sistema for integrar os resultados das subconsultas submetidas às fontes, ele irá verificar os metadados de cada resultado e, se necessário, converter o valor dos dados para o contexto definido de acordo com a consulta ou com o sistema.

Em se tratando de aplicações geográficas, outro caso interessante, fatores relacionados à escala corrente de manipulação dos dados, à disponibilidade de operadores espaciais específicos, à questão de representações múltiplas e à existência de sistemas de coordenadas diferentes são exemplos de informações contextuais que implicarão em diferentes resultados de consultas, de acordo com o tratamento fornecido a eles. Neste caso, o resultado da consulta pode realmente variar em diferentes situações, dependendo destes fatores percebidos ou inferidos em tempo de execução da consulta pelo sistema.

3.3.3.4 Um Exemplo de Aplicação - COntext INterchange Project (COIN)

Quando vários usuários realizam consultas, e estas podem ser respondidas a partir de múltiplas fontes, isso significa que dados semelhantes ou complementares podem estar associados a diferentes possíveis interpretações, ou seja, a diferentes contextos. O COntext INterchange Project - COIN permite a integração de diversas fontes de dados através de uma arquitetura de mediadores e, para tal, faz uso do conceito de contexto.

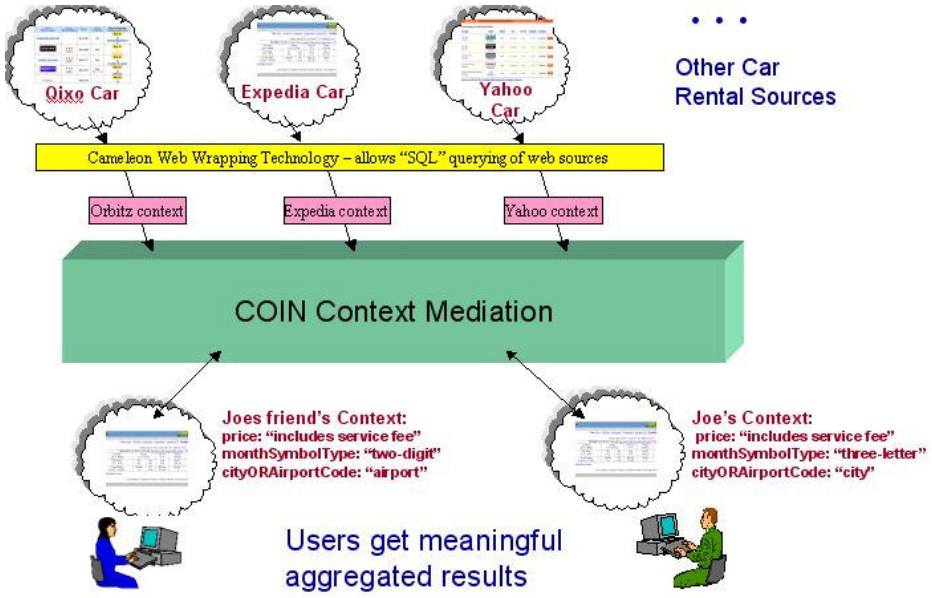

Figura 3.8 Sistema COIN

Para Firat [Firat 2003], os desafios a serem trilhados para a obtenção da integração de dados se encontram relacionados à extração dos dados a partir das fontes e da interpretação desses mesmos dados de acordo com o contexto associado. O projeto COIN [Firat 2003] define contexto como um conjunto de assertivas que alguém faz na tentativa de interpretar ou

entender os dados e classifica a heterogeneidade semântica como contextual, ontológica ou temporal. A primeira acontece quando duas ou mais representações existem para a mesma entidade do mundo real. A heterogeneidade ontológica diz respeito às diferenças esquemáticas entre as fontes e a temporal é ortogonal às duas anteriores e ocorre devido às mudanças tanto no nível dos dados quanto no nível do esquema.

Conforme a Figura 3.8, o sistema COIN permite que usuários realizem consultas sem se preocupar sobre quais contextos existem. O sistema automaticamente aplica o contexto relacionado àqueles dados a partir de um serviço chamado de mediador de contexto. A estratégia do COIN está baseada então na detecção e reconciliação de conflitos semânticos a partir do mediador de contexto. Para que este serviço seja realizado eficientemente, é necessário que cada fonte proveja a especificação lógica de como seus dados são interpretados e seus contextos associados. Além disso, devem especificar como estes conflitos devem ser resolvidos no momento em que surgirem. Vale salientar que os conflitos entre as fontes não são especificados a priori, mas dinamicamente quando acontecem [Firat 2003].

COIN utiliza ontologias, chamadas de modelos de domínio da aplicação, dentro de uma abordagem de ontologias combinadas, ou seja, a semântica de cada fonte é descrita por sua própria ontologia mas, para que estas sejam comparáveis, cada uma é organizada de acordo com um vocabulário comum compartilhado que é descrito como uma outra ontologia. Esta provê as funções de conversão a serem usadas quando os conflitos semânticos ocorrerem.

3.3.3.5 Vantagens

Como já comentado, a utilização de contexto pode facilitar a integração de dados, provendo um caráter incremental a este processo. O trabalho de Farquhar [Farquhar et al. 1995] apresenta a utilização de contexto como forma de representar o contexto semântico das fontes e gerar um contexto de integração a partir destes. Para tal, três tipos de contextos são definidos: (i) o contexto da fonte (*information source context*) - tradução direta do esquema do banco de dados (relacional) para os axiomas escritos em lógica de primeira ordem; o contexto semântico (*semantic context*) - busca resolver os conflitos semânticos através de técnicas de homogeneização; e o contexto de integração (*integrating context*) que contém axiomas que organizam sentenças dos diversos contextos semânticos. Metadados das fontes são traduzidos para assertivas em lógica de primeira ordem, ou seja, sentenças que são verdadeiras dentro de um contexto (no formato ist()), como por exemplo: ist (SC1, natural-size-units(x) = inch <=. product-type(x, television)), ou seja, se o contexto for televisão, a unidade será “inch”.

Dessa maneira, o contexto de cada fonte é definido e um contexto de integração é utilizado como um esquema global. O conceito de contexto também é empregado na definição dos requisitos dos usuários. A grande vantagem do trabalho de Farquhar é que novos axiomas podem ser inseridos sempre que novas fontes sejam acrescentadas ao sistema, assim como uma consulta realizada sobre o mesmo sistema poderá gerar resultados oriundos de diversos contextos (na verdade, de diversas fontes).

Algumas vantagens em utilizar contexto na integração de dados são [Kashyap, Sheth 1996a]: economia de representação, pois o contexto se torna um meio de focar num determinado objetivo; economia de raciocínio (inferência a partir do contexto); gerenciamento de informação inconsistente (a informação deve ser consistente dentro de um contexto); semântica flexível (dois objetos podem estar relacionados diferentemente em dois contextos distintos).

3.4 Considerações

Este capítulo apresentou aspectos semânticos que podem ser empregados na integração de dados com o objetivo de enriquecer representações e processos. O próprio metadado é empregado como forma de enriquecer a representação de dados (das fontes). Ontologias, por sua vez, permitem a uniformização de conceitos dentro de uma terminologia comum, ou seja, podem também ser empregadas na representação de metadados das fontes, criando um modelo comum.

Um exemplo de processo em sistemas de integração de dados que pode ser otimizado com aspectos semânticos é o processamento de consultas, desde a submissão da mesma até a etapa de integração de resultados. Neste caso, informações contextuais podem ser capturadas e/ou inferidas de acordo com o contexto do usuário, o contexto da consulta e o contexto do ambiente. Ontologias também podem ser usadas para prover vocabulário comum através de ontologias de domínio de referência. Na maioria dos domínios de conhecimento atuais, já é possível reutilizar ontologias padrões prontas, evitando retrabalho e permitindo homogeneização de termos. Ontologias também podem ser usadas para permitir inferência. Da mesma maneira, informações contextuais podem ser percebidas ou inferidas com o propósito de tornar os sistemas de integração mais adaptáveis e semanticamente mais ricos.

Capítulo 4

Sistemas P2P e os Gerenciadores de Dados

Um sistema *Peer-to-Peer* (P2P) é um ambiente de computação distribuída sem controle centralizado onde o software que é executado em cada ponto (*peer*) é equivalente em funcionalidade e tanto atua como cliente quanto servidor. Embora muito difundidos e úteis, sistemas P2P existentes funcionam geralmente para casos simples de compartilhamento de arquivos, como, por exemplo, para a troca de arquivos de música. Entretanto, buscar por "todas as músicas recentes de Madonna" não requer linguagens que apóiem buscas ou metadados complexos, sendo os formatos de busca existentes, assim, suficientes. Contudo, em outros cenários, como a troca de recursos específicos de áreas como educação, saúde, economia, entre outras, as pesquisas são bem mais complexas e necessitam de padrões para a descrição e busca dos metadados.

Se pensarmos em um cenário onde cientistas de instituições diferentes desejam cooperar e compartilhar suas pesquisas e experimentos, percebemos que dados mais complexos serão manipulados, assim como metadados serão necessários para prover seu entendimento. Dentro desta perspectiva, inicialmente desenvolvidos para permitir compartilhamento de dados não estruturados, sistemas P2P vêm evoluindo para sistemas de gerenciamento de dados estruturados através dos chamados *Peer Data Management Systems* ou PDMS. Um PDMS é um sistema de gerenciamento de dados que utiliza uma arquitetura P2P e busca aliar os benefícios de sistemas P2P, como a ausência de autoridade central, com o poder semântico dos bancos de dados [Zhao 2006]. Deste modo, a idéia é que cada ponto possa atuar como cliente realizando consultas e, ao mesmo tempo, como servidor, provendo resultados às consultas formuladas.

O objetivo deste capítulo é apresentar os sistemas P2P, suas características, arquiteturas e mecanismos de roteamento. Além disso, dentro deste contexto, apresentar os PDMS como arquiteturas distribuídas P2P propícias à integração de dados. Exemplos de PDMS serão também descritos, assim como um quadro comparativo será gerado como resultado desta avaliação.

4.1 Sistemas P2P

A tecnologia P2P não é de todo nova [Vallejos 2005]. Suas características básicas já podiam ser anteriormente encontradas em aplicações como a Usenet ou em sistemas DNS (*Domain*

Name System), ambas redes completamente descentralizadas e com alta escalabilidade. A Usenet, por exemplo, foi criada em 1979, como uma aplicação distribuída que provia *newsgroups*, onde arquivos eram trocados em batch sobre linhas telefônicas, sem nenhum tipo de centralização. Já os sistemas DNS surgiram por volta de 1983 como outra forma de prover compartilhamento de arquivos. São sistemas que conseguem aliar o conceito de uma aplicação P2P com um modelo hierárquico de informação.

Em sistemas DNS, os *namespaces* dos nomes DNS são construídos hierarquicamente [Vallejos 2005]. Esta hierarquia produz uma maneira simples de delegar responsabilidade para o banco de dados DNS que atua como servidor. Cada domínio possui um servidor de nomes (conhecido como *authority* ou autoridade) de hosts daquele domínio. Na busca por um endereço de um dado nome, o servidor pode consultar seu vizinho mais próximo ou delegar a consulta ao *authority*. Sucessivamente, a consulta pode ser delegada a uma autoridade sempre mais alta. Quando a resposta é obtida, ela é propagada de volta ao solicitante. Percebe-se então que servidores de nomes atuam tanto como servidores quanto como clientes e que existe um método natural de propagação de consultas através da rede. Ambas características são encontradas em ambientes P2P.

Apesar desta utilização, foi o surgimento da internet em 1994 que impulsionou a aplicação de tecnologias P2P. Inicialmente, nos anos 60 e 70, ainda na época da ARPANET, o modelo de prestação de serviços empregado na rede não estabelecia restrições entre as máquinas, ou seja, funcionava como um ambiente P2P [Kamienski et al. 2005]. Entretanto, aos poucos, a Internet foi perdendo essa característica, havendo a necessidade de especialização de funções, robustez e alta disponibilidade e passou a utilizar o modelo cliente-servidor como base. Assim, a *World Wide Web* foi concebida sobre uma arquitetura cliente-servidor, onde o cliente se conecta a um servidor conhecido, "baixa" o que lhe interessa e depois se desconecta [Vallejos 2005]. Neste tipo de operação, o cliente Web não necessita ter um endereço IP permanente, nem mesmo uma conexão contínua com a internet.

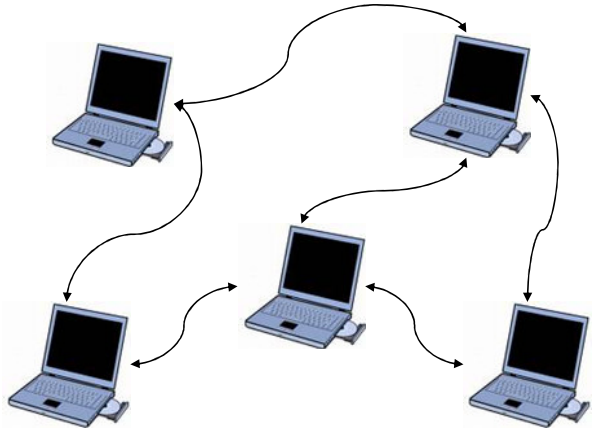

Fig. 4.1 Um Ambiente P2P

Neste sentido, a tecnologia P2P amadureceu com o intuito de mudar este cenário, pois não estabelece a necessidade de uma organização central ou hierárquica e dispõe aos seus

integrantes as mesmas capacidades e responsabilidades [Parameswaran et al. 2001]. Comparando a uma arquitetura cliente-servidor, onde o número de servidores é fixo, sistemas P2P são mais flexíveis e extensíveis, visto que, quando novos pontos entram na rede, a capacidade total do sistema aumenta, enquanto que nos ambientes clientes-servidores, a adição de novos clientes reduz o desempenho do mesmo [Zhao 2006]. Desta maneira, sistemas P2P surgiram para permitir que usuários interajam diretamente uns com os outros sem a intervenção de um servidor [Vallejos 2005], conforme ilustrado na Figura 4.1 [Brito 2005].

Sistemas P2P vêm se consagrando em diversas classes de aplicações: computação distribuída, troca de mensagens, trabalho colaborativo e compartilhamento de dados, principalmente, arquivos. Em especial, sistemas que permitem a troca de arquivos, como o Napster [Napster 2007], Gnutella [Gnutella 2007] e o Kazaa [Kazaa 2007] tornaram o paradigma P2P popular e difundido entre um conjunto bastante amplo de usuários. O Napster, por exemplo, logo no início de seu funcionamento conseguiu a marca de mais de 25 milhões de usuários.

Percebe-se que a idéia básica destes sistemas é garantir uma simplicidade de uso e de compartilhamento (arquivos e programas), de maneira justa: cada ponto, para poder acessar outros arquivos, deve primeiramente disponibilizar os seus; permitir a comunicação direta com outras pessoas por meio da internet, sem a necessidade de suporte de um servidor centralizado; e prover maior escalabilidade, tanto a nível do número de usuários quanto de poder de processamento e de recursos. Esta idéia parece casar exatamente com os interesses dos usuários: o uso mais natural e intuitivo da internet, melhor desempenho, maior disponibilidade, mecanismos de busca mais eficientes (resultados mais precisos num espaço maior de busca), portabilidade e ausência de um moderador de publicações de conteúdo.

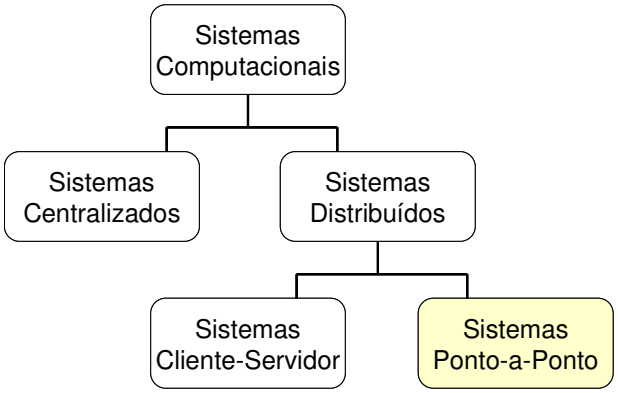

Figura 4.2 Classificação dos Sistemas Computacionais

Desta maneira, sistemas P2P são classificados como sistemas distribuídos, conforme a Figura 4.2. Vantagens obtidas com sua utilização incluem robustez em falhas, extensivo compartilhamento de recursos, auto-organização, balanceamento de carga, persistência de dados, anonimato, entre outras. Inúmeras definições sobre sistemas P2P vêm sendo estabelecidas, mas buscando uma condensação, apresentamos uma delas: “Sistemas e

aplicações P2P são sistemas distribuídos sem qualquer forma de controle centralizado ou hierarquia organizacional, onde o software que está sendo executado em cada nó é equivalente em funcionalidade." [Stoica et al. 2001]. No nosso enfoque, sistemas P2P consistem de uma ampla rede de pontos computacionais (*peers* ou nós de informação) interconectados que cooperam uns com os outros, trocando serviços e informações.

Em recente levantamento, Androutsellis-Theotokis e Spinellis [2004] apontam as características básicas de um sistema P2P:

- Pontos se conectam diretamente com outros pontos;
- Pontos são responsáveis por seus próprios dados;
- Pontos podem entrar e sair da rede a qualquer momento;
- Pontos podem atuar tanto como clientes quanto como servidores;
- Pontos são autônomos com respeito ao controle e estruturação da rede, ou seja, não existe autoridade central.

A título de ilustração, apresentamos abaixo o sistema Gnutella, um dos sistemas de maior sucesso na internet para troca de arquivos, através de um ambiente P2P.

O Gnutella é uma rede de compartilhamento de arquivos usada principalmente para a troca de músicas, filmes e softwares [Gnutella 2007]. É uma verdadeira rede P2P, pois trabalha realmente sem um servidor central que armazene informações sobre os arquivos que estão disponíveis na rede. Em vez disso, todas as máquinas da rede informam sobre os seus arquivos disponíveis e utilizam uma abordagem de busca distribuída para recuperá-los [Kamienski et al. 2005]. Desta forma, os arquivos são trocados diretamente entre os usuários.

No Gnutella, os pontos estão conectados via TCP/IP e executam um software que implementa um protocolo Gnutella. Para se conectar à rede, um ponto precisa saber previamente um endereço de um outro ponto que já participa da rede. Um dos grandes benefícios do Gnutella ser descentralizado é que se torna muito difícil a rede cair. Por isso, e de acordo com o site de compartilhamento de arquivos Sklick.com, a Gnutella é a segunda rede de troca de arquivos mais popular da Internet, perdendo apenas para a eDonkey 2000 [Wikipedia].

Nas subseções seguintes, os sistemas P2P serão classificados de acordo com sua arquitetura e mecanismos de roteamento comuns a estes sistemas serão mostrados.

4.1.1 Arquiteturas

De acordo com a literatura [Sung et al. 2005, Kamienski et al.], existem diversas categorias de arquiteturas de sistemas P2P. Uma das mais utilizadas, classifica os sistemas em [Lv et al. 2002]:

- Centralizados: sistemas que mantêm um índice central com informações atualizadas. Quando o usuário se conecta à rede ele, de fato, está se conectando a um ou mais servidores que atuam como um índice para busca dos dados disponíveis, assim o servidor que o recebeu irá atualizar seu índice de forma que os demais pontos possam

tomar conhecimento de sua entrada. Sistemas de compartilhamento de arquivos como o Napster [Napster 2007] e de troca de mensagens utilizam esta arquitetura.

- Descentralizados e Estruturados: não possuem um servidor centralizado de diretório de informações, mas têm uma estruturação significativa entre os nós. A topologia da rede é controlada e os documentos/arquivos são posicionados em locais que posteriormente tornam fácil a sua localização. Esse tipo de arquitetura em geral utiliza busca baseada em DHT (a ser explicada na seção 2.2). Essa arquitetura é utilizada pelos sistemas Chord [Stoica et al. 2001], Pastry [Rowstron, Druschel 2001] e CAN [Ratnasamy et al. 2001].

- Descentralizados e não Estruturados: não possuem servidor centralizado, nem controle preciso sobre a topologia e localização/busca de documentos. Exemplos da utilização desta arquitetura são as redes Gnutella [Gnutella 2007] e KaZaA [Kazaa 2007].

Uma outra classificação [Towsley 2003], baseada no tipo de busca provida pelo sistema, categoriza os sistemas P2P em três tipos:

- Busca centralizada: rede com um ponto central de busca e nós que consultam o ponto central para trocar informações diretamente entre os pontos; semelhante ao tipo centralizado da classificação anterior.

- Busca por inundação: rede com nós totalmente independentes, onde não existe um ponto central de falha. A maneira pela qual os nós se ligam à rede não é estruturada. Um nó que deseja entrar na rede simplesmente se conecta com outros nós já presentes na rede. As buscas, neste tipo de sistema, são feitas por mecanismos de inundação controlada por TTL (*time-to-live*) (a serem descritos na seção 4.2.2).

- Busca por tabela *hash* distribuída (DHT): rede onde os nós têm autonomia e usam uma tabela *hash* para separar o espaço de busca entre eles.

Fiorano [Fiorano 2003] compara e avalia diferentes topologias com respeito a aspectos como escalabilidade, desempenho, confiabilidade, entre outros, com o objetivo de determinar a arquitetura mais adequada para problemas que necessitam de computação distribuída. Em seu trabalho, ele apresenta três categorias de arquiteturas para sistemas P2P [Fiorano 2003]:

- Pura: nesta arquitetura inexiste um ponto central de controle, os pontos são autônomos e completamente interligados, podendo assim se comunicar diretamente uns com os outros, conforme a Figura 4.3. A maior vantagem deste tipo de arquitetura é a sua escalabilidade, pois qualquer nó pode entrar na rede e iniciar a troca de dados com outro nó. Sistemas puros tendem a ser mais tolerantes a falhas, visto que uma falha ou queda em um nó não afeta o restante do sistema. Um dos maiores representantes desse tipo de arquitetura é o sistema Gnutella [Gnutella 2007].

- Híbrida: em sistemas híbridos, a informação de controle é trocada com o servidor central, enquanto dados fluem entre os nós como na arquitetura pura. O servidor central atua como um agente de monitoração para todos os outros pontos e garante coerência das informações. Sistemas híbridos têm sido utilizados principalmente em aplicações críticas, geralmente limitadas a um pequeno conjunto de problemas

[Fiorano 2003]. Um exemplo de sistema que adota este tipo de arquitetura é o Groove [ ] – um sistema de gerenciamento de projetos colaborativos onde um servidor central controla toda a informação que está sendo compartilhada entre os pontos.

- Super-Ponto (*Super Peer*): neste tipo de arquitetura, existe a noção de Super-Ponto (*Super Peer*) - um ponto global que mantém um certo controle sobre os pontos que estão ligados a ele . Em outras palavras, são criadas sub-redes P2P e, para um ponto de uma sub-rede se comunicar com o ponto de outra, deve haver comunicação entre os Super-Pontos responsáveis por cada sub-rede. Em sistemas com Super-Ponto, a busca se torna bem mais rápida, se comparada com outras topologias, visto que o sistema é dividido em agrupamentos ou clusters cada qual com informações indexadas sobre seus pontos. Segundo Fiorano, uma busca que leva O(N) numa rede pura ou híbrida, irá levar O(N/M) (onde M é o número médio de pontos conectados a um super-ponto) numa arquitetura de super-pontos. Um exemplo de sistema que utiliza esta arquitetura é o Kazaa [Kazaa 2007].

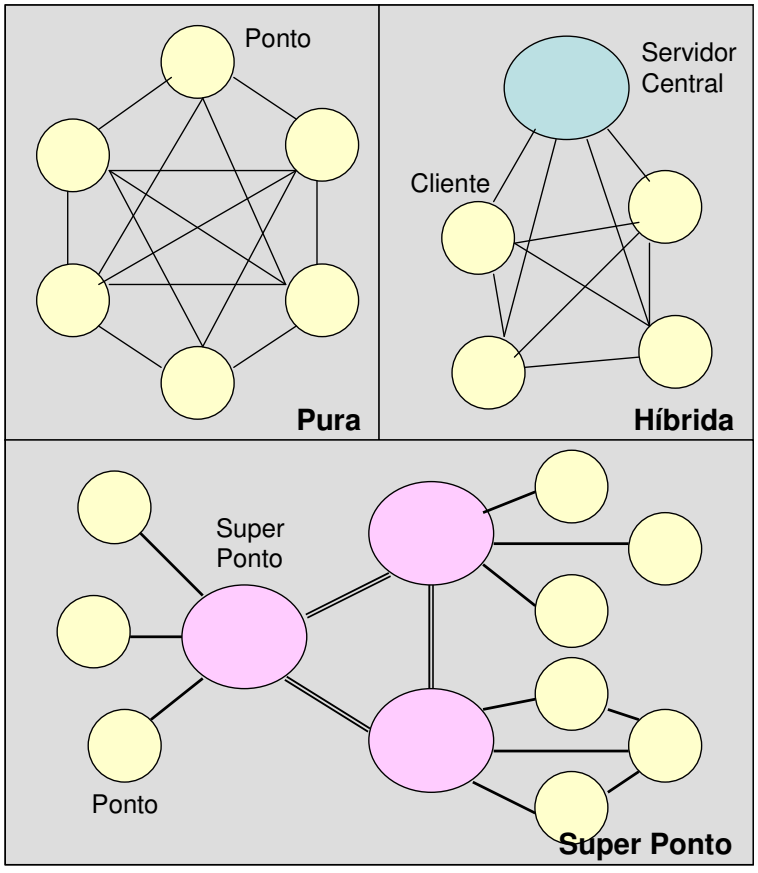

Fig. 4.3 Categorias de Arquiteturas [Fiorano 2003]

Embora clusters de super-pontos sejam considerados eficientes, escaláveis e gerenciáveis, um super-ponto pode se tornar um ponto de falha para seus pontos clientes. Este problema pode ser resolvido através do conceito de redundância de super-ponto (*super peer redundancy*), onde pontos reservas são definidos para assumirem a posição de super-ponto,

caso este tenha algum tipo de falha. Assim, uma arquitetura de super-ponto redundante que combina vantagens tanto de sistemas centralizados quanto de sistemas descentralizados é considerada a mais adequada para sistemas que precisem de computação distribuída [Fiorano 2003].

4.1.2 Roteamento

Um dos maiores objetivos de um sistema P2P é prover a busca por dados, onde buscar dados significa geralmente encontrar arquivos ou partes deles [Sung et al. 2005], através de consultas, geralmente por palavras-chave. Para encontrar os pontos que podem responder à consulta, mecanismos de roteamento (índices) são necessários e suas implementações podem variar de acordo com a categoria do sistema, como visto na seção anterior.

Algoritmos de busca e roteamento procuram otimizar o encaminhamento de uma mensagem de um ponto a outro. Algumas estratégias estão direcionadas a sistemas não estruturados, outras a sistemas estruturados, mas são geralmente baseadas em três modelos: centralizado, inundação e tabela *hash* distribuída (DHT).

Modelo Centralizado

Caracteriza-se pela conexão dos pontos a um diretório central onde são publicadas informações sobre o conteúdo que os pontos oferecem para compartilhamento. Deste modo, ao receber uma requisição, o índice (diretório) central escolhe o ponto no diretório que for mais adequado. Esse ponto pode ser o mais rápido e disponível, dependendo das necessidades do usuário [Kamienski et al. 2005]. Então, a troca de arquivos será efetuada diretamente entre os dois pontos.

Modelo de Inundação

Neste modelo, as buscas são feitas por mecanismos de inundação (*flooding*) controlada por um "tempo" - TTL (*time-to-live*). Assim, um nó que deseja buscar dados manda uma mensagem para todos os nós vizinhos a ele. Cada um de seus vizinhos avalia a mensagem e verifica se tem o dado procurado. Caso tenha, ele envia uma mensagem de volta ao nó de onde se originou a pesquisa informando o seu endereço IP. Caso não possua, ele incrementa o número de saltos da mensagem (*hop count*) e a repassa para todos os seus vizinhos. Quando o número de saltos ultrapassar o TTL definido, o encaminhamento de mensagens é interrompido. Logo, o TTL define o raio de ação da busca.

Modelo DHT

O modelo de tabelas *hash* distribuídas (DHT) é o mais recente. Neste tipo de modelo, um ID randômico é associado a cada ponto da rede que conhece um determinado número de pontos. Quando um documento é publicado num sistema P2P, um ID é associado ao documento baseado em uma *hash* de conteúdo dos documentos e no seu nome. Cada ponto então repassa o documento ao ponto cujo ID é mais próximo do ID do documento. Esse processo é repetido até que o ID do ponto atual seja o mais próximo do ID do documento.

Cada operação de roteamento também garante que uma cópia local do documento seja mantida. Quando um ponto solicita o documento de um sistema P2P, a requisição irá até o ponto com ID mais semelhante ao ID do documento. Esse processo continua até que uma

cópia do documento seja encontrada. Então, o documento é transferido ao ponto que originou a requisição, enquanto cada ponto que participou do roteamento permanecerá com uma cópia local do documento [Stoica et al. 2001]. Existem três algoritmos principais que implementam o modelo de DHT: Chord [Stoica et al. 2001], CAN [Ratnasamy et al. 2001] e Pastry [Rowstron et al. 2001].

Ao se realizar uma avaliação entre os modelos, observam-se vantagens e desvantagens em cada um. No caso do modelo centralizado, uma das grandes vantagens é a garantia de que se o dado estará disponível em algum dos pontos da rede, então ele será encontrado. Além desta, buscas por palavras-chave e múltiplos critérios são facilmente implementadas. A desvantagem, no entanto, é a alta susceptibilidade a falhas, devido aos pontos centrais que existem.

No modelo de inundação, a rede não se torna escalável porque é necessário um grande poder de processamento e largura de banda. Além disto, não há garantia de que os dados necessários serão acessados, pois o ponto que contém o dado pode estar mais longe que o TTL definido. Ao mesmo tempo, este modelo é o que mais reflete as propriedades de um sistema P2P.

No caso do modelo DHT, ele se mostra eficiente para comunidades grandes e globais, mas ao mesmo tempo ele apresenta um problema relacionado ao ID do documento. Este precisa ser conhecido antes que uma requisição do documento seja realizada. Assim, é mais difícil implementar uma busca nesse modelo que no modelo de inundação. Além disso, pode ocorrer a formação de "ilhas", onde a comunidade se divide em sub-comunidades que não possuem nenhuma ligação (*links*) entre si [Kamienski et al. 2005].

4.2 PDMS

Sistemas P2P já são uma realidade entre usuários da internet. Entretanto, se pensarmos em dados mais estruturados, percebemos uma lacuna relacionada ao gerenciamento de dados em ambientes P2P. A tecnologia de banco de dados evoluiu bastante ao longo dos últimos anos, saindo de arquiteturas centralizadas, para outras distribuídas, federadas, com esquemas globais únicos e para sistemas de integração de dados que fornecem transparência total de uso para seus usuários. Assim, é muito natural pensar em aliar as vantagens obtidas com a evolução das tecnologias de banco de dados com as vantagens dos sistemas P2P. Dessa união, surgiram os sistemas PDMS ou *Peer Data Management Systems*. A idéia é que usuários se beneficiem de toda a estrutura provida por sistemas P2P ao mesmo tempo em que obtenham as facilidades oriundas de SGBDs como linguagens de consulta apropriadas, modelos semânticos de suporte e facilidades de uso.

Como um típico sistema P2P, um PDMS ou Sistema Gerenciador de Dados em ambientes P2P herda todas as suas características, dentre elas o fato de cada ponto poder entrar e sair da rede a qualquer momento e a autonomia que cada um tem. Do ponto de vista de gerenciamento de dados, um PDMS deve lidar com questões como [Sung et al. 2005]:

- Localização dos Dados: pontos devem ser capazes de referenciar e localizar dados armazenados em outros pontos.

- Processamento de Consultas: dada uma consulta, o sistema deve ser capaz de descobrir os pontos que podem contribuir com dados relevantes à execução da consulta e processá-la eficientemente.
- Integração dos Dados: mesmo quando fontes de dados compartilhadas no sistema seguem diferentes esquemas ou representações, o sistema deve ainda ser capaz de acessar os dados, integrá-los e retornar os resultados.
- Consistência dos Dados: a consistência deve ser mantida em caso de replicação e uso de cache.

O princípio fundamental por trás do gerenciamento de dados é a independência dos mesmos que habilita usuários e aplicações a compartilharem dados num nível conceitual, abstraindo detalhes de implementação [Valduriez, Pacitti 2004]. PDMS são considerados uma extensão natural dos sistemas de integração de dados [Heese et al. 2005], que, por sua vez, já são uma extensão dos sistemas distribuídos.

SGBD distribuídos constituem uma tecnologia já utilizada, e os sistemas de integração de dados vêm sendo pesquisados ao longo dos últimos anos. PDMS, por sua vez, são tópicos de pesquisa recentes que carecem ainda de consolidação em diversos pontos. Fazendo uma pequena comparação técnica entre as três tecnologias, podemos apontar diferenças e semelhanças, como apresentado no Quadro 4.1, a partir da adaptação de [Ng et al. 2003].

Quadro 4.1: Comparação entre Sistemas Distribuídos, de Integração e PDMS

Tópico	SGBD Distribuído	Sistema de Integração de Dados	PDMS
Entrada e Saída de Nós	De forma controlada	De forma controlada	Podem entrar e sair a qualquer momento
Esquema Global	Existe e é compartilhado	Existe e é compartilhado	Não existe esquema único
Completude das Respostas	Dados nos nós são completos, por conseqüência, respostas às consultas são consideradas completas.	Dados nos nós são completos, mas fontes podem estar indisponíveis, o que pode produzir resultados incompletos.	Pontos podem estar desconectados ou não possuir todos os dados num dado momento, portanto uma consulta pode ter um resultado incompleto de dados.
Roteamento de Consultas	Consultas são roteadas para um conjunto pequeno de nós. A localização de cada nó é conhecida.	Consultas são roteadas para um conjunto de nós que pode não ser pequeno. A localização de cada nó deve ser conhecida.	Consultas podem ser roteadas para um número grande de pontos, geralmente a partir do "word-of-mouth", ou seja de vizinho para vizinho. A localização de cada ponto não é previamente conhecida.

Diante deste quadro e analisando as características dos PDMS, percebe-se que os pontos são altamente autônomos, em termos dos dados que eles armazenam, na forma como eles os descrevem (em seus esquemas) e na escolha dos pontos com os quais querem se comunicar [Roshelova 2004]. Em um PDMS, cada ponto está associado a um esquema que representa o seu domínio de interesse, e as relações semânticas entre os pontos são organizadas entre pares ou conjuntos pequenos de pontos. Percorrendo caminhos obtidos através dos mapeamentos, as consultas submetidas em um ponto podem obter dados relevantes e complementares de qualquer outro ponto da rede [Tatarinov, Halevy 2004].

As vantagens na utilização dos PDMS em detrimento dos sistemas distribuídos e/ou de integração são: a não existência de um único ponto de falha; a necessidade mínima de administração (a princípio, nenhuma); a larga quantidade de dados que é manipulada e usada para produzir resultados de consultas; dados podem ser replicados para se obter recuperação mais rápida, assim como resultados de consultas podem ser mantidos em cache local e reutilizados em consultas posteriores.

Ao se projetar um PDMS, alguns requisitos fundamentais são [Valduriez e Pacitti 2004]:

- Autonomia: um ponto autônomo deve ser capaz de entrar e sair do sistema a qualquer momento sem restrição. Deve também ser capaz de controlar os dados que armazena e quais outros pontos (confiáveis) podem armazená-los.
- Expressividade da consulta: a linguagem de consulta deve permitir ao usuário descrever os dados desejados num nível adequado de detalhes. A forma mais simples de consulta é busca por palavras, mas esta abordagem é adequada apenas para arquivos. Para dados mais estruturados, uma linguagem de consulta SQL-like é necessária.
- Eficiência: o uso eficiente de recursos de sistema P2P (largura de banda, poder computacional, armazenamento) deve resultar num custo menor, permitindo um maior processamento de consultas num determinado período de tempo (*throughput*).
- Qualidade do serviço: diz respeito à eficiência percebida pelo usuário, por exemplo, completude dos resultados de consultas, consistência dos dados, disponibilidade dos dados, tempo de resposta da consulta, entre outros.
- Tolerância a falhas: eficiência e qualidade do serviço devem ser providos mesmo na ocorrência de falhas. Dada a natureza dinâmica dos pontos que entram e saem a qualquer momento, uma solução viável é trabalhar com replicação de dados.
- Segurança: a natureza aberta de um sistema P2P torna o fator segurança um grande desafio, visto que muitos servidores podem não ser confiáveis. Para isso, é importante tratar questões como controle de acesso e reforçar questões relacionadas à propriedade intelectual.

Valduriez e Pacitti [Valduriez e Pacitti 2004] classificam os PDMS em não estruturados, estruturados e super-ponto. Em sistemas não estruturados, cada ponto pode se

comunicar diretamente com seus vizinhos. Sistemas estruturados são aqueles baseados em DHT e sistemas que fazem uso de super-pontos são aqueles onde alguns pontos podem realizar tarefas mais complexas como indexação, processamento de consultas e gerenciamento de metadados. Com base nesta classificação, os requisitos abordados acima são analisados para cada uma das categorias, como mostra o Quadro 4.2. Esta comparação gerada pode ajudar a analisar qual tipo de arquitetura pode ser mais indicada, de acordo com os critérios estudados.

Quadro 4.2: Requisitos de PDMS

Requisito	Não Estruturado	Estruturado	Super-Ponto
Autonomia	Alta	Baixa	Em média
Expressividade da Consulta	Alta	Baixa	Alta
Eficiência	Baixa	Alta	Alta
QoS	Baixa	Alta	Alta
Tolerância a Falhas	Alta	Alta	Baixa
Segurança	Baixa	Baixa	Alta

É grande o número de PDMS existentes, cada um com características e objetivos específicos. A seção seguinte apresenta exemplos de PDMS, focalizando principalmente em sua arquitetura, aspectos relacionados ao gerenciamento de dados e ao processamento de consultas. Ao término, um quadro comparativo será gerado com o objetivo de sumarizar as características analisadas.

4.2.1 PeerDB

O sistema PeerDB é um sistema que utiliza estratégias para suportar o compartilhamento de dados relacionais em ambiente P2P. Teve início a partir de um outro projeto conhecido como BestPeer [Ooi, Shu e Tan, 2003]. O BestPeer foi iniciado em 2000, na Universidade de Singapura, com o objetivo de estudar como tecnologias P2P poderiam ser empregadas para o gerenciamento de dados distribuídos.

Na plataforma BestPeer, a rede consiste de 2 tipos de entidades: um grande número de pontos; e um número menor de LIGLO servers (*Location Independent Global Names Lookup*) que provêem cada nó com um ID global único e mantêm o status de cada ponto (*on* ou *offline*). Construído no topo do BestPeer, o PeerDB é um sistema de gerenciamento de dados que visa integrar dados na sua forma relacional, possuindo forte dependência e interatividade com os usuários na submissão de consultas. Suas características principais são:

- Cada ponto é um sistema de gerenciamento de dados;
- Usuários podem compartilhar dados sem um esquema global compartilhado;
- Processamento de consultas é assistido por agentes;
- Pontos podem se "auto" reconfigurar e definir seus vizinhos; e

- Pontos que se habilitam a responder uma consulta são cotados a responder as consultas subseqüentes.

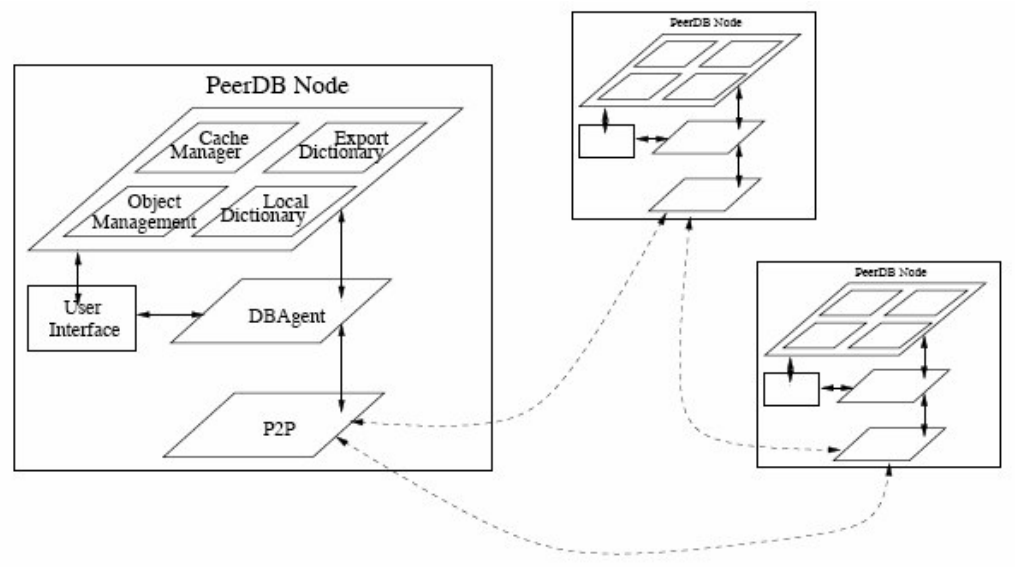

Fig. 4.4 Arquitetura do PeerDB

Sua arquitetura é composta de três camadas, conforme mostra a Figura 4.4:

- P2P *Layer*: provê capacidades P2P (por exemplo, facilita troca de dados e descoberta de recursos)
- *Agent Layer*: explora e mantém agentes que assistem o processamento das consultas. Assim, um agente denominado *Master Agent* gerencia as consultas dos usuários em cada ponto e é responsável por clonar e despachar *Worker Agents* para os pontos vizinhos, receber as respostas e apresentá-las ao usuário. Além disso, o *Master Agent* também monitora as estatísticas e gerencia políticas de reconfiguração. Nesta camada, existe também a interface de usuário – um ambiente amigável para usuários submeterem suas consultas (*SQL- like*), manter suas tabelas compartilhadas e manipular seus dados.
- *Data Management Layer*: provê capacidades de armazenamento e processamento dos dados. É formada por quatro componentes que são fracamente integrados. O primeiro deles é o Dicionário Local (*Local Dictionary*) que contém os metadados associados (esquema, palavras-chave) às relações locais. O segundo componente é o Dicionário de Exportação (*Export Dictionary*) que contém os metadados dos objetos que serão compartilhados com outros pontos. O terceiro componente é o Gerenciador de Cache (*Cache Manager*) que determina políticas de *caching* e substituição. Finalmente, o último componente é o Gerenciador de Objetos (*Object Management System*) que provê realmente o armazenamento, manipulação e recuperação dos dados no ponto.

Ponto	Nomes	Palavras-Chave
P1	Kinases	Proteína, humana
	SeqID	Chave, identificador, ID
	Comprimento	Comprimento
	SequenciaProteína	Seqüência, Seqüência Proteína
P2	Proteína	Proteína, annexin, zebrafish
	NumSeq	Número, identificador
	Comp	Comprimento
	Seqüência	Seqüência
P3	ProteinaKComp	Proteína, kinases, comprimento
	ID	Número, identificador
	SeqComprimento	Comprimento
	ProteinaKSeq	Proteína, seqüência
	ID	Número, identificador
	Sequencia	Seqüência
P4	Proteína	Proteína, kinases, annexin
	Nome	Nome
	Caracter	Características, funções

Fig. 4.5 Exemplo de Tabela

Um dos principais objetivos do PeerDB é permitir que seus usuários compartilhem seus dados (definindo o que é privado), sem fazer uso de um esquema global compartilhado. Assim, ele realiza mapeamentos entre esquemas de forma "on the fly", ou seja, dinamicamente à medida que consultas vão sendo realizadas. Para tal, utiliza uma abordagem baseada em Recuperação de Informação que faz uso de tabelas. Para cada tabela que é criada, metadados (palavras-chave + descrições) são mantidos. Estes metadados são referentes às relações e atributos dos pontos. Um exemplo de tabela é apresentado na Figura 4.5.

Assim, considerando o exemplo acima e que tenhamos os Pontos P1, P2, P3 e P4, suponhamos que a seguinte consulta (Q) seja formulada no ponto P1:

```
Q =  SELECT SeqId,SequenciaProteina
     FROM   Kinases
     WHERE comprimento > 30 ;
```

A estratégia *relation-matchings* adotada vai indicar que P2, P3 e P4 são relevantes para Q, visto que estes pontos possuem palavras-chave de relações compatíveis com aquela da consulta. Entretanto, percebe-se que P4 só tem associação com o nome da tabela, isto implica que, num grau de relevância, P4 estaria atrás de P2 e P3. Por outro ângulo, semanticamente, dados de P2 não são de interesse de P1 (pois não são Kinases), então, neste caso, o usuário vai decidir se P2 deve ser consultado.

O processamento de uma consulta é realizado em duas fases no PeerDB. Na primeira, a estratégia de *relation-matching* é aplicada com o objetivo de localizar tabelas candidatas. Estas tabelas são então retornadas ao ponto solicitante para que o usuário selecione as mais relevantes e para que estatísticas para processos futuros possam ser atualizadas. Na segunda fase, as consultas serão direcionadas aos pontos que contêm tais tabelas e as respostas são retornadas (e colocadas em cache).

Nas duas fases, agentes são utilizados: são eles que são enviados aos pontos e interagem com os SGBD e são eles os responsáveis por reescrever uma consulta em outro formato (*DBAgents*). Por exemplo, uma consulta sobre uma tabela pode ser reescrita em uma outra com junção entre várias tabelas. É importante salientar que uma consulta pode ser local ou remota.

4.2.2 Piazza

Piazza é um PDMS desenvolvido por pesquisadores da Universidade de Washington [Tatarinov et al. 2003]. Trata-se de um sistema que permite o compartilhamento de dados heterogêneos de forma distribuída e escalável. O sistema assume que usuários participantes estão interessados em compartilhar dados e dispostos a definir mapeamentos entre seus esquemas. Desta maneira, usuários formulam consultas sobre seu esquema de preferência e o sistema responde à consulta expandindo recursivamente todos os mapeamentos relevantes à mesma, recuperando, assim, os dados importantes que serão retornados aos usuários.

Pontos fortes no Piazza estão relacionados à definição dos mapeamentos e ao processamento de consultas. O primeiro é implementado através de um grafo arbitrário de esquemas interconectados, sendo alguns destes esquemas definidos virtualmente para propósitos de consulta e mapeamento. Uma consulta, por sua vez, é formulada pelo usuário a partir de tabelas de um esquema específico de ponto.

Existem dois tipos de mapeamentos entre esquemas no Piazza - Descrição do Ponto (*Peer Description*) e Descrição de Armazenamento (*Storage Description*), ambos mostrados na Figura 4.6. Descrição do Ponto se refere ao mapeamento que relaciona dois ou mais esquemas de pontos, os quais definem as correspondências entre as "visões de mundo" (esquemas exportados) dos diferentes pontos. Por outro lado, mapear os dados armazenados num ponto em seu esquema exportado é denominado de Descrição de Armazenamento, ou seja, é um mapeamento que relaciona um esquema armazenado a um esquema de ponto.

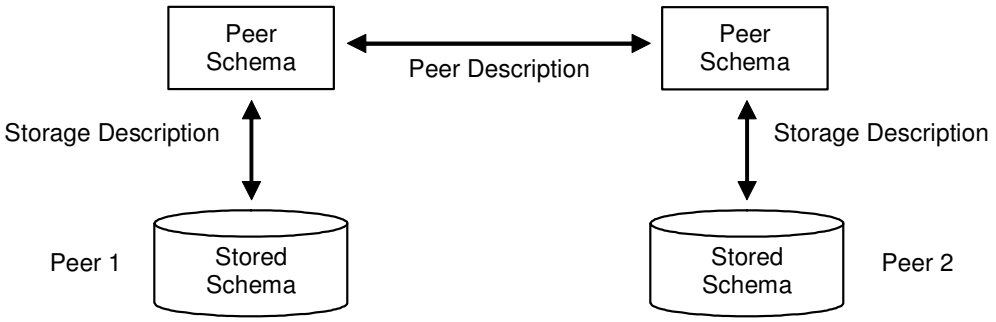

Fig. 4.6 Mapeamentos no Piazza [Tatarinov et al. 2003]

Para a construção dos mapeamentos, o Piazza se baseia no uso conjunto de heurísticas e algoritmos, que possuem como objetivo resolver problemas semânticos. Estes, por sua vez, são baseados em técnicas de aprendizagem de máquina e na exploração de experiências

anteriores, onde informações sobre mapeamentos válidos já existentes são utilizadas para o mapeamento de novos esquemas.

Para que uma consulta **Q** seja processada e seus resultados retornados ao usuário, o Piazza reformula **Q** em uma consulta $\mathbf{Q_0}$ que referencia apenas tabelas armazenadas. Depois avalia $\mathbf{Q_0}$ de forma a obter o resultado, através de técnicas de processamento de consultas adaptativa [Tatarinov et al. 2003]. O aspecto mais importante do processamento é, entretanto, a reformulação da consulta, cujo algoritmo implementado encontra-se apropriado para dados XML.

Este algoritmo aceita como entrada um conjunto de mapeamentos de pontos e descrições de armazenamento, além da consulta **Q** propriamente dita, e retorna como saída uma consulta reformulada $\mathbf{Q_0}$. Ele explora dois tipos de técnicas de reformulação: *unfolding* e *rewriting*. A primeira substitui um objetivo por um conjunto de sub-objetivos enquanto que a segunda substitui um conjunto de sub-objetivos por um único objetivo.

Através do envio de consultas reformuladas, um ponto pode obter os resultados de outros pontos que tenham as respostas. O algoritmo de reformulação ainda garante cortar resultados redundantes. Além disso, para identificar os pontos relevantes ao usuário, o Piazza constrói um índice para explorar informações sobre os reais dados de um ponto. Este índice é implementado como uma máquina de busca Web, diferentemente de outros sistemas que adotam uma estratégia descentralizada baseada em DHT.

4.2.3 Rosa – P2P

O sistema Rosa-P2P é uma evolução do sistema Rosa (*Repository of Objects with Semantic Access*) para um ambiente P2P. O sistema Rosa original é um sistema centralizado que tem como objetivo armazenar objetos de aprendizagem (LO) que representam conteúdos instrucionais [Brito 2005; Brito, Moura 2005]. Desta maneira, o Rosa-P2P busca integrar objetos de aprendizagem em um ambiente P2P, onde usuários podem efetuar consultas através de um Portal ou através de cada ponto participante da rede.

O sistema Rosa-P2P é baseado na topologia de super-ponto e utiliza uma estrutura semântica complexa denominada de vocabulário controlado que irá auxiliar na resolução de conflitos semânticos, sendo esta característica sua principal diferença em relação a outros sistemas como, por exemplo, o Edutella [Löser et al. 2003]. O sistema adota a estratégia de agrupamentos baseados em dois critérios importantes: assunto e localização. Desse modo, pontos com características comuns ficam próximos uns dos outros, o que vem a otimizar o processamento de consultas.

Sua arquitetura interna, composta de três módulos, é apresentada na Figura 4.7 e explicada a seguir:

- Módulo de interoperabilidade: realiza a formação e manutenção da rede P2P, estabelecendo conexões, atualizações de índices de roteamento, eleição de super-pontos e balanceamento da rede.
- Módulo de processamento de consultas: formado por dois componentes – a interface do usuário e o processador de consultas.

- Módulo de gerenciamento de dados: formado por dois componentes – vocabulários controlados e cache/integrador de dados.

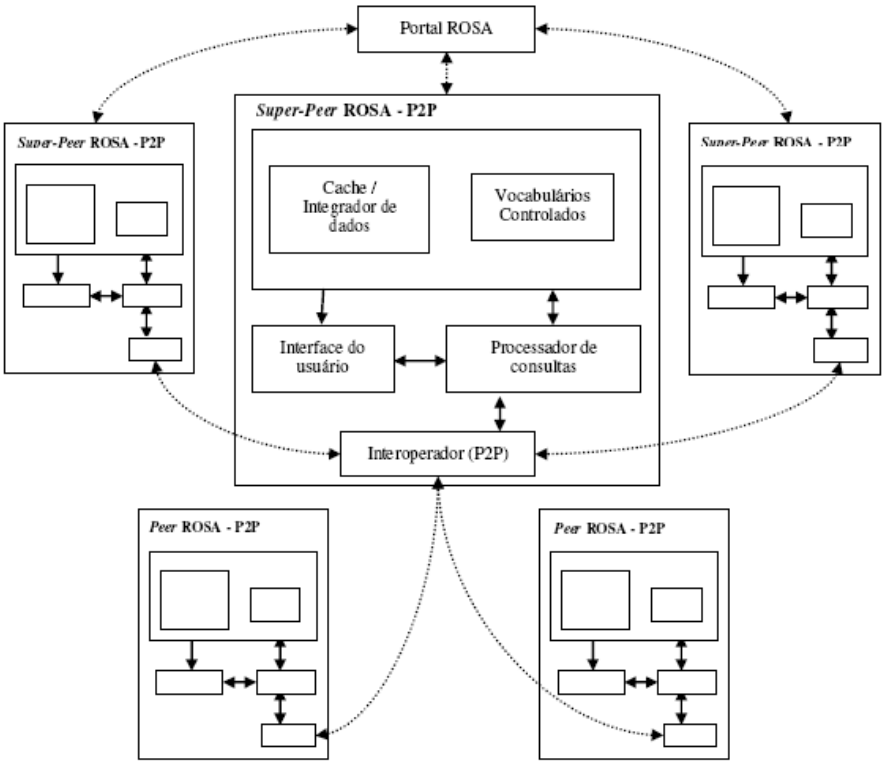

Fig. 4.7 Arquitetura Interna do Sistema Rosa-P2P

O sistema utiliza-se do conceito de vocabulários controlados para agregar valor semântico aos dados durante a integração dos mesmos. Estes vocabulários são utilizados em quatro propósitos básicos: na localização dos pontos relevantes à consulta; na reescrita da consulta pelos pontos receptores; na resolução de possíveis conflitos e na sugestão de opções e caminhos associados com a pesquisa. Para tal, são divididos em três tipos:

- Vocabulário controlado global: utilizado por todos os pontos do sistema.
- Vocabulário controlado local: especifica vocabulários de acordo com cada assunto, referente a um domínio de conhecimento.
- Vocabulário controlado de palavras-chave: apresenta-se como um conjunto de termos semanticamente relacionados, que permite detectar em tempo de execução da consulta qual o assunto relacionado.

Para o processamento de consultas, o sistema Rosa-P2P utiliza uma máquina de execução denominada MEC Rosa que possui uma álgebra específica, composta por operadores como seleção, projeção, navegação e junção. Como o uso do MEC ROSA inicialmente fica restrito a uma base local, foi estabelecida uma solução para sua utilização no

ambiente P2P: cada ponto processa a consulta da mesma maneira que o ponto solicitador, de forma autônoma, ou seja, a reescreve e gerencia sua execução. Assim, a estratégia de processamento de consultas é dividida em duas fases:

a) 1ª Fase

Consiste em identificar os pontos e/ou super-pontos capazes de respondê-la. Contadores (tempo limite para espera dos respectivos resultados e quantidade de pontos) são iniciados e gerenciados. Em outras palavras, primeiramente, localiza-se os super-pontos relevantes àquele domínio da consulta. Para tal, verifica-se se os descritores "título" e/ou "palavras-chave" fazem parte da consulta, caso afirmativo, esses valores serão comparados aos termos existentes no vocabulário controlado de palavras-chave. Uma vez encontrados, os respectivos assuntos são retornados e a consulta é reenviada aos pontos que tratam daquele domínio. Caso esses termos não sejam encontrados, o usuário será solicitado a fornecer algum termo que faça sentido à consulta.

b) 2ª Fase

Esta etapa consiste em reescrever a consulta de acordo com os termos dos vocabulários globais e locais, de modo que as consultas possam englobar um universo de dados mais extenso. Após sua reescrita, ela será processada e o resultado será retornado ao ponto ou super-ponto solicitador. Cada consulta será processada pelo MEC ROSA e ao término da execução de todas as sub-consultas ou ao termino dos contadores, os resultados mantidos em cache pelo solicitador devem ser integrados.

4.2.4 XPeer

XPeer é um PDMS baseado na arquitetura de super-pontos que implementa uma rede lógica baseada na semântica dos dados [Bellahsène et al. 2004]. A semântica exerce papel fundamental, pois é através dela que consultas são processadas e dados são compartilhados pelos pontos participantes. Especificamente, cada ponto pode atuar em um dos seguintes papéis na arquitetura:

- *Cluster peer*: trata-se de um mediador para um grupo de pontos de fontes de dados que compartilham esquemas relacionados semanticamente, conforme a Figura 4.8. O esquema mediado é denominado esquema do cluster. Cada esquema de ponto de dados é definido como uma visão sobre o esquema do cluster através de regras de mapeamento LAV. O *cluster peer* provê mapeamentos diretos entre seus pontos (o que, segundo os autores, dificulta a escalabilidade).
- *Domain peer*: um domínio é um conjunto de clusters que compartilham a mesma categoria de informação (ex: informações sobre os estudantes de todos os departamentos de uma universidade). O *domain peer* provê um serviço de diretório simples sobre os *cluster peers*, com informações sobre o conjunto de palavras-chave que refletem os dados sobre os quais eles podem responder.
- *Global peer*: serve como ponto de entrada para o sistema, localizando a qual domínio um ponto pertence. Para isso, o *global peer* mantém um registro de informações sobre os domínios, dentre elas a categoria da informação por eles providas.

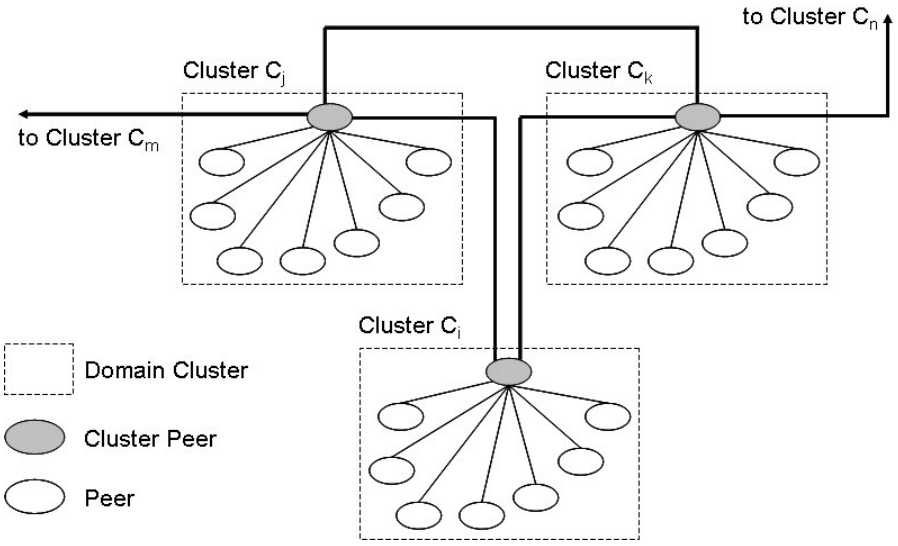

Fig. 4.8 Clusters na Arquitetura do XPeer [Bellahsène et al. 2004]

No XPeer, um ponto pode se juntar ao sistema com dois objetivos: formular consultas e prover, ao mesmo tempo, um esquema a ser compartilhado com outros pontos (atuando como cliente e servidor); ou apenas como cliente formulando consultas. O ponto que assume o papel de *cluster peer* será responsável por manter o esquema de mediação do cluster e por manter os mapeamentos entre o esquema de mediação e os pontos de dados.

A estrutura formada por *global*, *domain* e *cluster peers* reduz a necessidade de mapeamentos entre os pontos, formando uma hierarquia, o que aumenta a escalabilidade. Dessa maneira, *domain peers* e *global peers* são integrados num nível mais alto de abstração, através de palavras-chave. Como resultado, novas fontes de dados podem facilmente entrar e sair da rede, afetando apenas o cluster ao qual pertencem. Entretanto, assumindo um único *global peer*, um nível de *domain peers* e um único domínio para cada cluster peer torna a rede susceptível a falhas. Diante deste fato, uma nova implementação do XPeer, denominada iXPeer [Bellahsène et al. 2006] vem sendo implementada e faz uso do framework AutoMed [Boyd et al.2004] para facilitar esta extensão.

O Projeto de Geração Automática de Ferramentas Mediadoras para Integração de Bancos de Dados Heterogêneos (*Automatic Generation of Mediator Tools for Heterogeneous Database Integration* - AutoMed) [Boyd et al. 2004] usa uma metodologia baseada no formalismo BAV [McBrien, Poulovassilis 2003] onde o processo de integração é visto como uma seqüência de transformações reversíveis que modificam tanto o esquema como a extensão do banco de dados. Dessa maneira, o AutoMed provê uma implementação de uma abordagem baseada em mediador que permite mapeamentos bidirecionais entre fontes de dados e um mediador.

Através da utilização do AutoMed, um ponto de nível mais baixo na arquitetura iXPeer é qualquer fonte de dados (ponto de dados) que deseja prover acesso a suas informações. Um *cluster peer* é um único AutoMedPeer que armazena um meta banco de dados BAV, ou seja, uma rede BAV de todos os esquemas no cluster, juntamente com

detalhes relacionados a seu acesso. Cada *cluster peer* possui um esquema de cluster que torna público as informações disponíveis naquele cluster. Um *domain peer*, por sua vez, é tanto um P2PDirectory quanto um AutoMedPeer, gerenciando os *cluster peers* e permitindo que os mesmos publiquem seus esquemas de cluster. O serviço de P2PDirectory funciona como um serviço de diretório de metadados para a rede P2P, provendo meta informações a respeito dos pontos como, por exemplo, seu endereço IP.

Como mostrado na Figura 4.9, nesta nova implementação, não há distinção entre *domain peers* e *global peers*, o que faz com que *domain peers* possam compor qualquer estrutura hieráquica aninhada [Bellahsène et al. 2006], de maneira arbitrária e ficando livre de pontos de falhas. Embora não apresentado na Figura, *domain peers* e *cluster peers* podem participar de qualquer hierarquia de pontos.

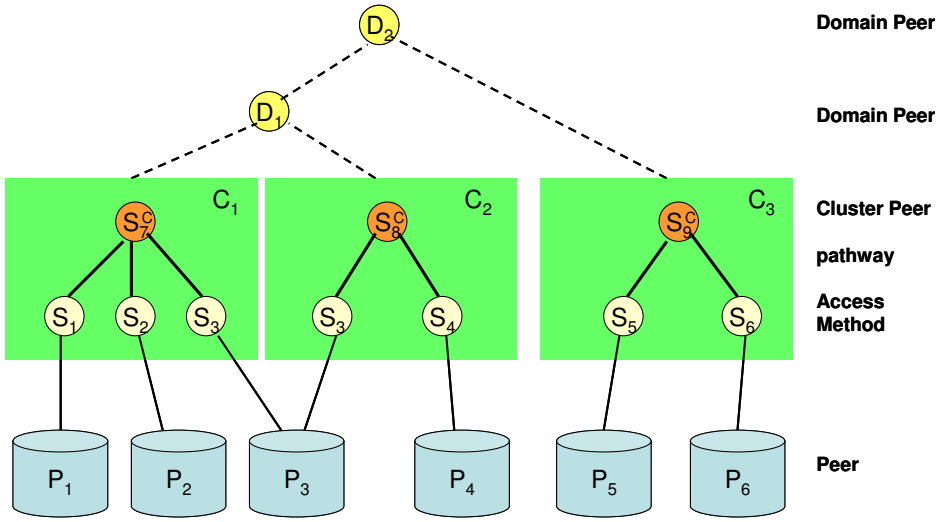

Fig. 4.9 iXPeer: extensão ao XPeer utilizando primitives do AutoMed [Bellahsène et al. 2006]

4.2.5 Helios/H_3

Helios é um sistema para descoberta e compartilhamento de conhecimento baseado em ontologia para sistemas distribuídos P2P [Castano, Montanelli 2005]. Helios é um ambiente multi-ontologias onde cada ponto provê sua própria ontologia e usa um associador semântico para identificar a afinidade semântica entre conceitos armazenados nas ontologias dos diversos pontos. Este associador semântico é, por sua vez, baseado no algoritmo H-Match [Castano et al. 2004] que realiza a associação a partir de aspectos lingüísticos e contextuais dos conceitos sendo comparados.

O projeto Helios (*Helios Evolving Interaction-based Ontology knowledge Sharing*) inicialmente fazia parte de um ambiente denominado H^3 que propunha construir uma *overlay network* entre pontos na qual cada ponto mantinha uma ontologia descrevendo o conhecimento que tinha da rede [Castano et al. 2003]. Esta abordagem permitia que pontos se aliassem a comunidades de interesse e compartilhassem suas informações com seus vizinhos.

Um ponto, segundo a arquitetura H^3, apresentado na Figura 4.10, seria composto de duas camadas:

- Camada de Conhecimento – *Helios*: o compartilhamento e a evolução do conhecimento são baseados nas *ontologias dos pontos*, ou seja, o conhecimento que um ponto traz à rede e o que ele tem da rede e nas *interações entre os pontos*, permitindo recuperação de conteúdo, aquisição e extensão do conhecimento de acordo com modelos de consulta e técnicas de alinhamento de ontologias.
- Camada de Comunicação – *Hermes*: suporta um processo de roteamento enriquecido semanticamente através de um componente de endereçamento baseado em ontologia.

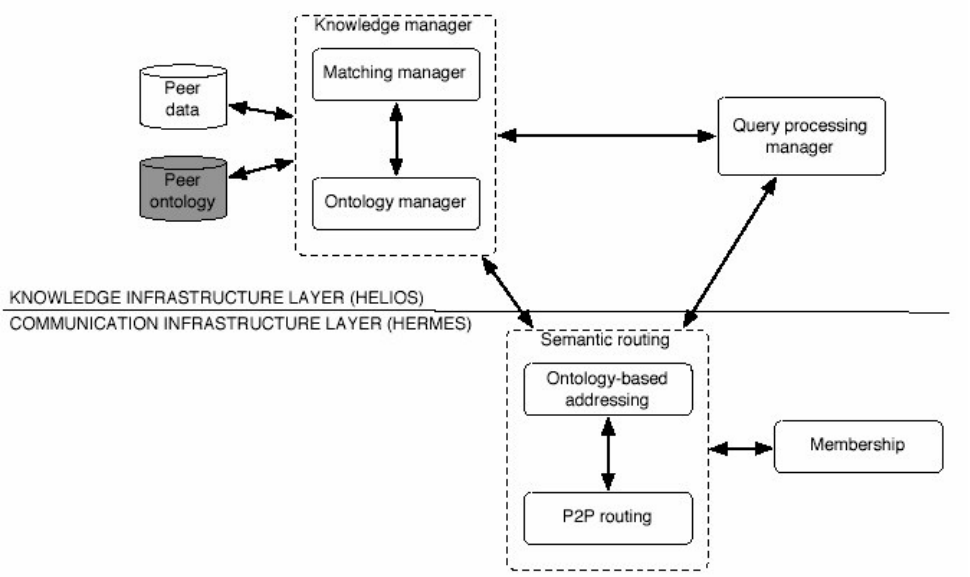

Fig. 4.10 Arquitetura de Referência de um Ponto H3 [Castano et al. 2003]

A camada de conhecimento é composta de três elementos: *gerenciador de processamento de consultas* – responsável por receber consultas, realizar o processamento e a composição da resposta; *gerenciador de associações* – realiza a comparação entre conceitos com o objetivo de encontrar correspondências; e o *gerenciador de ontologias* que suporta a criação e evolução de ontologias.

Três modelos diferentes de consulta são suportados pelo Helios:

- *Search model*: usado para encontrar conteúdos relacionados a um ou mais conceitos de interesse.
- *Probe model*: usado por um ponto interessado em estender seu conhecimento da rede; a resposta da consulta é um conjunto de metadados.
- *Probe/search*: permite que um ponto encontre tanto dados quanto metadados relacionados a um conceito. Com este tipo de consulta, um ponto pode realizar

uma atividade de busca e ao mesmo tempo aumentar o conhecimento que ele tem da rede.

Assim, quando um ponto recebe uma consulta de outro ponto, o gerenciador de processamento de consultas processa a consulta de modo a extrair o conceito buscado e o modelo de consulta requisitado.

A camada de comunicação é composta dos seguintes serviços: serviço de *membership* - invocado pelo ponto no *bootstrap* para descobrir outros pontos da mesma comunidade e também usado no *leave request*, para notificar os vizinhos; *Ontology-based addressing* – responsável pelo repasse da consulta, ou seja, se o ponto possui termos similares aos da consulta, então endereça a consulta, caso contrário, faz um broadcast; e o *Roteamento P2P* - mantém os pontos que possuem semântica comum juntos e faz o roteamento conjuntamente com o serviço de endereçamento baseado na ontologia.

Uma forma de compreender as interações entre os componentes internos de um ponto é por meio do envio e recebimento de uma consulta. O fluxo de interações referente ao envio de uma consulta é descrito abaixo:

a. Um ponto envia uma consulta **Q** sobre a rede H^3
b. O pedido chega ao *Query Processing Manager* que a reescreve em termos da ontologia do ponto (**Q´**).
c. **Q'** (reescrita) é encaminhada ao *Semantic Routing component*.
d. Este envia **Q'** apenas para os que podem responder, de acordo com a lista de pontos descobertos pelo *Knowledge Manager*.
    - O *Knowledge manager* determina caminhos de localização e os envia ao *Semantic Routing* que os utiliza para roteamento da consulta.
e. Se a abordagem dirigida a consultas é adotada, o ponto solicitante especifica qual estratégia de alinhamento deve ser usada.

O fluxo de interações referente ao recebimento de uma consulta é apresentado a seguir:

a. Quando um ponto recebe uma consulta **Q**, o *P2P routing component* a encaminha ao *Query Processing Manager*.
b. O *Query Processing Manager* realiza o processamento:
    - Uma consulta é transformada numa descrição ontológica de conceitos destino para associação com a ontologia do ponto. Dependendo do tipo de consulta (*search*, *probe* ou *search/probe*), dados e/ou metadados que tenham correspondência com a descrição serão obtidos.
    - Se nenhuma associação é encontrada, a consulta é descartada.
c. Se associações foram encontradas, a consulta juntamente com as respostas obtidas é encaminhada ao *Semantic Routing* que envia as respostas ao ponto solicitante

Mais recentemente, o Helios vem sendo estendido com o objetivo de permitir a auto-organização de comunidades semânticas a partir da combinação das descrições ontológicas dos interesses dos pontos juntamente com as técnicas de alinhamento de ontologias [Castano, Montanelli 2005]. Como resultado, pretende-se obter uma maior estruturação da rede através da agregação de pontos vizinhos a partir de um critério semântico, o que gera consultas mais completa e satisfatoriamente respondidas.

4.2.6 Humboldt Discoverer

O *Humboldt Discoverer* provê um índice semântico para uma arquitetura PDMS estendida para ambiente Web [Herschel, Heese 2005]. Através desse índice, um ponto localiza fontes de informação relevantes que não são alcançáveis por meio de caminhos de mapeamentos convencionais. Para tal, o *Humboldt Discoverer* implementa uma infraestrutura P2P mista, o que inclui a utilização de uma abordagem DHT juntamente com uma rede não estruturada que indexa esquemas de pontos através de tecnologias da Web Semântica.

A extensão a um PDMS ocorre através de três dimensões: semântica (*semantic dimension*), Web (*the Web dimension*) e qualidade (*the quality dimension*), como apresentadas na Figura 4.11.

A dimensão semântica provê uma forma alternativa de mapeamento entre esquemas de pontos. Assim, esta camada consiste de ontologias e mapeamentos entre elas, e um ponto deve ter mapeamentos de seu esquema local para uma ou mais ontologias, de modo a possibilitar o compartilhamento de seus dados.

A dimensão da Web é responsável por implementar a característica escalável objetivada pelo projeto, através do índice P2P que ajuda a localizar as fontes relevantes para uma consulta, mesmo que ainda não existam mapeamentos para elas. Para isso, cada ponto mantém o *concept store* que indexa os pontos vizinhos no nível de esquema.

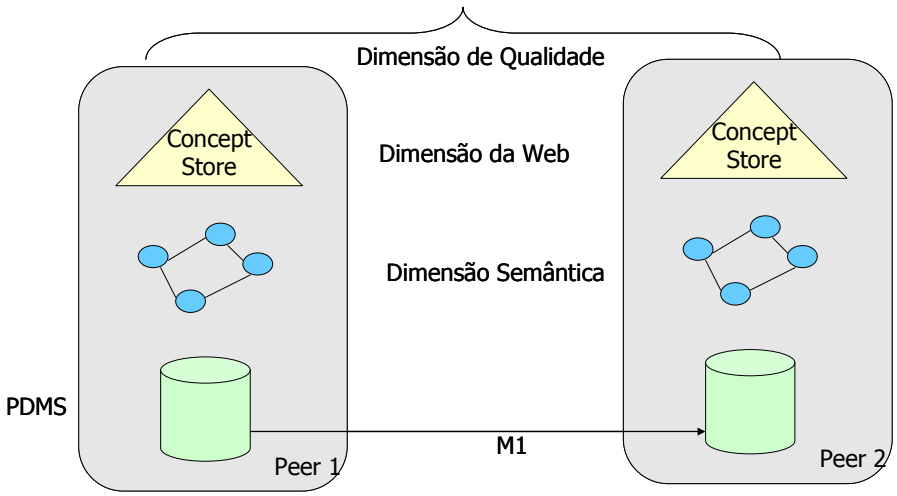

Fig. 4.11 Arquitetura Estendida (adaptado de [Herschel, Heese 2005])

A dimensão de qualidade influencia o processamento de consultas em todas as dimensões. No nível de PDMS e na dimensão semântica, a qualidade do resultado da consulta

é diretamente dependente dos caminhos de mapeamentos providos entre dois pontos. Na dimensão da Web, define-se uma métrica para localizar e realizar um ranking de pontos relevantes para responder à consulta.

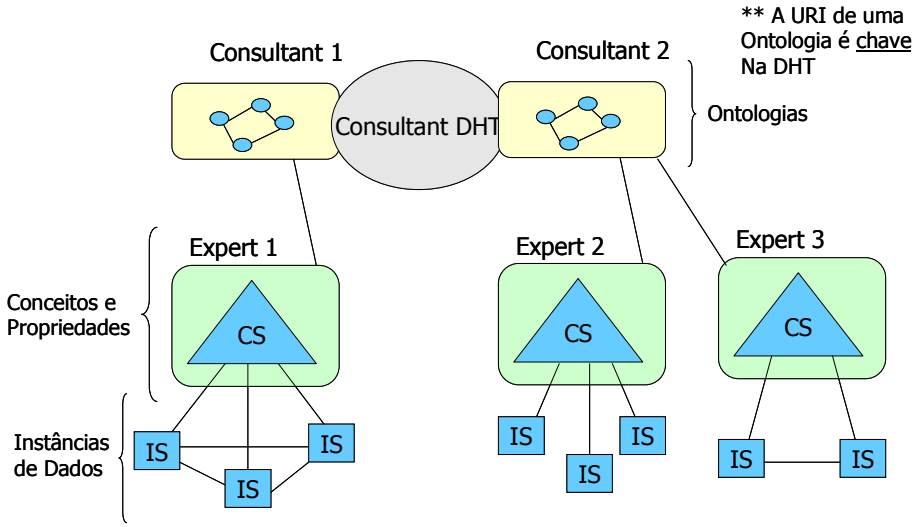

Fig. 4.12 Projeto da Rede P2P com o índice *Humboldt Discoverer* [Heese et al. 2005]

Mas, a grande vantagem deste PDMS estendido é o seu índice semântico denominado *Humboldt Discoverer*. O *Humboldt Discoverer* é um índice P2P que provê um *lookup* de pontos com conteúdos semelhantes e prioriza relevância em vez de eficiência. Para isso, a arquitetura mista que armazena informações de esquema (que não muda frequentemente) é mantida numa DHT (de alto custo para manter) em pontos que tenham sido definidos como confiáveis [Herschel, Heese 2005]. Por outro lado, dados que mudam freqüentemente, como dados de instâncias, são mantidos em sub-redes não estruturadas (menos custosas para manter). O Projeto da rede P2P com o índice *Humboldt Discoverer* é mostrado na Figura 4.12.

Nesta arquitetura, no topo, os nós *consultant* dentro da DHT são responsáveis por armazenar listas de nós especialistas (*expert*), assim como por armazenar uma ou mais ontologias. Os nós do tipo especialista mantêm as fontes de informações com suas respectivas ontologias e provêem capacidades de processamento de consultas dentro destas ontologias.

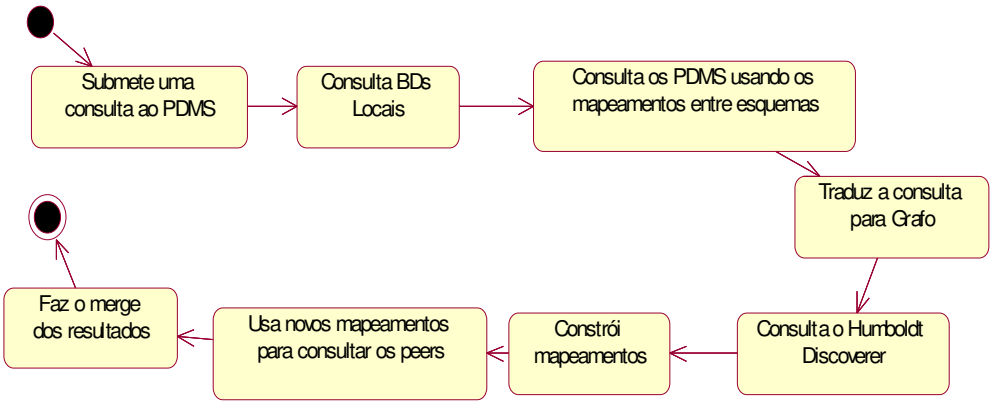

Fig. 4.13 Processamento de Consultas usando o *Humboldt Discoverer*

Dentro deste contexto, de modo genérico, o processamento de consultas é dividido em duas fases. Na primeira, a consulta é processada na camada PDMS, utilizando os mapeamentos já definidos entre os pontos (como mostrado na Figura 4.11). Se o resultado não é satisfatório, a consulta é processada de acordo com os mapeamentos entre as ontologias. Neste caso, a consulta é traduzida para um grafo e o *concept store* é consultado para buscar novas opções de pontos relevantes que possam contribuir. De modo mais detalhado, o processamento de consultas é realizado conforme as etapas da Figura 4.13, explicadas a seguir:

a. Um ponto recebe uma consulta do usuário escrita em SQL
b. Este ponto usa seu banco de dados local para responder à consulta.
c. O ponto identifica os mapeamentos de esquemas do PDMS e direciona a consulta com base nestes mapeamentos. Os pontos destino consultam seus bancos de dados e retornam o resultado.
d. Se a resposta combinada do PDMS é insuficiente, por exemplo, vazia, entra-se na segunda fase, ou seja, o ponto traduz a consulta num grafo de consulta através do mapeamento esquema-ontologia e consulta seu *concept store* para encontrar pontos relevantes e repassar a consulta.
e. O *concept store* retorna uma lista de pontos que podem contribuir.
f. Para cada ponto na lista gerada, constrói-se um mapeamento através da composição dos mapeamentos das ontologias que estão definidas na dimensão semântica.
g. Os mapeamentos compostos formam um novo caminho de mapeamentos no PDMS, podendo ser reutilizados no futuro.
h. Todos os resultados são combinados para produzir o resultado final.

4.2.7 Quadro Comparativo

Diversos são os PDMS atualmente em estudo e desenvolvimento. Cada um possui características específicas, mas em geral, muitas destas são comuns e buscam atender aos requisitos de simplicidade, escalabilidade e corretude de respostas de consultas. Como resultado da análise realizada sobre os sistemas acima descritos, o quadro comparativo 4.3 foi gerado onde as principais características e diferenciais são sumarizadas.

Projeto Característica	Instituição	Representação de Dados	Modelo de Arquitetura	Integração Semântica	Mapeamentos de Esquemas	Linguagem de Consulta	Processamento de Consultas	Características Particulares
PeerDB	Universidade de Singapura	Relacional	Pura	Metadados (palavras-chave)	Através da estratégia de *relation-matching*, usando palavras-chave para entidades e atributos	SQL	Agentes Inteligentes; *Relation-Matching* (Função de Similaridade); Forte dependência e interatividade com o usuário	Um ponto pode se "auto" reconfigurar e definir seus vizinhos; Gerenciamento de cache.
Piazza	Universidade de Washington e Pensilvânia	XML, RDF e DAML-OIL	Pura	Metadados; Estabelecidos aos pares; Explora a propriedade de transitividade; Utiliza GLAV	Heurísticas e Algoritmos; Armazenados em um repositório central	Xquery	Consulta reescrita com base em mapeamentos; Técnicas de aprendizagem de máquina e exploração de casos anteriores	-
Rosa – P2P	IME-RJ (Brasil)	XML e RDF	Super-Ponto	Vocabulários Controlados; Índices de Roteamento	A integração ocorre entre as próprias instâncias	RosaQL	Utiliza uma máquina de execução - MEC Rosa. Cada ponto processa a consulta da mesma maneira que o ponto solicitador, de forma autônoma, ou seja, a reescreve e gerencia sua execução.	Agrupamentos baseados em assunto e localização.
XPeer	Université de Montpellier	XML	Super-Ponto (*Peer*, *Cluster Peer, Domain Peer*, *Global Peer*)	Esquemas dos pontos devem se reportar a um esquema do super-ponto (cluster); Mapeamentos utilizam estratégia LAV/GAV; Extensão iXPeer já usa estratégia BAV	LAV/GAV e BAV (versão iXPeer)	XQuery	Consultas são reescritas pelos *cluster peers* e os resultados são integrados pelos *domain peers* via *cluster peers*.	Gerenciamento de metadados através de um modelo de repositório; Extensão iXPeer utiliza framework AutoMed.

Quadro Comparativo 4.3: Características dos PDMS

Projeto Característica	Instituição	Representação de Dados	Modelo de Arquitetura	Integração Semântica	Mapeamentos de Esquemas	Linguagem de Consulta	Processamento de Consultas	Características Particulares
Hélios/H_3	Universit`a degli Studi di Milano	H-Model (RDF, OWL e UML)	Pura	Mapeamentos entre ontologias			Modelos de Consulta (*Search*, *Probe* ou *Search/Probe*) podem retornar dados e/ou metadados que vão enriquecer o conhecimento que se tem da rede.	Ambiente multi-ontologias onde cada ponto provê sua própria ontologia e usa um associador semântico para identificar a afinidade entre conceitos.
Humboldt Discoverer	Universit¨at zu Berlin	Relacional	Mista com DHT e sub-redes não estruturadas	Ontologias e dimensões (semântica, web e qualidade)	Mapeamentos entre pontos e entre pontos e ontologias	SQL	Primeiramente, a consulta é processada na camada PDMS. Se o resultado não é satisfatório, a consulta é processada de acordo com os mapeamentos entre as ontologias. Neste caso, a consulta é traduzida para um grafo e o *concept store* é consultado para buscar novas opções de pontos que possam contribuir.	Extensão a um PDMS ocorre através de três dimensões: semântica, Web e qualidade; índice semântico provê um *lookup* de pontos com conteúdos semelhantes e prioriza relevância em vez de eficiência

Quadro Comparativo 4.3: Características dos PDMS

4.2.8 Outros Sistemas

Além dos sistemas descritos anteriormente, outros projetos de PDMS se encontram atualmente em pesquisa. Entre eles, podemos destacar: Hyperion [Arenas et al. 2003], Edutella [Löser et al. 2003], Appa [Valduriez, Pacitti 2004], CoDB [Franconi et al. 2004] e SEWASIE [Bergamaschi et al. 2005].

O projeto Hyperion [Arenas et al. 2003] vem sendo desenvolvido pela Universidade de Toronto e Ottawa, no Canadá. Seu principal objetivo é a especificação e o gerenciamento de metadados que permitam o compartilhamento e a coordenação de dados entre pontos autônomos e independentes. Para isso, implementa uma tabela de mapeamento que associa dados residentes em pontos diferentes, provendo mecanismos superficiais para o compartilhamento de dados. Além da tabela de mapeamento, o projeto Hyperion apresenta ainda outros elementos-chave como a coordenação de dados e o mecanismo de regras.

O Projeto Edutella [Löser et al. 2003] é uma aplicação em ambiente P2P para gerenciamento de dados baseado na topologia de super-pontos que visa o compartilhamento de recursos educacionais. Faz uso de RDF (*Resource Description Framework*) para representar os metadados referentes aos recursos disponíveis e do *framework* JXTA para prover a funcionalidade P2P. Utiliza ainda índices entre os super-pontos (SP/SP) e entre os super-pontos e seus pontos (SP/P) com o objetivo de prover roteamento e processamento de consultas.

O PDMS APPA tem como principais objetivos obter escalabilidade, disponibilidade e desempenho para aplicações avançadas [Valduriez, Pacitti 2004]. Para tal, é organizado em uma arquitetura em camadas, podendo ser implementado sobre diferentes topologias de rede (DHT, não estruturada ou super-ponto). As camadas de serviços do APPA são três: camada de rede P2P; camada de serviços básicos e camada de serviços avançados. A primeira provê independência de rede através de serviços que são comuns a todo tipo de rede, como atribuição de identificador de ponto, armazenamento baseado em chave e associação de pontos. A segunda camada provê gerenciamento e comunicação sobre a rede, incluindo gerenciamento de dados, de pontos e membros de grupos. A terceira camada, por sua vez, provê serviços para tratar os dados semanticamente, o que inclui gerenciamento de esquemas, replicação dos dados, processamento de consultas, caching e mecanismos de segurança.

O sistema CODB tem sido desenvolvido com base em quatro premissas essenciais: o papel das formulas de coordenação entre pontos para a migração de dados; a computação delegada aos pontos; a topologia da rede que pode mudar dinamicamente; a finalização e corretude de consultas, garantidas mesmo em caso de mudança na topologia da rede ou na falta das regras de coordenação [Franconi et al. 2004]. As regras de coordenação são definidas através do enfoque GLAV [Madhavan, Halevy 2003] e o sistema vem sendo desenvolvido sobre a plataforma JXTA.

O SEWASIE (**SE**mantic **W**ebs and **A**gentS in **I**ntegrated **E**conomies) é um sistema baseado em agentes e mediadores que tem por objetivo desenvolver uma máquina de busca que proveja acesso inteligente a fontes de dados heterogêneas na Web através de enriquecimento semântico [Bergamaschi et al. 2005]. Para isso, utiliza conceitos do

paradigma P2P onde pontos são chamados de SINodes e alguns agentes desempenham o papel de super-pontos (*brokering agents*). Assim, nós de informação (SINodes) são sistemas baseados em mediador que provêem uma visão virtual sobre as fontes de informação gerenciadas por meio deles. Cada SINode exporta uma ontologia que representa os metadados da sua visão virtual. Estas ontologias são então integradas pelos *brokering agents* que auxiliarão outros agentes (*query agents*) a executarem as consultas. Além disso, agentes de comunicação realizam tarefas de comunicação que formam a camada de rede.

4.3 Considerações

Este capítulo apresentou o conceito de sistemas P2P e de PDMS. Ambos constituem uma realidade atualmente, cada um com seus objetivos. Sistemas P2P são adequados ao compartilhamento de arquivos, troca de mensagens e trabalho colaborativo, com vários exemplos de aplicações (como Kazaa e Gnutella) que são verdadeiros sucessos entre usuários.

PDMS compõem uma nova classe de sistemas distribuídos, sendo considerados uma evolução dos sistemas de integração de dados, pois permitem a abstração dos dados distribuídos em fontes (no caso, pontos) autônomas e heterogêneas. Em especial, PDMS provêem escalabilidade e flexibilidade, sem a necessidade de autoridade central, nem de esquema global. Consultas são submetidas em pontos e propagadas por todos os pontos que podem contribuir com dados.

Exemplos de PDMS também foram descritos, através de suas características e propriedades arquiteturais. Como forma de sumarizar a análise realizada, geramos um quadro comparativo com as principais características e diferenciais entre eles. No capítulo 6 apresentaremos o PDMS (SPEED) que será a base para o desenvolvimento deste trabalho.

Capítulo 5

Processamento de Consulta em PDMS

Como visto em capítulos anteriores, um PDMS é um sistema de gerenciamento de dados que faz uso de uma arquitetura P2P. Seu objetivo principal é permitir a troca de dados, compartilhamento de informações e o processamento de consultas, sendo este último reconhecido como o principal serviço que um PDMS pode prover [Arenas et al. 2003]. Neste sentido, processar uma consulta em um sistema como esse significa prover capacidades de responder às consultas sobre uma rede arbitrária de pontos com esquemas locais e compartilhados e um conjunto de mapeamentos entre estes pontos.

A definição de uma estratégia de processamento de consultas em um PDMS é de vital importância para que os dados possam ser localizados, filtrados, recuperados e integrados de forma otimizada. De uma maneira simplificada, dada uma consulta **Q**, o sistema deve ser capaz de realizar pelo menos duas etapas básicas: (i) descobrir quais os pontos que possuem informações relevantes; e (ii) executar eficientemente a consulta **Q**. Entretanto, diversas estratégias de processamento de consultas podem ser implementadas, dependendo do tipo de sistema e da definição dos mapeamentos entre os pontos.

Este capítulo apresenta conceitos relacionados ao processamento de consultas em um PDMS. Para tal, apresenta uma introdução ao problema incluindo um exemplo motivador e aborda o conceito de processamento de consultas de maneira geral. Em seguida, detalha problemas e desafios relacionados à reformulação de consultas a partir de mapeamentos semânticos.

5.1 Introdução

Como uma evolução natural dos sistemas de integração de dados, os PDMS têm por objetivo básico permitir que usuários realizem consultas de forma transparente sobre diversos pontos (fontes de dados) heterogêneos e autônomos. Para isso, entretanto, não utiliza um esquema global, adotando geralmente estratégias baseadas em índices de consultas, metadados e mapeamentos entre pontos para conseguir executar as consultas.

Para que um ponto possa participar de um PDMS, ele deve publicar seu esquema de dados de modo que este possa ser visto pelos demais pontos da rede e, em contrapartida, o sistema deve prover mapeamentos deste ponto com outros. Através de mapeamentos ou da composição de mapeamentos, o sistema pode obter dados relevantes de qualquer ponto que esteja conectado [Tatarinov, Halevy 2004]. Assim, os mapeamentos entre pontos são fundamentais para o processamento de consultas, visto que sua utilização irá determinar a qualidade e o desempenho da execução das mesmas.

De acordo com as propriedades de flexibilidade e descentralização, um PDMS é adequado ao compartilhamento e troca de dados de diversos domínios como, por exemplo, dados compartilhados em pesquisas científicas [Ng et al. 2003]. Existem muitos cenários onde um PDMS pode ser útil: bioinformática, projetos institucionais, ensino, medicina, sistemas de informações geográficas, entre outros. Estes cenários são encontrados em projetos onde profissionais desejam compartilhar dados, cujas fontes podem ter interseções e/ou complementações e existe o propósito de interagir e produzir resultados de pesquisa mais amplos e completos.

Com o objetivo de contextualizar a problemática do processamento de dados em PDMS, apresentamos um exemplo de cenário de compartilhamento de dados voltado para um problema que instiga cientistas do mundo todo: o aquecimento global. Neste panorama, cientistas, meteorologistas e climatólogos de várias instituições de pesquisa têm se empenhado em estudar o fenômeno do aquecimento global, com suas causas e efeitos. Fenômenos naturais diversos como vulcões, ciclones, terremotos, degelos, nível dos oceanos estão associados a este fenômeno complexo. Assim, supondo dois pontos, com fontes de dados objeto-relacionais localizadas em diferentes universidades, temos os esquemas exportados de duas delas (**UP** – Portugal e **UB** - Brasil):

UP: UP.Fenômeno_natural(id, nome, tipo, gravidade, coordenadas)

UB: UB.Fenômeno_natural(id, nome, tipo, mortes, data_ocorrência)

Ao se efetuar uma análise sobre os esquemas, percebe-se que cada fonte possui informações próprias e que existem informações que somente são encontradas na fonte **UP** (coordenadas e gravidade) e outras na fonte **UB** (mortes e data_ocorrência). Além disso, mesmo que ambas as fontes possuam informações sobrepostas, seus conteúdos podem ser diferentes. Por exemplo, instâncias de ambas as fontes poderiam ser:

UP: (10, "Joaquina","Terramoto", "Alta", (23456.43, 67548.89)) e

UB: (32,"Joaquina","Terremoto","sim","20050811")

Percebemos que ambas as entidades se referem ao mesmo conceito do mundo real, apesar de heterogeneidades estruturais. Vale à pena então compartilhar tais informações entre os cientistas de ambas as universidades e produzir resultados mais completos para a pesquisa.

A Figura 5.1 mostra como consultas são propagadas e traduzidas em PDMS que adotam a topologia pura. Na maioria dos PDMS, um usuário de **UB** irá realizar uma consulta com base no esquema local de **UB**. Respostas são processadas em **UB** e a consulta é então reformulada e repassada para outros pontos através dos caminhos de mapeamentos na rede. Por exemplo, assumindo que existe um mapeamento $\mathbf{Map_{UB\_UP}}$ entre **UB** e **UP**, a consulta $\mathbf{Q_{UB}}$ será reformulada para $\mathbf{Q_{UP}}$, de acordo com o esquema de **UP**. $\mathbf{Q_{UP}}$ será processada no ponto **UP**. A consulta poderá ser reformulada para pontos adicionais vizinhos a **UP**, caso existam mapeamentos para eles. Resultados da consulta serão enviados de volta ao ponto **UB** (inicial) e integrados, depois das execuções nos pontos alcançados. Deste modo, o usuário que formulou a consulta $\mathbf{Q_{UB}}$ receberá resultados não somente de **UB**, mas de todos os pontos que contribuíram com respostas.

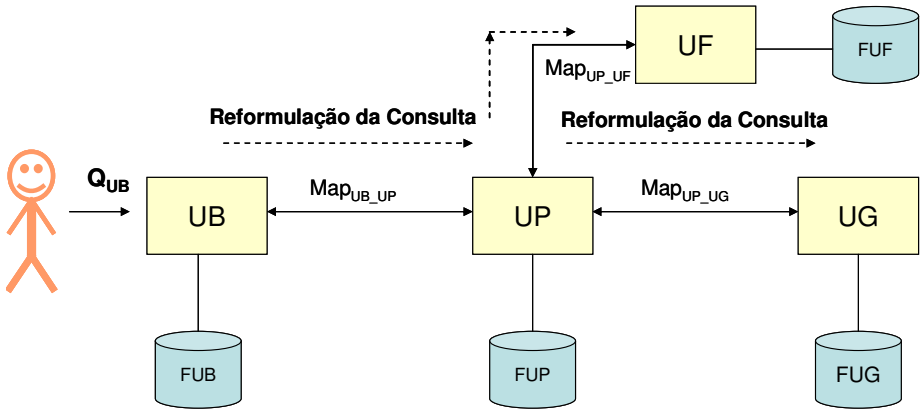

Fig. 5.1 Processamento de Consultas em um PDMS Genérico

Este exemplo demonstra aspectos importantes dos PDMS que têm forte influência sob o ponto de vista do processamento de consultas. Primeiramente, como a arquitetura para compartilhamento de dados é realmente descentralizada, não existe controle sobre os mapeamentos nem tampouco sobre o processamento de consultas, a não ser que extensões sejam providas para tal (como a utilização de TTL, por exemplo). Além disso, podemos considerar os PDMS como uma generalização dos sistemas de integração de dados, visto que alguns pontos funcionam como mediadores para outros pontos quando reformulam a consulta. Na Figura 5.1, por exemplo, UP atua como mediador para UF e UG. Neste sentido, problemas como a reformulação da consulta e a manutenção dos mapeamentos são comuns tanto a sistemas de integração de dados quanto a PDMS, entretanto, devido à dinamicidade e ausência de esquema global, estes problemas se tornam mais complexos em PDMS.

De maneira geral, o processamento de uma consulta está relacionado a quatro etapas fundamentais: análise, reformulação, otimização e execução. No caso de PDMS e sistemas de integração de dados, um dos problemas mais críticos está relacionado à reformulação da consulta, pois envolve a utilização dos mapeamentos de forma eficiente e pode levar em consideração aspectos semânticos como forma de otimização.

5.2 Processamento de Consultas

Desde os primórdios de utilização dos SGBD que o problema de processar uma consulta é fator-chave. O processamento de consultas consiste em transformar uma consulta definida em uma linguagem de manipulação de dados (DML) em um plano de execução e, em seguida, executar esse plano sobre o conteúdo na base de dados. Essa transformação deve fornecer corretude, de forma que o plano de execução final devolva os resultados requeridos na consulta de alto nível e eficiência, para que o tempo decorrido nesse processo não comprometa o tempo de resposta da consulta.

A seguir, descrevemos o processamento de consultas nos diversos tipos de tecnologias de bancos de dados e nos ambientes P2P.

5.2.1 Processamento de Consultas em Modelo Centralizado

O processamento de consultas escritas em linguagens declarativas (como, por exemplo, SQL ou XQuery) tem sido objeto de estudo desde a origem dos sistemas de bancos de dados relacionais. Segundo Katchaounov [Katchaounov 2003], o processamento de consultas é um termo coletivo que compreende todas as técnicas utilizadas para computar o resultado de uma consulta expressada através de uma linguagem declarativa. Diferentes técnicas podem ser usadas para processar, otimizar e executar consultas de alto nível no contexto de um SGBD. Mas, em geral, consultas são processadas de acordo com o diagrama ilustrado na Figura 5.2 que toma como exemplo o modelo relacional.

Primeiramente, o analisador léxico identifica os itens léxicos da consulta SQL, e a análise sintática verifica se a consulta está formulada de acordo com as regras sintáticas da linguagem de consulta, gerando uma árvore da consulta. Esta árvore é analisada pelo módulo de pré-processamento, de acordo com critérios semânticos, como, por exemplo, se uma relação realmente corresponde a relações existentes, se atributos e constantes são compatíveis com o tipo, entre outros. A árvore de consulta semanticamente correta é traduzida para uma representação interna.

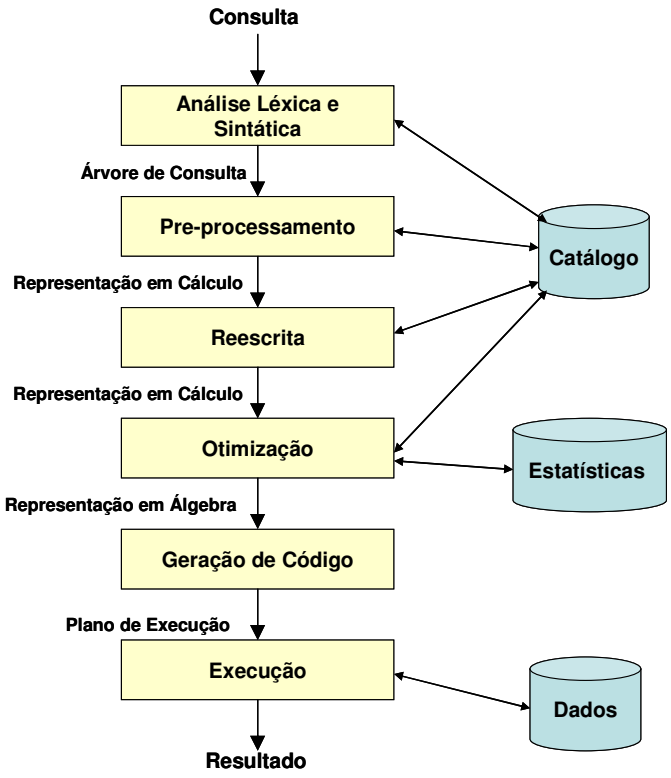

Fig. 5.2 Passos Típicos durante a execução de uma consulta em um SGBD

Neste ponto, inicia-se a otimização da consulta, realizada em duas etapas: a reescrita da consulta, onde uma nova representação é gerada (geralmente baseada no cálculo de predicados). Em seguida, a etapa propriamente dita de otimização é realizada, ou seja, a representação anterior da consulta é traduzida para um formato algébrico (álgebra relacional),

onde se torna mais eficiente. Para produzir um plano de execução de consulta (PEC) ótimo, a fase de otimização busca as expressões algébricas equivalentes que possam ser utilizadas para computar a consulta, atribui os algoritmos aplicáveis aos operadores e calcula o custo de execução de todos os operadores, de acordo com a seqüência exigida. Além disso, avalia a qualidade de cada plano gerado para escolher o mais adequado. Pode haver ainda uma outra fase onde a consulta em álgebra é transformada em código de mais baixo nível, para finalmente ser executada e possivelmente ser armazenada para uso posterior.

5.2.2 Processamento de Consultas em Modelos Distribuídos

Em sistemas distribuídos, fatores adicionais como o custo de transferência de dados na rede, o grau de homogeneidade/heterogeneidade, grau de autonomia e a existência ou não de um esquema global podem complicar ainda mais o processamento de consultas. Em sistemas com múltiplos bancos de dados, por exemplo, o processamento de consultas é realizado de acordo com três passos básicos [Evrendilek et al. 1995]: primeiramente, uma consulta global é decomposta em sub-consultas de forma que os dados necessários para cada sub-consulta estejam disponíveis em um banco de dados local; depois cada sub-consulta é traduzida para uma ou mais consultas do sistema da fonte local. Em seguida, os resultados retornados pelas sub-consultas são combinados para produzir a resposta final.

Em sistemas de integração de dados, o processamento de consultas pode diferir dependendo da abordagem utilizada. Na abordagem virtual, o mediador recebe a consulta do usuário e faz a decomposição em sub-consultas sobre as fontes de dados. Os resultados das sub-consultas enviados pelas fontes de dados são traduzidos, filtrados e combinados e a resposta final é retornada ao usuário. Na abordagem materializada, por sua vez, as consultas submetidas ao sistema de integração são avaliadas no repositório, o *data warehouse*, sem a necessidade de haver acesso às fontes de dados. É importante salientar que existem sistemas que combinam recursos de ambas as abordagens suportando o processamento de consultas virtuais e materializadas, como é o caso do Integra [Lóscio 2003, Batista 2003].

Analisando mais detalhadamente os sistemas de integração baseados em mediador (que usam a abordagem virtual), percebemos que fases são acrescentadas ou estendidas, se compararmos com o processamento de consultas mostrado num SGBD Relacional (Figura 5.2). Esta extensão é mostrada na Figura 5.3, onde a fase de decomposição é adicionada. A fase de decomposição identifica qual parte da consulta vai ser processada por qual fonte de dados. Esta fase de decomposição também se encontra presente nos sistemas federados ou com múltiplos bancos de dados. Entretanto, nestes sistemas, não é necessário considerar e tratar a capacidade computacional das fontes de dados.

Como resultado da decomposição, o plano original de execução de consultas é dividido em sub-consultas, cada uma executável em uma fonte de dados. Estas são submetidas aos *wrappers* que as traduzem de acordo com requisitos específicos para o tipo da fonte acessada. A fonte retorna, então, resultados em seu formato que são mapeados para o padrão do sistema. Estes resultados são combinados para produzir o resultado final.

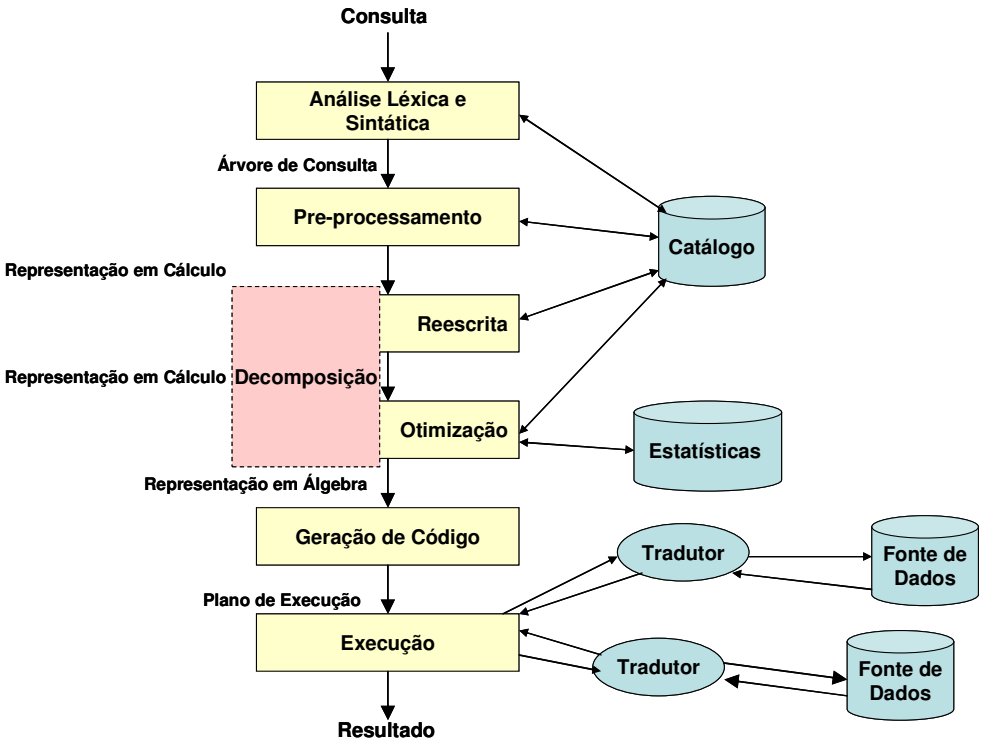

Fig. 5.3 Processamento de Consultas num Sistema baseado em Mediador (adaptado de [Katchanouv 2003])

Fazendo uma breve comparação em termos de otimização de consultas, o processamento de consultas tradicional, como mostrado na Figura 5.2, é realizado num ambiente cujas estatísticas são computadas *offline* e usadas estaticamente para otimizar a consulta que pode então ser executada. Isto é resultado do controle que existe sobre o banco de dados, tornando as informações do ambiente previsíveis e consistentes. Por outro lado, em sistemas de integração, muitas definições devem ser assumidas durante a otimização e as estatísticas calculadas dinamicamente, já que fontes podem estar indisponíveis ou não se ter completa informação sobre elas. Técnicas adaptativas têm sido pesquisadas com o objetivo de facilitar esta otimização [Ives et al. 2000].

5.2.3 Processamento de Consultas em Sistemas P2P

Um sistema P2P inclui uma grande rede heterogênea, autônoma, dinâmica e escalável onde pontos podem trocar e compartilhar dados e serviços de maneira descentralizada [Zhuge et al. 2005]. A motivação básica para utilização de tais sistemas é o compartilhamento de arquivos. Então, realizar uma consulta em um sistema P2P é, na verdade, encontrar um arquivo ou parte dele.

Sistemas P2P são classificados de acordo com sua arquitetura de rede, determinada pela localização dos arquivos e índices (onde estão armazenados) e/ou pela forma como são organizados para prover a realização de consultas ou buscas. Considerando este último critério, podemos dizer que sistemas P2P podem ser não estruturados ou estruturados. No primeiro caso, não há restrições quanto à forma como os dados são localizados, podendo ser

classificados em puros, puros com super-pontos (super Peers) ou híbridos [Sung et al.2005]. Os sistemas estruturados, por sua vez, são referenciados como sistemas que usam *Distributed Hash Tables* – DHT, apresentando ferramenta de *lookup* e de recuperação de dados. Sistemas que empregam DHT implementam endereçamento de nós/dados, protocolos de roteamento e protocolo de manutenção de estado de roteamento, provendo alta escalabilidade.

A localização de um arquivo, num sistema não estruturado, pode ser efetuada de duas maneiras: através da busca por palavra-chave (*keyword*) ou através da identificação do arquivo ou de partes dele. Por exemplo, no sistema BitTorrent [Sung et al. 2005], cada arquivo é mapeado para uma string cujo tamanho é um múltiplo de 20. A string pode ser dividida, onde cada parte atua como um índice para uma parte do arquivo.

Assim, o processamento de consultas em um sistema não estruturado é realizado através de dois métodos [Sung et al. 2005]:

- *Blind search*: arbitrariamente, as consultas são enviadas aos pontos sem nenhum conhecimento se a consulta será satisfeita, ou seja, o roteamento é feito de forma aleatória. A maneira mais simples de se executar o *blind search* é através da técnica de inundação (*flooding*), ou seja, a fim de propagar uma consulta para um certo número de pontos, cada ponto a envia para os seus vizinhos. A limitação desta varredura é normalmente feita através de um contador – TTL (*Time-to-Live*). Esta técnica apresenta a vantagem de rapidamente propagar a consulta, entretanto está limitada ao raio de ação definido pelo TTL.
- *Informed search*: consultas são enviadas aos pontos que têm uma alta probabilidade de satisfazer as mesmas. Para isso ser possível, os pontos armazenam metadados sobre os outros pontos. Um exemplo de estratégia que otimiza este tipo de busca é a que utiliza *Bloom Filter*. Um *Bloom Filter* contém palavras-chave descrevendo os arquivos que um ponto oferece, sendo frequentemente intercambiado entre os vizinhos. Desta maneira, cada ponto que o recebe, acrescenta suas próprias informações e o propaga para os demais.

Com o objetivo de melhorar o desempenho de técnicas aleatórias como a inundação, os conceitos de agrupamento (*clustering*) e indexação de pontos vêm sendo propostos [Zhuge et al. 2005]. A abordagem de agrupamento de pontos é utilizada para prover eficiência, como no sistema Edutella [Nejdl et al. 2003], por exemplo. No trabalho de Koloniari e Pitoura [Koloniari e Pitoura 2004] um roteamento baseado em conteúdo é proposto, ou seja, informações sobre a estrutura dos documentos são mantidas para facilitar o roteamento das consultas.

Sistemas estruturados utilizam mecanismos de localização dos arquivos escaláveis que se baseiam em tabelas *hash* distribuídas (DHT). Neste mecanismo, cada chave da tabela possui seu respectivo conteúdo armazenado por apenas um ponto. Quando um ponto deseja saber a localização de um arquivo, que é mapeado univocamente em uma chave, ele inicia uma busca que é roteada para o nó responsável por armazenar a respectiva chave. O ponto que possui a responsabilidade pela chave recebe a busca pela chave e responde com o seu valor, que é a localização do arquivo desejado.

Sistemas estruturados como Can [Ratsanamy et al. 2001], Chord [Stoica et al. 2001] e Pastry [Rowstron, Druschel 2001] usam DHT para roteamento. Estes sistemas são considerados muito eficientes no uso de recursos para a comunicação entre pontos, sendo altamente escaláveis. Contudo, não oferecem diretamente a possibilidade de pesquisar a informação com base no conteúdo, uma vez que assumem a existência de identificadores já conhecidos para o acesso a cada conjunto de dados disponíveis.

Sistemas baseados em DHT permitem consultas apenas por palavras-chave e associações exatas (*exact matches).* Algumas arquiteturas têm sido propostas para suportar consultas mais complexas (baseadas em DHT ou redes não estruturadas) como consultas por intervalo (*range queries*), consultas com múltiplos atributos (*multi-attribute query*), consultas com junção (*join queries*) e consultas por agregação (que fazem uso de operadores como *count*, *sum*, *average, grouping, minimum, maximum*).

5.2.4 Processamento de Consultas em PDMS

Realizando mais uma vez um paralelo entre sistemas de integração de dados e PDMS, observa-se que os primeiros podem ser compostos de vários e distribuídos mediadores especializados em algum domínio do conhecimento. Cada mediador integra apenas um conjunto de fontes disponíveis e compartilha suas abstrações com um nível mais alto de mediadores (uma hierarquia) e com aplicações [Katchanouv 2003]. Esta noção de mediadores distribuídos compondo uma hierarquia é uma visão totalmente compatível com as arquiteturas dos PDMS, principalmente com a topologia de super-ponto.

Na ausência de um esquema global, pontos podem desempenhar diversos papéis como *clientes*, formulando consultas, *servidores de dados* ou *mediadores*, reformulando consultas com base em mapeamentos entre pontos. Estes papéis têm associação com a topologia e a arquitetura definida para o sistema. Desta maneira, em arquiteturas puras, o processamento de consultas acontece da maneira como foi apresentado no exemplo motivador da Figura 5.1. Ou seja, uma consulta é iniciada em um ponto que a reformula para seus vizinhos. Estes também a reformulam para seus próprios vizinhos e assim segue por todos os caminhos de mapeamentos semânticos entre os pontos. Quando um ponto tem condições de responder à consulta, o faz e envia o seu resultado ao ponto que requisitou o processo. O ponto inicial, após todos os demais retornarem seus resultados, integra os mesmos e apresenta ao usuário um resultado final.

As principais vantagens da arquitetura pura sob o ponto de vista de processamento de consultas são: escalabilidade, pois qualquer ponto pode entrar e sair do sistema, sem interferir nos demais e sem interferir no andamento das consultas; inexistência de pontos de falhas, ou seja, respostas geralmente serão encontradas, mesmo que momentaneamente alguns pontos estejam indisponíveis, a não ser que o conjunto resposta seja realmente vazio. Por outro lado, as desvantagens observadas são: o algoritmo pode seguir por caminhos ineficientes, que não levem a resposta alguma, pode obter caminhos redundantes, resultando em consultas desnecessárias e, além destas, o tempo de resposta pode ser bastante alto.

Na arquitetura de super-ponto, este tipo de ponto age como um servidor dedicado a um conjunto de pontos, formando clusters semânticos, ou seja, um agrupamento de pontos que

têm um domínio comum de interesse. Assim, super-pontos são os responsáveis por manter índices de roteamento, gerenciar metadados e processar a consulta, agindo muitas vezes como mediadores. Em algums sistemas [Tatarinov et al. 2003; Bellahsène et al. 2004], inclusive, esquemas integrados são gerados no super-ponto e mapeamentos entre os super-pontos e os demais pontos do cluster são mantidos.

Nesta visão, quando uma consulta é submetida, geralmente ela é enviada ao super-ponto que identifica quais pontos dentro do cluster estão habilitados a responder a consulta. A partir daí, o super-ponto reformula a consulta de acordo com os mapeamentos e características específicas de cada ponto e a envia para que cada um a execute. O resultado é retornado ao super-ponto que integra todos eles e devolve ao ponto que iniciou a consulta. Este apresenta o resultado final ao usuário. Cada super-ponto pode repassar a consulta também a outros super-pontos que tenham semânticas comuns com o domínio daquela consulta. Assim, a consulta é propagada tanto vertical quanto horizontalmente na rede, de acordo com mapeamentos entre os pontos. Este processo é mostrado na Figura 5.4.

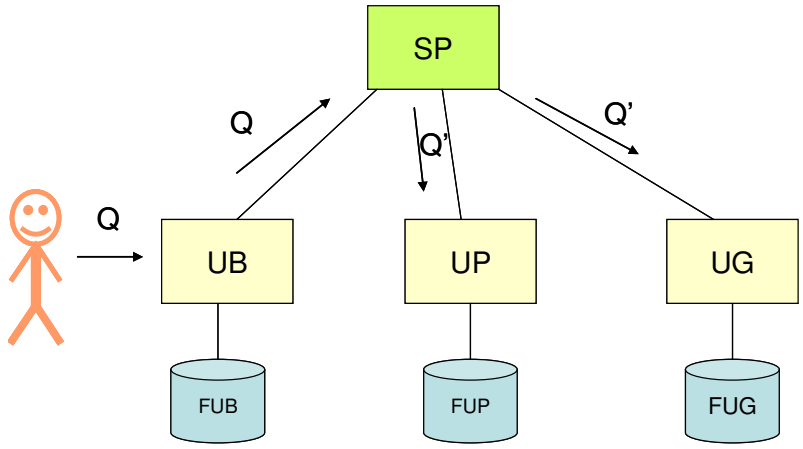

Fig. 5.4 Processamento de Consulta em PDMS com Super-Ponto

Vantagens da arquitetura de super-ponto, sob a ótica de processamento de consultas são: pontos que pertencem a domínios de conhecimento comuns são agrupados próximos uns aos outros sob um leve "controle" que geralmente se encarrega de realizar a integração dos resultados; o desempenho da execução da consulta é superior ao da topologia pura, visto que o sistema é particionado em pequenos grupos que têm interesses comuns, o que facilita o roteamento da consulta. Desvantagens desta abordagem incluem: super-pontos podem se tornar pontos críticos de falhas e, ao mesmo tempo, podem degradar o desempenho da resposta de uma consulta.

É interessante observar que a temática central em torno do processamento de consultas em ambientes de integração de dados, sejam eles baseados em esquema global ou em redes P2P, é a habilidade de responder consultas através de mapeamentos. Mais especificamente falando, a habilidade de reformular consultas com base em mapeamentos entre esquemas

(entre pontos num PDMS ou entre o esquema global e os esquemas das fontes, em sistemas de integração convencionais). A questão pode ser descrita assim [Karvounarakis 2005]:

Dada uma consulta **Q** sobre um esquema **A**, um esquema **B** e um mapeamento **M** entre eles, existe uma consulta **Q′** sobre **B** que retorne respostas "relevantes" de **Q**?

Para responder esta questão, é necessário aprofundar a discussão em torno de dois pontos fundamentais: analisar mecanismos eficientes para definição e manutenção dos mapeamentos e como estes interferem na reformulação da consulta.

5.3 Reformulação da Consulta

A reformulação da consulta é o processo pelo qual uma consulta num esquema de fonte de dados **A** é traduzida para o esquema da fonte de dados **B** de modo que possa ser compreendida por **B**. Desta maneira, o problema da reformulação de consultas consiste em encontrar uma consulta **Q′** para a fonte **B**, dada uma consulta **Q**, tal que:

- **Q′** contém respostas corretas;
- **Q′** $\subseteq$ **Q**;
- **Q′** provê todas as respostas possíveis para **Q**.

O processo de reformulação de uma consulta pode ser dividido em duas etapas: a reescrita da consulta que gera uma expressão de consulta (Q') e a resolução da consulta cujo resultado é o conjunto de todas as respostas possíveis para aquela expressão de consulta [Halevy 2000]. Para viabilizar as duas etapas, é necessário fazer uso de mapeamentos semânticos entre os esquemas que estão sendo empregados na reformulação. Os mapeamentos semânticos descrevem relacionamentos entre os termos usados em dois ou mais esquemas. Isto significa que soluções para o problema de reformulação de consulta devem incluir linguagens para especificação de mapeamentos e algoritmos que usem estes mapeamentos para resolver ou responder adequadamente as consultas.

Com o intuito de compreender melhor como a definição de mapeamentos interfere na reformulação da consulta, apresentamos a seguir conceitos que fundamentam sua especificação e utilização.

5.4 Mapeamentos Semânticos

Mapeamentos semânticos definem relacionamentos entre termos ou objetos em dois ou mais esquemas com o propósito de compartilhamento e integração dos dados [Sung et al. 2005]. Mapeamentos entre esquemas são sempre transitivos e, ao mesmo tempo em que estabelecem a maneira como os dados vão ser permutados num ambiente de integração, determinam como as consultas vão ser reformuladas entre diferentes esquemas. No caso dos sistemas convencionais que fazem uso de um esquema global, os mapeamentos são estabelecidos entre o esquema global e as fontes de dados. Nos PDMS, onde se torna inviável a manutenção de

um único esquema global, os mapeamentos são estabelecidos normalmente aos pares entre os pontos ou entre pequenos grupos de pontos.

Assim, em um ambiente PDMS, mapeamentos podem ser definidos de acordo com três modelos [Sung et al. 2005]:

a. Mapeamentos aos Pares: é a técnica mais simples e envolve o estabelecimento de mapeamentos entre pares de pontos. Graças à propriedade de transitividade, um ponto é capaz de acessar outros pontos através da rede semântica formada pelos mapeamentos ponto-a-ponto. Esta técnica é geralmente implementada na topologia pura.

b. Mapeamentos com Ponto de Mediação *(Peer-mediated Mappings)*: nesta abordagem, um ponto pode definir um mapeamento que relaciona um ou mais pontos, atuando como mediador destes.

c. Mapeamentos com Super-Ponto de Mediação *(Super-Peer mediated Mappings)*: em sistemas com super-ponto, esquemas de mediação (globais) podem ser definidos no nível do super-ponto, ou seja, deste nível para o nível dos pontos.

5.4.1 Conceitos Fundamentais

Para a expressão dos mapeamentos, é necessário adotar uma linguagem de definição. Considerando o modelo relacional (para fins didáticos), linguagens de expressão de mapeamentos são normalmente baseadas no conceito de consultas conjuntivas, por isso introduzimos alguns conceitos fundamentais relacionados a este formalismo, utilizando como base o trabalho de Ulman [Ulman].

Deste modo, ao considerarmos a regra **p(X,Z) :- a(X,Y) & a(Y,Z),** estamos nos referindo a **a** – um predicado EDB (*Extensional Database* ou tabela armazenada) e a **p** – um predicado IDB (*Intensional Database* ou predicado cuja relação é construída a partir de regras). Assumindo que **a** se refere a um arco (ou grafo), a interpretação correta para a regra acima descrita é "**p(X,Z)** é verdadeiro se existe um arco do nó **X** para o nó **Y** e também um arco de **Y** para **Z**", em outras palavras, **p** representa um caminho entre **X** e **Z**.

A regra é formada por um átomo – o cabeçalho (*head*), à esquerda do se (:-) e por zero ou mais átomos chamados de sub-objetivos (*subgoals*), à direita do se. O cabeçalho sempre deve possuir um predicado IDB, enquanto que os sub-objetivos podem ter tanto predicados IDB quanto predicados EDB.

Uma consulta conjuntiva **CQ** é, então, uma regra com sub-objetivos que são assumidos ter predicados EDB. Uma **CQ** é aplicada a relações EDB através da substituição de valores em todas as variáveis do corpo da regra. Se uma substituição torna todos os sub-objetivos verdadeiros, então a mesma substituição aplicada à cabeça é um fato inferido sobre o predicado da mesma.

Assumindo dados no modelo relacional, visões como consultas e estas no formato de consultas conjuntivas, usamos a notação seguinte como padrão para consultas:

$$Q(\bar{X}) :- p_1(\bar{X}_1), ..., p_n(\bar{X}_n)$$

Onde $\bar{X}$, $\bar{X}_1$, ..., $\bar{X}_n$, são variáveis tupla e $\bar{X} \subseteq \bar{X}_1 \cup \ldots \cup \bar{X}_n$. Os átomos $p_1(\bar{X}_1)$, ..., $p_n(\bar{X}_n)$ são chamados os sub-objetivos do corpo da consulta e $Q(\bar{X})$, o cabeçalho da consulta, como vimos acima. Neste sentido, dada uma consulta **Q** e uma instância de banco de dados **D**, **Q(D)** denota o resultado da avaliação de **Q** sobre **D**. Complementando estas definições, consideramos um esquema de um ponto **A** como $\mathbf{R_A}$ e as relações pertencentes a este esquema como $\mathbf{a}$, $\mathbf{a_1}$, ..., $\mathbf{a_n}$.

Por exemplo, uma consulta a uma relação professor que filtre os atributos nome e telefone seria escrita assim: *Q(nome,telefone) :- professor(nome, telefone)*. Em uma consulta a duas relações professor e endereço, filtrando os atributos nome e cidade e supondo a necessidade de uma junção entre ambas as relações, teríamos: *Q(nome, cidade) :- professor(nome), endereço(nome, cidade)*.

5.4.2 Formalizando Mapeamentos

Diferentes abordagens para definição de mapeamentos vêm sendo estudadas na literatura: *Global-as-View* (GAV) [Friedman et al. 1999], *Local-as-View* (LAV) [Lenzerini 2002], *Global-Local-as-View* (GLAV) [Madhavan, Halevy 2003] e *Both-as-View* (BAV) [McBrien, Poulovassilis 2003a]. Na abordagem GAV, um esquema destino é descrito como um conjunto de visões sobre esquemas fontes e na LAV, fontes são descritas como visões sobre um esquema destino. Estas duas abordagens são consideradas básicas e as demais são formalizações estendidas a partir destas. Com o intuito de compreender a semântica de um mapeamento, utilizaremos a abordagem GLAV para formalizar sua definição e apresentar conceitos relacionados.

No formalismo GLAV, um mapeamento semântico entre dois pontos A e B é especificado por um conjunto de fórmulas de mapeamentos, cada uma da forma $\mathbf{Q_A}(\bar{X}) \subseteq \mathbf{Q_B}(\bar{X})$, onde $\mathbf{Q_A}$ e $\mathbf{Q_B}$ são consultas conjuntivas sobre $\mathbf{R_A}$ e $\mathbf{R_B}$, respectivamente [Madhavan, Halevy 2003]. Denota-se, portanto, um mapeamento entre **A** e **B** como $\mathbf{M_{A \to B}}$.

Neste momento, é importante definir o conceito de *containment* e equivalência entre consultas: Uma consulta $\mathbf{Q_A}$ está contida em uma consulta $\mathbf{Q_B}$, denotada por $\mathbf{Q_A} \subseteq \mathbf{Q_B}$ se, para todas as instâncias do banco de dados **D**, o conjunto de tuplas retornado pela avaliação de $\mathbf{Q_A}$ sobre **D**, denotado por $\mathbf{Q_A(D)}$ é um subconjunto de $\mathbf{Q_B(D)}$. Duas consultas $\mathbf{Q_A}$ e $\mathbf{Q_B}$ são equivalentes se e somente se $\mathbf{Q_A} \subseteq \mathbf{Q_B}$ e $\mathbf{Q_B} \subseteq \mathbf{Q_A}$.

Dizemos, então, que uma instância de banco de dados $\mathbf{D_B}$ de $\mathbf{R_B}$ é consistente com um banco de dados $\mathbf{D_A}$ de $\mathbf{R_A}$ com respeito a uma fórmula de mapeamento $\mathbf{Q_A}(\bar{X}) \subseteq \mathbf{Q_B}(\bar{X})$, se o *containment* é verdadeiro quando $\mathbf{Q_A}$ e $\mathbf{Q_B}$ são avaliadas sobre $\mathbf{D_A}$ e $\mathbf{D_B}$, respectivamente. Portanto, uma instância de banco de dados $\mathbf{D_A}$ define um conjunto de instâncias $\mathbf{D_B}$ que são consistentes com $\mathbf{D_A}$ com respeito a toda fórmula de mapeamento em $\mathbf{M_{A \to B}}$.

As definições de *containment* e equivalência ajudam a compreender também o grau de precisão que um mapeamento tem, ou seja, quão completo é o conjunto de respostas obtido na resolução da consulta. Como resultado, mapeamentos podem ser classificados em três categorias. Assim, voltando a considerar um esquema $\mathbf{R_A}$ e um esquema $\mathbf{R_B}$, teremos :

- Se tanto $\mathbf{Q_A \subseteq Q_B}$ e $\mathbf{Q_B \subseteq Q_A}$, então $\mathbf{Q_A \equiv Q_B}$ e o mapeamento é denominado **exato** (*exact*). Este tipo de mapeamento provê exatamente os dados especificados na visão (consulta) associada.
- Se $\mathbf{Q_A \subseteq Q_B}$, o mapeamento é dito **reduzido** (*sound*). Neste caso, o mapeamento provê um subconjunto dos dados especificados na visão (consulta) associada.
- Se $\mathbf{Q_B \subseteq Q_A}$, o mapeamento é dito **completo** (*complete*). No mapeamento completo, obtém-se um super-conjunto dos dados especificados na visão (consulta) associada.

O problema de encontrar todas as respostas para uma consulta está relacionado também à noção de respostas corretas (*certain answers*) [Halevy 2000]. Esta definição diferencia os casos onde a extensão dos dados é tida como completa de casos onde ela é parcial. Então, dada uma consulta **Q** sobre $\mathbf{R_B}$, uma tupla $\mathbf{\bar{t}}$ é uma resposta correta para **Q** com respeito ao mapeamento $\mathbf{M_{A \rightarrow B}}$ se $\mathbf{t \in Q(D)}$ para cada $\mathbf{D \in D_B}$.

Finalmente, dado um mapeamento semântico e uma instância de $\mathbf{R_B}$, uma consulta **Q** vai ser respondida pela computação da reescrita maximamente-contida (*maximally-contained rewriting*) de **Q** sobre $\mathbf{R_B}$. A reescrita maximamente-contida é uma consulta $\mathbf{Q'}$ sobre $\mathbf{R_B}$ tal que $\mathbf{Q'(D_B)}$ é garantida ser o conjunto de todas as respostas corretas para **Q** seja qual for a instância $\mathbf{D_B}$.

Os formalismos GAV e LAV são casos especiais de GLAV [Madhavan, Halevy 2003]. GAV é obtido quando $\mathbf{Q_B}$ é um único átomo e não tem projeções e LAV é obtido quando essa situação se aplica a $\mathbf{Q_A}$. Como estes formalismos são básicos para a implementação de qualquer abordagem, veremos os dois em maior detalhe.

5.4.2.1 Global-as-View (GAV)

Nesta abordagem, elementos do esquema **B** são descritos em termos de consultas sobre esquemas de fontes $\mathbf{A_1, A_2, ... A_N}$. Desta maneira, para cada relação **B** em $\mathbf{R_B}$, existe um ou mais mapeamentos da forma $\mathbf{B \subseteq Q_A}$, ou $\mathbf{B(x) :- Q_A(x,y)}$, onde **y** são variáveis existenciais e $\mathbf{Q_A}$ uma consulta conjuntiva [Karvounarakis 2005]. Essencialmente, estas regras com **B** no cabeçalho determinam completamente o conteúdo de **B** com respeito à instância **I** de **A**. Então, para a reformulação da consulta, apenas é necessário substituir cada átomo relacional na consulta por sua definição em termos do esquema **A**.

Exemplo: Considere uma aplicação onde um ponto **A** tem informações sobre pesquisadores e os projetos realizados por eles: **A = {Proj_Membro(nome,projeto)}**. Em outro ponto, um esquema **B** possui pares de pesquisadores que trabalham em projetos comuns**: B = {MesmoProjeto(nome1, nome2, projeto)}**. O mapeamento GAV entre esquemas dos pontos **A** e **B** é definido da seguinte forma:

MesmoProjeto(x,y,p) :- Proj_Membro(x,p), Proj_Membro(y,p)

Este mapeamento expressa que todos os pares de pesquisadores cujos nomes apareçam relacionados ao mesmo projeto na relação **Proj_Membro** também aparecerão na mesma tupla de **MesmoProjeto**.

Considerando GAV em sistemas de integração que usam esquema global, para cada relação **R** do esquema global, temos associada uma consulta **Q** que obtém **R** a partir das fontes locais [Costa 2005]. Considerando **B** como o esquema global e $\mathbf{A_1}$, $\mathbf{A_2}$,..., $\mathbf{A_N}$ o conjunto de fontes de dados, um elemento de um esquema **B** está associado a uma visão sobre elementos de esquemas $\mathbf{A_1}$, $\mathbf{A_2}$, ..., $\mathbf{A_N}$. Ou seja, a semântica GAV é a mesma que a empregada em sistemas como PDMS, onde o mapeamento acontece geralmente entre dois pontos.

A reformulação de consultas em GAV é um processo relativamente direto. Uma vez que as relações no esquema destino **B** estão definidas em termos das relações do esquema fonte **A**, tudo o que se tem a fazer é decompor as definições das relações do esquema **B**. Assim, quando uma consulta é submetida em **B**, deve-se efetuar os seguintes passos:

a) Identificar quais relações estão sendo consultadas

b) Descobrir as definições dos mapeamentos (relação-> visões)

c) Reescrever e submeter a consulta para **A**

Exemplo: Considerando o exemplo anterior sobre membros de projetos, suponhamos as seguintes instâncias armazenadas no ponto de dados A:

a1 = {Proj_Membro('João Nunes', 'Bacias Hidrográficas'}.

a2 = {Proj_Membro('Zacarias Petruci', 'Bacias Hidrográficas'}.

a3 = {Proj_Membro('Maria Gomes', 'E-learning'}.

a4 = {Proj_Membro('Plinio Silva', 'E-learning'}.

Uma consulta **Q1= MesmoProjeto(x,y,p)** é submetida no ponto **B**. Esta consulta **Q1** vai ser reformulada sobre **A** de acordo com o mapeamento: **MesmoProjeto(x,y,p) :- Proj_Membro(x,p), Proj_Membro(y,p).**

O resultado de **Q1** será:

MesmoProjeto('João Nunes', 'Zacarias Petruci', 'Bacias Hidrográficas')

MesmoProjeto('Maria Gomes', 'Plinio Silva', 'E-learning')

Uma outra consulta **Q2** sobre **B** poderia ser: **Q2 = MesmoProjeto(x,y,'Bacias Hidrográficas').** Novamente, com base no mapeamento, o resultado será:

MesmoProjeto('João Nunes', 'Zacarias Petruci', 'Bacias Hidrográficas')

Em resumo, considerando um mapeamento GAV $\mathbf{M_{A \rightarrow B}}$, onde A e B são esquemas de pontos de dados (ou B é o esquema global e A um esquema de fonte, em sistemas de integração), podemos elencar algumas de suas propriedades (adaptado de [Costa 2005] e [Ka 2005]):

- Para toda relação **b** em **B**, um conjunto de relações de **A** é associado. Apenas dados provenientes das relações de **A** serão usadas para retornar instâncias da relação **b**.
- As relações de **B** podem ter mais instâncias do que as recuperadas de um ponto **A**.
- Mapeamentos GAV possuem grau de precisão exato ou reduzido.

O formalismo GAV é provavelmente o mais largamente utilizado e pesquisado em sistemas de integração que adotam esquema global. Exemplos de sistemas que fazem uso desta abordagem são o TSIMMIS [Chawathe et al. 1994], o SEWASIE [Bergamaschi et al. 2005] e o Integra [Lóscio 2003].

5.4.2.2 Local-as-View (LAV)

Nesta técnica, ocorre o inverso do enfoque GAV, isto é, elementos do esquema **A** (fonte) são descritos em termos de consultas sobre o esquema **B**. Neste sentido, para cada relação **A** em $\mathbf{R_A}$, existe um ou mais mapeamentos da forma $\mathbf{A \subseteq Q_B}$, onde $\mathbf{Q_B}$ é uma consulta conjuntiva. Em sistemas que usam esquema global, um mapeamento LAV é estabelecido também no sentido oposto ao GAV – das fontes para o esquema global. Isto significa que esquemas exportados das fontes são descritos como visões sobre classes virtuais do esquema global [Costa 2005].

Exemplo. Suponha que um esquema de ponto **B** possua uma relação com autores e seus artigos: **B = {Autor(nome, artigo)}**. Em outro ponto **A**, uma fonte apenas contém alguns (mas não necessariamente todos) dos pares de pesquisadores que são co-autores de artigos: **A = {CoAutor(nome1, nome2)}**. Um mapeamento LAV pode ser expresso da seguinte forma:

CoAutor(x,y) ⊆ Autor(x,p), Autor(y,p)

Ou seja, a visão da fonte A computa a junção das relações Autor(x,p) com Autor(y,p), do ponto B, de modo a obter os co-autores de um determinado artigo.

A reformulação da consulta a partir de mapeamentos LAV é bem mais complexa do que na abordagem GAV, pois os átomos que ocorrem na consulta aparecem no corpo dos mapeamentos (em vez de estarem no cabeçalho), dentro de uma conjunção de átomos que também contêm variáveis quantificadas existencialmente. Como resultado, estas regras não podem ser usadas diretamente, como no caso de mapeamentos GAV [Karvounarakis 2005]. Além disso, respostas maximamente contidas podem não ser garantidas.

Em sistemas que têm esquema global, o problema pode ser informalmente descrito assim [Costa 2005]: supondo uma consulta **Q** sobre o esquema global e um conjunto de visões $\mathbf{V_1, V_2, ..., V_N}$ sobre o mesmo esquema descritas nas fontes, é possível responder a **Q** usando apenas as respostas das visões $\mathbf{V_1, V_2, ..., V_N}$? Para compreender melhor esta situação, vejamos o seguinte exemplo:

Supondo que o esquema do ponto B do exemplo acima seja agora definido assim: **B = {Autor(nome, artigo, país)}** e o esquema do ponto A permaneça da mesma maneira: **A =**

{CoAutor(nome1, nome2)}. Com o mapeamento LAV: **CoAutor(x,y) ⊆ Autor(x,p), Autor(y,p),** é possível responder completamente a consulta **Q** sobre **B = Autor(nome, artigo, 'Brasil')**?

Como a restrição pais = 'Brasil' não é repassada para a visão estabelecida, os dados computados por esta visão serão todos aqueles daquela fonte, independentemente da condição definida, ou seja, todos os co-autores daquela fonte (A) serão retornados.

Consequentemente, no contexto de integração de dados, usando LAV, garante-se que respostas possíveis serão encontradas. Entretanto, restrições sobre o esquema destino (ou global) podem não ser repassadas às visões. Diante desta dificuldade, métodos vêm sendo propostos com o intuito de garantir reformulações maximamente contidas, como é o caso da técnica de regras inversas (*inverse rules*) [Duschka, Genesereth 1997]. Na técnica de regras inversas, acontece o seguinte:

- Para cada regra **r** e para cada conjunção no corpo de **r**, uma nova regra **r′** é criada, cujo cabeçalho é esta conjunção e cujo corpo é a cabeça de **r**.
- Como as conjunções de uma regra podem conter variáveis existencialmente quantificáveis que não podem aparecer no cabeçalho da regra, é necessário aplicar uma função de *skolem* antes da inversão. Para isso, troca-se todas as variáveis existenciais no corpo da regra por funções cujos parâmetros são as variáveis que aparecem no cabeçalho da regra.

De acordo com o exposto, podemos identificar alguma propriedades para os mapeamentos LAV [Lenzerine 2002]:

- Um mapeamento LAV $\mathbf{M_{A \rightarrow B}}$ não provê informação direta sobre os dados que satisfaçam plenamente a consulta sobre o esquema B;
- Para resolver uma consulta Q sobre B, é necessário inferir como usar o mapeamento $\mathbf{M_{A \rightarrow B}}$ de forma a obter dados de A. A tarefa de resolver consultas como um processo de inferência é similar a responder consultas com informação incompleta (*incomplete information*).
- Mapeamentos LAV possuem geralmente grau de precisão reduzido.

Exemplos de sistemas que usam LAV são o Nimble [Nimble 2006] e o XPeer [Bellahsène et al. 2004].

5.4.2.3 GAV x LAV

As estratégias GAV e LAV são consideradas fundamentais, pois sobre elas outras vêm sendo desenvolvidas. A decomposição da consulta na abordagem GAV se torna bem mais simples, pois as definições já existem no esquema onde está sendo submetida a consulta. A LAV, por sua vez, tem um maior grau de escalabilidade, pois é possível adicionar e remover novas fontes sem necessidade de alteração do esquema destino (ou global), entretanto a tarefa de decompor a consulta se torna bem mais complexa e pode envolver a necessidade de algoritmos extras como o de regras inversas.

5.4.2.4 Outras Estratégias

Além das estratégias GAV e LAV outras vêm sendo pesquisadas. A estratégia GLAV, como apresentada na formalização de mapeamentos, é uma generalização das estratégias anteriores, pois adota tanto GAV quanto LAV nas suas definições, combinando o poder de expressão de cada uma delas [Madhavan, Halevy 2003]. Nesta abordagem, problemas de restrições de direção nas associações entre os esquemas são removidos.

Neste sentido, um mapeamento GLAV é especificado da forma: $\mathbf{Q_A}(\bar{\mathbf{X}}) \subseteq \mathbf{Q_B}(\bar{\mathbf{X}})$, onde $\mathbf{Q_A}$ e $\mathbf{Q_B}$ são consultas conjuntivas. A resolução de consultas através de GLAV pode ser realizada pela redução à GAV ou LAV, dependendo da direção da reformulação.

As vantagens das estratégias LAV e GAV são complementadas na abordagem GLAV: enquanto LAV facilita a associação de várias fontes de dados a um ponto destino, GAV permite expressar junções sobre atributos que possam não estar presentes especificamente em uma fonte [Madhavan, Halevy 2003]. Como resultado, a estratégia GLAV é considerada por muitos autores a mais útil [Madhavan, Halevy 2003], tendo sido adotada em vários PDMS como o Piazza [Tatarinov et al. 2003] e o Humboldt Discoverer [Herschel, Heese 2005].

A estratégia Both-as-View (BAV) [McBrien, Poulovassilis 2003a], por sua vez, foi desenvolvida a partir do formalismo HDM (*Hipergraph Data Model)* [Poulovassilis, McBrien] e dos formalismos GAV e LAV. Nesta abordagem, esquemas são mapeados um ao outro usando uma seqüência de transformações chamadas *pathway* (caminhos). Estes caminhos são reversíveis, isto é, se existe um caminho $\mathbf{S_X} \rightarrow \mathbf{S_Y}$ de um esquema $\mathbf{X}$ para um esquema $\mathbf{Y}$, pode-se derivar o inverso $\mathbf{S_Y} \rightarrow \mathbf{S_X}$ e vice-versa. Além disso, a partir dos caminhos é possível extrair regras de mapeamento GAV, LAV e GLAV [McBrien, Poulovassilis 2006]. Exemplo de aplicação que faz uso do formalismo BAV é o Projeto AutoMed [Boyd et al. 2004]. O sistema iXPeer também utiliza este formalismo na extensão do PDMS XPeer.

5.5 Considerações sobre Reformulação de Consulta e Mapeamentos

A definição de uma boa estratégia de mapeamentos entre esquemas num sistema PDMS é vital para o bom funcionamento do mesmo. O conjunto de mapeamentos define a semântica do sistema e sua qualidade terá forte influência na qualidade dos resultados da consulta. Desta maneira, quando um usuário de um ponto submete uma consulta sobre um esquema, o PDMS expande recursivamente qualquer mapeamento relevante para a consulta em questão, recuperando dados dos pontos que podem contribuir, de acordo com uma das estratégias descritas acima.

A expansão dos mapeamentos ocorre através da formação de caminhos de mapeamentos semânticos, de acordo com as propriedades de transitividade, reversibilidade e composição. Entretanto, diferentes caminhos entre pares de pontos podem produzir diferentes conjuntos de respostas e assim, normalmente, um conjunto máximo de respostas somente é obtido através da varredura recursiva por todos os caminhos.

Encadear caminhos recursivamente se torna uma tarefa cara e, se não houver um tempo limite de varredura, o desempenho da consulta pode cair. Além disso, seguir por todos os caminhos leva a ineficiências [Tatarinov, Halevy 2004] como:

- O algoritmo pode seguir por muitos caminhos que poderiam ser desfeitos logo cedo.
- O algoritmo segue muitos caminhos que resultam em reformulações redundantes, ou seja, consultas desnecessárias.

Além disso, outro problema que deve ser tratado diz respeito à possibilidade de pontos poderem entrar e sair a qualquer instante. Portanto, o sistema deve ser capaz de verificar isso e usar os mapeamentos disponíveis naquele momento.

Pesquisas vêm sendo realizadas com o propósito de otimizar a reformulação de consultas num PDMS, através de técnicas como [Tatarinov, Halevy 2004]: poda e minimização (*pruning and minimization)* onde se busca evitar a execução de uma consulta de forma redundante, ou seja, que retorne um subconjunto de resultados de uma consulta executada anteriormente; e *pré-computação de caminhos semânticos,* onde os mapeamentos compostos podem ser pré-otimizados para remover redundâncias e, se a composição de mapeamentos resulta em uma reformulação vazia, então alguns caminhos já serão descartados.

A utilização de informação semântica também vem sendo estudada como forma de otimizar semanticamente a reformulação de consultas. O uso de conhecimento semântico em suas várias formas, incluindo meta-modelos, regras semânticas e restrições de integridade, pode otimizar as capacidades de transformação de consultas do usuário em outras semanticamente equivalentes que possam ser respondidas em menos tempo e/ou com menos recursos [Necib, Freytag 2005]. Este aspecto tem sido denominado de otimização semântica de consultas ou *semantic query optimization* [Necib, Freytag 2005].

Como forma de ilustração, consideremos o trabalho de Necib e Freitag [Necib, Freytag 2005], onde a reformulação de consulta é definida da seguinte maneira: dada uma fonte de dados **FD**, uma ontologia **O** e uma consulta de usuário **Q**, busca-se encontrar uma consulta reformulada **Q′** de **Q** usando **O** e um conjunto de regras semânticas de modo que **Q′** retorne respostas mais significativas ao usuário do que a consulta original **Q**.

Nesta definição, ontologias são o aspecto semântico usado para enriquecer o processo de reformulação. Especificamente falando, ontologias podem auxiliar a gerar consultas semanticamente reformuladas em três pontos básicos [Necib, Freytag 2004]:

- Expandir consultas de usuário através da alteração de condições de seleção usando sinônimos para os termos em uso;
- Substituir condições da consulta com outras condições que são semanticamente equivalentes;
- Reduzir o escopo de consultas, restringindo seu contexto.

Aplicar conceitos semânticos na reformulação de consultas em PDMS constitui ainda um desafio, mas, ao mesmo tempo, pode contribuir para que o processo seja realizado com

mais qualidade e precisão, ajudando inclusive na pré-computação de caminhos semânticos e na escolha de caminhos.

5.6 Considerações

A propagação de uma consulta num PDMS é complexa e desafiante. Em um SGBD distribuído, existe a noção de corretude e completude de resultados de consultas. Isto significa que resultados são corretos, de acordo com a consistência dos dados e completos no sentido de que os nós possuem as respostas e estão disponíveis para a consulta. Nos PDMS, em contrapartida, não existe a noção de esquema global e os pontos têm conhecimento apenas de parte da rede, ou seja, de um subconjunto dos pontos que são seus vizinhos ou que compõem um cluster. Estes fatores levam a resultados muitas vezes aproximados para as consultas.

Um aspecto crucial é a escolha de uma estratégia para definição dos mapeamentos e como estes são utilizados ou compostos para permitir a reformulação e propagação da consulta entre os pontos. Conceitos oriundos de Bancos de Dados como visões, restrições de integridade, relacionamentos, reutilização de resultados de consultas e aspectos semânticos podem auxiliar neste processo.

Este capítulo apresentou o processamento de consultas na ótica de diversas tecnologias de Bancos de Dados e de PDMS. Em especial, procurou detalhar e contextualizar os problemas relacionados à reformulação da consulta, e como a definição de estratégias de mapeamentos pode ajudar neste processo.

Capítulo 6

Proposta de Tese

Este capítulo tem por objetivo apresentar a proposta deste trabalho de acordo com os problemas e desafios descritos nos capítulos anteriores. Para isso, na seção 6.1 descrevemos o sistema SPEED – base deste trabalho e sobre o qual nossa proposta será desenvolvida. Na seção 6.2, mostramos como a semântica, através de metadados, ontologias e contexto, pode enriquecer serviços no SPEED, em especial aqueles relacionados ao gerenciamento de mapeamentos e reformulação de consultas. Na seção 6.3, apresentamos uma visão geral da Linguagem de Mapeamentos Semânticos – LMS. Na seção 6.4, apresentamos um processo geral de reformulação de consulta baseada em semântica – foco deste trabalho. Por fim, discutimos nossas contribuições e descrevemos a metodologia a ser aplicada.

6.1 O Sistema SPEED

O **SPEED** - **S**emantic **PEE**r-to-Peer **D**ata Management System é um sistema gerenciador de dados para ambiente P2P que adota uma topologia mista, empregando os conceitos de super-ponto e DHT [Pires 2007]. A rede DHT é usada para auxiliar pontos que possuem interesses comuns a encontrar um ao outro e formar comunidades semânticas. Dentro de uma comunidade, pontos são organizados numa topologia de super-ponto.

O sistema SPEED vem sendo pesquisado e desenvolvido com o propósito de prover soluções para problemas críticos de gerenciamento de dados em sistemas P2P, como conectividade, mapeamentos, processamento de consultas e qualidade de serviços. Para isso, utiliza semântica como base para o desenvolvimento e gerenciamento de seus serviços. A seguir, a arquitetura do SPEED é apresentada juntamente com seus principais componentes funcionais.

6.1.1 A Arquitetura do SPEED

Como mostrada na Figura 6.1, três tipos diferentes de pontos são considerados no SPEED: pontos de dados, pontos de integração e pontos semânticos. **Pontos de dados** são fontes que desejam compartilhar dados com outros pontos no sistema. Na Figura 6.1, $\mathbf{I_1D_1}$ e $\mathbf{I_1D_2}$ são exemplos de pontos de dados. Pontos de dados são agrupados de acordo com seus interesses (domínio) em clusters semânticos.

O conceito de "interesse" é essencial dentro da arquitetura do SPEED. O "interesse" é na verdade a representação semântica dos dados exportados por um ponto de dados, ou seja, o que ele deseja disponibilizar para os demais pontos. Assim, a semântica está associada a este interesse.

Cada cluster semântico está associado a um interesse comum entre pontos de dados e possui um **ponto de integração** que é responsável por tarefas como indexação de metadados, processamento de consultas e integração dos dados. Pontos de integração são pontos de dados com alta disponibilidade e poder computacional. Por exemplo, o ponto $\mathbf{I_1}$ é um ponto de integração do cluster semântico composto pelos pontos de dados $\mathbf{I_1D_1}$, $\mathbf{I_1D_2}$ e $\mathbf{I_1D_N}$.

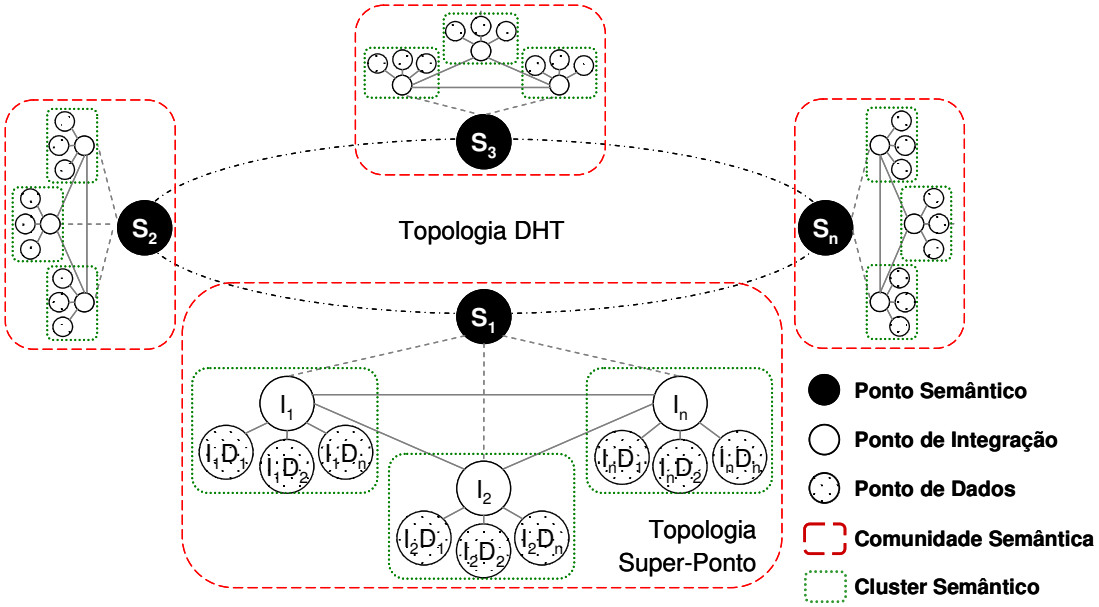

Fig. 6.1 Arquitetura Geral do Sistema SPEED

Pontos de integração se comunicam com um **ponto semântico** que é responsável por armazenar e oferecer uma ontologia padrão de domínio específico como, por exemplo, educação, saúde, geografia. Desta maneira, uma comunidade semântica é composta de clusters que possuem interesses e semânticas comuns. Quando um ponto de dados entra no sistema, o ponto semântico identifica um cluster apropriado onde o ponto deve ser conectado. Na Figura 6.1, $\mathbf{S_1}$ é um exemplo de ponto semântico.

Nesta abordagem, um ponto de integração nomeia seus respectivos clusters semânticos, enquanto que um ponto semântico nomeia suas comunidades semânticas correspondentes. Assim, de acordo com a Figura 6.1, temos o cluster semântico $\mathbf{I_1}$ (a partir do ponto de integração homônimo) e a comunidade semântica $\mathbf{S_1}$ (a partir do ponto semântico homônimo). Deste modo, neste exemplo, $\mathbf{S_1}$ é composto pelos clusters semânticos $\mathbf{I_1}$, $\mathbf{I_2}$ e $\mathbf{I_N}$.

Como visto na Figura 6.1, a arquitetura do SPEED é considerada mista, pois faz uso de DHT e, ao mesmo tempo, emprega o conceito de super-ponto. A rede DHT é, na verdade, composta pelos pontos semânticos, ou seja, pontos confiáveis, com boa largura de banda e que permanecem na rede por longos períodos de tempo. A função da DHT é ajudar pontos com interesses comuns a encontrarem um ao outro e facilitar a formação das comunidades semânticas.

Como a topologia de super-ponto é também empregada, existe uma maior exploração da heterogeneidade dos pontos participantes. O conceito de cluster aliado ao super-ponto responsável por ele quebra a rede em porções mais fáceis de gerenciar. Algumas tarefas como indexação de metadados, processamento de consultas e integração de dados que demandam alto poder computacional e conhecimento dos pontos envolvidos são realizadas pelos pontos de integração. Clusters, por sua vez, provêem um ambiente mais adequado às técnicas de associação dos esquemas exportados e, assim como as comunidades, são individualmente mais estáveis do que a rede inteira.

Quando um ponto entra no SPEED, ele é vinculado a uma comunidade e a um cluster compatível com seus interesses, ou seja, com seu esquema exportado. Este esquema exportado é mapeado para uma notação ontológica e mapeamentos são definidos entre as ontologias dos pontos com a ontologia de referência do cluster. Mapeamentos são também definidos entre ontologias dos pontos de integração, de modo que consultas possam ser processadas tanto no nível do cluster quanto no nível da comunidade. Como resultado destas definições, obtém-se qualidade nos mapeamentos, assim como o processamento de consultas se torna mais eficiente, pois o sistema é dividido em espaços menores de busca. Na seção 6.1.3, os aspectos essenciais do SPEED serão discutidos.

6.1.2 Principais Componentes Funcionais

Como visto na seção anterior, um ponto pode assumir diferentes papéis: ponto de dados, ponto de integração ou ponto semântico. Como ilustrados na Figura 6.2, cada um possui peculiaridades e componentes funcionais diferentes, complementando-se de modo a compor um ambiente transparente de consultas a fontes autônomas e distribuídas através dos pontos.

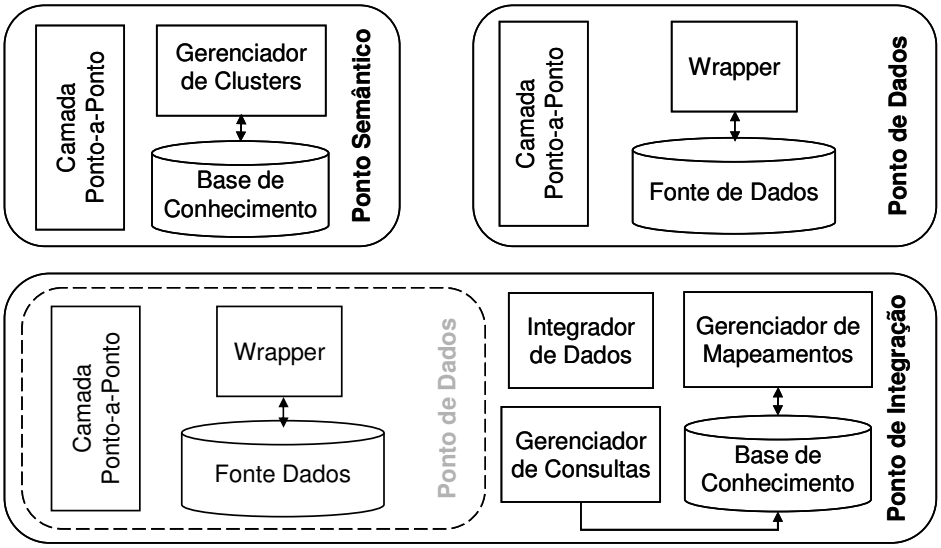

Fig. 6.2 Componentes Funcionais dos Pontos

Os pontos de dados correspondem a uma fonte de dados. Os dados serão compartilhados com os demais pontos através dos mapeamentos semânticos. Cada ponto de dados é composto de:

- Camada P2P: provê a comunicação com um ponto de integração.
- Wrapper: responsável por traduzir dados da fonte para o modelo de dados comum e pela tradução de consultas da linguagem de consulta comum para a linguagem de consulta específica do ponto e vice-versa.
- Fonte de Dados Local: armazena os dados locais a serem compartilhados com os outros pontos de acordo com o esquema exportado.

Um ponto de integração é um ponto de dados especial dentro de um cluster semântico que possui recursos computacionais extras. Este ponto possui um conhecimento detalhado dos pontos associados a ele. O conhecimento utilizado e mantido pelo ponto de integração é armazenado numa Base de Conhecimento. Como todo ponto de integração é também um ponto de dados, percebe-se na Figura 6.2 que ele possui todos os componentes de um ponto de dados e os estende através de serviços mais especializados como gerenciamento de mapeamentos, gerenciamento de consultas e integração de dados. Desta maneira, um ponto de integração acrescenta ou estende os seguintes componentes:

- Camada P2P: permite a comunicação com pontos de dados, pontos de integração dentro da mesma comunidade semântica e com o ponto semântico vinculado a ele.
- Base de Conhecimento (BC): armazena informações úteis para a realização de tarefas como associação de esquemas exportados, processamento de consultas e integração de dados. A Base de Conhecimento inclui informações sobre: (i) a ontologia de referência do cluster; (ii) os pontos vinculados a ele, como o esquema exportado e informações de disponibilidade; (iii) mapeamentos da ontologia de referência do cluster com as ontologias que representam os esquemas exportados dos pontos daquele cluster; (iv) mapeamentos entre a ontologia de referência do cluster com outras pertencentes a outros clusters (ou seja, mapeamentos entre pontos de integração) da mesma comunidade; (v) informações contextuais como, por exemplo, aquelas referentes aos pontos e às consultas em execução.
- Gerenciador de Consultas: processa consultas enviadas ao ponto de integração. Quando uma consulta chega ao ponto de integração, este módulo utiliza o conhecimento presente na **BC** para identificar quais pontos de dados são capazes de responder à consulta. O conjunto retornado de pontos de dados é denominado de pontos de dados relevantes. A consulta é, então, reformulada e enviada a estes pontos relevantes. Em paralelo, o gerenciador de consultas propaga a consulta original para outros pontos de integração dentro da mesma comunidade. Esquemas exportados e mapeamentos semânticos são usados para reformular consultas. A seção 6.3 apresenta maiores detalhes sobre o processo de reformulação de consultas no SPEED.

- Integrador de Dados: integra resultados da consulta retornados dos pontos de dados e dos pontos de integração. Em seguida, envia o resultado final para que o ponto que originou a consulta o apresente ao usuário.
- Gerenciador de Mapeamentos: gera e mantém mapeamentos semânticos entre esquemas exportados (mapeados para ontologias) com a ontologia de referência do cluster e entre ontologias de pontos de integração.

O ponto semântico é na verdade um serviço que deve estar sempre *online*. Na hierarquia de pontos, ele atua como um servidor de ontologias padrões específicas de domínios, como a UMLS para a área médica [UMLS 2007], por exemplo. Estas ontologias de domínio são usadas para enriquecer semanticamente esquemas exportados de pontos de dados e, eficientemente, distribuir pontos de dados dentro de comunidades semânticas e clusters. O ponto semântico também funciona como ponto de entrada para a comunidade correspondente, ou seja, quando um ponto entra no sistema, realiza-se uma comparação entre a ontologia que representa o esquema exportado com a ontologia do domínio de cada comunidade. O resultado que demonstrar maior compatibilidade entre as ontologias define em qual comunidade o ponto estará vinculado. Os componentes de um ponto semântico são:

- Camada P2P: permite a comunicação com pontos de integração que participam do mesmo domínio semântico e com outros pontos semânticos existentes no sistema.
- Base de Conhecimento: armazena informações usadas para manter uma comunidade semântica, incluindo a ontologia padrão de domínio e a descrição dos pontos de integração da comunidade.
- Gerenciador de Clusters: responsável pela conectividade dos pontos que desejam entrar no sistema. Para tal, identifica o cluster apropriado para cada ponto solicitante. Também coordena a auto-organização dos pontos e o balanceamento de carga dos clusters.

6.1.3 Aspectos Essenciais no SPEED

O sistema SPEED vem sendo desenvolvido como um meio de tratar problemas de integração de dados dentro do paradigma de distribuição P2P. Neste sentido, problemas inerentes ao gerenciamento de dados como a entrada e saída dos pontos, a localização de pontos capazes de responder consultas, a reformulação da consulta durante sua propagação e a criação e manutenção dos mapeamentos entre os pontos, são gerenciados. A diferença crucial entre o SPEED e outros PDMS é a forte utilização da semântica em todos os níveis: semântica dos dados, semântica dos pontos, semântica do domínio, semântica da comunidade.

No SPEED, o conteúdo de cada ponto e seu contexto semântico é usado para construir uma estrutura de rede semântica, ou seja, pontos com domínios comuns devem se conhecer e se juntar para que a rede seja eficientemente organizada. Desta maneira, a semântica é inicialmente utilizada:

i. Na definição dos clusters semânticos – conjuntos de pontos associados a um mesmo domínio de conhecimento;

ii. Na organização das comunidades semânticas – conjunto de clusters semânticos associados ao mesmo ponto semântico;

iii. Na utilização de ontologias de domínio;

iv. No enriquecimento das ontologias de referência dos clusters;

v. Nos mapeamentos entre pontos de dados e pontos de integração e entre pontos de integração;

vi. Na reformulação de consultas.

Estes três últimos pontos são tópicos relacionados com este trabalho e esta proposta. O uso de conhecimento semântico em suas várias formas, incluindo meta-modelos, regras semânticas e restrições de integridade, pode otimizar as capacidades de geração de mapeamentos e reformulação de consultas de modo a obter respostas mais completas e mais significativas.

Dentro deste escopo, aspectos essenciais e prioritários a serem tratados no SPEED são:

- A questão da conectividade, ou seja, a entrada e saída de pontos e o que isto acarreta do ponto de vista do gerenciamento de dados. Esta questão envolve também o gerenciamento e manutenção dos clusters semânticos e comunidades semânticas e é tópico de pesquisa que está sendo tratado no trabalho de Doutoramento de Pires [Pires 2007].
- A definição dos mapeamentos entre as ontologias que representam os esquemas exportados e as ontologias de referência dos clusters, assim como entre os pontos de integração. Este tópico é parte essencial deste trabalho, tendo em vista que as consultas serão reformuladas com base nos mapeamentos existentes.
- O índice semântico de consultas que será definido e mantido de forma a prover a identificação dos pontos capazes de respondê-las. Este índice será organizado a partir da ontologia do cluster e também faz parte deste trabalho.
- O processamento de consultas é um dos serviços mais importantes e envolve um longo processo cujas etapas podem ser sintetizadas em submissão da consulta, análise da consulta, identificação dos pontos capazes de respondê-la, reformulação de acordo com mapeamentos, execução, integração e apresentação dos resultados. Este trabalho apresenta um processo geral que envolve tais etapas, mas focaliza nos aspectos relacionados à reformulação da consulta através dos mapeamentos e de forma enriquecida semanticamente.

6.2 Enriquecimento Semântico no SPEED

Ao considerarmos requisitos para a construção de um PDMS, aspectos funcionais como localização dos dados, conectividade, compartilhamento de esquemas exportados, mapeamentos entre pontos e processamento de consultas são prontamente identificados como essenciais. Entretanto, se analisarmos o ponto de vista de um usuário, alguns requisitos não funcionais podem ser percebidos.

O usuário utiliza um PDMS com um objetivo fundamental: obter respostas às suas consultas, de forma transparente e rápida. A princípio ele busca um sistema com fontes diversas, espalhadas e autônomas porque o que ele obtém utilizando apenas sua fonte local pode não ser suficiente. Em outras palavras, ele busca respostas mais completas e significativas para o que ele necessita naquele momento. Ele pode também desejar responder questões muito complexas, que envolvam critérios e condições diferentes e que, para isso, seja necessária a combinação de dados de várias fontes.

No exemplo apresentado no Capítulo 5, cientistas do mundo inteiro desejam compartilhar informações sobre o problema do aquecimento global. A utilização de um PDMS é muito adequada a este tipo de aplicação, pois os conjuntos de dados de cada instituição podem ter interseções ou complementações que propiciem respostas mais amplas para todos os cientistas. Ou seja, um resultado mais completo é obtido.

Por outro lado, este exemplo demonstra também um aspecto fundamental: a busca pelo compartilhamento de informações está sempre relacionada a um domínio de conhecimento em particular, ou seja, a semântica dos dados é o fator chave para determinar com quais pontos (instituições) se vai interagir. Como conseqüência, a semântica do domínio em questão é determinante na criação das comunidades de pesquisa. A semântica de cada ponto, sua disponibilidade e capacidades específicas são também essenciais e, de acordo com a dinamicidade presente em tais ambientes, primordiais na identificação de pontos habilitados a responderem consultas.

Para se obter semântica dos pontos, dos dados, das consultas e do domínio de conhecimento em questão, três conceitos podem ser empregados: metadados, ontologias e informações contextuais. O SPEED utiliza os três conceitos como meio de obter semântica e prover serviços mais relevantes, como veremos a seguir.

6.2.1 Metadados no SPEED

Como descrito no Capítulo 3, metadados são descrições de dados armazenados em fontes de dados e são utilizados para fins de recuperação e manutenção. Os metadados tratam a interoperabilidade em nível de gerenciamento, facilitando a integração de dados. No SPEED, metadados são utilizados para descrever os conteúdos das fontes dos pontos e podem também ser utilizados para descrever resultados parciais de consultas.

Cada ponto de dados está relacionado a uma fonte de dados. O esquema exportado de um ponto de dados é na verdade o conjunto de metadados que descrevem aquela fonte. Com o objetivo de garantir um modelo padrão para gerenciamento e utilização desses metadados, os esquemas exportados serão mapeados para uma notação formal baseada em ontologia.

Assim, quando um ponto entra no sistema, inicialmente o sistema mapeia seu esquema exportado para uma ontologia. Em seguida, verifica em qual comunidade e em qual cluster aquele ponto deve ficar. Considerando o exemplo apresentado no Capítulo 5 sobre o aquecimento global, tomemos como esquemas exportados do pontos UP e UB, as seguintes relações:

Ponto UP: UP.Fenômeno_natural(id, nome, tipo, gravidade, coordenadas)

Ponto UB: UB.Fenômeno_natural(id, nome, tipo, mortes, data_ocorrência)

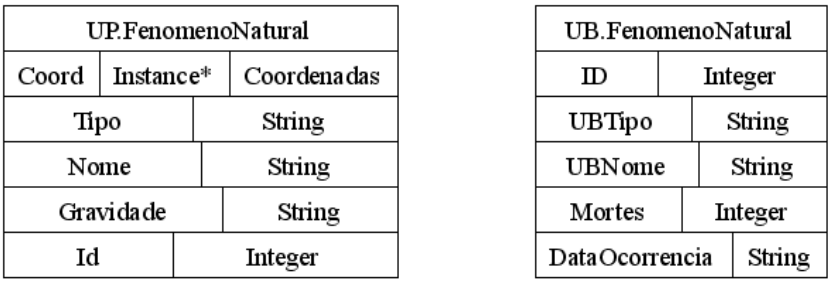

Fig. 6.3 Esquemas Exportados em Notação Ontológica

Utilizando a notação OntoViz (plug-in da ferramenta Protègé)[1], cada esquema exportado é mapeado para as ontologias mostradas na Figura 6.3. Assim, os metadados de cada ponto serão manipulados através destas ontologias.

6.2.2 Ontologias no SPEED

O uso de ontologias tem sido reconhecido como uma abordagem efetiva para promover a interoperabilidade entre fontes distribuídas, na resolução de heterogeneidades de dados no nível sintático e semântico. A utilização de ontologias em PDMS deu origem ao termo OPDMS ou *Ontology-based P2P Data Management Systems*, como descrito no trabalho de Xiao e Cruz [Xiao, Cruz 2006]. Considerando este termo, podemos classificar o SPEED como um OPDMS, pois o conceito de ontologias é fundamentalmente utilizado para a definição e aplicação de semântica em todos os seus serviços. Desta maneira, o SPEED aplica o conceito de ontologias com os seguintes propósitos:

i. Ontologias são utilizadas para descrever os metadados das fontes ligadas aos pontos de dados, como mostrado na seção anterior. Ou seja, ontologias são usadas como representação padrão de metadados conceituais, o que resolve heterogeneidades sintáticas entre pontos diversos, mas heterogeneidades semânticas podem ainda existir.

ii. Ontologias de domínio são utilizadas como referência de termos, com o objetivo de resolver problemas de heterogeneidade semântica. O SPEED usa ontologia de domínio tanto no cluster quanto na comunidade.

iii. Ontologia enquanto técnica de formalização é utilizada para representar informações contextuais a serem manipuladas no sistema.

Percebemos que o conceito de ontologia é aplicado sob a ótica dos três critérios semânticos aplicados ao sistema: metadados, ontologia e contexto. Como metadados foram explicados na seção anterior e contexto será explicado na posterior, iremos nos deter agora nos mecanismos relacionados à criação e evolução das ontologias de referência.

[1] http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz#nid6CS

A comunidade semântica usa uma ontologia padrão disponível para um determinado domínio de conhecimento. Esta ontologia é específica do domínio, mas é abrangente no que diz respeito ao seu vocabulário, incluindo termos e associações que serão utilizados como referência semântica pelos clusters da comunidade. Esta ontologia de domínio também é usada para identificar onde um ponto de dados que está entrando no sistema será vinculado. A ontologia do ponto é comparada às ontologias de domínio das comunidades para este fim. Ela é mantida na base de conhecimento do ponto semântico, como mostra a Figura 6.4.

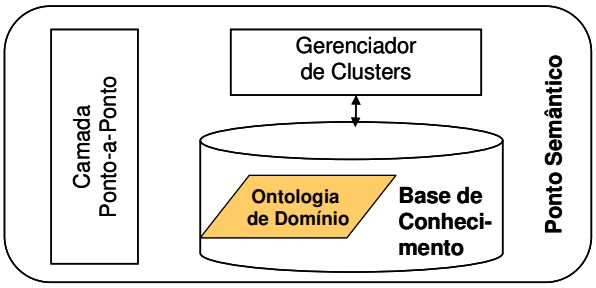

Fig. 6.4 Ontologia de Referência no Nível da Comunidade Semântica

A ontologia de referência do cluster (ORC) é a união de todos os esquemas exportados pelos pontos de dados daquele cluster, ou seja, a união das ontologias que os representam. Consequentemente, ela atua como um vocabulário compartilhado dos termos daquele cluster, inter-relacionando grupos de termos que têm associações semânticas. Ela é mantida na base de conhecimento do ponto de integração, conforme mostra a Figura 6.5.

A idéia é que a ORC seja criada a partir do primeiro ponto de dados que é associado ao cluster e, à medida em que outros pontos vão sendo adicionados ao cluster, a ontologia vai sendo enriquecida com os conceitos oriundos de cada ontologia de ponto, sendo normalizada de acordo com os termos da ontologia de domínio da comunidade. A evolução desta ontologia deve ser tratada, pois além de atuar como referência de conceitos e propriedades, esta ontologia de referência é usada como índice semântico para as consultas.

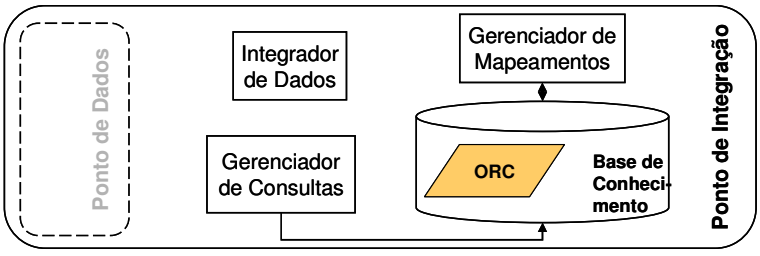

Fig. 6.5 Ontologia de Referência no Nível do Cluster (ORC)

O objetivo do índice semântico de consultas é apontar para os pontos habilitados a responder consultas. Quando uma consulta é submetida, seus termos são normalizados de acordo com a ontologia de referência do cluster e depois os mapeamentos entre esta ontologia

e as ontologias dos pontos são identificados. Como resultado, uma lista de pontos candidatos a responder a consulta é gerada. Estes pontos irão receber a consulta reformulada de acordo com seu próprio esquema exportado e a executarão.

6.2.3 Contexto no SPEED

O contexto pode ser usado de diversas maneiras com o intuito de capturar a semântica relevante de uma entidade e seus relacionamentos, sendo geralmente composto de conceitos, regras, proposições e/ou suposições associados a uma situação específica. Exemplos de informações contextuais dentro do escopo de integração de dados são os metadados [Souza et al. 2006]. No caso do SPEED, o contexto pode ser aplicado ao domínio de conhecimento, às entidades e termos, aos mapeamentos, ao ambiente (diante de sua dinamicidade) e a tarefas como, por exemplo, o processamento de consultas e integração de resultados.

Este trabalho tem como uma de suas metas tornar o SPEED sensível ao contexto, ou seja, o SPEED irá considerar, além das informações explícitas configuradas ou fornecidas pelos usuários, aquelas armazenadas em uma base de conhecimento contextual, aquelas inferidas por meio de raciocínio e/ou ainda aquelas percebidas a partir do ambiente. Com base nas informações contextuais, o SPEED terá condições de enriquecer a criação de mapeamentos e o processamento de consultas, tornando-os processos adaptados ao contexto corrente. Como resultado, respostas de consultas mais significativas e relevantes serão retornadas ao usuário, pois estarão relacionadas com a necessidade corrente do usuário ou com aspectos associados à consulta e/ou mapeamentos.

Como um sistema sensível ao contexto, o SPEED requer que as informações contextuais sejam trocadas e utilizadas por diferentes entidades, como agentes humanos e de software, dispositivos e serviços, de acordo com uma mesma compreensão semântica. Entretanto, alguns aspectos devem ser considerados quando se avalia técnicas para representar contexto [Souza et al. 2006]: o modelo deve ser portável; o modelo deve possuir ferramentas para edição, checagem de tipos e conversão entre diferentes formatos; o modelo deve ser formal o suficiente para facilitar sua definição e permitir reusabilidade e o modelo deve prover mecanismos de raciocínio.

Diante destas premissas, optamos por utilizar ontologia como meio de representar informações contextuais no SPEED. A ontologia de contexto do SPEED é na verdade uma extensão da ontologia de contexto para integração de dados geográficos apresentada em Souza, Salgado e Tedesco [Souza et al. 2006], sendo uma das contribuições específicas deste trabalho. A seguir, apresentamos a ontologia especificada para o SPEED através de seus conceitos fundamentais. Em seguida, como forma de validar os mesmos, apresentamos alguns exemplos de utilização baseados no exemplo motivador da Seção 5.1.

6.2.3.1 A Ontologia de Contexto no SPEED

Nesta seção, apresentamos nossos primeiros passos na construção de uma ontologia para representação de contexto de acordo com o escopo de integração de dados em PDMS e dos requisitos essenciais do SPEED para este trabalho – definição de mapeamentos e reformulação de consultas. Esta ontologia vem sendo desenvolvida em camadas através da

ferramenta Protégè 3.2[2]. A idéia das camadas vem do trabalho [Souza et al. 2006] onde especificamos uma ontologia genérica (*upper*) que pode ser reutilizada em qualquer domínio de aplicação como, por exemplo, Sistemas Colaborativos, Computação Ubíqua, Interação Humano-Computador e Hipermídia Adaptativa. A partir da ontologia genérica, especificamos uma ontologia de integração de dados que compunha o nível intermediário (*middle)* e, finalmente, definimos uma ontologia relativa ao domínio específico, no caso ilustrado pela integração de dados geográficos. Esta ontologia é apresentada no Anexo A.

A ontologia de contexto para o gerenciamento de dados em PDMS também é organizada em camadas e vem sendo desenvolvida como uma extensão da ontologia para integração de dados geográficos. Como a ontologia completa envolve muitos conceitos e associações entre eles, apresentamos uma taxonomia dos conceitos na Figura 6.6, abstraindo, neste primeiro momento, as associações. Por razões de espaço e de legibilidade, mostramos a taxonomia em notação UML[3]. De acordo com a Figura 6.6, a ontologia genérica está codificada em azul, a intermediária em amarelo e a específica em rosa.

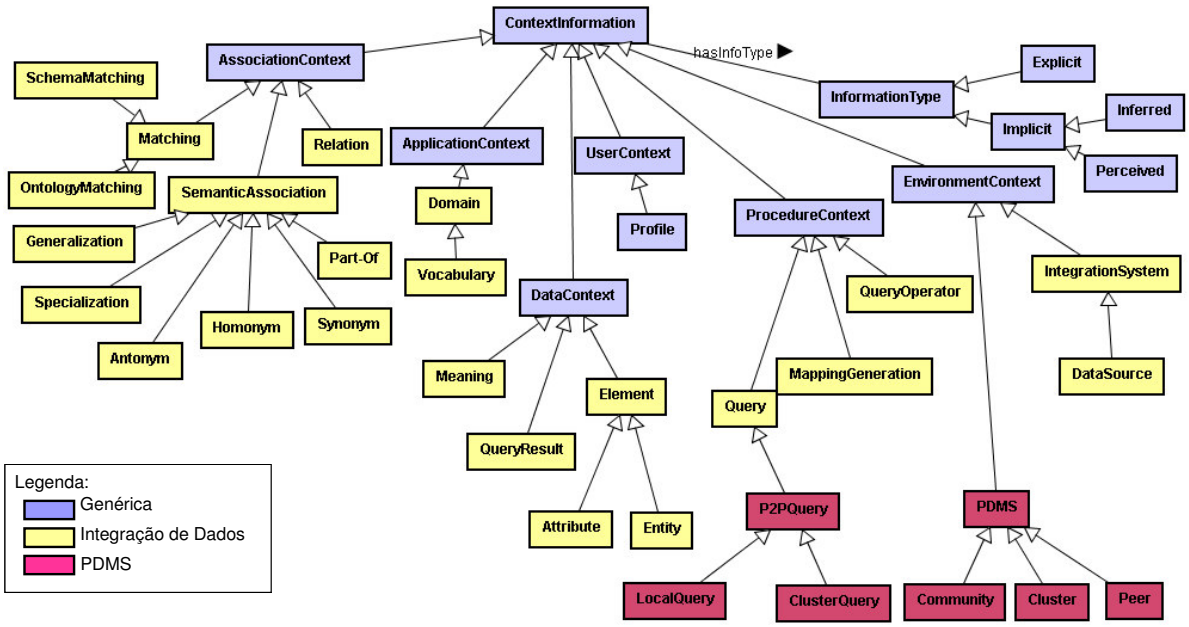

Fig. 6.6 Taxonomia de Informações Contextuais no SPEED

A ontologia genérica foi ampliada com novos conceitos identificados (ambiente, aplicação), mas pode, da mesma maneira, ser reutilizada na especificação de uma ontologia de contexto para qualquer domínio de aplicação. A camada intermediária (em amarelo) contém conceitos próprios do escopo de integração de dados e a camada específica (em rosa) acrescenta aqueles inerentes ao gerenciamento de dados em ambientes PDMS. Em especial,

[2] http://protege.stanford.edu/
[3] http://www.uml.org/

acrescentamos nesta última camada conceitos próprios do SPEED, como os tipos de pontos, tipos de consultas, entre outros. A vantagem de utilizarmos camadas é justamente a flexibilidade de podermos acoplar e desacoplar cada uma delas de acordo com a necessidade do domínio de aplicação em questão.

A ontologia de contexto com todos os seus conceitos e associações é apresentada na Figura 6.7. A seguir, descrevemos cada um dos conceitos da ontologia de contexto, sua semântica e seus principais slots.

Context Information: este é o conceito raiz da ontologia. É classificado em seis sub-conceitos: *UserContext*, *DataContext*, *AssociationContext*, *ProcedureContext*, *ApplicationContext* e *EnvironmentContext*. Juntamente com *Information Type*, estes meta-conceitos compõem a ontologia de contexto genérica.

UserContext: contém informações sobre o usuário, seu perfil (Profile e Preferences), sua identificação (UserID) e localização (UserLocation). Dependendo do tipo de consulta e da interface em uso, o usuário pode definir suas preferências sobre a maneira como uma consulta deve ser apresentada.

DataContext: refere-se a todas as informações contextuais relacionadas aos dados. É classificado em Element, Meaning e Query Result. O conceito Element (Entity ou Attribute) é um dos principais conceitos do escopo de integração de dados, sendo composto dos seguintes slots: elementID, elementName, hasSemanticAssociation, inPeer e hasMeaning (descoberta quando se identifica o termo correspondente da ontologia de referência de domínio).

AssociationContext: está relacionado aos relacionamentos que podem ocorrer na integração de dados, como associações semânticas (SemanticAssociation) e mapeamentos (Matching). Estas informações são exemplos de informações contextuais inferidas, pois são derivadas de acordo com um conjunto de regras e condições. Associações semânticas (semantic associations) representam relacionamentos que acontecem no mundo real e são utilizados para determinar o grau de similaridade que uma entidade (ou atributo) tem em relação à outra (a outro). Mapeamentos (Matching) são o resultado da geração de mapeamentos entre esquemas ou ontologias que representam esquemas, constituindo informações contextuais essenciais na reformulação da consulta.

ProcedureContext: um procedimento é uma coleção de ações ordenadas. A idéia, neste caso, é prover contextualização das etapas realizadas com o intuito de resolver um problema (ou tarefa). Cada etapa é executada sob a ótica de um conjunto de circunstâncias que compõem o contexto da execução e podem prover adaptação. No escopo aqui detalhado, um procedimento (procedure) pode ser o processo de geração de mapeamentos (mapping generation), uma consulta (query) ou um operador de consulta (QueryOperator).

EnvironmentContext: informações contextuais do ambiente estão relacionadas à dinamicidade do mesmo, podendo ser diferentes de acordo com características próprias de cada um. Nesta ontologia, classificamos o ambiente em dois: IntegrationSystem, sistema de integração convencional que faz uso de esquema global e PDMS, neste caso, o SPEED. Vale salientar que dependendo do domínio da aplicação, este ambiente muda.

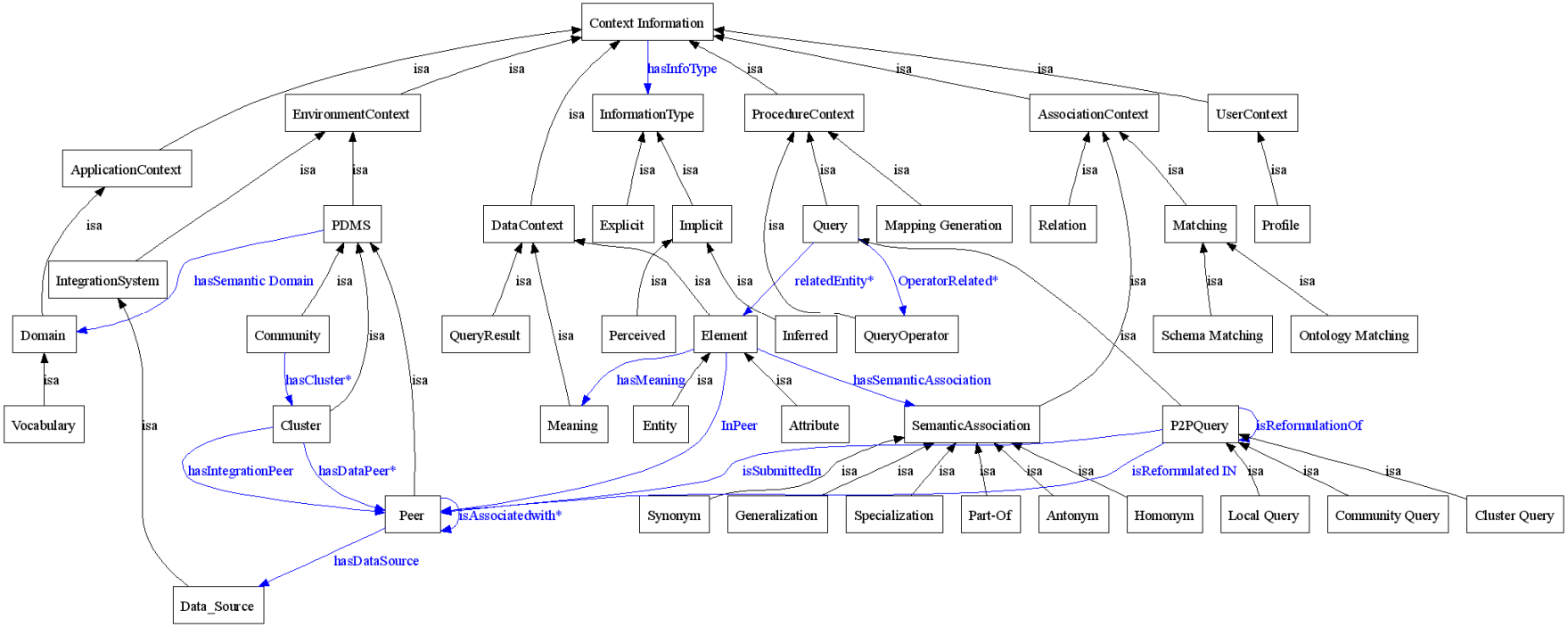

Fig. 6.7 Ontologia de Contexto para o SPEED

Dentro do escopo de PDMS e do SPEED, o contexto do ambiente inclui o contexto dos pontos (*peers*), o contexto dos clusters e o contexto da comunidade. O conceito PDMS possui um *slot* hasSemanticDomain que é herdado pelos três sub-conceitos. O conceito Peer acrescenta os seguintes *slots*: PeerID, PeerName, Role, Available, DataModel, hasDataSource, isAssociatedWith e Localization. O conceito Cluster, por sua vez, acrescenta ClusterID, has IntegrationPeer, hasDataPeer, enquanto que o conceito Community acrescenta CommunityID, hasCluster. Outros *slots* poderão ser incluídos de acordo com a demanda dos serviços que forem sendo contextualizados.

ApplicationContext: o domínio da aplicação implica em características e necessidades diferentes de contexto, passando a ser uma informação essencial a ser tratada. Uma informação de domínio (Domain) é uma informação contextual que, no escopo corrente, está relacionada ao ponto, aos clusters e à comunidade. Cada domínio possui um vocabulário (Vocabulary) próprio que é composto de termos (SetTerm), geralmente presentes em uma ontologia de domínio.

Todas as informações contextuais estão relacionadas a um tipo (Information Type) que pode ser explícito ou implícito, sendo o relacionamento hasInfoType herdado por toda a hierarquia de InformationContext. Uma informação contextual explícita é obtida a partir de fontes estáticas, como um *profile* ou um arquivo de configuração. Uma informação implícita, por sua vez, pode ser percebida a partir da dinamicidade do ambiente ou inferida através de algum processo de raciocínio. Por exemplo, o tipo de associação semântica entre dois elementos será inferido a partir da comparação semântica de ambos e do grau de similaridade apresentado.

Utilizando o exemplo apresentado na Seção 5.1 sobre o aquecimento global, realizamos um estudo de caso sobre a ontologia de contexto. Para isso, instâncias têm sido incluídas com o objetivo de validar os meta-conceitos e conceitos definidos. Como resultado, obtivemos algumas visões da ontologia na forma de conceitos, *slots* e instâncias que serão mostradas a seguir. Todos os exemplos são mostrados na notação OntoViz (Protégé plug-in)[4], onde instâncias são associadas aos seus conceitos através do relacionamento **io** (instance of) e subtipos são associados aos seus supertipos através do relacionamento **isa**.

Na Figura 6.8, apresentamos uma visão parcial dos conceitos Element (Entity), Peer e Meaning, mostrando algumas instâncias e os relacionamentos entre os conceitos e entre as instâncias também. O conceito Entity, por exemplo, possui duas instâncias oriundas de pontos diferentes (P1 e P2): UP.Terramoto e UB.Terremoto. Ambas instâncias são sinônimas e possuem o significado de “Terremoto”.

[4] http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz#nid6CS

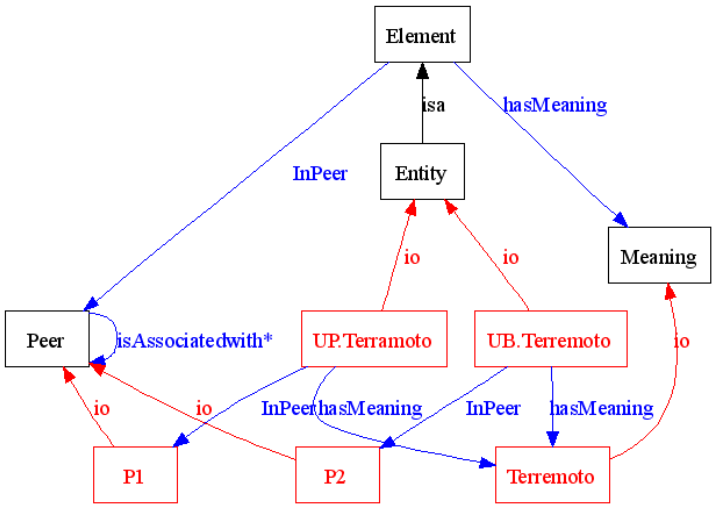

Fig. 6.8 Uma Visão Parcial de Entity, Peer, Meaning e suas Instâncias

Detalhando esta visão, podemos apresentar os *slots* destes conceitos, apresentando seus valores correntes, como mostra a Figura 6.10.

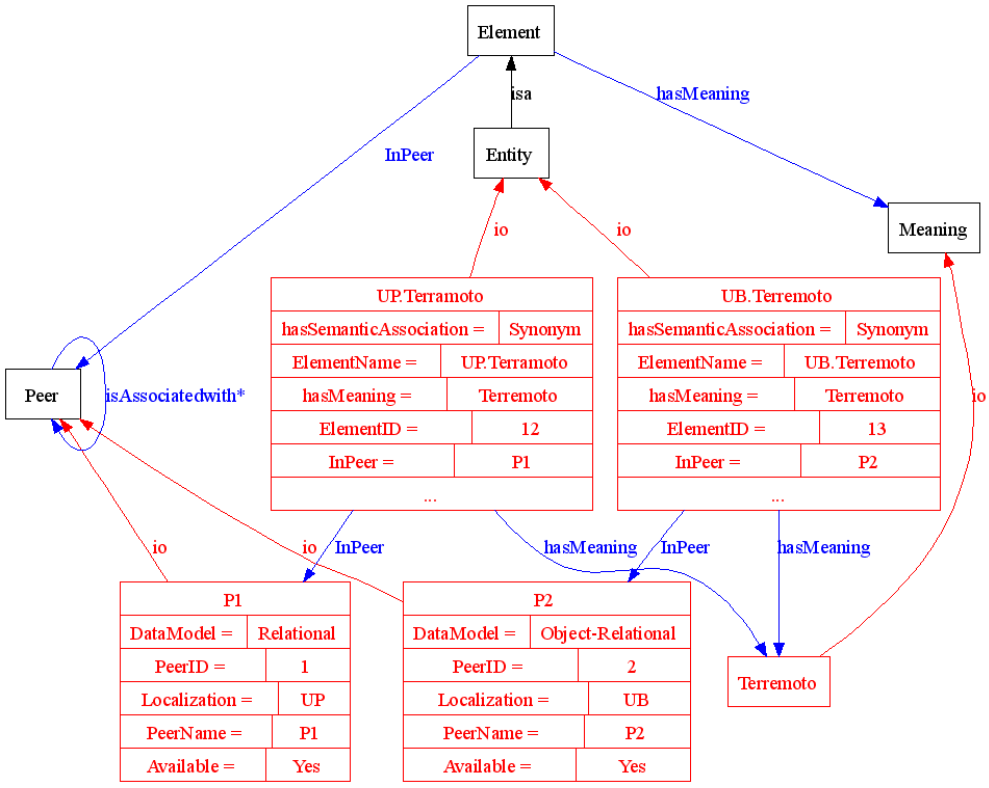

Fig. 6.9 Visão Parcial de Peer, Entity e Meaning acrescentando *slots* e valores correntes.

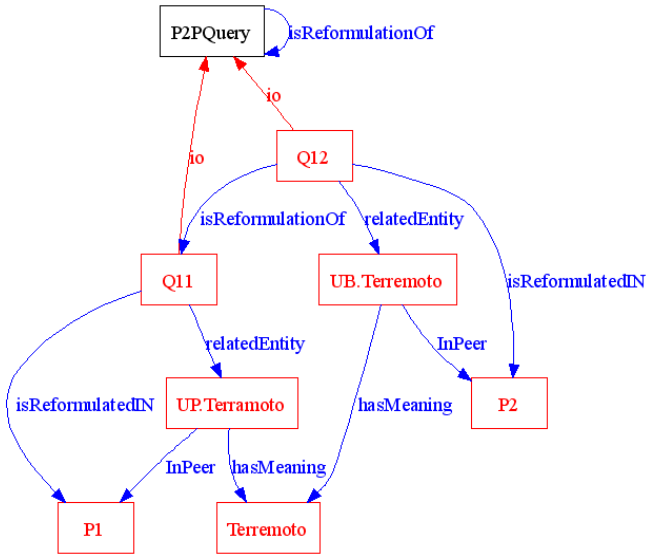

Fig. 6.10 Visão Parcial de P2PQuery com a Reformulação de Q11 para Q12

O próximo exemplo considera uma consulta submetida no ponto P1 que é reformulada para o ponto P2. Ou seja, a Figura 6.11 mostra instâncias do conceito P2PQuery (Q11 e Q12), relacionadas às entidades das consultas, seu termo de referência (significado) e a reformulação de Q11 para Q12.

Inicialmente, esta ontologia vem sendo desenvolvida com o propósito de facilitar a reformulação de consultas no SPEED, mas depois ela pode ser estendida e usada para outras tarefas. A idéia é identificar quais informações pertencentes ao escopo de gerenciamento de dados em ambientes P2P podem ser classificadas como contexto e que tipos de contextos devem ser considerados para otimizar a reformulação da consulta. Através da ontologia de contexto, iremos compor regras de inferência que habilitem a descoberta de contexto implícito a partir de contextos explícitos.

6.3 Definição dos Mapeamentos

No nosso trabalho, ontologias são usadas para uniformemente representar os pontos de dados heterogêneos. Como já explicado, cada cluster vai gerenciar uma ontologia de referência – a ORC que vai ser a união de todos os esquemas exportados daquele cluster. Da ORC para as ontologias dos pontos (OP), vamos definir mapeamentos semânticos que possibilitarão a reformulação das consultas e sua resolução.

Este trabalho irá propor um formalismo para a definição dos mapeamentos semânticos do SPEED, a princípio denominado **LMS – L**inguagem de **M**apeamentos **S**emânticos. A idéia é que a LMS seja expressiva o suficiente para representar a identificação do mapeamento, o relacionamento semântico que ele denota, identificando seu tipo, seu grau de precisão e os elementos das ontologias que são utilizados para sua definição. A LMS vai utilizar como base semântica a meta-ontologia de mapeamentos apresentada na Figura 6.11.

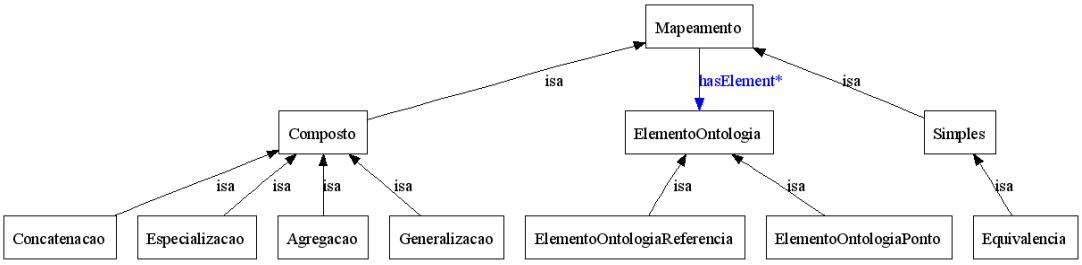

Fig. 6.11 Meta-ontologia de Mapeamentos

Um mapeamento é um relacionamento entre dois elementos provenientes de esquemas ou ontologias diferentes. Ele pode ser basicamente representado da seguinte forma: $\mathbf{E_A}$ **<operador>** $\mathbf{E_B}$ onde $\mathbf{E_A}$ é um elemento (entidade ou atributo) de um esquema **A** e $\mathbf{E_B}$ um elemento de um esquema **B** e **operador** representa um relacionamento semântico entre estes elementos.

Um mapeamento simples é aquele onde os lados do relacionamento são formados de entidades ou atributos simples, enquanto que um mapeamento composto é aquele onde existem grupos de entidades no lado direito do relacionamento semântico.

Um mapeamento simples do tipo *Equivalência*, denotado por $\mathbf{E_{OP}} \equiv \mathbf{E_{ORC}}$, ocorre quando $\mathbf{E_{OP}}$ e $\mathbf{E_{ORC}}$ são semanticamente equivalentes, ou seja, eles descrevem o mesmo conceito do mundo real.

Um mapeamento é do tipo *Agregação* quando existe uma composição de elementos, ou seja, um elemento $\mathbf{E_{ORC}}$ é composto de outros elementos $\mathbf{E_{OP1}}$, $\mathbf{E_{OP2}}$, ..., $\mathbf{E_{OPN}}$. Este mapeamento é denotado como $\mathbf{E_{ORC}} <> \mathbf{E_{OP1}}, \mathbf{E_{OP2}}, ..., \mathbf{E_{OPN}}$.

Quadro 6.1: Exemplos de Mapeamentos

Mapeamentos	Tipo	Elemento da ORC	Elementos das Ontologias dos Pontos
Professor ≡ P1.Prof Professor ≡ P2.Instrutor	Equivalência	Professor	P1.Prof P2.Instrutor
Professor.nome ≡ P1.Prof.nome + P1.Prof.sobrenome	Concatenação	Professor	P1.nome P1.sobrenome
Professor ⊇ P1.ProfVisitante, P1.ProfSubstituto	Generalização	Professor	P1.ProfVisitante P1.ProfSubstituto
P1.ProfSubstituto ⊆ Professor	Especialização	Professor	P1.ProfSubstituto
Computador <> P1.Teclado, P1.Mouse, P1.Gabinete	Agregação	Computador	P1.Teclado P1.Mouse P1.Gabinete

Já um mapeamento é do tipo *Generalização* quando $\mathbf{E_{ORC}} \supseteq \mathbf{E_{OP1}}, \mathbf{E_{OP2}}, ..., \mathbf{E_{OPN}}$, ou seja, um elemento $\mathbf{E_{ORC}}$ é um supertipo de um conjunto de outros elementos (subtipos) $\mathbf{E_{OP1}}$, $\mathbf{E_{OP2}}$, ..., $\mathbf{E_{OPN}}$. Em contrapartida, dizemos que um mapeamento denota *Especialização* se um elemento $\mathbf{E_{OP1}} \subseteq \mathbf{E_{ORC}}$, isto é, $\mathbf{E_{OP1}}$ é um subtipo de $\mathbf{E_{ORC}}$.

Se um elemento $\mathbf{E_{ORC}} \equiv \mathbf{E_{OP11}} + \mathbf{E_{OP12}} + \mathbf{E_{OP13}}$, isto significa que $\mathbf{E_{ORC}}$ é a *Concatenação* dos elementos $\mathbf{E_{OP11}}$, $\mathbf{E_{OP12}}$ e $\mathbf{E_{OP13}}$. O Quadro 6.1 mostra exemplos de mapeamentos semânticos, de acordo com esta classificação.

Desta maneira, um mapeamento será definido como uma 5-upla: ⟨**ID, $\mathbf{E_{OP}}$, $\mathbf{E_{ORC}}$, Type,GP**⟩, onde

- **ID** é um identificador único para o mapeamento;
- $\mathbf{E_{OP}}$ e $\mathbf{E_{ORC}}$ são elementos das ontologias dos pontos e da ontologia de referência do cluster, respectivamente;
- **Type** se refere à classificação semântica do mapeamento: equivalência, especialização, agregação, generalização e concatenação;
- **GP** é o grau de precisão de um mapeamento: se ele é reduzido, completo ou exato. Esta informação está relacionada às instancias das entidades que compõem o mapeamento e pode ser usada para enriquecê-lo.

Cada mapeamento do SPEED vai obedecer a uma formatação deste tipo (que será ainda ampliada e formalizada) e será uma instância da meta-ontologia de mapeamentos acima especificada. Os mapeamentos serão gerados através de um processo de *ontology matching* entre as ontologias ORC e a ontologia do ponto [Shvaiko, Euzenat 2005].

A definição dos mapeamentos possui uma grande relevância no SPEED porque todo o processo de reformulação e execução da consulta depende dos mapeamentos gerados. Da forma como está sendo especificado, o mapeamento está também sendo enriquecido semanticamente, visto que seu relacionamento semântico está sendo incluído e seu grau de precisão também. Estas informações, por exemplo, vão se transformar em informações contextuais e regras serão associadas a elas com o intuito de otimizar a reformulação e a resolução da consulta. Por exemplo, para indicar uma escala de prioridade de envio de consultas reformuladas aos pontos de dados de um cluster, pode-se utilizar o grau de precisão do mapeamento. Assim, uma regra pode ser especificada: SE mapeamento isExato

ENTÃO prioridade = 3

SENÃO SE mapeamento isReduzido

ENTÃO prioridade = 2

SENÃO SE mapeamento isCompleto

ENTÃO prioridade = 1;

Este tipo de raciocínio pode ser utilizado para podar ramos da árvore que representa os nós da consulta, pode evitar caminhos redundantes e pode ser ampliado para realmente prover uma pré-computação de caminhos semânticos.

A definição de mapeamentos irá também considerar questões relacionadas à reversibilidade de mapeamentos, composição e transitividade dos mesmos, de modo a otimizar mais ainda a formação de caminhos semânticos sua utilização.

6.4 Reformulação de Consultas no SPEED

Considerando a arquitetura do SPEED, uma consulta pode ser submetida num ponto de dados ou num ponto de integração. Ao formular uma consulta, em qualquer das duas opções indicadas, ele terá a opção de escolher dois tipos de consulta:

- *Local Query*: na consulta denominada local, apenas os metadados do esquema exportado do ponto são apresentados ao usuário. Ele formula a consulta com base no vocabulário presente nesta visão. Entretanto, a consulta é propagada por todos os clusters capazes de respondê-la, ou seja, por todos os pontos que possuam semântica comum. O usuário receberá como resposta o conjunto de resultados, retornados por todos os pontos, que serão integrados. Muito provavelmente, ele terá mais informações do que se tivesse realizado uma consulta apenas sobre sua fonte local.
- *Cluster Query*: ao escolher a opção de cluster query, o vocabulário presente na ontologia de referência do cluster (ORC) é disponibilizado ao usuário, de modo que ele vai formular sua consulta com base no conjunto de metadados disponíveis naquele cluster. O maior benefício desta opção é que o usuário terá uma visão enriquecida de todos os pontos acoplados àquele cluster, já que a ontologia de referência do cluster inclui o conjunto de termos daquele cluster. Consequentemente, ele mesmo pode formular uma consulta mais complexa, completa e envolvendo os critérios que melhor lhe convier relacionados à semântica daquele cluster. O resultado será mais amplo e poderá lhe trazer perspectivas novas de análises. Da mesma forma que a *local query*, a *cluster query* também é propagada por todos os clusters capazes de respondê-la.

Um aspecto essencial no processamento de consultas do SPEED é a identificação de pontos capazes de responder às consultas. Para isso, é necessário um mecanismo de índice que proveja a lista de pontos candidatos, ora denominada LPC, determinando os pontos que possuem dados para responder aquela consulta. O conceito de índice semântico de consultas será implementado no SPEED a partir da ORC [Herschel, Heese 2005].

Como visto, a ORC é criada e enriquecida à medida em que novos pontos vão entrando no cluster e disponibilizando seus esquemas exportados em formato de ontologia. Para isso, é realizado um processo de *merge* entre a ontologia do ponto com a ORC. O resultado é a ORC enriquecida com conceitos próprios do ponto recém-incluído. O SPEED deverá prover mecanismos para tratamento da evolução da ORC, de acordo com o balanceamento dos clusters.

O ponto de integração é o responsável por manipular o índice de consultas. Quando este ponto recebe uma consulta, ele identifica os termos da consulta e os compara com os termos ontológicos (da ORC); daí, a partir dos mapeamentos e do índice de consultas ele identifica os pontos candidatos a respondê-la (LPC). Em seguida, reformula a consulta de acordo com os mapeamentos, reenvia para cada um dos pontos, recebe os resultados e os integra. Ao término, envia o resultado final para o ponto que iniciou a consulta.

A utilização de pontos de integração reduz a sobrecarga de processamento do sistema, pois as consultas são direcionadas apenas aos clusters e pontos que possuem realmente dados sobre aquele domínio. Por outro lado, é necessário se trabalhar com pontos de integração

candidatos a assumirem o posto de pontos de integração, em caso de falhas. Para isso, este ponto candidato se manterá replicado como todas as informações gerenciadas pelo ponto de integração: índice de consultas, mapeamentos, ontologias e descrição dos pontos.

6.4.1 Descrição Geral do Processo

Neste trabalho, propomos um processo geral para a execução de consultas no SPEED, desde a sua submissão até o resultado final. A idéia é aplicar aspectos semânticos como metadados, ontologias e contexto, em todo o processamento de consultas. Os metadados são usados para recuperar informações dos pontos de dados e ontologias são aplicadas em vários níveis: na representação dos esquemas exportados, nas ontologias de referência do domínio (tanto do cluster quanto da comunidade), como índice de consultas e na representação de informações contextuais.

Como o processamento de consultas é uma tarefa dinâmica, informações contextuais podem contribuir para otimizar caminhos de mapeamentos semânticos a serem trilhados, determinar formatos a serem empregados, indicar conversões de valores e organizar resultados.

A seguir, apresentamos o processo genérico a ser trabalhado, sumarizado graficamente na Figura 6.12. No decorrer do trabalho, iremos focalizar os aspectos relacionados à reformulação da consulta a partir dos mapeamentos, levando em consideração aspectos semânticos que possam contribuir, em especial, aqueles obtidos através de informações contextuais.

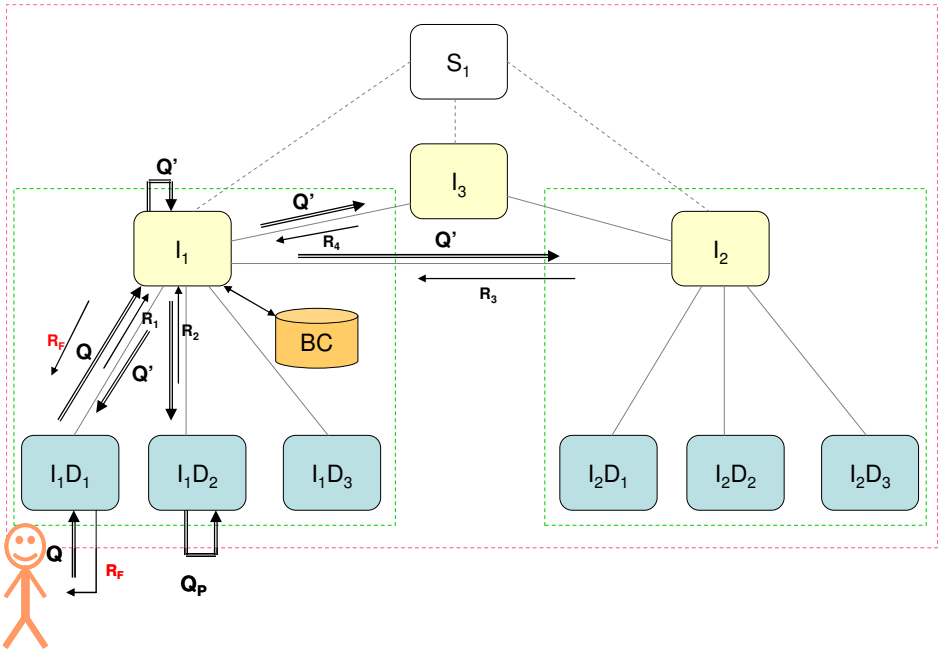

Fig. 6.12 Processo de Execução de Consulta no SPEED

Processo Geral

1. **Usuário expressa a consulta Q no ponto de dados ou no ponto de integração**
    a. A consulta pode ser local ou no cluster
        i. Se for local, a consulta é feita com base no esquema exportado pelo ponto
        ii. Se for no cluster, a consulta é expressa com base na ontologia de referência do cluster (ORC)
    b. Se a consulta estiver no ponto de dados, ela é repassada para o ponto de integração.
2. **O ponto de integração normaliza a consulta Q de acordo com a ORC, gerando Q'.**
3. **O ponto de integração identifica o contexto da consulta**
    a. Informações Contextuais: termos ontológicos (*hasOntologicalTerm*), operadores (*OperatorRelated*), modelo de dados (*QueryModel*), ponto de submissão (isSubmittedIn), restrições (*hasRestriction*), entre outros.
4. **O ponto de integração analisa a consulta Q',** o índice de consultas e os mapeamentos.
    a. Identifica os pontos candidatos a respondê-la dentro do cluster: lista de pontos candidatos – **LPC.**
    b. Analisa o contexto dos mapeamentos: tipo do mapeamento (*mappingType*), precisão (*precisionDegree*), necessidade de composição (*MappingComposition*). Infere a pré-computação de caminhos semânticos.
        i. Enriquece a LPC com uma escala de prioridade dos pontos para reformulação da consulta (LPC'). A idéia é obter um valor que determine o grau de precisão de cada mapeamento (ver regra mostrada na seção 6.3). Com isso, pode-se organizar uma ordem de envio das consultas, evitando mandar logo uma consulta para um ponto que pouco vai contribuir.
        ii. Checa-se possibilidade de redundância entre caminhos de mapeamentos e a necessidade de composição dos mesmos.
    c. Identifica os pontos de integração que podem receber a consulta **Q'– LPI**. Estes pontos de integração irão reenviá-la para seus pontos de dados.
5. **O ponto de integração analisa o contexto de cada ponto pertencente às listas LPC' e LPI**

a. Informações contextuais: disponibilidade (*available*), modelo do ponto (*DataModel*), operadores compatíveis (*compatibleOperators*), domínio (*hasSemanticDomain*), papel (*Role*), entre outros.

b. Resultado: LPC' e LPI atualizadas.

6. A partir da LPC' e de acordo com mapeamentos, para cada ponto determinado:

a. Consulta é reformulada através do mapeamento ou da composição/transitividade dos mapeamentos

i. Consulta reformulada: $\mathbf{Q_P}$

b. Consulta $\mathbf{Q_P}$ é executada no ponto

c. Resultado da consulta $\mathbf{Q_P}$ $\mathbf{(R_1)}$ é enviado ao ponto de integração.

7. A partir da LPI, de acordo com ordem de prioridade dos pontos de integração:

a. Envia a consulta **Q'** para cada ponto de integração, de acordo com mapeamentos.

b. Cada ponto de integração que recebe a consulta **Q'** realiza o mesmo sub-processo apresentado em 4 e 6.

c. Resultados das consultas são retornados ao ponto de integração responsável.

8. Ponto de Integração responsável recebe resultados ($\mathbf{R_1}$...$\mathbf{R_N}$) das consultas:

a. Dos pontos de dados do cluster

b. Dos demais pontos de integração relevantes para a consulta.

9. Ponto de integração responsável:

a. Analisa contexto do resultado

b. Integra resultados

c. Monta Resultado final ($\mathbf{R_F}$)

d. Envia resultado final ao ponto que iniciou a consulta

e. Salva consulta e resultados em Cache

f. Atualiza estatísticas

10. Ponto que iniciou a consulta:

a. Recebe resultado final ($\mathbf{R_F}$)

b. Apresenta resultado final ($\mathbf{R_F}$) ao usuário

6.4.2 Um Exemplo

Como forma de ilustrar o processo descrito, apresentamos um exemplo simples de execução de consulta. Tomemos como base um cluster composto de dois pontos de dados P_1 e P_2 e um ponto de integração I_1. A ontologia de referência do cluster (ORC) está representada na Figura 6.13, enquanto Os esquemas exportados dos pontos de dados estão representados na Figura 6.14.

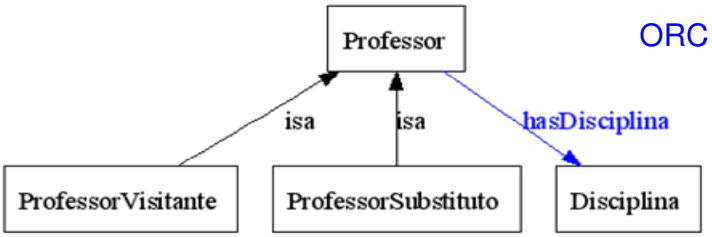

Fig. 6.13 Ontologia de Referência do Cluster

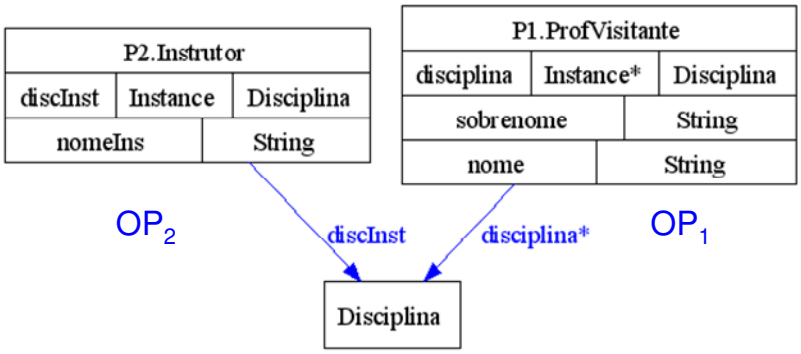

Fig. 6.14 Ontologias dos Pontos P_1 e P_2

1ª etapa – Submissão:

A partir do ponto P_2, o usuário submete a consulta local Q = Select nome From P2.Instrutor Where discInst like 'Banco de Dados'; Q é então repassada ao ponto de Integração I_1.

2ª Etapa – Normalização:

Q -> Q' = Select nome From Professor Where hasDisciplina 'Banco de Dados';

3ª Etapa: Contextualização da consulta Q':

A consulta é contextualizada com os valores correntes das informações percebidas ou inferidas, ou seja, onde ela é submetida, sua identificação, modelo de consultas, termos ontológicos, operadores e entidades associadas e se tem restrições especificadas. Estas informações são mostradas abaixo:

Q'	
isSubmittedIn =	P2
QueryID =	Q'
hasOntologicalTerm =	Professor
	Professor.Nome
	Professor.Disciplina
QueryModel =	Relational
OperatorRelated =	Selection
	Projection
	Like
hasRestriction =	No
relatedEntity =	Professor

4ª Etapa: Geração da Lista LPC

LPC = {P1, P2}

Mapeamentos:

Professor ≡ P1.ProfVisitante

Professor ≡ P2.Instrutor

Professor.nome ≡ P1.ProfVisitante.nome + P1.ProfVisitante.sobrenome

Professor ⊇ P1.ProfVisitante

Considerando que todos os mapeamentos são exatos, a LPC acrescenta a prioridade 3 para cada ponto, resultando em LPC = {P1 (3), P2 (3)}. Isto significa que os pontos P1 e P2 serão os primeiros a receberem a consulta reformulada. Se houvessem outros pontos, estes poderiam ser escalados para um próximo envio, dependendo da análise de fatores como redundância e *containment*.

5ª Etapa: Análise do contexto dos pontos P_1 e P_2

As informações contextuais dos pontos P1 e P2 são capturadas de acordo com os valores correntes, como mostrado abaixo:

P1	
Available =	Sim
PeerID =	1
hasDataSource =	Or.D1
DataModel =	Relacional
PeerName =	P1
isAssociatedwith =	I1
Localization =	Brasil

P2	
Available =	Sim
PeerID =	2
hasDataSource =	MSQL.D2
DataModel =	Objecto-Relacional
PeerName =	P2
isAssociatedwith =	I1
Localization =	Espanha

6ª Etapa: Reformulação

P1: Q_P = Select P1.ProfVisitante.Nome + P1.ProfVisitante.Sobrenome From P1.ProfVisitante Where P1.Disciplina like ' Banco de Dados';

P2: Q_P = Select P2.Instrutor.Nome From P2.Instrutor Where DiscInst like 'Banco de Dados';

7ª Etapa: Resultados de P_1 e P_2 são retornados para I_1

8ª Etapa: I_1 analisa contexto do resultado. Como não tiveram restrições quanto ao resultado, I_1 integra os mesmos e gera um resultado final R_F. Envia R_F ao ponto que iniciou a consulta; salva o resultado em Cache e atualiza estatísticas.

9ª Etapa: P_2 recebe R_F e o apresenta.

6.5 Contribuições

Este trabalho está inserido dentro do projeto SPEED - um PDMS baseado em semântica. Dentro deste escopo, um aspecto essencial ao sistema é a reformulação da consulta baseada em semântica, foco do nosso trabalho. Visando desenvolver uma estratégia de reformulação de consulta baseada em semântica, serão desenvolvidos modelos, técnicas e algoritmos para definição e utilização dos mapeamentos, para criação e utilização do índice de consultas e para a estratégia de reformulação da consulta. A idéia é otimizar a reformulação da consulta através de informações contextuais. Neste sentido, as contribuições deste trabalho podem ser descritas como segue:

- Definição e Utilização de Mapeamentos

  Uma das contribuições deste trabalho será a formalização da definição de mapeamentos dentro do SPEED. Isto inclui a criação de uma linguagem para definição de mapeamentos semânticos, aqui denominada LMS – Linguagem de Mapeamentos Semânticos, que será especificada formalmente. Esta linguagem deverá ser expressiva o suficiente para permitir a definição de mapeamentos entre elementos de ontologias (ORC e Ontologias dos pontos de dados), o que implica em tratar características próprias de ontologias e de mapeamentos entre ontologias. A definição de mapeamentos mais expressivos visa possibilitar extrair semântica dos mesmos na hora da reformulação da consulta. A semântica extraída será usada para otimizar o processo de reformulação.

- Criação e Utilização do índice de Consultas a partir da Ontologia de Referência do Cluster

  A ontologia de referência do cluster é criada e enriquecida à medida em que novos pontos são incluídos no cluster. A nossa contribuição está focada na criação e utilização de um índice semântico de consultas a partir desta ontologia. Para isso, teremos de analisar como extrair desta ontologia o índice e como realizar sua evolução à medida em que a ontologia seja alterada (saída ou entrada de pontos).

- Definição e Implementação da Estratégia de Reformulação de Consulta

A definição e implementação de uma estratégia para reformulação de consulta no SPEED é a maior contribuição deste trabalho. O diferencial entre esta estratégia e outras está relacionado à utilização de semântica em todas as etapas do processo como forma de otimização. Desta maneira, a estratégia será especificada, implementada e validada, de forma iterativa e incremental, gerando versões contínuas.

- Aplicação de Contexto na Otimização da Reformulação da Consulta

Uma das contribuições deste trabalho é tornar o SPEED sensível ao contexto no que diz respeito à reformulação da consulta. Isto significa que para trabalhar com contexto, primeiramente, é necessário definir o modo como as informações contextuais vão ser representadas e quais regras serão necessárias para prover raciocínio. No nosso trabalho, uma ontologia de contexto vem sendo desenvolvida com o propósito de possibilitar tal aplicação [Souza et. al 2006]. A ontologia de contexto do SPEED será ampliada com regras e conceitos, validada através de cenários diferentes de formulações e reformulações de consultas e utilizada para otimizar o processo de reformulação da consulta.

6.6 Metodologia

Este trabalho será desenvolvido através de estudos, pesquisas, discussões, especificação, desenvolvimento e testes de algoritmos, bem como de estudos de caso que validem a proposta. O desenvolvimento do trabalho será executado em seis etapas conforme descrição a seguir:

I. Analisar aspectos semânticos a serem aplicados - metadados, ontologias e contextos e como eles podem efetivamente otimizar semanticamente a reformulação da consulta.

II. Finalizar e Validar a Ontologia de Contexto.
    a. Instanciar a ontologia através de estudos de caso que demonstrem diferentes cenários de utilização.

III. Estudar Estratégias para Definição de Mapeamentos
    a. Analisar critérios de expressividade, semântica, tratamento de visões, enfoques de reformulação de consulta, aspectos próprios de ontologias, entre outros.

IV. Definir Formalismo para Expressão dos Mapeamentos
    a. Estender e formalizar a Linguagem de Mapeamentos Semânticos - LMS

V. Definir Estratégia de Criação e Manutenção do Índice de Consultas
    a. A partir da criação e evolução das ontologias dos clusters

VI. Estudar Técnicas para Reformulação de Consulta em ambientes PDMS

VII. Definir o Processo de Reformulação de Consultas no SPEED
    a. Estabelecer Processo Geral

b. Propor estratégia para reformulação de consulta com base nos aspectos semânticos estudados
    i. Modelar esta estratégia
    ii. Implementá-la no sistema SPEED de forma incremental e iterativa
    iii. Testar e validar a estratégia

VIII. Escrever Artigos que possam referendar o andamento da pesquisa

IX. Escrever a Tese

X. Defender a Tese

As atividades descritas acima contemplam o trabalho da data desta proposta adiante. Algumas delas já foram iniciadas, outras dependem de definições a serem realizadas.

6.7 Cronograma

Nesta seção, é apresentado o cronograma para realização do trabalho proposto, no Quadro 6.2. Para tal, o cronograma apresenta as atividades já realizadas e estabelece prazos para cada uma das etapas descritas anteriormente que serão desenvolvidas.

Atividades Realizadas (04/2005 a 03/2007)

I. Créditos em Disciplinas

II. Levantamento do Estado da Arte

III. Definição da Ontologia de Contexto para Integração de Dados Geográficos

IV. Preparação da Qualificação e Proposta de Tese

Atividades a serem Realizadas (04/2007 a 02/2009)

V. Estudo de Aspectos Semânticos

VI. Finalização e Validação da Ontologia de Contexto para PDMS.

VII. Estudo de Estratégias para Definição de Mapeamentos

VIII. Definição do Formalismo para Expressão dos Mapeamentos

IX. Estabelecimento da Estratégia de Criação e Manutenção do Índice de Consultas

X. Estudo de Técnicas para Reformulação de Consulta em ambientes PDMS

XI. Definição do Processo de Reformulação de Consultas no SPEED

XII. Escrita de Artigos que possam referendar o andamento da pesquisa

XIII. Escrita da Tese

XIV. Defesa da Tese

Quadro 6.2 Cronograma de Atividades

Atividade	2005											
	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
I												
	2006											
I												
II												
III												
IV												
XII												
	2007											
IV												
V												
VI												
VII												
VIII												
XII												
	2008											
VIII												
IX												
X												
XI												
XII												
XIII												
	2009											
XIV												

Referências

Almeida (2001). “Definições e estrutura de metadados”. Disponível em http://www.eci.ufmg.br/mba/text/repr_1_4_1.PDF. Acessado em janeiro de 2007.

Almeida M., Bax M. (2003) “Taxonomia para Projetos de Integração de Fontes de Dados Baseados em Ontologias”. Anais do V do Enancib (Encontro Nacional de Pesquisa em Ciência da Informação), 2003.

Amaral A. (2005). “Resolvendo alguns conflitos semânticos em Sistemas de Integração de Dados”. Trabalho de Conclusão de Curso, CIn/UFPE.

Androutsellis-Theotokis, S., and D Spinellis, D., (2004), A survey of peer-to-peer content distribution techonologies, ACM Computing Survey 36(12) pp335–371.

Arenas, M., Kantere, V., Kementsietsidis, A., Kiringa, I., Miller, R. J., e Mylopoulos, J. (2003) “The Hyperion Project: From Data Integration to Data Coordination”. ACM SIGMOD Record 32, 3.

Arens Y., Hsu C., Knoblock C. A. (1996). “Query processing in the SIMS information mediator “. In Advanced Planning Technology. AAAI Press, California, USA, 1996.

Batista C. (2003) “Otimização de Acesso em um Sistema de Integração de Dados através do Uso de Caching e Materialização de Dados”. Dissertação de Mestrado, CIN/UFPE.

Bazire M., Brézillon P. (2005) “Understanding Context Before Using It”. 5th International and Interdisciplinary Conference, CONTEXT 2005, Paris, France, July 5-8, 2005. Proceedings: Lecture Notes in Computer Science 3554 Springer 2005, ISBN 3-540-26924-X.

Bellahsène Z., Lazinitis C., McBrien P.J. and Rizopoulos N. (2006). “iXPeer: Implementing layers of abstraction in P2P Schema Mapping using AutoMed”. In IWI'2, co-located with WWW'2006.

Bellahsène, Z., Roantree, M., and King, N. (2004) “Services for Large Scale P2P Networks”. In Proc. of European Research Consortium for Informatics and Mathematics News Journal (ERCIM’04).

Beneventano D., Bergamaschi S., Castano S., et al. (2000) “Information Integration: the MOMIS Project Demonstration”. Proc. of the VLDB2000 26th Int. Conference on Very Large Databases, Cairo, Egypt, Sept. 2000.

Bergamaschi S., Quix C., Jarke M. (2005). “The SEWASIE EU IST Project”', SIG SEMIS Bulletin, Vol. 2, No. 1, Feb. 2005. Disponível em http://www.sewasie.org/sewasie-documents.htm

Bernstein P., Giunchiglia F., Kementsietsidis A., Mylopoulos J., Serafini L. and Zaihrayeu I. (2002) “Data management for peer-to-peer computing: A vision”. In Proc. of the 5th Int.Workshop on the Web and Databases (WebDB 2002), 2002.

Borst P. (1997) “Construction of Engineering Ontologies for Knowledge Sharing and Reuse”. PhD thesis, Tweente University, 1997.

Bouzy, B., Cazenave, T. (1997) "Using the Object Oriented Paradigm to Model Context in Computer Go", In: Proceedings of Context'97, Rio de Janeiro, Brazil.

Boyd M., Kittivoravitkul S., Lazanitis C., McBrien P., Rizopoulos N. (2004) “AutoMed: A BAV Data Integration System for Heterogeneous Data Sources”. CAiSE 2004: 82-97

Brézillon P. (1999) “Context in Problem Solving: A Survey”. The Knowledge Engineering Review, 14(1): 1-34.

Brézillon, P. (2003) "Representation of Procedures and Practices in Contextual Graphs", In: The Knowledge Engineering Review, v. 18, n. 2, pp. 147-174.

Brito G., Moura A. M. (2005). “Integração de Objetos de Aprendizagem no Sistema ROSA-P2P”. SBBD 2005: 235-249

Brito, G. (2005) “Integração de Objetos de Aprendizagem no Sistema ROSA-P2P”. MSc Dissertation, IME-RJ.

Bunningen, A. (2004) "Context Aware Querying - Challenges for data management in ambient intelligence", Technical Report TR-CTIT-04-51, Centre for Telematics and Information Technology, University of Twente, Enschede.

Calvanese D., Giacomo G., Lembo D., Lenzerini M. and Rosati R. (2004) “What to ask to a peer: Ontology-based query reformulation”. In Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004), 2004.

Cantele R., Adamatti D., Grigas M., Sichman J. (2004). "Reengenharia e ontologias: análise e aplicação". In: Anais do I Workshop de Rede Semântica (WWS2004), Brasília, 2004.

Castano S. and Antonellis V. (1999). “A discovery-based approach to database ontology design”. Distributed and Parallel Databases - Special Issue on Ontologies and Databases, 7(1), 1999.

Castano S. and Montanelli S. (2005) “Semantic Self-Formation of Communities of Peers”. In Proc. of the ESWC Workshop on Ontologies in Peer-to-Peer Communities, Heraklion, Greece, 2005.

Castano S., Ferrara A., Montanelli S. and Racca G. (2004) “Semantic Information Interoperability in Open Networked Systems”. In Proc. of the Int. Conference on Semantics of a Networked World (ICSNW 2004), Paris, France, June 2004.

Castano S., Ferrara A., Montanelli S., Pagani E., Rossi G.P. (2003). “Ontology-Addressable Contents in P2P Networks”. SemPGRID Workshop.

Catarci T. and Lenzerini M. (1993). “Representing and using interschema knowledge in cooperative information systems”. Journal of Intelligent and Cooperative Information Systems, pages 55–62, 1993.

Chamberlin D., Florescu D., Robie J., Simeon J. and Stefanascu M.(2001). “ XQuery: A query language for XML”. Technical report, World Wide Web Consortium, February 2001. Disponível em http://wwww.w3.org/TR/xquery/.

Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Y. Papakonstantinou Y., Ullman J. and Widom J. (1994) “The TSIMMIS Project: Integration of Heterogeneous Information Sources”. In: Proceedings of the 10th Meeting of the Information Processing Society of Japan. pp. 7—18

Chen, G., Kotz, D. (2000) "A Survey of Context-Aware Mobile Computing Research", Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Costa T. (2005). “O Gerenciador de Consultas de um Sistema de Integração de Dados”. Dissertação de Mestrado, CIN/UFPE.

Cruz I., Xiao H. (2005). “The Role of Ontologies in Data Integration”. Journal of Engineering Intelligent Systems", vol. 13, num. 4, 2005.

Dey, A. K., Salber, D., Abowd, G. D. (2001) "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", In: Human Computer Interaction, v. 16, Special Issue on Context-Aware Computing, pp. 97-166.

Do H. and Rahm E. (2002). “COMA - a system for flexible combination of schema matching approaches”. In Proc. of VLDB, 2002.

Duschka O., Genesereth M. (1997). “Answering recursive queries using views”. In Proceedings of ACM Symposium on Principles of Database Systems, pages 109–116.

Elmasri R., Navathe S. (2005) “Sistemas de Banco de Dados”. 4 ed. São Paulo: Pearson Education, 2005.

Evrendilek C., Dogac A., Nural S. and Ozcan F. (1995). "Query Decomposition, Optimization and Processing in Multidatabase Systems", in Proc. of Workshop on Next Generation Information Technologies and Systems, Naharia, Israel, June, 1995.

Fagin R., Kolaitis P., Popa L. and Tan W. (2004). “Composing schema mappings: Second-order dependencies to the rescue”. In Proc. Of PODS, pages 83–94, 2004.

Farquhar A., Dappert A., Fikes R., Pratt W. (1995) “Integrating Information Sources Using Context Logic”. In Proc. of AAAI Spring Symposium on Information Gathering from Distributed Heterogeneous Environments, 1995.

Felicíssimo C., Breitman K. (2004). “Uma Estrategia para o Alinhamento Taxonomico de Ontologias“. In. Proceedings of the 1st Brazilian Workshop on Semantic Web (WWS'2004), Brasilia, October 22, 2004.

Fensel D. et al. (1998) “ONTOBROKER – the Very High Idea“. Em: Proceedings of the 11th International Florida Artificial Intelligence Research Society Conference. Sanibel Island, Florida, USA, May 1998.

Fiorano. (2003) “Super-Peer Architectures for Distributed Computing”. White Paper, Fiorano Software, Inc.

Firat A. (2003). “Information Integration Using Contextual Knowledge and Ontology Merging”. Doctoral Thesis, The Sloan School of Management, Massachusetts Institute of Technology. September 2003.

Franconi E. , Kuper G., Lopatenko A. and Zaihrayeu I. (2004) “The coDB robust peer-to-peer database system”. In Proc. 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing, 2004.

Friedman M., Levy A. e Millstein T. (1999) “Navigational plans for data integration”. In AAAI ’99/IAAI ’99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, pages 67–73, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

Goh, C.: Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. Ph.D. Thesis, MIT Sloan School of Management, 1996.

Gruber T. (1995). “Toward principles for the design of ontologies used for knowledge sharing”. 1995.

Gruber, T. (1993) "A Translation Approach to Portable Ontologies", In: Knowledge Acquisition, v. 5, n. 2, pp. 199-220.

Grüninger M., Fox M. S. (1994). “The Role of Competency Questions in Enterprise Engineering”. Workshop on Bechmarking, Theory and Practice. Trondheim, Norway.

Guarino N. (1997) “Understanding and building, using ontologies”. Int. J. Hum.-Comput. Stud. 46(2): 293-310 (1997)

Guimarães C. (2005). “Introdução a Linguagens de Marcação: HTML, XHTML, SGML, XML”. Disponível em http://www.dcc.unicamp.br/~celio/inf533/docs/markup.html. Acessado em Janeiro de 2007.

Gupta S., Mumick I. (1995) “Maintenance of Materialized Views: Problems, Techniques and Applications”. IEEE Data Engineering Bulletin. 18(2): pp 3-18. June, 1995.

Halevy A. Y. (2001). “Answering queries using views: A survey“. The VLDB Journal, 10(4):270– 294, 2001.

Halevy A., Franklin M. and Maier D. (2005) “From Databases to Dataspaces: a New Abstraction for Information Management”. ACM SIGMOD Record, December 2005.

Halevy A., Rajaraman A. and Ordille J. (2006).“Data integration: the teenage years”. In. Proceedings of the 32nd international conference on Very large data bases - Volume 32, Pages: 9 – 16, 2006.

Halevy, A. Y. (2000). “Theory of Answering Queries Using View“. ACM Special Interest Group on Management of Data Record 29(4), 40--47.

Heese R., Herschel S., Naumann F., and Roth A. (2005) “Self-extending peer data management”. In G. Vossen, F. Leymann, P. C. Lockemann, and W. Stucky, editors, Proceedings of the German Conference on Datenbanksysteme in Business, Technologie und Web, volume 65 of LNI. GI, March 2005.

Held, A., Buchholz, S., Schill, A. (2002) "Modeling of Context Information for Pervasive Computing Applications", In: Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL, USA.

Herschel S., Heese R. (2005) “Humboldt Discoverer: A semantic P2P index for PDMS”. Proceedings of the International Workshop Data Integration and the Semantic Web, Porto, Portugal, 2005.

Ives Z. G., Levy A. Y.,Weld D. S., Florescu D., Friedman M.(2000) “Adaptive Query Processing for Internet Applications”. IEEE Data Engineering Bulletin, Vol.23, No.2, pp.19-26, 2000.

Jakobovits R. (1997) “Integrating Autonomous Heterogeneous Information Sources”. Dept. of Computer Science & Engineering, University of Washington. 15 de Julho de 1997.

Kalfoglou, Y. and Schorlemmer, M. (2003). “Ontology mapping: the state of the art”. The Knowledge Engineering Review 18(1) pp. 1-31.

Kamienski C., Souto E., Rocha J., Domingues M., Callado A., Sadok D. (2005). “Colaboração na Internet e a Tecnologia Peer-to-Peer”. In: XXV Congresso da Sociedade Brasileira de Computação – SBC2005. 25 a 29 Jul. São Leopoldo – RS.

Karvounarakis G. (2005). “Answering queries across mappings”. Disponível em http://www.seas.upenn.edu/~gkarvoun/publications/queries+mappings.pdf, acessado em Março de 2007.

Kashyap V., Sheth A. (1996a) "Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach." VLDB Journal 5, no. 4 (1996): 276--304. 367

Kashyap V., Sheth A. (1996b) “Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies”. In Cooperative Information Systems: Current Trends and Directions, M. P . Papazoglou and G. Schlageter, Eds. London: Academic Press, 1996, pp. 139-178. (1996)

Katchaounov, T. (2003) “Query Processing for Peer Mediator Databases”. Doctoral thesis, Uppsala University, Sweden.

Koloniari G., Pitoura E. (2004) “Content-based Routing of Path Queries in Peer-to-Peer Systems”. EDBT 2004.

Lenzerini M. (2002). “Data integration: a theoretical perspective”. In Proceedings of ACM Symposium on Principles of Database Systems, pages 233–246, New York, NY, USA, 2002. ACM Press.

Levy A., Rajaraman A., and Ordille J. (1996). “Querying Heterogeneous Information Sources Using Source Descriptions”. In Proceedings of the International Conference on Very Large Databases (VLDB), 1996.

Levy A.: Combining Artificial Intelligence and Databases for Data Integration. Artificial Intelligence Today, 1999, pp. 249-268.

Lóscio, B. (2003) “Managing the Evolution of XML-based Mediation Queries”. PHD Thesis, Federal University of Pernambuco, Brazil.

Löser, A, Siberski, W., Wolpers, M., and Nejdl, W. (2003) “Information Integration in Schema-Based Peer-to-Peer Networks”. In Proc. of the Conference on Advanced Information Systems Engineering.

Lv, Q., et al. (2002) “Search and Replication in Unstructured Peer-to-Peer Networks”. 16o ACM International Conference on Supercomputing(ICS'02), Junho de 2002.

Madhavan J. and Halevy A. Y. (2003) “Composing mappings among data sources”. In VLDB, pages 572–583, 2003.

Marietto M., David N., Sichman J., Coelho H. (2002) “Infraestrutura para a Construção de Ontologias”. Working Paper 01-2002. Escola Politécnica da Universidade de São Paulo.

McBrien P.J. and Poulovassilis A. (2003a) “Data integration by bi-directional schema transformation rules”. In Proc. 19th International Conference on Data Engineering - ICDE'03. IEEE, 2003.

McBrien P.J. and Poulovassilis A. (2003b) “Defining peer-to-peer data integration using both as view rules”. In Proc. DBISP2P, at VLDB'03, Berlin, Germany, 2003.

McBrien P.J. and Poulovassilis A. (2006). “P2P query reformulation over Both-as-View data transformation rules”. In Proceedings of DBISP2P 2006 .

McCarthy, J. (1993). “Notes on formalizing context”. In Proceedings of the Thirteenth International Joint Conference in Artificial Intelligence, 555-560. San Mateo, California: Morgan Kaufmann.

Mena, E., Kashyap, V., Sheth, A., Illarramendi, A. (1996) “OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies”. Conference on Cooperative Information Systems 41 14{25).

Mika, P., Akkermans, H. (2005) "Towards a New Synthesis of Ontology Technology and Knowledge Management", In: The Knowledge Engineering Review, v. 19, n. 4, pp. 317-345.

Mori A. e Carvalho C. L. (2004). “Metadados no contexto da Web Semântica”. Trabalho Técnico. Disponível em http://www.inf.ufg.br/~cedric/DWeb.html. Acessado em janeiro de 2007.

Necib C. B. and Freytag J. (2004) “Using Ontologies for Database Query Reformulation”. Proceedings of the 18th Conference on Advances in Databases and Information Systems (ADBIS'04), Budapest, Hungary, 2004.

Necib C. B. and Freytag J. (2005) “Query Processing Using Ontologies”. Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE'05), Porto, Portugal, 2005.

Ng, W. S., Ooi, B., Tan, K., and Zhou, A. (2003) “PeerDB: a P2P-based System for Distributed Data Sharing”. (2003) In Proc. of 19th International Conference on Data Eng. (ICDE).

Nimble Technology, Inc (2006). The Nimble Integration Suite White Paper. Disponível em http://www.nimble.com/solutions/literature/. Acessado em Julho/2006.

Noy, N. F., Musen, M. A.: The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, International Journal of Human-Computer Studies, 2003.

Ooi, B., Shu, Y., and Tan, K. (2003) “Relational Data Sharing in Peer-based Data Management Systems”. ACM SIGMOD Record, 3 2(3).

Parameswaran M., Susarla A., Whiston A. (2001). “P2P Networking: An Information-Sharing Alternative”. IEEE Computer, Julho de 2001.

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.

Power, R.: Topic Maps for Context Management. In International Symposium on Information and Communication Technologies (ISICT 2003), pp. 199-204.

Rahm, E., Bernstein, P. (2001) “A survey of approaches to automatic schema matching”. The VLDB Journal 10:334-350.

Ratnasamy S., Francis P., Handley M. and Karp R. (2001). “A scalable content-addressable network”. In Proceedings of SIGCOMM 2001.

Roshelova A. (2004) “A Peer-to-Peer Database Management System”. Disponível em http://eprints.biblio.unitn.it/archive/00000585/01/albena.pdf

Rouse C. (2006). “Schema Matching in a Peer-to-Peer Database System”. Master Thesis in Computer Science, University of Cape Town. February 2006.

Rowstron A. and Druschel P. (2001). "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. Int'l Conf. Distributed Systems Platforms, 2001.

Rundensteiner, E., Koeller, A. and Zhang, X. (2000). “Maintaining Data Warehouses over Changing Information Sources”. Communications of the ACM, Special Section on System Integration, 2000.

Ruzzi M. (2004). “Data Integration : state of the art, new issues and research plan”.

Shvaiko P. and Euzenat J. (2005). “A Survey of Schema-based Matching Approaches”. Journal on Data Semantics, 2005.

Souza, D., Salgado, A. C., Tedesco, P. (2006) "Towards a Context Ontology for Geospatial Data Integration", In: Second International Workshop on Semantic-based Geographical Information Systems (SeBGIS'06), Montpellier, France.

Stefanidis K., Pitoura E., Vassiliadis P. (2005). “On Supporting Context-Aware Preferences in Relational Database Systems”. In Proc. of the first International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP’2005), in conjunction with MDM 2005, Cyprus.

Stoica I., Morris R., Karger D., Kaashock M. and Balakrishman H. (2001) “Chord: A scalable peer-to-peer lookup protocol for internet applications”. In Proceedings of the ACM SIGCOMM, pages 149--160, San Diego, California, Agosto de 2001.

Strang, T., Linnhoff-Popien, C. (2004) "A Context Modeling Survey", In: Workshop on Advanced Context Modeling, Reasoning and Management, In: 6th International Conference on Ubiquitous Computing, Nottingham/England.

Sung, L. G. A., Ahmed, N., Blanco, R., Li, H, Soliman, M. A., and Hadaller, D. (2005) “A Survey of Data Management in Peer-to-Peer Systems”. School of Computer Science, University of Waterloo.

Tatarinov, I., Halevy, A. (2004) “Efficient query reformulation in peer-data management systems”. In Proc. of SIGMOD Conference.

Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X., Kadiyska, Y., Miklau, G., and Mork, P. (2003) “The Piazza Peer Data Management Project”. ACM SIGMOD Record, Vol. 32.

The CIMI Standard (2007). Disponível em http://www.ukoln.ac.uk/metadata/desire/overview/rev_04.htm. Acessado em janeiro de 2007.

The DublinCore Standard web site (2007), http://dublincore.org/. Acessado em Janeiro de 2007.

The FGDC Standard web site (2007), www.fgdc.gov/. Acessado em janeiro de 2007.

The Gnutella System web site (2007), http://www.gnutella.com/. Acessado em março de 2007.

The HELIOS Project web site. http://islab.dico.unimi.it/helios/.

The Kazaa System web site (2007), http://www.kazaa.com/us/index.htm. Acessado em março de 2007.

The MARC Standard web site (2007), www.loc.gov/marc/ Acessado em janeiro de 2007.

The Napster System web site (2007), http://www.napster.com/. Acessado em março de 2007.

The UMLS web site (2007), http://umlsinfo.nlm.nih.gov/. Acessado em março de 2007.

The W3C Metadata (2007), www.w3.org/Metadata/. Acessado em janeiro de 2007.

Towsley D. (2003) “Peer-peer Networking”. Tutorial no SBRC2003, disponível em http://www.cin.ufpe.br/~gprt/gtp2p/tutoriais/p2p03-tutorial_towsley.pdf.

Ullman, J. D., (1997). “Information Integration Using Logical Views”. In Proc. of ICDT’97, vol.1186 of LNCS, pp.19-40, Springer-Verlag, 1997.

Unknown Author, (2003), Introduction to P2P networks, P2P networks project, http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/Intro.html, acessado em 15 de Janeiro de 2007.

Uschold M. and Jasper R. (1999). “A Framework for Understanding and Classifying Ontology Applications”. In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5), Stockholm, Sweden, Aug. 1999. http://sunsite.informatik.rwthaachen. de/Publications/CEUR-WS/Vol-18/.

Uschold M., Gruniger M. (1996). “Ontologies: Principles, methods and applications”. KnowledgeEngineering Review, 11(2):93–155, 1996.

Valduriez, P., Pacitti, E. (2004) “Data Management in Large-scale P2P Systems”. In Proc. of Int. Conf. on High Performance Computing for Computational Science (VecPar 2004), Valencia, Spain.

Vallejos J. (2005). “Mobile Actors Supporting Reconfigurable Applications in Open Peer-to-Peer Networks”. Master Thesis. Vrije Universiteit Brussel, 2005.

Vieira V. (2006). “Gerenciamento de Contexto em Sistemas Colaborativos”, Monografia de Qualificação e Proposta de Tese de Doutorado, CIn, Universidade Federal de Pernambuco, PE, Recife, Brasil.

Vieira V., Souza D., Salgado, A. C., Tedesco, P. (2006). “Uso e Representação de Contexto em Sistemas Computacionais”. In: Cesar A. C. Teixeira; Clever Ricardo G. de Farias; Jair C. Leite; Raquel O. Prates. (Org.). Tópicos em Sistemas Interativos e Colaborativos. São Carlos: UFSCAR, 2006, v. , p. 127-166.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S. (2001) “Ontology-based Integration of Information - a Survey of Existing Approaches”. In Stuckenschmidt, H., ed., IJCAI-01 Workshop: Ontologies and Information Sharing, pp. 108--117. 2001.

Wang, X. H., Gu, T., Zhang, D. Q., Pung, H. K. (2004) "Ontology based context modeling and reasoning using OWL", In: Workshop on Context Modeling and Reasoning at II IEEE International Conference on Pervasive Computing and Communication, Orlando, Florida.

Wiederhold G. (1992) “Mediators in the architecture of future information systems”. IEEE Computer, 25(3):38 – 49, 1992.

Xiao H., Cruz I. (2004) “RDF-based Metadata Management in Peer-to-Peer Systems”. In The 2nd IST Workshop on Metadata Management in Grid and P2P System (MMGPS 2004).

Xiao H., Cruz I. (2006). “Ontology-based Query Rewriting in Peer-to-Peer Networks”. In Proc. of the 2nd International Conference on Knowledge Engineering and Decision Support, 2006.

Xu L. and Embley D.W. (2002) “Combining the Best of Global-as-View and Local-as-View for Data Integration”. November 2002.

Young W. A. “Evaluation of Peer-to-Peer Database Solutions”. Disponível em http://www.tonyyoung.ca/cs654paper.pdf

Zacarias, M., Pinto, H. S., Nunes, J., Tribolet, S. (2004) "Redes de Conhecimento em Engenharia Organizacional: O Imperativo dos Contextos de Ação", Caderno de Biblioteconomia Arquivística e Documentação Cadernos BAD, v. 001.

Zhao J. (2006). “Schema Mediation and Query Processing in Peer Data Management Systems”. Master Thesis, The University Of British Columbia, October, 2006.

Zhuge H., Liu J., Feng L., Sun X., e He C. (2005) “Query Routing in a Peer-to-peer Semantic Link Network”. Computational Intelligence, 21 (2) (2005) 197-216.

Anexo A

Ontologia de Contexto para Integração de Dados Geográficos

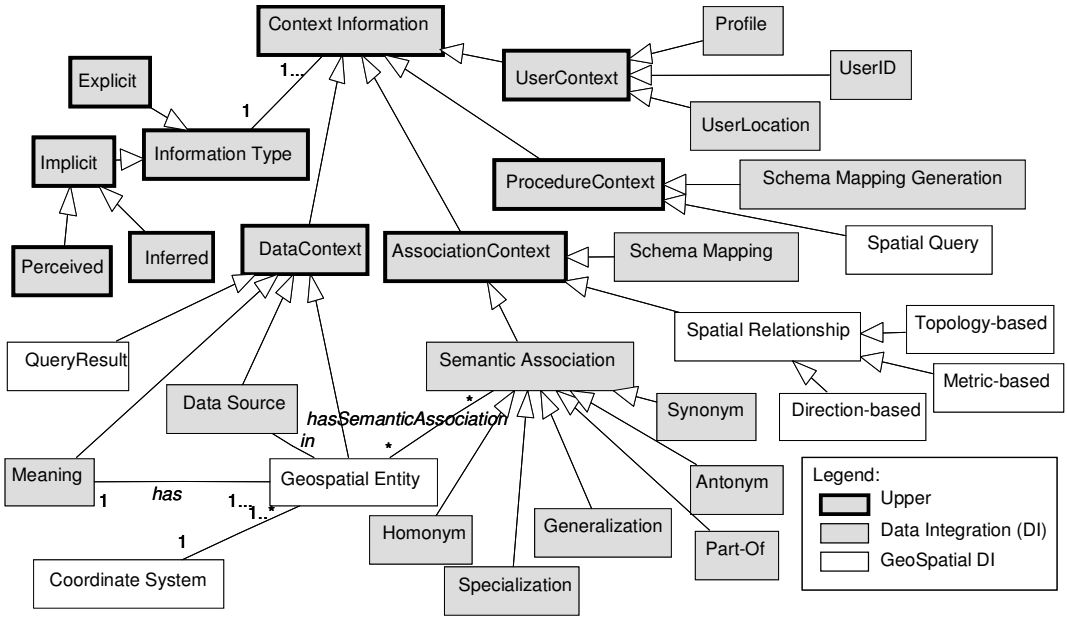

Fig. A.1 Ontologia de Contexto para Integração de Dados Geográficos [Souza et al. 2006]