Leonardo Jose Silvestre
Cristiano Biancardi
Alvaro C. P. Barbosa

Sistemas para Integração de Dados

Vitória - ES

2004

Sumário

Lista de Figuras

1 *Sistemas para Integração de Dados*

Atualmente, vivemos em uma época na qual a assim chamada sociedade da informação demanda um completo acesso para a informação disponível, que é freqüentemente heterogênea e distribuída (WACHE et al., 2001). Para tanto, sistemas de integração de dados vêm sendo desenvolvidos, na tentativa de provêr aos usuários uma visão única, uniforme e homogênea, para um grande número de fontes de dados heterogêneas, desenvolvidas independentemente. O objetivo maior desses sistemas é liberar o usuário de ter que localizar as fontes de dados, interagir com cada uma isoladamente e combinar os dados das múltiplas fontes de dados manualmente (HAKIMPOUR; GEPPERT, 2001; HALEVY, 2003). Desde o surgimento dos sistemas de gerenciamento de banco de dados, o problema de integração de dados tem sido reconhecido como ubíquo e criticamente importante (MILLER et al., 2001), e vem demandando pesquisa na área de Banco de Dados por mais de uma década (SHETH; LARSON, 1990; LITWIN; MARK; ROUSSOPOULOS, 1990; SILBERSCHATZ; ZDONIK, 1997; ABITEBOUL et al., 2003; HALEVY, 2003). O advento da *Web* fez crescer dramaticamente a necessidade por mecanismos flexíveis e eficientes para prover formas integradas de manipular múltiplas fontes heterogêneas de informação, já que a *Web* está se tornando a principal infraestrutura para publicação e disseminação de informação em organizações públicas e privadas, tanto no nível interno quanto no externo (CASTANO et al., 2002). Além de integração de dados na *Web*, as aplicações de sistemas de integração de dados também envolvem gerenciamento de dados em grandes organizações, compartilhamento de dados entre agências governamentais e grandes projetos científicos como, por exemplo, pesquisas biológicas e astronomia (HALEVY, 2003).

2 *Sistemas relacionados com Ontologias*

Neste capítulo são apresentados trabalhos de integração de dados heterogêneos e distribuídos que utilizam ontologias. Tais trabalhos foram divididos entre os tradicionais, mais referenciados na literatura, e os mais recentes, com propósito de investigar como esses sistemas lidam com metadados, de forma a permitir um maior embasamento para a definição dos serviços e componentes do ambiente de metadados do CoDIMS.

O problema de integração de dados tem recebido grande atenção da comunidade de banco de dados, tendo sido estudado desde o início da décad de 80 (ZIEGLER; DITTRICH, 2004). São encontrados, na literatura, diversos sistemas e propostas tentando solucionar tal problema. O aparente vasto número de soluções pode, erroneamente, indicar uma área de pesquisa já resolvida. Pelo contrário, é uma área cada vez mais importante e onde não existe uma solução geral que seja adequada ou que se ajuste aos diversos problemas de integração, o que se constata pelo surgimento de novas propostas. Dentre elas, diversos utilizam ontologias para a definição de metadados, buscando uma maior semântica para os mesmos.

A seguir, são apresentadas as arquiteturas de alguns sistemas de integração de dados que utilizam ontologias: os mais referenciados na literatura e também os mais recentes. Além destes, outros foram analisados, mas não serão aqui apresentados devido às semelhanças com outros sistemas ou por utilizarem outras abordagens. Nosso objetivo foi investigar a forma como esses sistemas representam os metadados e lidam com as ontologias, portanto, a apresentação das arquiteturas dos mesmos dará enfoque a esses aspectos, com o propósito de nos permitir um embasamento maior para a definição dos serviços e componentes do ambiente de metadados do CoDIMS.

2.1 SIMS

O SIMS (*Services and Information Management for decision Systems*) (ARENS; KNOBLOCK, 1992; ARENS et al., 1993; ARENS; KNOBLOCK; SHEN, 1996) é um sistema para integração de informações desenvolvido no *Information Sciences Institute* da Universidade da Califórnia do Sul (EUA). Sua abordagem explora um modelo semântico do domínio do problema para integrar informações de várias fontes de informação, tais como bancos de dados, bases de conhecimento, programas etc.

Para proceder a integração, o SIMS necessita de uma ontologia que descreva o domínio no qual as informações a serem tratadas se encontram, assim como a estrutura e os conteúdos das fontes de informação. Essa ontologia é chamada de modelo do domínio, e é uma descrição declarativa dos objetos e atividades possíveis no domínio de aplicação, como vistos por um usuário típico. Para uma fonte de informação ser incorporada ao sistema, deve-se primeiramente criar um modelo para a mesma. Este indica o modelo de dados usado, a linguagem de consulta, localização na rede, tamanho estimado, freqüência de atualização etc., e descreve seus campos em termos do modelo de domínio. Posteriormente, seus conceitos e regras são relacionados aos conceitos e regras correspondentes do modelo do domínio. O usuário formula consultas usando termos do domínio da aplicação, através de instruções Loom (REFERÊNCIA).

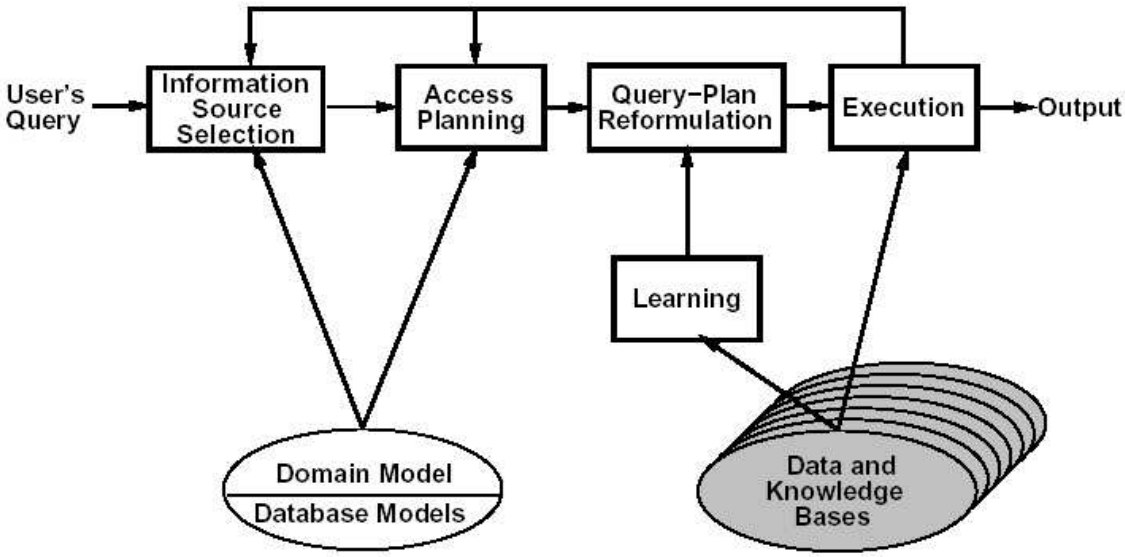

Figura 1: Funcionamento do SIMS.

A figura 1 (ARENS et al., 1993) apresenta um diagrama que resume o funcionamento do SIMS.

2.2 ONTOBROKER

O ONTOBROKER (DECKER et al., 1999; FENSEL et al., 1998; DECKER et al.,) é um sistema para integração de páginas Web desenvolvido na Universidade de Karlsrube, na Alemanha. Seu principal objetivo é extrair, inferir e gerar metadados específicos de domínio para integrar páginas Web, usando uma ontologia de domínio que reflete o consenso de um grupo de usuários Web.

Sua arquitetura, apresentada na figura 2, é composta pelas seguintes partes:

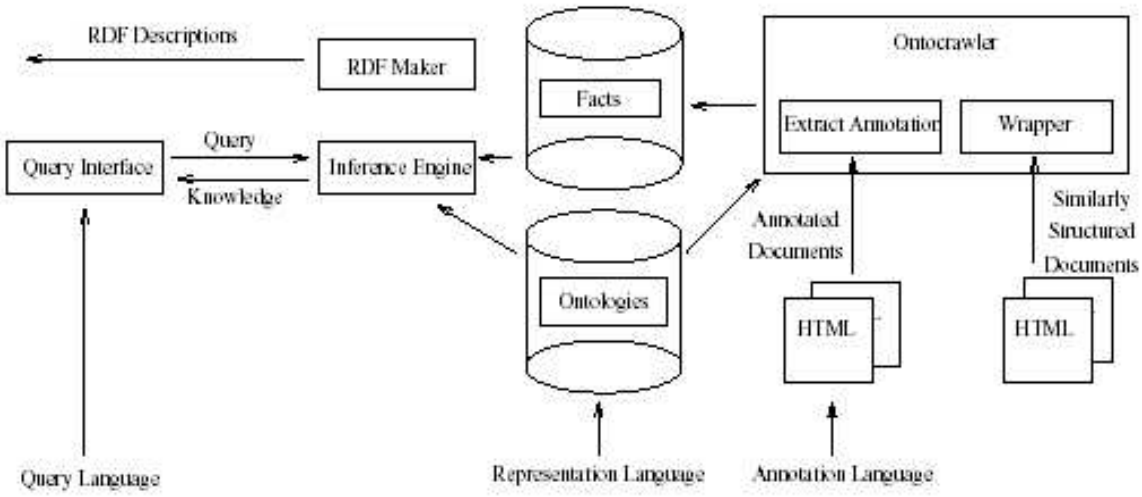

Figura 2: Arquitetura do ONTOBROKER.

- *Ontologias*, que são a parte principal do sistema;
- *Ontocrawler*, que extrai conhecimento formal de páginas HTML;
- *Máquina de inferência*, que explora a semântica formal da linguagem de representação e possibilita inferência automática;
- *RDF-Maker*, que explora a máquina de inferência e gera uma representação RDF da informação que pode ser inferida da ontologia;
- *Interface de consulta*, que possibilita formulação interativa de consultas enquanto se navega pela ontologia.

ONTOBROKER utiliza RDF/RDFS para representar seus metadados, anotados nos recursos Web, e *Frame-Logic* (*F-Logic*) (KIFER; LAUSEN; WU, 1990) é utilizada como base da máquina de inferência, ou seja, *F-Logic* é usada quando o sistema está respondendo consultas baseadas em uma ontologia.

2.3 OBSERVER

O OBSERVER (MENA et al., 1996) é um sistema para processamento de consultas desenvolvido na Universidade da Geórgia (EUA). OBSERVER tem como objetivo principal o processamento de consultas em sistemas de informação globais, baseado em interoperabilidade entre ontologias pré-existentes. Ele utiliza metadados para capturar o conteúdo das informações dos repositórios. O usuário efetua consultas sobre o sistema expressando suas necessidades de informação utilizando descrições de metadados representados utilizando *Description Logics* (BRACHMAN; SCHMOLZE, 1985).

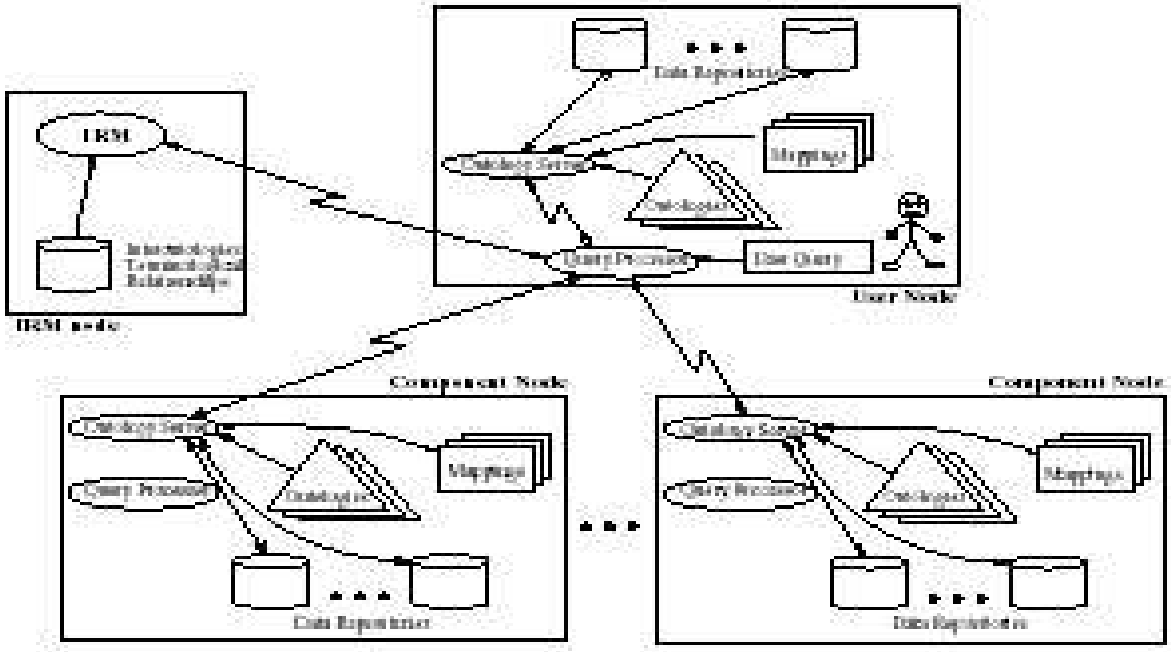

Figura 3: Arquitetura do OBSERVER.

As consultas de usuários são re-escritas de forma a se preservar a semântica, o que é feito pela substituição de termos com termos sinônimos de diferentes ontologias. Termos *hyponym and hypernym* também podem ser usados e a perda de informação medida.

A arquitetura do OBSERVER, que pode ser observada na figura 3, é composta de:

- *Processador de Consultas*, responsável por, dada uma consulta expressa em DLs usando termos de uma dada ontologia, navegar por outras ontologias componentes e traduzir termos da consulta em termos das mesmas, preservando a semântica da consulta do usuário;
- *Servidor de Ontologias*, que provê informações sobre as ontologias ao Processador de

Consultas, tais como as definições dos termos e os mapeamentos entre cada termo na ontologia e estruturas nos repositórios de dados;

- *Gerenciador de Relacionamentos Inter-ontologias* (*Interontology Relationships Manager, IRM*), que contém relacionamentos sinônimos relacionando os termos em várias ontologias;

- *Ontologias*, sendo que cada uma delas é um conjunto de termos de interesse em um domínio de informação particular, sendo expressada em DLs.

2.4 MOMIS

O MOMIS (BENEVENTANO et al., 2001a, 2001b; BENEVENTANO; BERGAMASCHI, 2004) *Mediator EnvirOnment for Multiple Information Sources* é um *framework* para extração e integração de informações de fontes de informação heterogêneas, desenvolvido pela universidade de Modena e Reggio Emilia, na Itália. Ele implementa uma metodologia semi-automática para integração de dados que segue a abordagem *Global As View* (GAV). O resultado do processo de integração é um esquema global, chamado *Global Virtual View* (GVV), o qual representa a informação contida nas fontes, sendo uma ontologia que representa o domínio das fontes integradas. Como modelo de dados comum, MOMIS utiliza uma linguagem orientada a objetos chamada $ODL_I{}^3$, que é uma evolução da linguagem ODL, padrão para SGBDs OO. As consultas de usuário também são feitas utilizando-se a linguagem $ODL_I{}^3$. MOMIS utiliza CORBA (OMG, 1997) para comunicação entre seus componentes. Os mapeamentos entre a GVV e os esquemas são mantidos em uma tabela.

A figura 4 apresenta a arquitetura do MOMIS, a qual é formada por:

- *Construtor de Ontologias*, que processa e integra descrições recebidas dos *wrappers* para derivar o esquema global compartilhado. Isso é feito interagindo com diferentes módulos de serviço: ODB-Tools, que é um ambiente para inferência em banco de dados orientado a objetos, que RELIES em DLs; WordNet, que é um banco de dados lexico, o qual suporta o mediador na construção dos relacionamentos; e a ferramenta ARTEMIS, que executa a operação de *clustering* para geração da GVV;

- *Repositório de Metadados*, o qual armazena as ontologias geradas peo construtor de ontologias;

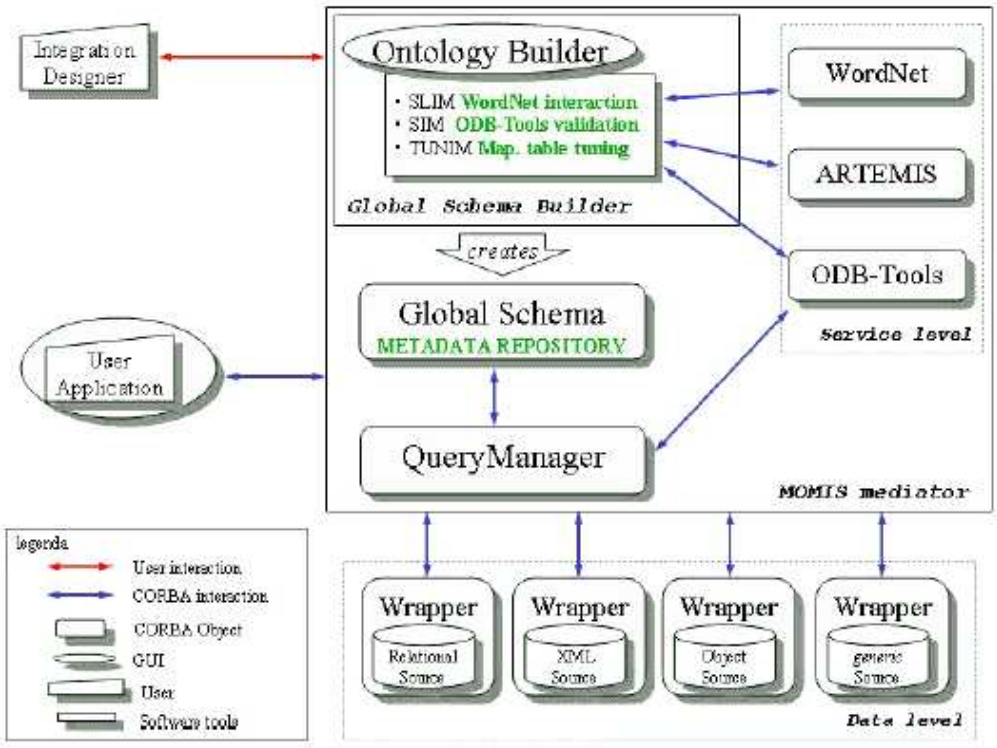

Figura 4: Arquitetura do MOMIS.

- *Gerente de Consultas*, responsável por executar o processamento e a otimização das consultas. Ele gera, a partir da consulta do usuário, consultas $\text{ODL}_I{}^3$, que são enviadas para os *wrappers*, e também sintetiza os resultados para apresentação ao usuário;
- *Wrappers*, localizados no *data level*, e responsáveis pela tradução entre o modelo de dados da fonte e o modelo de dados do mediador, no caso $\text{ODL}_I{}^3$.

2.5 YACOB

O YACOB (KARNSTEDT et al., 2003) é um mediador desenvolvido na universidade de Halle-Wittenberg, na Alemanha. Ele é um mediador baseado em ontologias que usa conhecimento de domínio modelado, representado em termos de conceitos, propriedades e relacionamentos, para integrar dados heterogêneos da Web. O YACOB foi originalmente desenvolvido para permitir acesso integrado a bancos de dados, disponíveis na Web, de artefatos culturais perdidos ou roubados durante a segunda guerra mundial. Suas ontologias são definidas em RDFS.

A figura 5 apresenta a arquitetura do YACOB, que é composta pelos seguintes com-

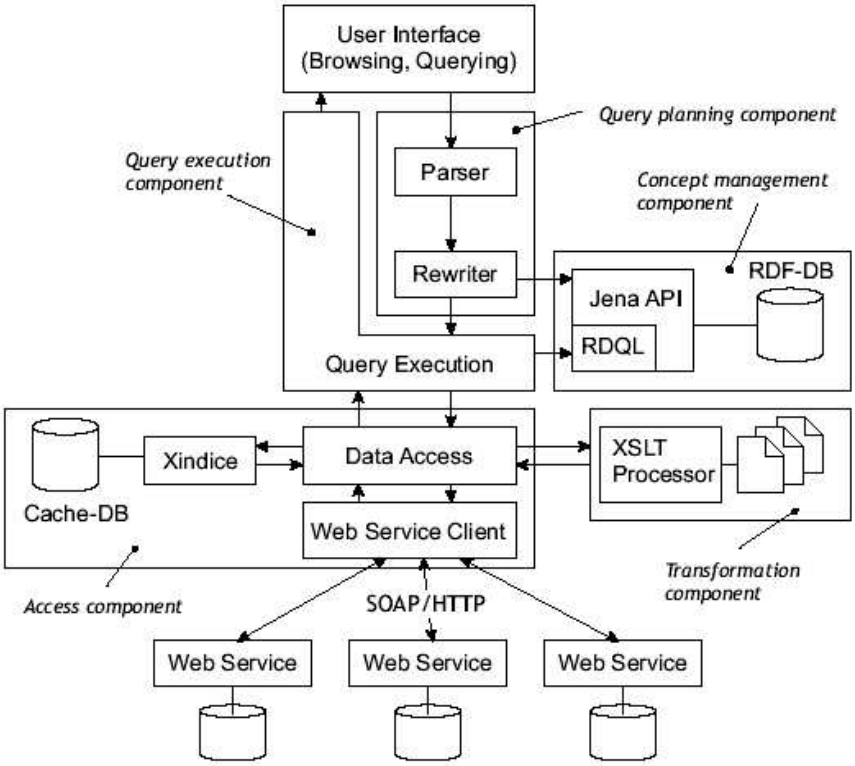

Figura 5: Arquitetura do YACOB.

ponentes:

- *Componente de Acesso* (*Access Component*): oferece serviços para que o mediador acesse os *wrappers*. Estes recebem consultas XPath, traduzem para o modelo de cada fonte e retornam o resultado como um documento XML de uma DTD arbitrária. Tal comunicação é feita usando SOAP, já que os *wrappers* são Web Services. O componente de acesso também possui uma cache que armazena os resultados das consultas em um banco de dados local, para permitir o uso desses dados em consultas subseqüentes;

- *Gerente de Conceitos* (*Concept Manager*): provê serviços para armazenar e recuperar metadados (conceitos e seus mapeamentos) em termos de um grafo RDF;

- *Componentes de Planejamento e Execução de Consultas* (*Query Planning* e *Query Execution*): processam as consultas globais, expressas em XPath;

- *Componente de transformação* (*Transformation Component*): responsável por transformar, de acordo com o modelo global, os resultados retornados das fontes.

2.6 TAMBIS

The TAMBIS (*Transparent Access to Multiple Bioinformatics Information Sources*) (GOBLE et al., 2001; PEIM et al.,) é um sistema para permitir acesso transparente a fontes de informação de bioinformática desenvolvido na universidade de Manchester, Reino Unido. Ele utiliza uma ontologia do domínio de biologia molecular e bioinformática, representada usando uma *description logic* e gerenciada por um servidor de terminologias, para permitir consultas, através de um esquema global, a diversas fontes de informação.

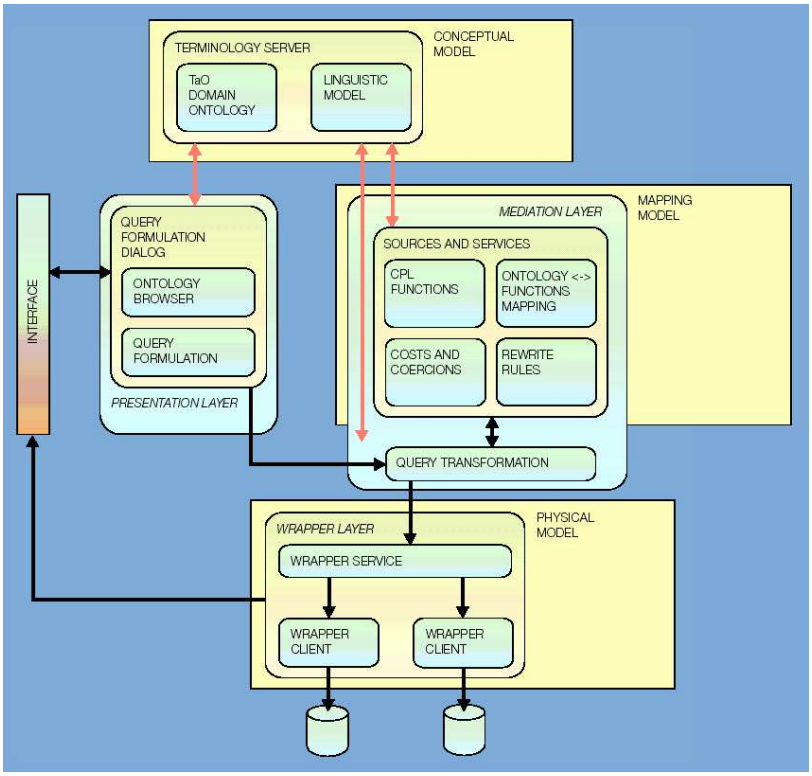

Figura 6: Arquitetura do TAMBIS.

Sua arquitetura, apresentada na figura 6, é composta por cinco componentes principais, organizados em um modelo *wrapper*-mediador clássico de três camadas: apresentação, mediação e wrapper. O servidor de terminologias (*Terminology Server*) armazena e gerencia a ontologia de domínio e o modelo linguístico, e é extensivamente utilizado na reformulação e na transformação de consultas. A camada de apresentação (*Presentation Layer*) é responsável por prover a formulação das consultas. A camada de mediação (*Mediation Layer*) é responsável por lidar com os mapeamentos, utilizados para a reescrita/transformação de consultas. A camada de *wrapper* (*Wrapper Layer*) trata dos *wrappers*, provendo acesso às fontes e transformação entre modelos.

2.7 Suwanmanee

O trabalho de Suwanmanee (SUWANMANEE; BENSLIMANE; THIRAN, 2005), desenvolvido na universidade de Lyon, França, com o objetivo de integrar fontes de dados heterogêneas no contexto da Web Semântica. Ele utiliza uma abordagem ontológica para integração de dados, utilizando construções nativas da linguagem OWL para representar metadados e mapeamentos.

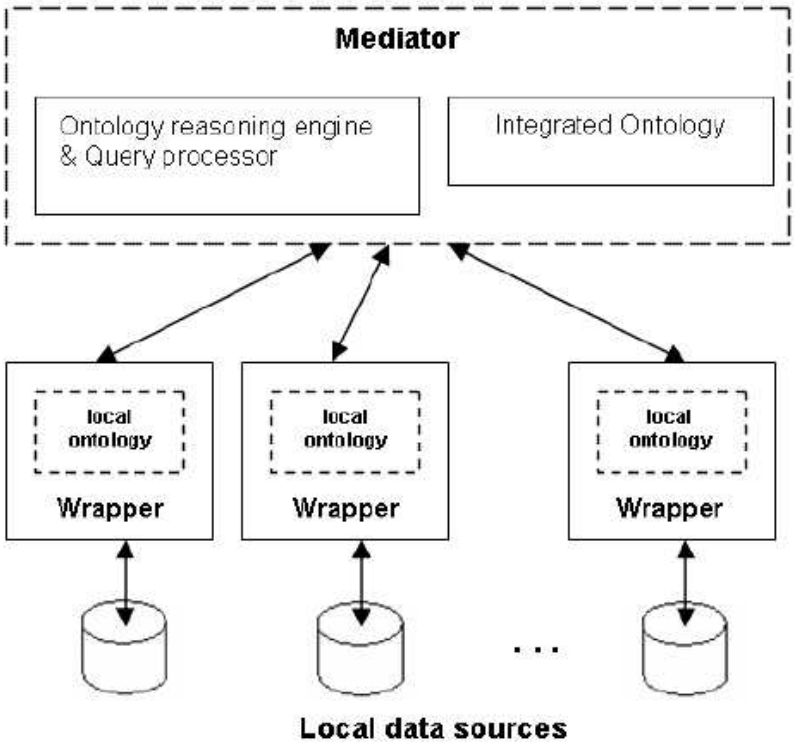

Figura 7: Arquitetura do Trabalho de Suwanmanee.

A arquitetura do sistema, apresentada na figura 7, é composta pelas camadas de fontes (*source layer*), *wrapper* e mediação (*mediating layer*). A camada de fontes corresponde às fontes de dados estruturados, como bancos de dados, repositórios XML e documentos Web. A camada de *wrapper* contém wrappers para cada fonte de dados local. Cada *wrapper* provê uma ontologia OWL representando a fonte e formas de acessar e consultar a mesma. A camada de mediação contém o mediador que possibilita interoperabilidade entre as fontes locais. Uma de suas principais funções é integrar ontologias locais, para garantir acesso global às fontes. Ele contém uma máquina de inferência que lida com as ontologias e os mapeamentos, e um processador de consultas.

3 Sistemas relacionados com Grid

Na literatura são encontradas diversas propostas e sistemas voltados para o problema de integração de dados heterogêneos e distribuídos, como por exemplo: MOMIS, IBIS, Le Select/Agora, Prometheus, MOCHA, CoDIMS/CoDIMS-G, HEROS (sistemas acadêmicos); DB2ii, Denodo (sistemas comerciais [1].) etc.

Este capítulo tem por objetivo apresentar/analisar as arquiteturas de alguns dos principais sistemas existentes, relacionados com Grid. Além destes, outros foram estudados, mas não serão mostrados devido a utilizarem outras abordagens ou por serem sememlhantes aos sistemas aqui apresentados.

3.1 CoDIMS-G

O CoDIMS-G(FONTES et al., 2004) é um sistema de integração de dados e programas gerado a partir de uma configuração do CoDIMS(BARBOSA; PORTO; MELO, 2002). Também pode ser visto como um serviço de integração de dados e programas para Grid o qual fornece para os usuários um acesso transparente para dados e programas distribuídos no Grid, assim como gerenciamento e alocação de recursos dinâmicos. Nele também é proposto um novo algoritmo de escalonamento de nodos e projetada uma máquina de consulta distribuída adaptativa para o ambiente de Grid. Assim, no CoDIMS-G os usuários podem expressar consultas que combinam processamento distribuído de dados com a invocação de programas sobre os mesmos.

A arquitetura do CoDIMS-G pode ser observada na figura 8. Nela, o componente Controle (*Control*) que é a essência do ambiente CoDIMS (BARBOSA, 2001), armazena, gerencia, valida e verifica uma instância de configuração. O Controle orquestra a comunicação entre os componentes. Usuários acessam o serviço através uma aplicação *Web*. A

[1] É importante ressaltar que os sistemas comerciais foram influenciados diretamente por resultados de pesquisas no meio acadêmico. A vasta maioria dos produtos de integração de dados foram desenvolvidos por membros da comunidade de pesquisa, seja na forma de criação de companhias, seja por comercialização de tecnologias a partir de laboratórios de pesquisa industrial (HALEVY, 2003)

requisição de um usuário é encaminhada para o componente Controle, que a envia para o componente Analisador (*Parser*). Este transforma a requisição do usuário em uma representação de grafo de consulta. O Otimizador de Consulta (*Query Optimizer* (QO)) recebe o grafo e gera um plano de execução de consulta distrubuído (DQEP), usando um modelo de custo baseado em estatísticas de dados e programas armazenadas no Gerenciador de Metadados (*Metadata Manager* (MM)). Para DQEPs que incluem operações para a avaliação de programas de uso paralelo, o otimizador chama o componente Escalonador (*Scheduler* (SC)). Este último acessa o MM para capturar o histórico de desempenho de execução de nodos do grid e custos de avalização de programas usuário, além de estatísticas dos resultados intermediários. Baseado nas estatísticas coletadas, o SC aplica um algoritmo de escalonamento de nodos do grid, para encontrar um sub-conjunto de nodos a serem alocados pelo gerenciador da máquina de execução (*Query Engine Manager* (QEM)), onde serão executados os programas do usuário. O QEM é responsável por implantar os serviços da máquina de execução de consulta (QE) nos nodos especificados no DQEP, e gerenciar os seus ciclos de vida durante a execução da consulta. O QEM gerencia o desempenho, em tempo real, dos QEs por consultar as estatísticas de vazão de dados e decidir por realocação, em tempo de execução, das QEs com um plano re-otimizado. Os QEs são implementados como instâncias do *framework* de execução de consulta (AYRES; PORTO; MELO, 2004). Cada QE recebe um fragmento de um completo DQEP, e é responsável pela instanciação dos operadores e pelo controle de execução, incluindo a comunicação com outros fragmentos para recuperação das tuplas. Como parte do DQEP, o operador *scan* acessa as fontes de dados com *wrappers* que preparam os dados de acordo com o formato de dados comum.

O CoDIMS-G está voltado para o desenvolvimento de soluções para problemas de aplicações científicas, tal como o desenvolvimento de um serviço para suportar um estágio de processamento de aplicações de visualização científica que simulam um fluxo de fluído através de um caminho (PORTO et al., 2004). Assim, o CoDIMS-G não é uma maneria mais geral de adaptar o CoDIMS ao ambiente de Grid.

3.2 OGSA-DAI

O projeto OGSA-DAI(*Open Grid Service Architecture Data Access and Integration*) (OPEN..., 2004a) visa a construção de um *middleware* para ajudar no acesso e integração de dados a partir de fontes de dados via grid. Ele permite a representação de banco de dados como serviços grid, que são então acessíveis a partir de outras máquinas de uma

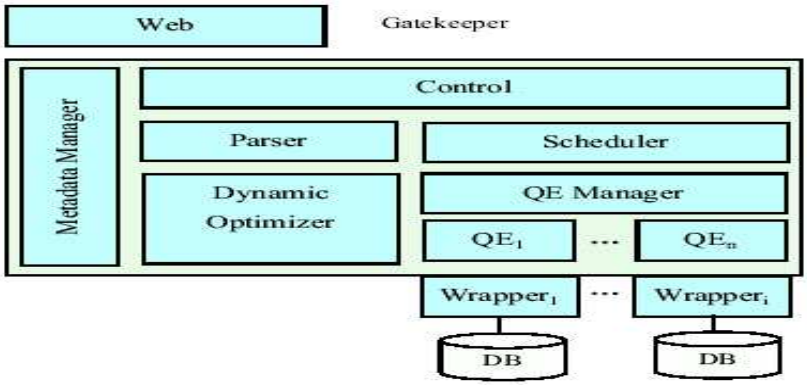

Figura 8: Arquitetura do CoDIMS-G.

maneira segura e seamless (transparente, sem emenda)????. Com isso o OGSA-DAI possibilita expor as fontes de dados para o mundo baseado na abordagem de serviços grid OGSA (FOSTER, 2002) sendo também baseado em cima do Globus Tookit (GLOBUS, 2004). A figura 9 descreve a arquitetura de serviço do OGSA-DAI.

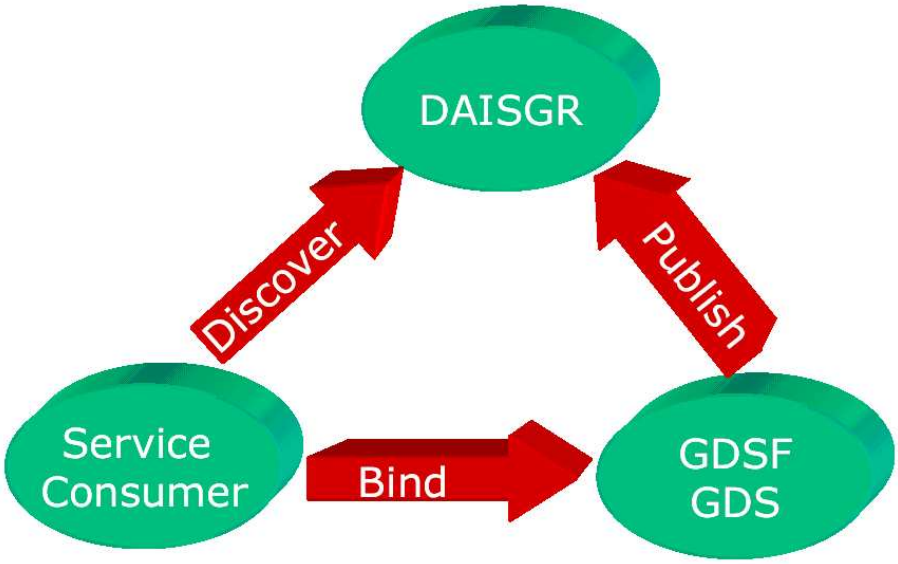

Figura 9: Arquitetura de Serviço do OGSA-DAI.

OGSA DAI suporta vários recursos de dados homogêneos ou heterogênes, tal como banco de dados relacional, XML e até mesmo arquivos de sistema, para serem acessados através de uma interface de acesso a dados baseado em um serviço Web (QI, 2004).

Três principais tipos de serviços são definidos para descoberta e acesso a recursos: *Data Access and Integration Service Group Registry*(DAISGR), *Grid Data Services Factories*(GDSF) e o *Grid Data Services*(GDS). O DAISGR é um serviço persistente que fornece a informação necessária para clientes descobrirem serviços e recursos dinamica-

mente. O DAISGR lista as GDSFs que fornecem um método persistente para a criação de GDSs. A figura 10 representa este procedimento.

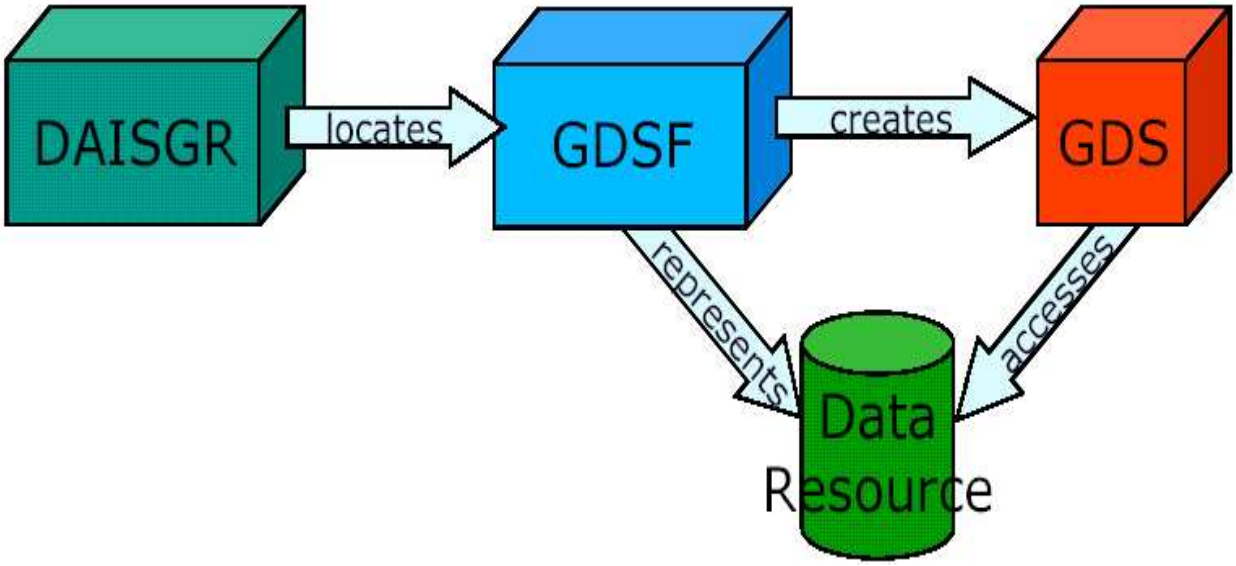

Figura 10: Serviços OGSA-DAI.

O GDS fornece uma representação transiente e *stateful* de alguns recursos de dados ou serviços sob os quais um número de atividades pre-definidas podem ser executadas (MUNRO, 2004). Ele representa um link primário para o banco de dado sendo acessado, podendo ser usado para funções de acesso, integração e entrega de dados. Assim, uma instância de GDS pode interagir com um banco de dados para criar, recuperar, atualizar ou remover dados. O GDS é a contribuição central do OGSA-DAI (ALPDEMIR et al., 2003b). O cliente interage com o GDS por enviar um documento escrito em XML chamado de *Perform*. Este documento contém descrições de uma ou mais atividades que o GDS deverá executar. A figura 11 ilustra o *framework* básico do OGSA-DAI para a interação com um cliente.

O OGSA-DAI, pode ser visto como um *wrapper*, possuindo funcionalidades como: comunicação com a fonte de dados; comunicação entre diferentes OGSA-DAIs; importação do esquema da fonte a qual está ligado; capacidades de criar, recuperar, atualizar ou remover dados; além de desfrutar das vantagens de estar em um ambiente de grid. Contudo, ele não possui a funcionalidade de tradução entre modelos, ou seja, ele tem que receber a consulta já no modelo da fonte a qual ele está ligado. Além disso, o mesmo não é capaz de notificar, de uma maneira automática, alterações feitas nos esquemas das fontes, o que é um problema, dado que o usuário irá fazer consultas com base nos esquemas antigos, podendo gerar erros que vão desde a acessar um campo que não mais existe ou que existe

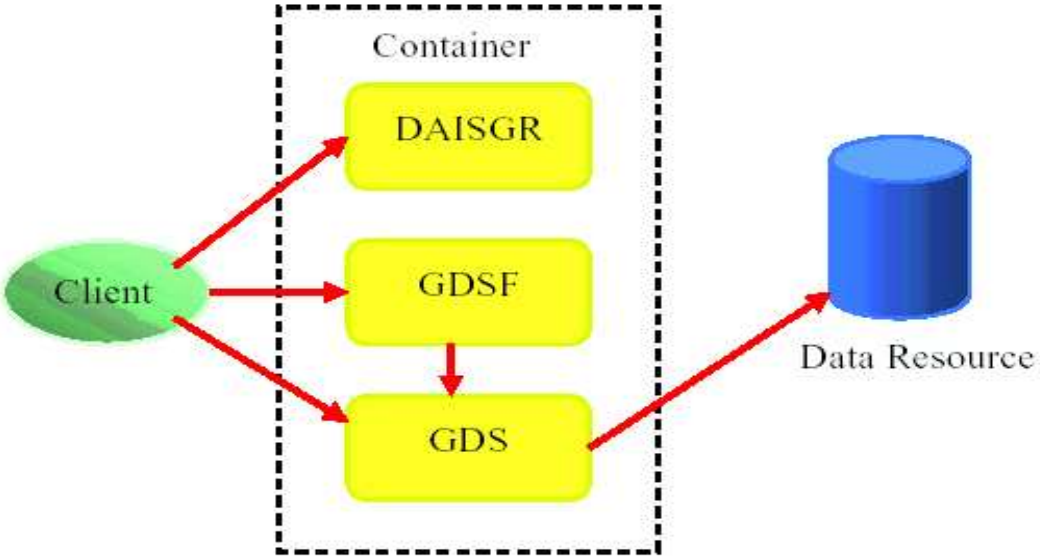

Figura 11: *Framework* OGSA-DAI: Interação cliente - OGSA-DAI.

com um novo nome, ou mesmo não pesquisar novos campos adicionados que poderiam ser de grande utilidade.

3.3 OGSA-DPQ

OGSA-DPQ (*Open Grid Service Architecture - Distributed Query Processor*) (OPEN..., 2004b) é um *framework* de integração de dados e orquestração de serviços no qual é possível fazer a coordenação e a incorporação de serviços web para efetuar análise de dados dentro da recuperação de dados (ALPDEMIR et al., 2003a).

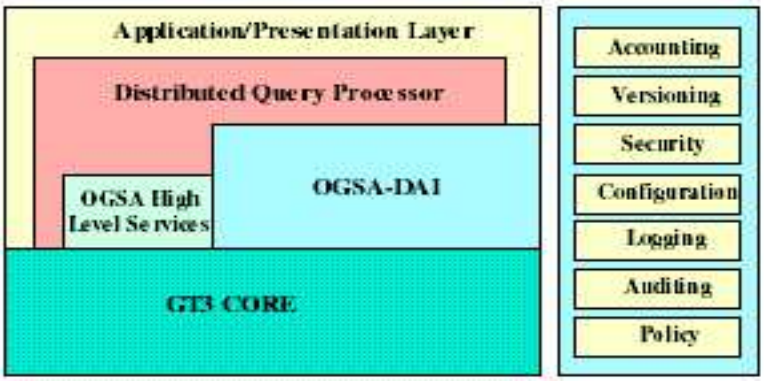

Figura 12: Camada arquitetural do OGSA-DPQ.

Como pode ser observado na figura 12 o OGSA-DPQ usa os serviços oferecidos pelo *framework* OGSA-DAI para acessar, de forma homogênea, as fontes de dados potencial-

mente heterogênas. Em um nível menor temos a camada GT3, que é usada tanto pelo OGSA-DAI quanto pelo OGSA-DPQ para criação de instâncias, acesso do estado e gerenciamento do tempo de vida das instâncias de serviços. A figura 13 mostra a interação entre os *frameworks* OGSA-DPQ e OGSA-DAI.

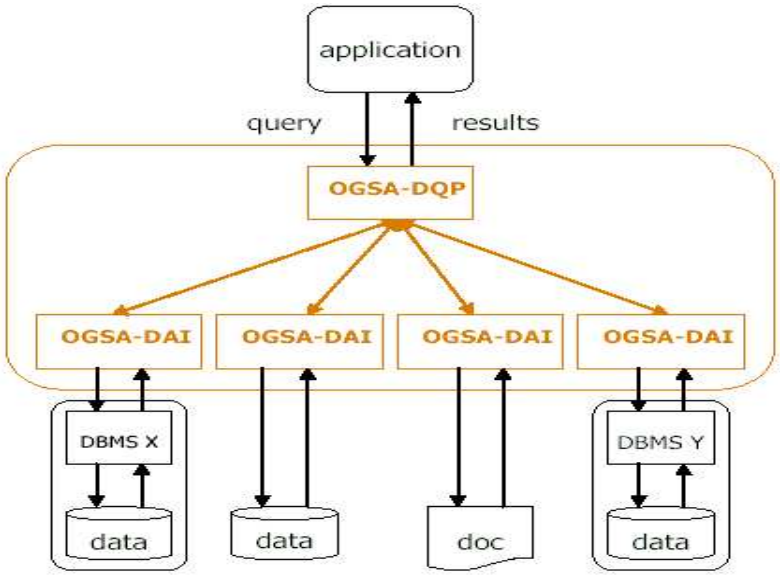

Figura 13: Interação entre os *frameworks* OGSA-DPQ e OGSA-DAI.

No OGSA-DPQ, instâncias de serviço de execução de consulta são implantadas em nodos de grid para implementar paralelismo de programas de usuário. Operadores algébricos, tal como, *exchange* e *operation-call*, implementam comunicação entre nodos e invocação de programas de usuário, respectivamente (FONTES et al., 2004).

Para cumprir com as suas funções, OGSA-DPQ fornece dois serviços: O *Grid Distributed Query Service* (GDQS) e o *Grid Query Evaluation Service* (GQES). O GDQS é construído sobre o Polar* (SMITH et al., 2002) encapsulando as funcionalidades de compilação e otimização. O GDQS fornece uma interface primária para o usuário e atua como um coordenador entre a máquina de otimização/compilação de consultas e as instâncias GQES. O GQES, por outro lado, é usado para avaliar um sub-plano de consulta associado para ele pelo GDQS. O número de instâncias GQES e sua localização no grid é especificado pelo GDQS baseado nas decisões tomadas pelo otimizador e representado como um escalonamento para partições de consulta (sub-planos). As instâncias GDES são criadas e escalonadas dinamicamente e suas interações coordenadas pelo GDQS.

Um cliente, assim como no OGSA-DAI, pode submeter uma consulta embutida dentrode um documento XML chamado *Perform*. Essa consulta é compilada e otimizada dentro de um plano de execução de consulta, cujas partições são escalonadas para serem

executadas em diferentes instâncias GQES. O GDQS usa esta informação para criar instâncias GQES em seus nodos de execução e dispacha os sub-planos para serem executados em seus respectivos GQES. GQES podem interagir com outras instâncias GDS para obter dados. Os resultados se propagam através das instâncias GQES até alcançarem o cliente. Esta visão geral das interações durante uma execução de consulta poder ser observada na figura 14.

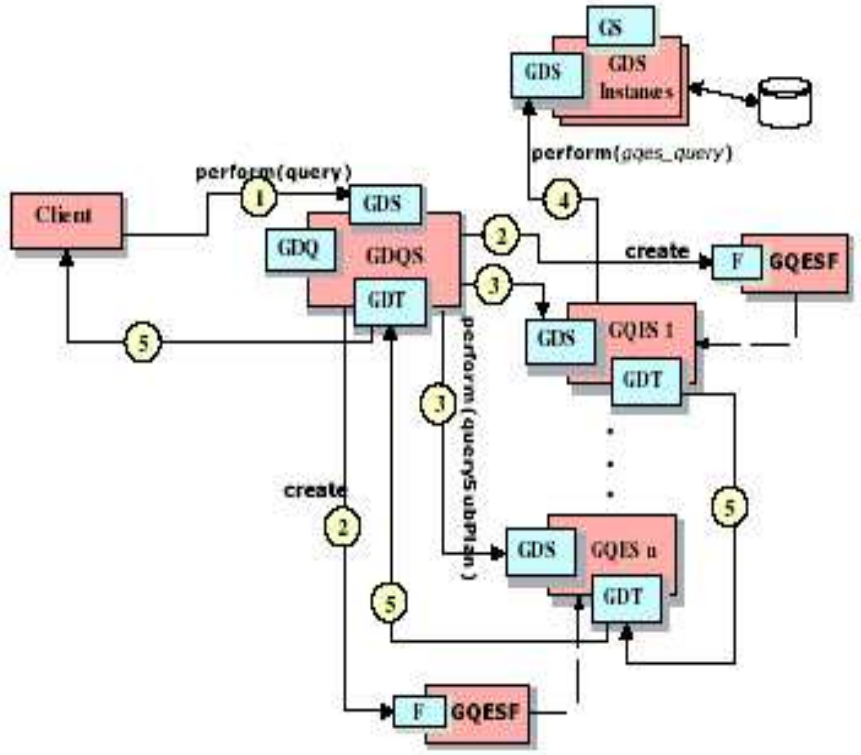

Figura 14: Interações durante uma execução de consulta.

Assim como o OGSA-DAI, o OGSA-DPQ possui algumas restrições que devem ser explicadas. Dentre elas estão as que foram herdadas do próprio OGSA-DAI, já que o mesmo é utilizado pelo OGSA-DPQ, e outras pertencentes somente ao OGSA-DPQ. Uma delas está relacionada com a obtenção dos esquemas das fontes de dados, pois nenhuma integração de esquema e resolução de conflitos é suportada durante a importação dos esquemas. Os esquemas importados são simplesmente acumulados e mantidos localmente. Isso dificulta a vida do usuário final, dado que ele terá grandes dificuldades para integrar "visualmente"os esquemas e resolver conflitos semânticos e/ou estruturais entre os mesmos. Um outra restrição é que o plano de execução de consulta gerado pelo OGSA-DPQ, o qual é estático (FONTES et al., 2004). Além disso, existe a possibilidade de um ou mais nodos alocados para este plano estático ficarem sobrecarregados, ou a fonte com a qual eles se comunicam não está mais respondendo. Neste sentido, no OGSA-DPQ, não é

possível fazer uma análise, em tempo de execução, dos nodos alocados no grid para que, se necessário, seja feita uma realocação dos mesmos e a geração de um novo plano.

3.4 MOCHA

MOCHA *Middleware Based On a Code SHipping Architecture* (MIDDLEWARE...,) é um sistema *middleware* projetado para interconectar fontes de dados distribuídas sobre uma grande área de rede (RODRIGUEZ-MARTINEZ; ROUSSOPOULOS, 2000a). MOCHA é construído baseado na noção de que um *middlware* para um ambiente distribuído de larga escale deveria ser auto-extensível (RODRIGUEZ-MARTINEZ; ROUSSOPOULOS, 2000b). Isso significa que novos tipos de dados e operadores específicos de uma aplicação necessário para o processamento de uma consulta são implantados para *sites* remotos de uma maneira automática pelo sistema *middleware*. No MOCHA isto é realizado através do envio de classes Java implementando estes tipos ou operadores para o *site* remoto, onde eles possam ser usados para manipular dados de seu interesse. Todas as classes Java são primeiro armazenadas em um ou mais repositórios de dados, a partir do qual são posteriormente recuperados e implantados baseado em sua necessidade. O objetivo principal da idéia de implantação automática de código é suprir a necessidade por componente de processamento específicos da aplicação em *sites* remotos que não os fornece. Assim, MOCHA se capitaliza em sua habilidade de implantação de código automática para fornecer um serviço de processamento de consulta eficiente. Por enviar código para operadores de consulta, MOCHA pode produzir planos eficientes que colocam a execução de poderosos operadores de redução de dados (filtros) nas fontes de dados diminuindo com isso o tráfego que dados entre o *site* e o sistema.

A motivação do MOCHA é que, os dados armazenados em diversos *sites* são baseados em tipos de dados complexos e que a *Web* tem se tornado, de fato, uma interface de usuário para aplicações em rede. Assim, os usuários necessitam de uma solução que permita, facilmente, integrar os clientes baseados em *Web* e visualizar os dados disponibilizados pelos mesmos.

A arquitetura do MOCHA, mostrada na figura 15, é composta de quatro componente principais: Aplicação Cliente - um *applet*, *servlet* ou uma aplicação Java usada para submeter consultas para o sistema.; Coordenador de Processamento de Consultas (QPC) - fornece serviços tal como análise de consulta, otimização de consulta, escalonamento de operador de consulta, gerenciamento de catálogo e execução de consulta, além de ser

o responsável por implantar todas as funcionalidades necessárias para o cliente e para os *sites* remotos a partir dos quais os dados serão extraídos.; Fornecedor de Acesso aos dados (DAP) - fornece ao QPC um mecanismo de acesso uniforme para uma fonte de dados remota, e executa alguns dos operadores na consulta (aqueles que filtram os dados sendo acessados).; e o Servidor de Dados - armazena um conjunto de dados para um site em particular.

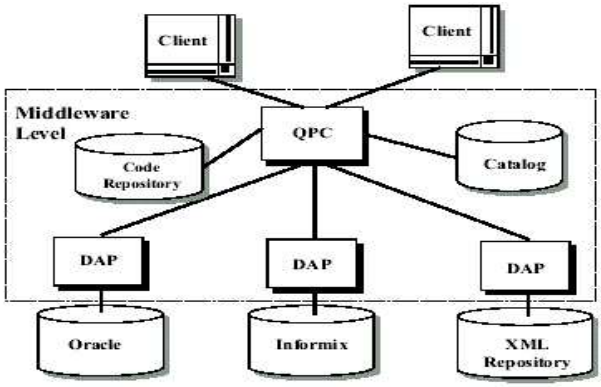

Figura 15: Arquitetura do MOCHA.

No MOCHA, pode-se enxergar os DAP como sendo *wrappers*, possuindo funcionalidades mais elaboradas do que apenas fazer a tradução entre modelos e a comunicação com as fontes de dados. O mesmo tem capacidade de processamento de consulta, podendo utilizar tipos de dados e operadores específicos de uma aplicação, necessários para o processamento de uma consulta, os quais são implantados nos DAPs de uma maneira automática pelo sistema *middleware*. Contudo, os mesmos não possuem a capacidade de se comunicarem entre si para promover o paralelismo na execução da consulta e com isso aumentar a eficiência do sistema. Além disso, os DAPs são previamente instalados em máquinas, no próprio *site* remoto ou próximo a ele, onde irão executar. Como as máquinas onde os DAPs estão instalados podem ficar sobrecarregadas e os DAPs não podem ser reinstalados em tempo de execução, o sistema integrador poderá ter sua execução afetada, dado que o QPC poderá ficar parado esperando os dados proveniente de um DAP que está com o processamento lento ou até mesmo parado.

Referências

ABITEBOUL, S. et al. *The lowell database research self assessment*. May 2003. Disponível em: <http://research.microsoft.com/~Gray/lowell/>. Acesso em: 06 set. 2004.

ALPDEMIR, N. et al. Ogsa-dqp: A service-based distributed query processor for the grid. In: . [S.l.]: UK e-Science All Hands Meeting Nottingham. EPSRC, 2003.

ALPDEMIR, N. et al. Service-based distributed querying on the grid. In: *1st International Conference on Service Oriented Computing (ISOC)*. [S.l.]: Springer Verlag, 2003. p. 467–482.

ARENS, Y. et al. Retrieving and integrating data from multiple information sources. *International Journal of Cooperative Information Systems*, v. 2, n. 2, p. 127–158, 1993. Disponível em: <citeseer.ist.psu.edu/arens93retrieving.html>.

ARENS, Y.; KNOBLOCK, C. A. Planning and reformulating queries for semantically-modeled multidatabase systems. In: *Proc. of the ISMM International Conference on Information and Knowledge Management CIKM-92*. Baltimore, MD: [s.n.], 1992. p. 92–101.

ARENS, Y.; KNOBLOCK, C. A.; SHEN, W.-M. Query reformulation for dynamic information integration. *J. Intell. Inf. Syst.*, Kluwer Academic Publishers, v. 6, n. 2-3, p. 99–130, 1996. ISSN 0925-9902.

AYRES, F.; PORTO, F.; MELO, R. An extensible query execution engine for supporting new query execution models. In: . [S.l.]: To appear in LNCS, 2004.

BARBOSA, A. C. P. *Middleware para Integração de Dados Heterogêneos Baseado em Composição de Frameworks*. Tese (Doutorado) — PUC-Rio, Brasil, Maio 2001. In Portuguese.

BARBOSA, A. C. P.; PORTO, F.; MELO, R. N. Configurable data integration middleware system. *Journal of the Brasilian Computer Society*, p. 12–19, 2002.

BENEVENTANO, D.; BERGAMASCHI, S. The momis methodology for integrating heterogeneous data sources. In: *IFIP World Computer Congress*. Tolouse, France: [s.n.], 2004. Disponível em: <http://www.dbgroup.unimo.it/prototipo/paper/ifip2004.pdf>. Acesso em: 10 nov. 2004.

BENEVENTANO, D. et al. The MOMIS approach to information integration. In: *ICEIS (1)*. [s.n.], 2001. p. 194–198. Disponível em: <citeseer.ist.psu.edu/beneventano01momis.html>. Acesso em: 06 set. 2004.

BENEVENTANO, D. et al. *MOMIS: Overview*. 2001. Disponível em: <http://dbgroup.unimo.it/Momis/tour/two.html>.

BRACHMAN, R. J.; SCHMOLZE, J. G. An overview of the kl-one knowledge representation system. *Cognitive Science*, v. 9, n. 2, p. 171–216, 1985.

CASTANO, S. et al. Ontology-based integration of heterogeneous xml datasources. In: *Proc. of 10th Italian Symposium on Advanced Database Systems - SEBD'02*. Isola d'Elba, Italy: [s.n.], 2002. p. 27–41.

DECKER, S. et al. Ontobroker: Ontology based access to distributed and semi-structured information. In: AL., R. M. et (Ed.). *Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand*. Kluwer Academic Publishers, Boston, 1999. Disponível em: <http://www.aifb.uni-karlsruhe.de/WBS/Publ/1999/ds8_sdeetal_1999.pdf>.

DECKER, S. et al. *The Technical Core of Ontobroker*. Disponível em: <citeseer.csail.mit.edu/149833.html>.

FENSEL, D. et al. *Ontobroker: How to make the WWW intelligent*. 76128 Karlsruhe, Germany, JAN 1998.

FONTES, V. et al. Codims-g: a data and program integration service for the grid. In: *Proceedings of the 2nd Workshop on Middleware for Grid Computing*. [S.l.]: ACM Press, 2004. p. 29–34.

FOSTER, I. *What is the Grid? A Three Point Checklist*. July 2002. GRIDToday. Disponível em: <http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>.

GLOBUS. *The Globus Toolkit*. 2004. Disponível em: <www.globus.org>. Acesso em: 30 nov. 2004.

GOBLE, C. A. et al. Transparent access to multiple bioinformatics information sources. *IBM Syst. J.*, IBM Corp., v. 40, n. 2, p. 532–551, 2001. ISSN 0018-8670.

HAKIMPOUR, F.; GEPPERT, A. Resolving semantic heterogeneity in schema integration: an ontology based approach. In: *Proceedings of the international conference on Formal Ontology in Information Systems*. USA: [s.n.], 2001. v. 2001, p. 297–308.

HALEVY, A. Y. Data integration: A status report. In: *Proceedings of 10th Conference on Database Systems for Business Technology and the Web (BTW 2003)*. Germany: [s.n.], 2003.

KARNSTEDT, M. et al. Semantic caching in ontology-based mediator systems. In: *Proceedings of the 3rd International Workshop of the GI Working Group "'Web und Datenbanken"', Berliner XML-Tage 2003, Berlin, 13. Oktober 2003*. [S.l.: s.n.], 2003. p. 155–169.

KIFER, M.; LAUSEN, G.; WU, J. *Logical Foundations of Object-Oriented and Frame-Based Languages*. [S.l.], 1990.

LITWIN, W.; MARK, L.; ROUSSOPOULOS, N. Interoperability of multiple autonomous databases. *ACM Comput. Surv.*, ACM Press, v. 22, n. 3, p. 267–293, 1990. ISSN 0360-0300.

MENA, E. et al. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In: *Conference on Cooperative Information Systems*. [s.n.], 1996. p. 14–25. Disponível em: <citeseer.ist.psu.edu/mena96observer.html>. Acesso em: 13 set. 2004.

MIDDLEWARE Based On a Code SHipping Architecture (MOCHA). Disponível em: <http://www.cs.umd.edu/projects/mocha/>. Acesso em: 03 dec. 2004.

MILLER, R. J. et al. The clio project: Managing heterogeneity. v. 30, n. 1, p. 78–83, March 2001. Disponível em: <citeseer.ist.psu.edu/miller01clio.html>.

MUNRO, C. *Developing a C Client Toolkit for OGSA-DAI*. Dissertação (Mestrado) — University of Edinburgh, 2004.

OMG, O. M. G. *CORBA - Common Request Brokering Architecture*. 1997. Disponível em: <http://www.corba.org/>. Acesso em: 25 set. 2004.

OPEN Grid Services Architecture - Data Access and Integration - OGSA-DAI. 2004. Disponível em: <http://www.ogsadai.org.uk>. Acesso em: 30 nov. 2004.

OPEN Grid Services Architecture - Distributed Query Processor - OGSA-DPQ. 2004. Disponível em: <http://www.ogsadai.org.uk/dqp/>. Acesso em: 02 dec. 2004.

PEIM, M. et al. Query processing with description logic ontologies over object-wrapped databases. In: *Proceedings of the 14th International Conference on Scientific and Statistical Database Management, year = 2002, pages = 27-36, ee = http://computer.org/proceedings/ssdbm/1632/16320027abs.htm, bibsource = DBLP, http://dblp.uni-trier.de, publisher = IEEE Computer Society*. [S.l.: s.n.].

PORTO, F. et al. *CoDIMS - A Middleware System to Support Visualization Applications in a Grid*. Petrópolis, RJ, Brazil, 2004.

QI, K. *Data integration scenarios in OGSA-DAI*. Dissertação (Mestrado) — University of Edinburgh, 2004.

RODRIGUEZ-MARTINEZ, M.; ROUSSOPOULOS, N. Automatic deployment of application-specific metadata and code in mocha. In: *EDBT'00: Proceedings of the 7th International Conference on Extending Database Technology*. [S.l.]: Springer-Verlag, 2000. p. 69–85. ISBN 3-540-67227-3.

RODRIGUEZ-MARTINEZ, M.; ROUSSOPOULOS, N. Mocha: a self-extensible database middleware system for distributed data sources. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. [S.l.]: ACM Press, 2000. p. 213–224. ISBN 1-58113-217-4.

SHETH, A. P.; LARSON, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, ACM Press, v. 22, n. 3, p. 183–236, 1990. ISSN 0360-0300.

SILBERSCHATZ, A.; ZDONIK, S. Database systems-breaking out of the box. *SIGMOD Rec.*, ACM Press, v. 26, n. 3, p. 36–50, 1997. ISSN 0163-5808.

SMITH, J. et al. Distributed query processing on the grid. In: *Proceedings of the Third International Workshop on Grid Computing*. [S.l.]: Springer-Verlag, 2002. p. 279–290. ISBN 3-540-00133-6.

SUWANMANEE, S.; BENSLIMANE, D.; THIRAN, P. Owl-based approach for semantic interoperability. In: *Preceedings of AINA 2005*. [S.l.]: IEEE Computer Science Press, 2005.

WACHE, H. et al. *Ontology-based integration of information - a survey of existing approaches*. 2001. Disponível em: <citeseer.ist.psu.edu/wache01ontologybased.html>.

ZIEGLER, P.; DITTRICH, K. R. Three decades of data integration - all problems solved? In: JACQUART, R. (Ed.). *18th IFIP World Computer Congress (WCC 2004), Volume 12, Building the Information Society*. Toulouse, France, August 22-27: Kluwer, 2004. (IFIP International Federation for Information Processing, v. 156), p. 3–12.