FERNANDO RYOJI KAKUGAWA

INTEGRAÇÃO DE BANCOS DE DADOS HETEROGÊNEOS
UTILIZANDO GRADES COMPUTACIONAIS

São Paulo
2010

FERNANDO RYOJI KAKUGAWA

INTEGRAÇÃO DE BANCOS DE DADOS HETEROGÊNEOS
UTILIZANDO GRADES COMPUTACIONAIS

Dissertação apresentada a Escola
Politécnica da Universidade de São Paulo
para a obtenção do título de Mestre em
Engenharia Elétrica

São Paulo
2010

FERNANDO RYOJI KAKUGAWA

INTEGRAÇÃO DE BANCOS DE DADOS HETEROGÊNEOS
UTILIZANDO GRADES COMPUTACIONAIS

Dissertação apresentada a Escola
Politécnica da Universidade de São Paulo
para a obtenção do título de Mestre em
Engenharia Elétrica

Área de concentração:
Engenharia de Computação e Sistemas
Digitais

Orientadora:
Profa. Doutora Liria Matsumoto Sato

São Paulo
2010

Aos três anjos que Deus colocou em minha vida

AGRADECIMENTOS

Agradeço a Deus pela graça da vida, a oportunidade de realizar um sonho, fazer aquilo que sinto muito prazer e por colocar três anjos em minha vida. O primeiro anjo é a minha mãe Rosa Kakugawa pelos valores na vida e a procura de fazer sempre o que é correto. O segundo anjo, a minha esposa Andréa Kakugawa pelo suporte emocional e incentivo incondicional para a realização deste trabalho. O terceiro anjo, a minha orientadora Profa. Liria Sato pelos seus ensinamentos, a sábia condução deste trabalho e a sua dedicação aos seus alunos.

As pessoas que foram fundamentais durante esta jornada, Francisco Massetto e Calebe Bianchini por semearem a chama do mundo acadêmico dentro de mim durante a graduação, Augusto Gomes Jr por acreditar no meu potencial e me apresentar ao LAHPC, Darlon Vasata e Mathias Brito pela afeição que nasceu no convívio dentro do laboratório. A amizade que tenho deles é um grande legado produzido por este trabalho.

Ao Prof. Mario Donato, por me receber no laboratório e iniciar os primeiros passos. Costumo dizer que durante o processo de entrada na Poli/USP para trabalhar no LAHPC, o Augusto abriu as portas, o Prof. Mario me recebeu e a Profa. Liria me adotou.

Ao LAHPC pela infraestrutura oferecida e aos demais membros os quais não estão citados aqui, mas sem sombra de dúvidas contribuíram com este trabalho construindo um ambiente de solidariedade e muito bom humor.

“Para cultivar a sabedoria, é preciso força interior. Sem crescimento interno, é difícil conquistar a autoconfiança e a coragem necessária. O impossível torna-se possível com a força de vontade.”

Tenzin Gyatso

RESUMO

Bancos de dados normalmente são projetados para atender a um domínio específico de uma aplicação, tornando o acesso aos dados limitado e uma tarefa árdua em relação à integração de bancos e compartilhamento de dados. Existem várias pesquisas no intuito de integrar dados, como a criação de softwares específicos para uma determinada aplicação e até soluções mais radicais como refazer todos os bancos de dados envolvidos, demonstrando que ainda existem questões em aberto e que a área está longe de atingir soluções definitivas. Este trabalho apresenta conceitos e estratégias para a integração de bancos de dados heterogêneos e a implementa na forma do DIGE, uma ferramenta para desenvolver sistemas de banco de dados integrando diferentes bancos de dados relacionais heterogêneos utilizando grades computacionais. O sistema criado permite o compartilhamento de acesso deixando os dados armazenados em seu local de origem, desta forma, usuários do sistema acessam os dados em outras instituições com a impressão de que os dados estão armazenados localmente. O programador da aplicação final pode acessar e manipular os dados de forma convencional utilizando a linguagem SQL sem se preocupar com a localização e o esquema de cada banco e o administrador do sistema pode adicionar ou remover bancos de forma facilitada sem a necessidade de solicitar alterações na aplicação final.

Palavras-chave: Sistemas Distribuídos, Bancos de Dados Distribuídos, Grade Computacional, Conversão de Esquema.

ABSTRACT

Databases are usually designed to support a specific application domain, thus making data-access and data-sharing a hard and arduous task when database integration is required. Therefore, research projects have been developed in order to integrate several and heterogeneous databases systems, such as specific-domain application tools or even more extreme solutions, as a complete database redefinition and redesign. Considering these open questions and with no definite answers, this work presents some concepts, strategies and an implementation for heterogeneous databases integration. In this implementation, the DIGE tool was developed to provide access to heterogeneous and geographically distributed databases, using Grid computing, which store locally its data, appearing to the user application, the data are stored locally. By this way, programmers can manipulate data using conventional SQL language with no concern about database location or its schema. Systems Administrators may also add or remove databases on the whole system without need to change the final user application.

Keywords: Distributed Systems, Distributed Databases, Grid Computing, Schema Conversion

LISTA DE ILUSTRAÇÕES

Figura 2.1 – Camadas de uma arquitetura em grade.....	23
Figura 2.2 – Funcionamento Genérico do OGSA-DAI.....	27
Figura 2.3 - Funcionamento do OGSA-DAI.....	28
Figura 3.1 – Arquitetura CoDIMS	30
Figura 3.2 – Arquitetura GISE	31
Figura 3.3 – Visão geral do OGSA-DQP	32
Figura 3.4 - Arquitetura do GDIS.....	33
Figura 3.5 – Arquitetura Mediador-Adaptadores	34
Figura 3.6 – A Arquitetura hierárquica aplicada no projeto	35
Figura 4.1 – Um Ambiente Integrando Bancos de Dados Distribuídos	40
Figura 4.2 – A Arquitetura Proposta.....	42
Figura 4.3 – Exemplo de um ambiente de execução	44
Figura 4.4 – Execução de uma consulta	45
Figura 5.1 – Arquitetura DIGE.....	48
Figura 5.2 – Workflow padrão OGSA-DAI para consulta em um banco de dados	50
Figura 5.3 – Workflow executado pelo SelectQCONG.....	51
Figura 5.4 – Interligando activities para o SelectQCONG	52
Figura 5.5 – Arquivo de configuração para expor um recurso.....	53
Figura 5.6 – Arquivo de configuração para um conjunto de recursos	54
Figura 5.7 – Execução de consulta em banco exposto no mesmo SA.....	55
Figura 5.8 – Delegando a execução da consulta para outro SA	55
Figura 5.9 – Arquitetura do QCON.....	57
Figura 5.10 – Exemplo de consulta para QCON	57
Figura 5.11 – Análise de uma consulta com QCON.....	57
Figura 5.12 – Decomposição de uma consulta pelo QCON.....	58
Figura 5.13 – Arquivo de configuração para o mapeamento de esquema	58
Figura 5.14 – Decomposição de uma consulta pelo QCON.....	59
Figura 5.15 – Nova consulta gerada pelo QCON.....	59
Figura 6.1 – Configuração Utilizada nos Testes.....	60
Figura 6.2 – Arquivo de configuração para mapeamento do recurso sp1	64
Figura 6.3 – Tempo de execução <i>select</i> para um banco utilizando acesso direto	66

Figura 6.4 – Tempo de execução <i>select</i> para um banco delegando a consulta para outro SA	67
Figura 6.5 – Tempo de execução de um <i>select</i> em mais de um banco utilizando acesso direto	68
Figura 6.6 – Tempo de execução de um <i>select</i> para mais de um banco delegando a consulta para outro SA.....	70
Figura 6.7 – Comportamento dos tempos de execução utilizando acesso direto	70
Figura 6.8 - Comportamento dos tempos de execução delegando a consulta a outro SA	71
Figura 6.9 – Tempo de execução para <i>delete</i> , <i>insert</i> e <i>update</i> , utilizando banco Postgres na máquina GAS.....	72
Figura 6.10 - Tempo de execução para <i>delete</i> , <i>insert</i> e <i>update</i> , utilizando banco MySQL na máquina GAS	72
Figura 6.11 – Topologia da Aplicação	75
Figura 6.12 – Funcionamento recursos sao paulo e campinas	75
Figura 6.13 – Funcionamento recurso <i>ambos</i>	76
Figura 6.14 – Tela de login do sistema	76
Figura 6.15 – Resultado de uma Consulta	77

LISTA DE TABELAS

Tabela 4.1 – Resumo das características dos trabalhos relacionados.....	46
Tabela 5.1 – Bancos de dados compatíveis.....	48
Tabela 6.1 – Máquinas e Ferramentas utilizadas.....	63
Tabela 6.2 - Características de cada recurso.....	63
Tabela 6.3 – Modelo global adotado	64
Tabela 6.4 – Consultas baseadas no esquema global utilizado pelo DIGE	64
Tabela 6.5 – Desvio padrão para os tempos da Figura 6.3.....	66
Tabela 6.6 – Desvio padrão para os tempos da Figura 6.4.....	67
Tabela 6.7 – Desvio padrão para os tempos da Figura 6.5.....	69
Tabela 6.8 – Desvio padrão para os tempos da Figura 6.6.....	70
Tabela 6.9 – Desvio padrão para os tempos da Figura 6.9.....	72
Tabela 6.10 – Desvio padrão para os tempos da Figura 6.10.....	72

LISTA DE ABREVIATURAS E SIGLAS

AntLR	Another Tool for Language Recognition
AST	Abstract Syntax Tree
CoDIMS	Configurable Data Integration Middleware
DIGE	Data Integration using Grid Environment
DIMS	Data Integration Middleware
DML	Data Manipulation Language
DQEP	Distributed Query Execution Plan
DQP	Distributed Query Processing
EPUSP	Escola Politécnica da Universidade de São Paulo
FD	Fonte de Dados
GDIS	Grid Data Integration System
GISE	Grid Integration Service
GT4	Globus Toolkit version 4
HTTP	Hyper Text Transfer Protocol
JDBC	Java Database Connectivity
JSP	Java Server Pages
LAHPC	Laboratory of Architecture and High Performance Computing
LNCC	Laboratório Nacional de Computação Científica
MM	Metadata Manager
OGF	Open Grid Forum
OGSA	Open Grid Service Architecture
OGSA-DAI	Open Grid Service Architecture - Data Access Integration
OGSI	Open Grid Service Infrastructure
P2P	Peer to Peer
QCON	Query CONverter
QCON-G	Query CONverter for Grid
QE	Query Engine
QEM	Query Engine Manager
SA	Servidor de Acesso
SGBD	Sistema Gerenciador de Banco de Dados
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structure Query Language

W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	eXtensible Markup Language

SUMÁRIO

1. INTRODUÇÃO	16
1.1. Objetivo	18
1.2. Motivação	19
1.3. Metodologia	20
1.4. Organização do Texto	21
2. CONCEITOS E TECNOLOGIAS UTILIZADAS	22
2.1. Conceitos	22
2.1.1. Grades Computacionais	22
2.1.2. Web Services	24
2.1.3. Bancos de Dados Distribuídos	24
2.1.4. Integração de Bancos de Dados	25
2.2. Tecnologias Utilizadas	26
2.2.1. Globus Toolkit	26
2.2.2. OGSA-DAI	27
2.3. Considerações do capítulo	28
3. TRABALHOS RELACIONADOS	29
3.1. CoDIMS	29
3.2. GISE	30
3.3. OGSA-DQP	32
3.4. GDIS	32
3.5. Arquitetura Mediadores e Adaptadores	33
3.6. Arquitetura de um Sistema para Integração de Bancos de Dados com Suporte a Replicação Utilizando a Tecnologia de Grades Computacionais	34
3.7. Considerações do capítulo	35
4. ARQUITETURA PARA INTEGRAÇÃO DE BANCOS DE DADOS	38
4.1. A Arquitetura Proposta	41

4.2. O Tratamento da Heterogeneidade entre Bancos.....	44
4.3. Análise Comparativa	45
5. DESCRIÇÃO DA IMPLEMENTAÇÃO	47
5.1. Componente Persistência	48
5.2. Componente Cliente	48
5.2.1. OGSA-DAI Client	49
5.2.2. Framework DIGE	50
5.2.3. Driver JDBC OGSA-DAI	52
5.3. Componente Integrador	52
5.3.1. QCON-G	55
5.3.2. QCON.....	56
6. TESTES REALIZADOS	60
6.1. Metodologia utilizada nos testes	60
6.2. Análise dos Resultados.....	65
6.3. Protótipo de uma Aplicação	73
6.3.1. Estudo de Caso	73
6.3.2. A Aplicação Desenvolvida	74
7. CONCLUSÕES e TRABALHOS FUTUROS	78
7.1. Conclusões	78
7.2. Trabalhos Futuros	79
REFERÊNCIAS.....	80
Apêndice A – Esquema global utilizado para o Hemosystem	85
Apêndice B – Bancos de dados integrados pelo Hemosystem	86
Apêndice C – Arquivos de configuração para Hemosystem	89
Apêndice D – Arquivos de configuração de recursos.....	93

1. INTRODUÇÃO

A globalização é uma forma de integração econômica, social e cultural, e em um processo contínuo vem aproximando as pessoas, interligando o mundo e diminuindo distâncias. Corroborando com este processo, a partir do advento da Internet, a possibilidade de acesso à informação atingiu novos patamares permitindo que dados espalhados geograficamente pudessem ser acessados com maior facilidade.

Bancos de dados, normalmente, são projetados para atender especificamente a um domínio de uma aplicação ou a um sistema de informação (Ozsu e Valduriez, 1999). Desta forma, apesar da disponibilidade tecnológica para o acesso a dados, o banco estará atendendo somente ao domínio específico para o qual fora designado. Somando a este fato o processo de globalização, existe uma crescente necessidade em interligar bancos de diversas instituições, integrando dados armazenados em diferentes locais como em empresas, centros de pesquisas, cidades e países diferentes, permitindo o acesso compartilhado aos dados (Elmagarmid, Rusinkiewicz e Sheth, 1999).

Neste contexto, o conceito de computação em grade (*grid computing*) (Foster e Kesselman, 2004) vem sendo muito difundido nos últimos anos, tanto no meio acadêmico como nas empresas. O maior objetivo deste conceito é atender a necessidade em utilizar sistemas de computação distribuída, possibilitando o compartilhamento e a coordenação de diversos recursos heterogêneos (computadores, instrumentos científicos, bancos de dados, etc.) distribuídos entre diferentes Organizações Virtuais (Foster, Kesselman e Tuecke, 2001).

O desafio em integrar bancos de dados heterogêneos, com o intuito de transparecer o conjunto de bancos como um único banco, permitindo o acesso compartilhado, está no desenvolvimento de mecanismos que provêm a troca de informações entre os diferentes bancos, onde cada um utiliza seu esquema para ser representado em um sistema de informação (Navathe, Elmasri e Larson, 1986). Existem várias pesquisas no intuito de integrar dados, como a criação de softwares específicos para uma determinada aplicação ou até mesmo estratégias radicais como refazer todos os bancos de dados envolvidos correndo riscos de inviabilidade

como tempo elevado para implementação e os altos custos envolvidos (Holanda, 2005) (Leão, Valente, *et al.*, 2004).

Este trabalho apresenta uma arquitetura para integração de bancos de dados heterogêneos e a implementa na forma de uma ferramenta aqui denominada DIGE (Data Integration using Grid Environment). Esta arquitetura está baseada em conceitos de grade computacional, gerando sistemas que integram diferentes bancos de dados heterogêneos, dispersos geograficamente, permitindo o acesso descentralizado, respeitando as políticas de acesso local. A ferramenta permite criar um ambiente que provê o acesso compartilhado aos dados através de configurações referentes aos bancos a serem integrados e consultas SQL convencional, utilizando um esquema global.

Essas características permitem ao usuário final deste sistema, ter seus dados, os quais estão armazenados localmente em cada instituição, disponibilizados em diferentes locais. Com a utilização desta ferramenta programadores podem implementar uma aplicação final para acessar e manipular os bancos de forma convencional utilizando a linguagem SQL sem a necessidade de se preocupar com localização e esquema de cada banco.

Para tanto, na construção do sistema utilizando a ferramenta DIGE, é necessário apenas configurar o SA (Servidor de Acesso) informando a localização e o esquema local dos bancos envolvidos. Estas características permitem o acesso ao sistema de bancos integrado através de múltiplos SA's, fornecendo através de uma interface, que pode ser disponibilizada em mais de um SA, a capacidade de realizar o acesso a diversos bancos de forma simultânea, utilizando uma única consulta, independentemente do esquema lógico local de cada banco de dados. Outro ponto relevante é que esta característica permite ao administrador do sistema integrado adicionar e ou remover bancos de forma facilitada sem a necessidade de informar as alterações ou modificar a aplicação final.

Este tipo de transparência é possível no sistema, pois a complexidade envolvida na localização e na diferença entre os esquemas lógicos são tratados em um SA, onde estas características são configuráveis. Estas configurações são armazenadas em dois arquivos, o primeiro contém a localização precisa do banco e as informações para acessá-lo, e o segundo o mapeamento do esquema global para o esquema local de cada banco. A configuração referente à localização dos bancos

é tratada no SA onde é uma consulta é requisitada, e o tratamento para o esquema lógico de cada banco é realizado no SA onde o banco está exposto.

Desta forma, a partir de uma consulta enviada ao SA é verificada a localização de todos os bancos envolvidos, e caso um destes bancos esteja exposto em outro SA, esta consulta será delegada. Ao acessar o banco para realizar a consulta, a ferramenta utiliza uma conversão de esquema, que converte uma consulta utilizando um esquema global para o esquema local, de forma compreensível ao banco de destino. Estas configurações devem ser realizadas no SA onde o banco está exposto, permitindo que consultas sejam realizadas de forma distribuída e paralela.

A abordagem adotada permite que a aplicação final não necessite sofrer alterações caso o sistema de integração gerado adicione ou remova algum banco. Desta forma a aplicação continua acessando a interface disponibilizada da mesma maneira, sem a necessidade de conhecer detalhes a respeito da quantidade de bancos envolvidos, a localização dos bancos e seus esquemas locais.

1.1. Objetivo

O objetivo deste trabalho é a proposta de uma arquitetura baseada em conceitos de grade computacional para integração de bancos de dados heterogêneos, distribuídos geograficamente. Esta arquitetura deve permitir que os diversos bancos integrados em um sistema, transpareçam como se fossem um único banco, escondendo as complexidades envolvidas tanto em sua localização como em sua heterogeneidade. Esta arquitetura foi implementada na forma de uma ferramenta que será denominada DIGE.

Tais características permitirão que aplicações construídas com base nesta ferramenta possam ter a inclusão ou a remoção de bancos de dados sem a necessidade de alterações em seu código, visto que os dados poderão ser acessados através de um esquema global, ocultando a heterogeneidade entre os esquemas locais, e permitindo o acesso ao bancos através de uma consulta a apenas um SA (Servidor de Acesso) que se necessário delegará a consulta a outros SA's, sem a necessidade de tratar no programa a localização e a forma de acesso aos bancos pertencentes ao sistema.

1.2. Motivação

A motivação deste trabalho surgiu no desenvolvimento de um projeto para integração de dados entre sistemas de saúde, mais especificamente para área de hemofilia, onde o foco principal era o compartilhamento de dados entre as diversas instituições. Como resultado do projeto, foi desenvolvida uma ferramenta para a integração de bancos de dados com suporte a replicação (Brito, 2009) (Brito, Kakugawa, *et al.*, 2009).

Normalmente, cada instituição possui seu sistema de informação produzindo seus dados de forma isolada, pois estes sistemas são projetados de forma autônoma, procurando atender necessidades específicas para o domínio de uma aplicação, sem se preocupar com hardware, software, padrões ou formas de integração tornando o ambiente heterogêneo (Leão, Valente, *et al.*, 2004).

Assim, os sistemas de informações que as autarquias públicas vêm disponibilizando para o sistema de saúde público têm atuado de forma discreta, basicamente restritos na captura de dados da população para eleger suas prioridades nas ações em gestão (Holanda, 2005). Porém a utilização destes dados visando auxiliar outras áreas como no apoio às atividades dos profissionais em saúde ou mesmo para um atendimento mais eficiente ao paciente ainda é pouco difundida (Barbosa, 2006). Para isso existe a necessidade de fazer com que cada um destes bancos de dados permita o acesso compartilhado e juntas, formem uma grande base de dados distribuídos geograficamente, procurando um atendimento mais eficiente aos pacientes. Outro aspecto interessante desta abordagem é fornecer aos os gestores administrativos informações sobre cada banco de dados, ou mesmo de forma total, onde as informações poderiam ser consolidadas.

Estas necessidades verificadas durante a execução do projeto é a motivação para a construção de uma ferramenta que integrasse bancos de dados com esquema distinto, o qual facilitaria o processo de integração utilizando os bancos de dados que cada instituição vem utilizando, pois estes possuíam similaridades e as diferenças basicamente estariam na representação sintática de seus esquemas lógicos.

Outro aspecto relevante nesta abordagem é que os sistemas gerados podem permitir solucionar alguns problemas relacionados a bancos legados, permitindo que estes possam ser integrados ao sistema e seus dados compartilhados.

1.3. Metodologia

Para o desenvolvimento deste trabalho é importante que conceitos e a definição de uma arquitetura para integração sejam bem definidos. Para essa tarefa é imprescindível uma revisão bibliográfica a respeito do estado da arte para os tópicos citados.

Desta forma foram verificados os requisitos necessários para viabilizar este tipo de sistema, quais os obstáculos para o seu desenvolvimento, que possa realizar a integração e o compartilhamento de dados entre organizações distintas, onde fossem respeitadas as políticas locais no que se refere a acesso a dados e a forma de disponibilizar dados autorizados. Após esta etapa, formas de como contornar os problemas considerando as restrições de cada organização envolvida, foram idealizadas.

Foram realizados levantamentos para identificar quais as tecnologias e ferramentas seriam adequadas para a solução do problema em questão, bem como sua viabilidade. Em seguida, a partir das tecnologias escolhidas, foram desenvolvidas propostas e estratégias para a manipulação de dados dos bancos distribuídos em ambiente heterogêneo.

Como forma de validação, foi desenvolvida a implementação da ferramenta e a realização de testes de funcionalidades, verificando se os mecanismos desenvolvidos funcionam de forma adequada, assim como testes de desempenho verificando o tempo de execução dos mecanismos desenvolvidos e por fim um protótipo de aplicação baseado nas necessidades verificadas em um estudo de caso que utilizou a ferramenta desenvolvida.

1.4. Organização do Texto

O restante deste trabalho está organizado da seguinte maneira:

- No Capítulo 2 são apresentados os conceitos envolvidos a respeito de grade computacional, *web services* e bancos de dados distribuídos, assim como as ferramentas utilizadas no desenvolvimento do sistema, como o Globus Toolkit e o OGSA-DAI.
- O Capítulo 3 apresenta alguns trabalhos existentes em integração de banco de dados, os quais abordam principalmente a utilização da tecnologia de grade computacional e o conceito de integração utilizando mediadores.
- O Capítulo 4 apresenta a ferramenta desenvolvida neste trabalho, considerando e descrevendo as estratégias adotadas para a integração de bancos de dados.
- No Capítulo 5 é descrito o DIGE, uma implementação da ferramenta desenvolvida, apresentando alguns detalhes de implementação da arquitetura e de seus componentes conforme os conceitos e estratégias apresentados no Capítulo 4.
- O Capítulo 6 apresenta os testes realizados, a descrição do ambiente de testes, a metodologia utilizada, os resultados obtidos e um protótipo de aplicação utilizando o sistema de bancos integrados.
- No Capítulo 7 são apresentadas as conclusões atingidas com este trabalho, e sugestões para trabalhos futuros e alguns detalhes a serem implementados nesta ferramenta.

2. CONCEITOS E TECNOLOGIAS UTILIZADAS

Neste capítulo é apresentada uma visão geral para melhor compreensão a cerca dos conceitos relacionados a este trabalho, assim como as justificativas para as tecnologias adotadas na implementação da ferramenta.

2.1. Conceitos

Os subtópicos a seguir irão descrever os conceitos mais relevantes utilizados neste trabalho para uma melhor compreensão do texto.

2.1.1. Grades Computacionais

O termo grade (do termo em inglês “grid”) surgiu em meados da década de 1990 como uma forma de caracterizar uma infraestrutura de computação distribuída para ser utilizada em problemas complexos no campo da engenharia e da ciência (Foster e Kesselman, 2004). Este nome foi escolhido devido à sua semelhança com a rede de energia elétrica (*electric grid*) que oferece energia sob demanda, deixando transparente a complexidade envolvida na produção e na malha de distribuição da energia elétrica. Já em uma grade de computadores são oferecidos serviços e recursos computacionais sob demanda tornando transparente a organização dos computadores distribuídos geograficamente.

Esta forma de organização permite o compartilhamento de diversos recursos computacionais entre seus colaboradores. Estes recursos podem ser supercomputadores, armazenamento de dados, telescópio, bancos de dados entre outros. Este tipo de ambiente requisita a coordenação dos recursos compartilhados, resolvendo problemas de forma dinâmica em diversas localidades como se fosse um único recurso local (Foster, Kesselman e Tuecke, 2001). O compartilhamento deve ser bem controlado, definindo de forma clara e cuidadosa o que pode ser compartilhado e entre quem.

Para que um ambiente de grade computacional trabalhe de maneira associada, algumas características são importantes como a interoperabilidade, escalabilidade e a portabilidade entre sistemas. No intuito de garantir tais

propriedades, a comunidade de *grid computing* vem empenhando esforços em pesquisas para a utilização de protocolos de comunicação baseados em padrões de arquitetura aberta (Foster, Kesselman e Tuecke, 2001). A Figura 2.1 ilustra uma arquitetura sugerida por Foster (Foster e Kesselman, 2004) para os componentes fundamentais de uma grade computacional, identificando cada camada com seus componentes para propósito geral.

Figura 2.1 – Camadas de uma arquitetura em grade

realizar o compartilhamento e serem mediados através do gerenciamento da grade. Esses recursos (físicos ou lógicos) podem ser de diversos tipos como supercomputadores, dispositivos de armazenamento, redes, bancos de dados, sensores, entre outros.

A camada *resource and connectivity protocols* define os protocolos de comunicação e autenticação que serão utilizados na grade. Os recursos por sua vez são as implementações destes protocolos, realizando chamadas para a camada *fabric*, para o acesso ou a manipulação dos serviços disponíveis.

A camada *collective services* é responsável por protocolos de informações gerais a respeito da grade, como dados referentes à estrutura, estados dos recursos e os protocolos de gerenciamento para negociar o acesso uniforme aos recursos.

A camada *user applications* é a camada final da arquitetura responsável por operar o sistema permitindo a comunicação com as demais camadas. Desta forma, ela irá obter as credenciais de acesso para manipular arquivos utilizando a camada *resource and connectivity*, para consultar informações a respeito da grade tais como onde os recursos estão localizados, o status de execução da sua requisição

utilizando a camada *collective services* e o acesso a algum supercomputador ou a um sistema de armazenamento utilizando a camada *fabric*.

2.1.2. Web Services

Web services (Booth, Haas e McCabe, 2004) são softwares desenvolvidos para prover interoperabilidade entre máquinas. Para cumprir estes objetivos, permite a independência de plataforma e linguagens de programação, interagindo, de maneira definida, através de suas descrições, utilizando comunicação baseada em protocolos abertos e amplamente conhecidos na web. Desta forma *web services* possibilita a interoperabilidade entre computadores tornando possível a execução de serviços remotos.

O World Wide Web Consortium (W3C, 2010), vem despendendo esforços no intuito de estabelecer padrões, o que vem fazendo do *web services* uma forma amplamente utilizada para integração de sistemas.

O *web services* está baseado em dois padrões, o WSDL (Web Services Description Language) (Chinnici, Moreau, *et al.*, 2007), fornecendo “como” e “onde” o serviço está disponibilizado, e o SOAP (Simple Object Access Protocol) (Gudgin, Hadley, *et al.*, 2007), responsável pela troca de mensagens. Estas mensagens são encapsuladas em um protocolo HTTP para o transporte. Apesar do W3C não definir um padrão e citar em sua documentação, o HTTP é um protocolo consolidado na internet e muito utilizado nos *web services*.

2.1.3. Bancos de Dados Distribuídos

De acordo com Ozsu (Ozsu e Valduriez, 1999), um sistema de banco de dados distribuídos é uma coleção de vários bancos de dados logicamente inter-relacionados distribuídos através de uma rede de computadores. Um sistema gerenciador de banco de dados distribuído é um software que permite o gerenciamento do sistema de banco de dados distribuído de forma transparente.

Uma arquitetura de banco de dados distribuídos pode ser classificada baseada em fatores como distribuição, heterogeneidade, autonomia e

interoperabilidade. Apesar de alguns autores na área ainda divergirem a respeito destes fatores, iremos considerar os seguintes:

- *Distribuição* – a distribuição refere à forma como os dados estão armazenados fisicamente sobre múltiplos bancos de dados. Estes dados podem ser particionados e distribuídos de forma horizontal, onde os registros de uma tabela são distribuídos entre os bancos de dados, ou vertical onde as colunas são distribuídas ou de forma mista utilizando as técnicas de ambos os modelos.
- *Heterogeneidade* – a heterogeneidade pode acontecer de diversas formas em um sistema de bancos de dados distribuídos. As diferenças podem ocorrer desde hardware, sistemas de arquivos a SGBDs, esquemas, estruturas e formas de dados. Usualmente, um sistema para ser considerado heterogêneo basta não ter uma característica de um sistema homogêneo.
- *Autonomia* – a autonomia refere ao aspecto de como estes diversos bancos de dados são controlados. Para verificar o nível de autonomia dos bancos existem diversos fatores a serem considerados como: a autonomia de cada projeto do banco de dados em sua concepção, a autonomia de comunicação para decidir como e quando responder à requisição de outros bancos, a autonomia em executar transações que lhe são submetidas conforme sua conveniência.
- *Interoperabilidade* – a interoperabilidade é a capacidade que sistemas possuem em solicitar e receber pedidos uns aos outros entre si.

2.1.4. Integração de Bancos de Dados

A integração de bancos de dados envolve o processo pelo qual os dados dos bancos participantes, possam ser integrados conceitualmente para formar uma definição coesa de um banco de dados múltiplo (Ozsu e Valduriez, 1999).

Basicamente existem dois tipos de abordagem para integração de bancos de dados: a integração por esquema global e a integração por esquema federado (Elmagarmid, Rusinkiewicz e Sheth, 1999). Na abordagem para integração através de esquema global, o esquema local de cada banco são combinados e mapeados em um único esquema integrado. Na abordagem de esquema federado, cada banco

integrado oferece um esquema para exportação, que é uma parte de seu esquema total, mapeando os dados que deseja compartilhar através de *views*.

As abordagens para integração estão baseadas em fragmentação de dados, o qual é formado basicamente por três formatos; horizontal, vertical e híbrido para particionar o armazenamento de dados. A fragmentação horizontal divide uma relação separando suas tuplas em dois ou mais fragmentos. A fragmentação vertical é a decomposição de uma relação e seus atributos em dois ou mais relações fragmentando os atributos. E finalmente a forma híbrida o qual procura mesclar estas duas abordagens (OZSU e VALDURIEZ, 1999).

Este trabalho adota a abordagem de integração por esquema global de dados fragmentados na horizontal, o qual permite formular uma consulta que irá abranger os diversos bancos de dados que pertencem ao sistema. Esta abordagem exige que administradores de sistema de bancos locais interajam com o administrador do sistema de bancos integrados.

2.2. Tecnologias Utilizadas

Para implementar a proposta deste trabalho, algumas tecnologias foram escolhidas, como o GT4 que provê a plataforma de grade computacional, e o OGSA-DAI que oferece o acesso aos bancos de dados. Os tópicos a seguir, descrevem estas tecnologias.

2.2.1. Globus Toolkit

O Globus Toolkit 4 (GT4) (Foster, 2005) é um *middleware* que oferece suporte e infraestrutura para o desenvolvimento de aplicações para computação distribuída orientada a serviço. Possui uma gama de recursos onde desenvolvedores podem contar como auxílio para implementar seus serviços. Entre suas características podem ser citados a segurança, gerenciamento de recursos e acesso, execução de processos, transferência de arquivos entre recursos, conjunto de políticas administrativas entre outros.

O GT4 foi escolhido por ser amplamente utilizado pela comunidade acadêmica na criação de infraestrutura para a computação em grade e também pela

sua boa documentação. Parte desta credibilidade pode ser atribuída pelo fato de sua interface de serviços serem oferecidas em SOA (MacKenzie, Laskey, *et al.*, 2006), como o GridFTP (Allcock, Bresnahan, *et al.*, 2005) que é um componente para a transferência de arquivos através de uma grade computacional. É importante citar que o GT4 oferece um serviço de autenticação de máquinas e usuários da grade, como parte do seu mecanismo de segurança.

A sua implementação está baseada na arquitetura OGSA (Open Grid Service Architecture) (Foster, Kesselman, *et al.*) que propõe requisitos para integração e gerenciamento de serviços em uma organização virtual. Esta arquitetura permite baixo acoplamento como provedor de serviços, permitindo que mudanças no código fonte do serviço não reflitam em mudanças na sua interface ou na forma de acesso. O OGSA é implementado por mecanismos da especificação OGSi (Open Grid Service Infrastructure) (Tuecke, Czajkowski, *et al.*, 2003), onde são definidos os protocolos, interfaces, notificações, políticas de acesso, entre outros.

2.2.2. OGSA-DAI

O OGSA-DAI (Hong, Hume, *et al.*, 2007) é um *middleware* para integração de dados em uma infraestrutura de grade baseado no GT4, mas podendo ser utilizado sobre o Apache Tomcat (Apache Tomcat, 2010), OMII-UK (OMII-UK, 2010), gLite (Laure, Gr, *et al.*, 2006) entre outros. É um *framework* extensível que permite o acesso a fontes de dados heterogêneas através de web service (Figura 2.2), permite o acesso a bancos de dados relacionais, bancos baseados em XML e também em sistemas de arquivos, executando *workflows* baseados em dados realizando o acesso, integração, transformação e entrega de dados dentro de um ambiente de grade computacional.

Figura 2.2 – Funcionamento Genérico do OGSA-DAI (OGSA-DAI, 2010)

A Figura 2.3 ilustra o funcionamento do OGSA-DAI, onde um cliente envia seu *workflow* ou uma requisição para o componente DRES (Data Request Execution Service), que é a interface para o acesso ao sistema através de um *web service*, o qual permite utilizar o DRER (Data Request Execution Resource). O DRER é o componente onde o *workflow* é executado, acessando os bancos de dados através do *data resource* e retornando o resultado da operação requisitada de forma segura e transparente em relação ao tipo de fonte de dados e sua localização.

Figura 2.3 - Funcionamento do OGSA-DAI

Uma das limitações do OGSA-DAI, é que as fontes de dados que são integradas em um mesmo *data resource* devem possuir o mesmo esquema restringindo suas possibilidades, pois não possui nenhum tipo de tratamento para o acesso a esquemas diferentes entre os bancos de dados.

2.3. Considerações do capítulo

Neste capítulo foram apresentados os conceitos e as ferramentas para a tecnologia escolhida. Grades computacionais e *web services* vêm convergindo para uma mesma direção oferecendo uma interface para serviços computacionais que interoperam entre plataformas distintas de maneira transparente. Para esta tarefa, o GT4 vem disponibilizando diversos recursos computacionais na forma de *web services* denominando este tipo de interface como *grid services*.

Um destes recursos, o acesso e a manipulação a bancos de dados, oferecido pelo GT4 é denominado de *data service*, e é gerenciado pelo OGSA-DAI, o qual interliga e permite o acesso aos diversos bancos de dados distribuídos que compõe o sistema.

3. TRABALHOS RELACIONADOS

A problemática envolvida na integração de bancos de dados é um tema de pesquisa atual possuindo vários problemas. Existem várias questões a serem resolvidas, devido à complexidade envolvida na heterogeneidade entre banco de dados, impossibilitando alguma solução geral ou que seja adequada aos diversos problemas existente a cerca do tema integração. Como uma forma de ilustrar a dimensão do problema, Ozsu (Ozsu e Valduriez, 1999) descreve uma taxonomia com doze tipos de arquiteturas de banco de dados e os classifica conforme características de autonomia, heterogeneidade e distribuição de dados. Para cada grau destes itens existem inúmeros problemas específicos a serem solucionados.

Dentro deste escopo, este trabalho encontra-se na arquitetura que possui autonomia de integração restrita, onde uma única imagem do banco de dados está disponível para os usuários que desejam compartilhar os dados, distribuição não hierárquica ou completamente distribuída, onde cada máquina pode comunicar com outras máquinas para executar consultas e a heterogeneidade do sistema, onde existe diferenças entre hardware e software no sistema.

Neste capítulo são apresentados alguns trabalhos desenvolvidos na área de integração de dados heterogêneos, onde estes trabalhos possuem semelhanças e diferenças em suas abordagens e objetivos os quais serão descritos nos tópicos a seguir.

3.1. CoDIMS

O CoDIMS (Barbosa, Porto e Melo, 2002) é um trabalho inicialmente realizado pela PUC-Rio e atualmente desenvolvido pela Universidade Federal do Espírito Santo - UFES. É um *middleware* para a integração de dados que permite integrar diferentes fontes de dados utilizando diversos modelos de dados internamente. Atualmente o CoDIMS foi estendido para atuar em ambientes de grades computacionais (Biancardi, 2005), e para permitir esta característica a camada *wrapper* foi desacoplada da camada de integração no intuito de desonerar o processamento concentrado neste local. Desta forma a camada *wapper* está baseada no Globus Toolkit 3 (GT3) na forma de um *web services*, auxiliando no processo de integração

de dados heterogêneos e distribuídos. Alguns projetos de pesquisa vêm utilizando esta ferramenta entre eles um projeto no LNCC para disponibilizar um serviço de dados científicos em um ambiente de grade denominando esta infraestrutura de CoDIMS-G (Fontes, Schulze, *et al.*, 2004).

A arquitetura dos componentes do CoDIMS está descrita na Figura 3.1, onde o componente Controle é responsável pela coordenação do sistema e pela gerência de execução de serviços representados por um *workflow*. Para tanto, o Controle é o local onde o ambiente é configurado, repassando as mensagens recebidas aos outros componentes. A partir de uma consulta global recebida pela interface, verifica-se as meta-informações do sistema integrador no componente Gerência de Metadados, o componente Processamento de Consultas interpreta e analisa a consulta produzindo subconsultas que serão submetidas ao componente Acesso aos Dados que envia ao respectivo *wrapper* de cada fonte de dados.

Figura 3.1 – Arquitetura CoDIMS (Barbosa, 2001)

Como o CoDIMS é um ambiente configurável, os componentes citados no parágrafo anterior, são obrigatórios e os demais componentes podem ser utilizados conforme a necessidade de cada aplicação. A integração é realizada através da forma vertical de dados, onde a composição de um registro se dá através de um conjunto de colunas dos diversos bancos envolvidos.

3.2. GISE

O GISE (Grid Integration SErvice) (Gallo, Gagliardi, *et al.*, 2007) é um sistema desenvolvido pela Universidade Católica de Santos e foi desenvolvido para ser

utilizado no projeto IntegraEPI, cujo objetivo é a criação de um sistema de vigilância epidemiológica da dengue para uma área geograficamente distribuída. Foi concebido para integração de múltiplas fontes de dados utilizando uma plataforma de grade computacional.

A arquitetura está baseada em componentes conforme descrito na Figura 3.2 onde o mediador atua na requisição de consulta, fragmentando-a em várias consultas para cada fonte de dados. O conjunto de sub-consultas inicia um processo de três passos;

- Identifica-se onde está a fonte de dados a ser consultada.
- Monta-se uma matriz na qual cada fonte de dados representa uma linha e o número de colunas é o número de campos envolvido em cada banco de dados alimentando o conversor estrutural.
- Finalmente cada subconsulta irá realizar sua consulta no formato SQL com os parâmetros e operações modificados.

Figura 3.2 – Arquitetura GISE (Gallo, Gagliardi, et al., 2007)

O GISE baseia o seu mediador junto ao cliente, onde a consulta é fragmentada em subconsultas, conforme a quantidade de fontes de dados, e as repassa aos Tradutores específicos, realizando assim o acesso a cada fonte de dados. Após receber os diversos resultados das subconsultas, os dados serão montados, processados e enviados ao cliente.

3.3. OGSA-DQP

O OGSA-DQP (Open Grid Service Architecture – Distributed Query Processor) (Alpdemir, Mukherjee, *et al.*, 2003), é um componente que estende as funcionalidades do OGSA-DAI, realizando consultas sobre diversos recursos OGSA-DAI, orquestrando a utilização destes recursos. De uma forma geral o DQP é um *middleware* entre o cliente e o OGSA-DAI, conforme a Figura 3.3, o qual pode ser considerado um mediador utilizando os recursos OGSA-DAI como *wrappers*. Desta forma o DQP pode planejar, agendar e executar consultas em paralelo através de diversos recursos distintos de forma transparente.

Figura 3.3 – Visão geral do OGSA-DQP

O DQP trabalha com bancos de dados com esquemas diferentes, realizando consultas simultâneas entre eles, dividindo-a em partições que serão transformadas em várias subconsultas os quais serão enviadas aos bancos de dados de destino em um formato compreensível. Para esta tarefa, oferece um serviço que articula como organizar esta distribuição utilizando técnicas e análises de bancos de dados paralelos (Lynden, Mukherjee, *et al.*, 2009).

Para que o programador possa realizar consultas através de diversos bancos com esquema diferente, ele deve conhecer o esquema do banco de destino para realizar a consulta. Outro ponto é esta na abordagem utilizada a qual visa a integração de dados distribuídos na vertical, ou seja, integra colunas de diferentes tabelas e organiza-as como uma nova tabela.

3.4. GDIS

O GDIS (Grid Data Integration System) (Comito, Gounaris, *et al.*, 2009) é um trabalho desenvolvido na Universidade da Calabria e propõe a integração de fontes

de dados distribuídos sobre uma grade computacional adicionando novas funcionalidades ao DQP (Alpdemir, Mukherjee, *et al.*, 2003), descentralizando o ponto de mediação proposto no trabalho original.

Cada ponto do sistema pode assumir o papel de uma, duas ou todas as tarefas do GDIS, onde tais tarefas são descritas como:

- Provedores de dados fornecendo as fontes de dados.
- Mediadores que proveem o mapeamento entre esquemas.
- Clientes fazendo requisição de consultas.

Na Figura 3.4 está a arquitetura básica do GDIS, onde a camada GDI descentraliza a mediação realizada pelo DQP utilizando uma abordagem P2P e um otimizador de consultas para auxiliar o DQP no processo de fragmentar a consulta para as diversas fontes de dados.

Figura 3.4 - Arquitetura do GDIS (Comito e Talia, 2004)

Como o GDIS está baseado no DQP, ele também está baseado na integração de dados distribuídos na vertical, onde as colunas de diferentes tabelas são concatenadas para constituir uma nova tabela.

3.5. Arquitetura Mediadores e Adaptadores

O mediador para processamento de consultas é um trabalho desenvolvido pela Universidade Federal do Rio de Janeiro (Figueiredo, Braganholo e Mattoso, 2007), e propõe uma arquitetura para executar consultas XQuery sobre base de dados XML distribuídas e fragmentadas, conforme a Figura 3.5. A arquitetura do sistema oferece suporte para uma visão global a partir do uso do mediador, responsável pela

execução de consultas distribuídas, ocultando detalhes de localização e formas como os dados estão fragmentados. Assim, uma consulta submetida ao Mediador sobre uma visão global é fragmentada em diversas subconsultas os quais são executadas nos Adaptadores acoplados aos bancos de dados remotos. Os resultados das subconsultas são retornados pelos Acopladores ao Mediador que constrói um resultado final.

Figura 3.5 – Arquitetura Mediador-Adaptadores
(Figueiredo, Braganholo e Mattoso, 2007)

A arquitetura Mediador-Adaptadores é uma abordagem para realizar consultas em base de dados fragmentadas. Está baseada no processamento de consultas distribuídas em base de dados XML, realizando o tratamento para a execução de subconsultas em um Mediador centralizado, o qual personaliza a consulta para serem enviadas aos Adaptadores e em seguida recebe os diversos resultados para compor um resultado final.

3.6. Arquitetura de um Sistema para Integração de Bancos de Dados com Suporte a Replicação Utilizando a Tecnologia de Grades Computacionais

A arquitetura apresentada por Brito (Brito, 2009) realiza a integração e replicação de bancos de dados de forma hierárquica e configurável. A abordagem hierárquica está relacionada ao fluxo de replicações, onde as operações realizadas em um banco de dados tem esta operação propagada aos níveis mais altos da hierarquia.

Esta arquitetura foi aplicada em um projeto denominado Arquitetura de Sistemas Abertos e Distribuídos para o Compartilhamento de Informação e Serviços de Saúde. O objetivo do projeto foi identificar uma arquitetura de sistemas que

auxilia na gestão de informações hospitalares desenvolvendo um protótipo de sistema baseado no compartilhamento de informações entre unidades, utilizando a tecnologia de grades computacionais (Brito, Kakugawa, *et al.*, 2009).

A Figura 3.6 ilustra a arquitetura hierárquica aplicada ao protótipo do projeto, mostrando o fluxo de replicação dos dados. A figura ilustra também quais componentes da arquitetura devem ficar em cada instituição para que a integração possa ser realizada e os dados possam ser compartilhados.

Figura 3.6 – A Arquitetura hierárquica aplicada no projeto (Brito, 2009)

A solução desenvolvida permite o acesso compartilhado aos dados entre as diversas instituições, mas como o projeto está baseado no OGSA-DAI para o acesso aos bancos, todos os bancos do sistema devem ser idênticos, ou as diferenças devem ser tratadas na aplicação do sistema, gerando uma limitação nesta abordagem de integração.

3.7. Considerações do capítulo

Neste capítulo foram apresentados alguns trabalhos relacionados com integração de bancos de dados utilizando a tecnologia de grades computacionais, no intuito de realizar um estudo a cerca do estado da arte e apontar as diferenças para o trabalho proposto. Uma forma comum verificada foi que em todos os trabalhos a abordagem de integração está fundamentada no conceito de Mediador (*mediator*) e Adaptador (*wrapper*), e a maioria está voltada para integrar colunas de diversos bancos.

O *middleware* OGSA-DAI oferece mecanismos que permitem trabalhar somente com bancos de dados que possuam o mesmo esquema lógico. Esta

característica restringe suas funcionalidades em permitir somente o acesso a fontes de dados caso elas sejam heterogêneas. Isto implica que os programadores de aplicações devem conhecer cada fonte de dados e as manipule de forma coerente.

O GISE trabalha com uma abordagem a qual utiliza o mediador próximo ao cliente fazendo com que as configurações a respeito de onde estão os dados e qual o esquema de cada banco seja realizada pelo cliente. Trabalha com a abordagem de integração de dados distribuídos de forma vertical, ou seja, as colunas das tabelas estão divididas entre os diversos nós da grade. Outro ponto está na centralização do mediador gerando um ponto de gargalo e sujeito a falhas no sistema.

O DPQ é um componente OGSA-DAI, desenvolvido de forma personalizada, visa integrar bancos de dados através da orquestração dos diversos recursos de dados envolvidos, permitindo a consulta através de bancos com esquema diferentes. Mas a integração realizada pelo DQP é de forma vertical, ou seja, integra as colunas de diferentes tabelas e o programador para realizar a consulta necessita conhecer o esquema de todos os bancos envolvidos na consulta.

O GDIS visa aperfeiçoar as funcionalidades do DQP inserindo uma abordagem P2P descentralizando o mediador e a introdução de um novo algoritmo para a otimização de consultas. Como está baseado no DQP, realiza a integração vertical de dados.

Já o CoDIMS, é um ambiente para integração que pode configurar seus componentes conforme a necessidade da aplicação. Utiliza uma abordagem centralizada para utilizar o mediador (componente Controle), e os dados são integrados de forma vertical, onde o conjunto de colunas dos diversos bancos resulta em um registro.

A arquitetura Mediador-Adaptador, apesar de trabalhar com fontes de dados XML, mostra que a abordagem *mediator-wrapper* para integrar dados é também utilizada em outros tipos de fontes de dados deixando bem claro seu funcionamento.

A arquitetura proposta por Brito (Brito, 2009) ainda que vise à integração de dados, o sistema possui algumas limitações exigindo esquema idêntico entre os bancos integrados. Outro aspecto que merece comentário é que o foco maior da arquitetura está em fornecer mecanismos que propiciem a replicação dos dados.

Neste trabalho, aqui apresentado, é proposta uma abordagem que difere das demais mencionadas, pois trata da distribuição de linhas das tabelas entre os diversos bancos. Visa diminuir a complexidade envolvida com bancos de dados distribuídos e heterogêneos na visão de uma aplicação cliente, deixando transparente a localização e as características particulares de cada banco de dados. Para que isso seja possível, a transformação das consultas deve ser realizada o mais próximo possível dos bancos envolvidos, tornando este processo transparente.

4. ARQUITETURA PARA INTEGRAÇÃO DE BANCOS DE DADOS

A integração de bancos de dados é um tema complexo de pesquisa o qual está longe de atingir uma solução que seja definitiva pela complexidade envolvida nas diversas situações particulares que cada banco possui. Conforme verificado no Capítulo 3, onde foram apresentados alguns trabalhos, é possível notar a complexidade envolvida no tema integração de dados, os quais cada trabalho procura atender a uma determinada necessidade.

O foco para a realização deste trabalho foi definir e implementar uma arquitetura que realize a integração horizontal, ou seja, permitindo que uma tabela tenha seus registros particionados e espalhados em diferentes bancos de dados distribuídos. Tal abordagem permite a distribuição dos dados, descentralizando o seu armazenamento, assim como a possibilidade de realizar consultas de forma distribuída, dividindo o processamento total exigido em uma consulta entre as diversas máquinas que compõem o sistema integrado. Esta é a essência do sistema construído utilizando a arquitetura aqui proposta, permitindo a autonomia de cada sistema de banco de dados pertencente ao sistema integrado.

Baseado nos princípios citados no parágrafo anterior, o novo sistema produzido deve gerar benefícios às diversas esferas que irão interagir. Usuários finais, os quais terão suas informações disponibilizadas em outros locais diferentes ao de sua origem, administradores institucionais que poderão visualizar os dados de forma hierárquica, desenvolvedores de aplicações que irão programar utilizando um esquema global sem se preocupar com a localização de bancos e o administrador do sistema integrado que poderá adicionar ou remover bancos de forma simples, sem a necessidade de alterações nas aplicações.

Um exemplo dos benefícios que esta arquitetura de integração poderia oferecer seriam as autarquias públicas, onde cada município, estado e união possuem dados que poderiam ser compartilhados e o armazenamento permanecendo no local de origem. Informações a respeito dos cidadãos é uma boa forma de ilustrar este exemplo, onde os cidadãos, no papel do usuário final, poderiam ir para outras cidades e ter suas informações disponibilizadas para um atendimento em algum serviço público sem a necessidade de recadastro. O governo

estadual, no papel de administrador institucional, verifica como as prefeituras atendem a população. O desenvolvedor da aplicação que acessa os dados não necessita conhecer os esquemas de cada banco de cada cidade e a sua localização e o administrador do sistema integrado pode adicionar ou remover cidades sem a necessidade de alterar a aplicação final com as modificações realizadas.

O projeto OGSA-DAI (Antonioletti, Atkinson, *et al.*, 2005), dentro de sua linha de pesquisa, visa permitir o acesso a bancos de dados em diferentes formatos, como o modelo relacional, baseado em XML e sistemas de arquivos, permitindo o acesso simultâneo às fontes de dados integradas. Para que o acesso simultâneo ocorra, as fontes de dados devem possuir o mesmo esquema lógico, limitando suas possibilidades. Uma vez que OGSA-DAI permite o acesso a diversas fontes de dados, o tratamento em relação a heterogeneidade entre esquemas deve ser realizado na aplicação final do sistema. Para tanto, o programador da aplicação, deve conhecer cada esquema e localização do banco a ser integrado controlando o envio de requisições para cada fonte de dados assim como tratar cada resposta de todas as requisições.

Um comentário relevante a respeito do OGSA-DAI, é que apesar de possuir um *framework* para manipular o sistema integrado construído, demanda um custo elevado para o seu aprendizado para utilização do ambiente e de seus recursos para manipular os dados. Existe uma documentação produzida pelos seus idealizadores no site do projeto (OGSA-DAI, 2010), mas este extenso documento ainda carece de uma nova forma de organização a qual permita localizar itens de forma mais intuitiva.

Desta forma, desenvolveu-se neste trabalho, uma arquitetura de suporte à construção de sistemas de bancos de dados, implementado na forma de uma ferramenta denominada DIGE (Database Integration using Grid Environment), que integram diferentes bancos de dados relacionais distribuídos com esquemas lógicos heterogêneos. Os bancos a serem integrados podem utilizar os gerenciadores de bancos de dados relacionais existentes no mercado, e podem estar localizados em diferentes domínios administrativos. Para tanto, serão interconectados oferecendo ao usuário final, através de uma interface, uma forma transparente de acesso aos diversos bancos de forma simultânea, ocultando a complexidade envolvida em conectar os bancos distribuídos e manipulando seus diferentes esquemas lógicos.

Além disso, o sistema proporciona autonomia entre os diversos domínios administrativos envolvidos na integração e compartilhamento dos dados respeitando as políticas de acesso local de cada domínio.

Os bancos de dados podem ser integrados de diferentes maneiras, conforme representado na Figura 4.1, onde bancos de dados podem estar no mesmo ou em Domínio Administrativo diferente. Bancos de cada domínio administrativo aplicam suas políticas locais para gerenciar seus bancos. Para um banco de dados participar do sistema de bancos integrados para o compartilhamento de dados entre os diferentes domínios administrativos, o banco deve estar exposto, no mínimo, em um SA (Servidor de Acesso), que gerencia as requisições de acesso enviadas por outros domínios administrativos. O SA é a infraestrutura que permite a comunicação entre os bancos de dados e o *middleware* desta grade, podendo acessar um banco de dados diretamente ou delegando a consulta a outro SA.

Figura 4.1 – Um Ambiente Integrando Bancos de Dados Distribuídos

Observando a Figura 4.1, o Cliente 2 está utilizando um SA no DA1 (Domínio Administrativo 1), que pode acessar de forma direta o banco em seu domínio e também aos bancos de DA2. Os bancos em DA2, apesar de não possuírem um SA local, podem ser integrados ao sistema utilizando o SA em DA1. Já o Cliente 1 utiliza o SA em DA5 para acessar o banco local de forma direta e os bancos localizados

nos outros Domínios Administrativos através de outros SA's. A Figura 4.1 mostra os clientes utilizando o SA de forma estática, mas eles podem utilizar qualquer um dos outros SA's pertencentes ao sistema de integração.

Esta arquitetura permite ao sistema de integração inserir ou remover bancos facilmente, visto que para que um novo banco de dados seja integrado ao sistema basta expô-lo como um novo recurso computacional em um SA. Isto também é válido para a remoção. Após uma inserção ou remoção de bancos, esta característica permite que as aplicações continuem trabalhando da mesma forma anterior, utilizando a mesma interface para o acesso aos dados sem a necessidade de alterar seu código fonte. Para tornar esta característica possível, o SA onde o banco de dados está exposto fica responsável por indicar a localização, permitir o acesso e converter o esquema. Esta característica também é válida para a conversão de consultas, onde a configuração fica somente no SA onde o banco foi exposto não necessitando estar presente em outros SA's.

Desta forma o acesso é realizado de duas maneiras:

- O SA manipula diretamente os bancos de dados nele expostos.
- Caso um banco esteja exposto em outro SA, a consulta é delegada a este SA.

Os subtópicos a seguir descrevem a arquitetura proposta e o funcionamento do sistema de bancos de dados integrados.

4.1.A Arquitetura Proposta

O objetivo para o desenvolvimento da ferramenta proposta é a formulação de uma arquitetura distribuída, onde cada banco de dados participante mantenha o armazenamento local, trabalhando de forma cooperativa, dispensando técnicas como a extração para um novo armazenamento ou a centralização dos dados. Assim como o armazenamento deve ser distribuído, o acesso aos dados deve também seguir esta premissa, deixando o sistema livre de ponto único sujeito a falhas e passível de se tornar um gargalo prejudicando o desempenho.

Baseado nestas premissas, a Figura 4.2 ilustra a arquitetura da ferramenta. A partir de uma consulta SQL em um formato que denominaremos de esquema global, o usuário envia esta consulta a um SA pertencente a grade computacional. O SA de

posse da consulta irá verificar se o banco de dados o qual deseja fazer a consulta está exposto em seu domínio. Caso o banco esteja exposto em seu domínio, o SA envia a consulta ao banco para que a operação seja realizada. Caso o banco esteja exposto em um outro SA, esta consulta será delegada ao SA onde o banco está exposto, delegando a execução da consulta.

Figura 4.2 – A Arquitetura Proposta

Este tipo de abordagem permite à ferramenta distribuir o acesso aos bancos envolvidos em uma consulta, onde cada SA é responsável pelo acesso aos bancos que estão expostos nele. Com esta distribuição no acesso aos bancos é possível distribuir a execução da consulta, fazendo com que cada SA execute a parte que lhe cabe, processando o Conversor, modificando a consulta para o esquema do banco de dados local.

Assim, é possível que tanto o acesso, a execução e a conversão da consulta sejam realizados de forma distribuída, permitindo ao Componente Cliente uma abstração da complexidade envolvida nesta distribuição, sem a necessidade de conhecer a comunicação entre os SA's e nem os esquemas de cada banco de dados. A arquitetura está dividida em três componentes, os quais serão comentados a seguir.

- **Componente Cliente**

Camada que possui as conexões necessárias para realizar a comunicação com os SA's. Uma aplicação para realizar sua consulta, acessa os SA's através deste componente.

- **Componente Integrador**

Esta camada, com os softwares instalados em cada SA, é o núcleo da ferramenta. Nesta camada é onde se encontram os *middleware* para grade, as configurações necessárias para o acesso individual a cada banco de dados, a comunicação necessária para delegar uma consulta a bancos que estão expostos em outros SA's, as configurações a respeito do esquema de cada banco. É também onde são realizadas as configurações para que os diversos bancos de dados possam se comportar como uma única fonte de dados oferecendo uma interface para o acesso aos diversos bancos simultaneamente, assim como a conversão de consulta, de forma a tornar compreensível ao banco desejado.

- **Componente Persistência**

Esta camada é o local onde se realiza o armazenamento, a modificação e a extração de dados.

Uma forma de descrever como estes componentes estariam instalados em um ambiente de execução está ilustrado na Figura 4.3. Esta figura mostra um exemplo hipotético de aplicação da ferramenta. Os usuários utilizariam um browser conectado a um portal em um servidor web para solicitar e visualizar os dados. Cada servidor web, para executar sua aplicação, necessita do componente Cliente para acessar os SA's e realizar a consulta.

O componente Integrador são os SA's utilizando os *middleware* para grade para comunicarem entre si distribuindo a consulta para que cada um deles execute-a sobre seus bancos. O conversor de consultas trata a heterogeneidade entre os esquemas e finalmente o acesso ao banco para a execução da consulta.

E o componente Persistência que é composto pelos bancos de dados locais.

Figura 4.3 – Exemplo de um ambiente de execução

4.2.O Tratamento da Heterogeneidade entre Bancos

Para manipular a heterogeneidade entre os bancos de dados envolvidos, o SA utiliza um conversor de consultas que gera uma nova consulta para os bancos de dados da grade, convertendo uma consulta SQL para o formato que cada banco envolvido na operação possa compreender. Isto permite ao usuário utilizar uma única consulta para os diversos bancos envolvidos permitindo um acesso uniforme aos dados, passando a impressão de utilizar um único banco de dados disponível na grade através de um esquema global.

A Figura 4.4 ilustra a execução deste tipo de operação, onde SA1 recebe uma consulta no formato do esquema global, verifica onde os bancos estão expostos para executar a consulta. Para BD1 e BD2, que estão expostos nele, fragmenta esta consulta em subconsultas, verifica qual o esquema local de cada banco, converte a consulta e a executa em cada um dos seus bancos. Para BD3, que está exposto em outro SA, SA1 encaminha a consulta para SA2 que irá realizar o mesmo procedimento, verificando qual o esquema local de BD3, convertendo e executando a consulta em um formato compreensível por BD3.

Durante o processo de conversão, é realizada uma análise da consulta SQL, onde são verificados erros de sintaxe e se todos os dados descritos na consulta estão no formato do esquema global. O sistema então separa todos os nomes e operadores para serem mapeados para o esquema do banco de dados de destino.

Conforme a Figura 4.4, podemos observar que SA1 recebe uma consulta e identifica que BD1 e BD2 estão expostos nele, identificando que a tabela *padrão* e as colunas *nome* e *cpf*, descrito no esquema global, estão se referindo respectivamente a coluna *paciente* e *cod_cpf* da tabela *lista* do BD1, o qual utiliza um SGBD MySQL. O mesmo processo se aplica ao BD2, que está utilizando um SGBD PostgreSQL, onde receberá a consulta com o nome da tabela *padrão* convertido para *cadastro* e os nomes da coluna *nome* para *cliente* e *cpf* para *num_cpf*. A consulta a BD3, o qual utiliza um SGBD DB2, será delegada a SA2, o qual no processo de conversão verifica que a tabela *padrão* está referindo a *pessoa* e as colunas *nome* e *cpf* serão convertidos para *name* e *id_cpf*. Desta forma a consulta é executada em um formato compreensível por cada banco de dados.

Figura 4.4 – Execução de uma consulta

O aspecto atraente nesta proposta está na localização do conversor de consultas, que está localizado junto ao SA onde o banco de dados está exposto. Esta opção esconde as particularidades do esquema de cada banco, possibilitando ao desenvolvedor da aplicação que utilizará o sistema com os bancos de dados integrados preocupar-se somente com o esquema global, sem a necessidade de configurar e indicar todas as fontes de dados desejada.

4.3. Análise Comparativa

A abordagem proposta por este trabalho para a integração de bancos e o acesso compartilhado aos dados difere dos demais trabalhos relacionados, conforme pode ser visualizado na Tabela 4.1.

A arquitetura deste trabalho propõe a integração de dados fragmentados na horizontal, onde os registros estão distribuídos entre os diversos bancos que compõe o sistema de integração. O SA conceitualmente está realizando o trabalho de mediação entre a aplicação final e o conjunto de bancos de dados integrados, desta forma a arquitetura utiliza múltiplos SA's possibilitando o acesso descentralizado e realizando consultas simultâneas de forma distribuída.

Uma característica relevante está em oferecer transparência em relação ao acesso aos dados assim como a localização de cada banco. A transparência em relação ao acesso a bancos com esquema heterogêneo está na utilização de um esquema global de dados, onde a conversão de consultas é realizada no SA onde o banco está exposto, corroborando com a abordagem de múltiplos SA's, proporcionando o processamento desta tarefa de forma distribuída. A abordagem de múltiplos SA's permite também transparecer a localização dos bancos, onde o SA verifica quais os SA's estão envolvidos no processo, delegando a consulta.

Tabela 4.1 – Resumo das características dos trabalhos relacionados

Trabalho	Fragmentação dos Dados	Mediador	Bancos Integrados	Conversão de Consultas	Transparência
CoDIMS	Vertical	Centralizado	Heterogêneo	Sim	Completa
GISE	Vertical	Centralizado	Heterogêneo	Sim	Parcial
OGSA-DQP	Vertical	Centralizado	Heterogêneo	Não	Parcial
GDIS	Vertical	Distribuído	Heterogêneo	Não	Parcial
Arquitetura Mediadores e Adaptadores	XML	Centralizado	Heterogêneo	Sim	Completa
Arquitetura de (BRITO, 2009)	Horizontal	Distribuído	Homogêneo	Não	Completa
Arquitetura proposta por este trabalho	Horizontal	Distribuído	Heterogêneo	Sim	Completa

5. DESCRIÇÃO DA IMPLEMENTAÇÃO

Para o bom entendimento deste capítulo serão definidos alguns termos que serão frequentemente utilizados, para que possam ser contextualizados. O termo *activity* (atividade) está relacionado a uma tarefa específica que o sistema deve realizar. Um conjunto de *activities* organizados em uma ordem para execução será definido como um *workflow* (fluxo de trabalho). O termo expor banco de dados é relacionado a disponibilizar um banco de dados na forma de um recurso computacional, permitindo o acesso e sua manipulação em um ambiente de grade computacional.

A ferramenta desenvolvida neste trabalho foi denominada DIGE (Data Integration using Grid Environment) e está baseada na tecnologia de computação em grade utilizando o OGSA-DAI (OGSA-DAI, 2010), uma ferramenta que expõe bancos de dados relacionais heterogêneos em uma grade computacional. Este trabalho adiciona funcionalidades que tornam transparentes a forma de utilizar um sistema de bancos de dados integrados construído através do DIGE, permitindo que bancos expostos como um recurso computacional dentro da grade, possam se comportar como uma única fonte de dados lógica, tornando transparente à aplicação final o tratamento em relação a heterogeneidade entre esquemas lógicos e o acesso a diversos bancos de dados.

Em conceitos de sistemas distribuídos (Tanenbaum e Steen, 2008), é recomendada a utilização de padrões para permitir ao sistema um maior nível de interoperabilidade. A opção em utilizar o OGSA-DAI está no fato de ser um projeto concebido e implementado baseado em protocolos de comunicação descritos pela OGSi (Tuecke, Czajkowski, *et al.*, 2003), os quais utilizam padrões abertos e estabelecidos pela comunidade de *grid computing* regidos pela OGF (Open Grid Forum) (OGF).

A arquitetura do DIGE está ilustrada conforme a Figura 5.1. A arquitetura é composta de três componentes; o componente Cliente que está baseado no *framework* DIGE ou o Driver JDBC OGSA-DAI que fazem chamadas ao OGSA-DAI *Client Toolkit*, o componente Integrador baseado no GT4 com o OGSA-DAI utilizando as *activities* do QCON-G e o componente Persistência baseado em

bancos de dados relacionais disponíveis no mercado. Nos próximos tópicos estes componentes serão descritos com maiores detalhes.

Figura 5.1 – Arquitetura DIGE

5.1. Componente Persistência

O componente Persistência é responsável pelo acesso, armazenamento, manipulação e recuperação dos dados nas máquinas. É constituído pelos sistemas de bancos de dados, autônomos e dispersos geograficamente os quais serão integrados. Este componente oferece suporte aos principais SGBDs relacionais, tais como os representados na Tabela 5.1.

Tabela 5.1 – Bancos de dados compatíveis

Banco de Dados	Versão
MySQL	5.0.15
IBM DB2	8.1 FixPack 5 (Express-C)
Microsoft SQL Server	2000
Oracle	10g Enterprise Edition Release 10.2.0.1.0 – Production
PostgreSQL	8.1.4

5.2. Componente Cliente

O componente Cliente foi desenvolvido como forma de permitir que a aplicação final realize a comunicação para interagir com o sistema de bancos integrados construído. Visando facilitar esta comunicação, é disponibilizado por este componente um *framework* que encapsula os procedimentos necessários, que demandam certo tempo de aprendizado.

Estes procedimentos são desde chamadas de classes para a comunicação como a manipulação de *activities* e a construção do *workflow*. O OGSA-DAI oferece diversas *activities* para determinadas funcionalidades, entre as principais estão, executar *query*, executar *update* e notificar status da requisição. Para o DIGE foram desenvolvidas novas *activities* para a execução de tarefas específicas como a conversão de consulta, executar *query* em um banco, executar *query* em vários bancos de dados com esquema heterogêneo de forma simultânea, executar *update* com conversão, organizar resultados em tuplas. As *activities* desenvolvidas para o DIGE serão detalhadas no tópico componente Integrador.

Entretanto, para oferecer flexibilidade na programação, é oferecido ao programador utilizar também a interface OGSA-DAI Client ou o Driver JDBC OGSA-DAI. O componente foi desenvolvido para permitir que o programador da aplicação final tivesse algumas opções para o acesso aos sistemas de bancos integrados através do DIGE.

Os subtópicos a seguir descreve cada tipo de opção disponibilizada:

5.2.1. OGSA-DAI Client

A primeira opção é continuar utilizando a interface *client* oferecida pelo OGSA-DAI. Para que o programador possa manipular esta interface, ele deve ter conhecimento de como programar neste ambiente, como manipular as *activities* para montar um *workflow*.

É importante comentar que o uso da interface permite que as funcionalidades nativas do OGSA-DAI possam ser utilizadas assim como as novas *activities* implementadas pelo DIGE, permitindo manipular o ambiente de forma personalizada caso seja conveniente.

Utilizando esta interface, o programador deve conhecer quais as classes que irão permitir o acesso ao serviço disponibilizado pelo GT4, assim como as *activities* necessárias para o processo e a montagem de um *workflow*. A seguir, é descrito os passos necessários para utilizar esta interface:

- Instancia um servidor
- Informa o local do servidor

- Instancia um pedido de requisição
- Informa o nome do serviço
- Informa o nome do recurso
- Instancia as *activities* desejadas
- Informa a consulta pretendida
- Informa qual *activity* deve ter sua saída conectada a entrada de outra *activity*
- Monta um *workflow*
- Envia a requisição pedindo a execução do *workflow* para o SA (Servidor de Acesso)
- Recebe uma lista informando o estado das requisições e a resposta

A Figura 5.2, ilustra um *workflow* simples construído utilizando o OGSA-DAI para realizar uma consulta do tipo *select* para um único banco de dados.

```

01 // Create the workflow
02 PipelineWorkflow pipeline = new PipelineWorkflow();
03 pipeline.add(sqlQuery);
04 pipeline.add(TupleSplit);
05 pipeline.add(deliverToRequestStatus);

```

Figura 5.2 – Workflow padrão OGSA-DAI para consulta em um banco de dados

5.2.2. Framework DIGE

O DIGE disponibiliza o uso de um *framework* para que as aplicações manipulem o sistema de bancos construído, através de classes específicas desenvolvidas em Java para realizar consultas. Este *framework* torna transparente detalhes de uma implementação utilizando o OGSA-DAI *Client*, assim como a escolha e o funcionamento das *activities* e o comportamento do *workflow* construído, consistindo basicamente em três classes:

- *SelectQCONG* – para realizar uma consulta do tipo *select* em um recurso que possua somente um banco de dados.
- *SelectBagQCONG* – para realizar uma consulta do tipo *select* de forma simultânea em um recurso que possua mais de um banco de dados.
- *UpdateQCONG* – para realizar uma consulta do tipo *insert* ou *update* ou *delete* para um recurso que possua somente um banco de dados.

A facilidade em construir uma aplicação utilizando este *framework* está no fato do programador instanciar uma das classes que atenda sua necessidade e informar três características para executar uma consulta no ambiente, os quais são:

- `location` – informa a localização do SA que deseja utilizar
- `resource` – informa o recurso desejado dentro do SA
- `query` – informa a consulta a ser realizada

Para as classes que realizam uma consulta do tipo *select* (`SelectQCONG` e `SelectBagQCONG`), o retorno para a consulta é uma instância da classe tipo `ResultSet` da linguagem Java, que é um conjunto de objetos com as informações requisitadas, como resultado da consulta. Para a classe `UpdateQCONG`, que realiza consultas do tipo *insert*, *update* e *delete*, o retorno é uma instância da classe `RequestStatus` com a informação se a operação foi ou não realizada com sucesso.

O *framework* é responsável por criar um *workflow*, constituído de algumas *activities*, cujo objetivo é acessar os bancos de dados para executar um comando DML (Data Manipulation Language) da linguagem SQL (Structured Query Language). Este *workflow* é enviado a um SA para o acesso ao sistema de banco de dados para sua execução. A Figura 5.3 ilustra um *workflow* montado pela classe `SelectQCONG`, para executar uma consulta em um banco de dados.

01	<code>// Workflow para executar a consulta em um banco</code>
02	<code>PipelineWorkflow workflow = new PipelineWorkflow();</code>
03	<code>workflow.add(sqlQueryQCONG);</code>
04	<code>workflow.add(tupleToWebRowSet);</code>
05	<code>workflow.add(deliverToRequestStatus);</code>

Figura 5.3 – Workflow executado pelo `SelectQCONG`

Outro procedimento que merece comentário está nos mecanismos de funcionamento das *activities*, organizando suas entradas e saídas para que um *workflow* possa ser montado e executado de maneira correta. A Figura 5.4, ilustra de forma breve, as *activities* utilizadas no *workflow* e como elas estão interligadas para a execução do processo. A linha 03 mostra a *activity* `sqlQueryQCONG` recebendo em sua entrada uma expressão contendo uma consulta, na linha 05 sua saída alimenta a entrada da próxima *activity* a ser executada no *workflow*, e na linha 07, o mesmo procedimento ocorre para a ultima *activity* de execução no *workflow*.

```

01 //Interligando as activities
02
03 sqlQueryQCONG.connectExpressionInput(query.getExpression());
04
05 tupleToWebRowSet.connectDataInput(sqlQueryQCONG.getDataOutput());
06
07 deliverToRequestStatus.connectInput(tupleToWebRowSet.getResultOutput());

```

Figura 5.4 – Interligando activities para o SelectQCONG

Comparando a forma de utilização da interface *client* do OGSA-DAI e o *framework* do DIGE, fica claro o porquê do desenvolvimento do *framework*, que torna transparente todo o procedimento exigido pelo OGSA-DAI para a execução da consulta.

5.2.3. Driver JDBC OGSA-DAI

E por fim, outra forma de acessar é utilizando o Driver JDBC OGSA-DAI (Brito e Sato, 2008), desenvolvido para permitir interoperabilidade entre uma grade computacional e uma aplicação desenvolvida em Java. O *driver* foi desenvolvido para que os programadores desta linguagem ficassem confortáveis no desenvolvimento de aplicações utilizando as chamadas do JDBC para acessar aos dados, visto que, poucos programadores estão familiarizados com o acesso a bancos disponibilizados em uma grade computacional. Ele consiste em implementações das interfaces especificadas pela SUN, permitindo que aplicações desenvolvidas em Java, o desenvolvedor ao utilizar o ambiente do OGSA-DAI, não necessite conhecer detalhes de uma implementação em grade computacional, bastando conhecer o acesso via JDBC a uma base de dados qualquer (nesta situação, uma fonte de dados disponibilizada na grade). A opção pela compatibilidade com o *driver* JDBC está no fato dos *middlewares* para grades oferecerem bibliotecas de programação para a linguagem Java.

5.3. Componente Integrador

O componente Integrador é o núcleo do DIGE, recebendo uma requisição de consulta do componente Cliente, verificando onde o banco de dados se encontra, realizando a conversão de consulta para em seguida encaminhar a consulta ao

banco desejado. Este componente foi projetado seguindo o princípio de distribuir o armazenamento dos dados e o processamento de uma requisição.

Para o seu desenvolvimento, foi utilizado ferramentas como o Globus Toolkit versão 4 (GT4) e o OGSA-DAI como descritos no tópico 2.2 do Capítulo 2. Este componente será responsável por receber um *workflow* do cliente e coordenar a sua execução. Para que esta arquitetura funcionasse conforme os objetivos do DIGE, foram desenvolvidas novas *activities* internas para o OGSA-DAI, possibilitando o acesso às bases de dados heterogêneas através de um conversor de consultas.

O GT4 é responsável pela segurança do sistema e de fornecer uma interface para acessar o serviço que processa uma requisição para dados (Foster, Kesselman, *et al.*, 2002). Para utilizar os recursos que permitem manipular dados, esta requisição é encaminhada ao OGSA-DAI que indica a localização do banco de dados, permite o acesso e executa o *workflow* organizado pelo cliente (Antonioletti, Atkinson, *et al.*, 2005). Estas operações são recebidas como mensagens HTTP que encapsulam mensagem SOAP transportando os dados sobre o *workflow*.

Cada SA deve ter este componente implantado, o qual será responsável pelas configurações sobre a localização dos bancos e a respeito de seus esquemas. A Figura 5.5 ilustra o arquivo de configuração para expor um banco de dados em um SA, onde as informações como o mapeamento do banco para um recurso é identificado através do *resource.id*, e a localização do banco através do *db.uri*.

```
resource.id= nome do recurso  
db.product= nome do SGBD  
db.vendor= fabricante do SGBD  
db.version= versão do SGBD  
db.uri= local do banco de dados  
db.driver= nome do driver  
resource.jar.dir= caminho para o driver  
user= nome do usuário do SGBD  
password= senha do usuário
```

Figura 5.5 – Arquivo de configuração para expor um recurso

Cada SA é responsável pelos bancos que estão expostos nele, desta forma o recurso pode ser configurado para indicar um outro recurso que esteja em um outro SA. Este tipo de configuração permite delegar a execução da consulta ao SA onde o

banco está realmente exposto, distribuindo o processamento para executar a consulta e sua conversão entre todos os SA's envolvidos no sistema.

Finalmente, um recurso também pode ser configurado como um conjunto de recursos, como ilustrado na Figura 5.6, onde um recurso foi nomeado como Agrupados e permite o acesso aos recursos Database1, Database2, Database3 e Database4 de forma simultânea. Neste exemplo ao enviar uma requisição ao recurso Agrupado, existe um tratamento para que esta requisição seja enviada a todos que fazem parte deste recurso.

```
id=Agrupados
type=uk.org.ogsadai.DATA_RESOURCE
CONFIG
    Database1=Database1
    Database2=Database2
    Database3=Database3
    Database4=Database4
END
ACTIVITIES
    SQLBagConvert=SQLBagConvert
END
```

Figura 5.6 – Arquivo de configuração para um conjunto de recursos

Veja que do ponto de vista “acessar um recurso”, a utilização sempre será a mesma, através do nome do recurso, independente se este recurso disponibiliza o acesso a um ou mais bancos de dados, os quais podem ser locais ou em outros SA's.

Esta abordagem adotada permite que o componente Cliente utilize um SA sempre da mesma maneira, abstraindo a complexidade envolvida sobre o recurso no que diz respeito a quantidade de bancos envolvidos em um recurso, onde se localiza cada banco de dados, a distribuição da requisição e da conversão de consulta. Esta característica de manipular um recurso permite a inserção ou remoção dos bancos de maneira facilitada, onde a aplicação final continua utilizando o mesmo recurso da mesma maneira, independentemente da quantidade de bancos configurados.

Os subtópicos a seguir descrevem as novas *activities* desenvolvidas para manipular os dados, como o acesso aos bancos envolvidos é realizado e como é realizada a conversão de consulta.

5.3.1. QCON-G

O QCON-G (Query CONverter for Grid) foi desenvolvido para prover o acesso simultâneo a bancos de dados relacionais, com esquemas lógicos diferentes, expostos em um ambiente de grade computacional. Como forma de permitir manipular dados de forma transparente aos diversos bancos de dados que compõem o sistema, as requisições são enviadas a um SA, e propagada aos outros SA's envolvidos na consulta, permitindo as operações de manipulação de dados prevista pela DML (Data Manipulation Language) do SQL tais como *select*, *insert*, *update* e *delete*.

Um dos pontos atraentes de toda a arquitetura está na localização do tratamento para manipular a heterogeneidade entre esquemas. Assim como a execução de uma requisição é realizada nos SA's que expõe os bancos, a conversão de consulta é realizada no instante anterior ao acesso real ao banco pretendido, permitindo que os SA's possam ser manipulados de forma uniforme entre eles.

Desta forma, o cliente envia uma consulta SQL para um recurso específico em um SA, que de posse destas informações verifica o local dos bancos de destino. Caso o banco desejado esteja exposto em outro SA, a requisição será encaminhada para este outro SA o qual executará a requisição conforme ilustrado na Figura 5.8. Caso o banco esteja exposto neste SA, ele executa a requisição, verificando qual é o esquema lógico local para converter a consulta e executar no banco de dados desejado conforme a Figura 5.7.

Figura 5.7 – Execução de consulta em banco exposto no mesmo SA

Figura 5.8 – Delegando a execução da consulta para outro SA

Como o OGSA-DAI trabalha somente com bancos de dados que possuem o mesmo esquema lógico, foram criadas novas *activities* adicionadas ao OGSA-DAI

permitindo executar consultas através de diversos bancos de dados distribuídos com esquema diferente. Assim foram desenvolvidas as seguintes *activities*:

- `ConvertQCONG` – *activity* específica para realizar conversão de consultas. Recebe uma consulta DML e personaliza conforme o banco de dados de destino.
- `SQLQueryQCONG` – *activity* para realizar consulta do tipo *select*, utilizando a conversão de consultas, especificamente em um determinado banco de dados.
- `SQLBagQCONG` – *activity* para realizar consulta do tipo *select* de forma simultânea para vários bancos de dados. Através da informação de um conjunto de recursos, propaga a requisição para executar no SA responsável pelo banco. Esta *activity* implementa internamente a *activity* `SQLQueryQCONG`, a qual irá realizar a consulta em um banco específico.
- `UpdateQCONG` – *activity* para realizar consultas do tipo *insert*, *update*, *delete*, utilizando a conversão de consultas, especificamente em um determinado banco de dados.
- `QCONGActivityException` – *activity* para informar uma exceção gerada pelas *activities* do QCON-G.

As *activities* de conversão analisam a consulta utilizando o QCON (Query CONverter) verificando as características locais do banco desejado modificando a consulta. Para cada banco de dados exposto como um recurso de grade, as informações pertinentes ao novo banco devem ser informadas em um arquivo de configuração utilizado pelo QCON para modificar a consulta.

5.3.2. QCON

O QCON (Query CONvert) funciona a partir de uma consulta SQL e a informação de qual recurso deseja utilizar. Como este conversor é solicitado internamente à *activity* `SQLQueryQCONG` e `UpdateQCONG`, as quais utilizam em suas consultas, recursos que contenham somente um banco de dados. O nome do recurso é necessário para que o conversor saiba qual é o mapeamento do esquema local para o esquema global.

Este componente possui uma arquitetura baseada em cinco módulos conforme a Figura 5.9, composta por Analisador, Decomposição, Mapeamento, Repositório de Esquema e Conversor.

Figura 5.9 – Arquitetura do QCON

A seguir será apresentado o funcionamento de cada módulo do QCON. Para ilustrar este funcionamento será considerada a seguinte consulta ilustrada na Figura 5.8, a qual será executada em um banco exposto em um recurso chamado **teste**.

```
select * from table where column1=100;
```

Figura 5.10 – Exemplo de consulta para QCON

- **Conversor** – responsável pelo controle do QCON. Neste módulo, a consulta SQL é recebida e encaminhada ao módulo Analisador, e no final do processo constrói uma nova consulta baseada nos fragmentos enviados pelo módulo de Mapeamento.
- **Analisador** – responsável pela análise léxica e sintática de uma consulta. Para realizar esta análise, foi construída uma gramática livre de contexto utilizando regras do padrão SQL-99 para as operações da DML. Caso haja algum erro na construção da consulta SQL, será gerada uma exceção informando que a consulta submetida contém erro. A Figura 5.11 ilustra que o componente analisou e reconheceu a consulta verificando se não há erros na sua construção.

select	*	from	table	where	column1	=	100
--------	---	------	-------	-------	---------	---	-----

Figura 5.11 – Análise de uma consulta com QCON

- **Decomposição** – após a análise da consulta, ela será fragmentada em símbolos, para em seguida serem separados entre palavras reservadas da linguagem SQL, nomes de tabelas, nomes de colunas, operadores, constantes e dados. A Figura 5.12 ilustra que as palavras que compõem a consulta foram separadas e categorizadas.

Figura 5.12 – Decomposição de uma consulta pelo QCON

- **Repositório de Esquema** – local onde os esquemas dos bancos locais e o modelo do esquema global estão armazenados. Estas informações ficam armazenadas em um arquivo de configuração, ilustrado na Figura 5.13, onde para evitar conflitos de homônimos cada coluna deve ter seu nome associado a sua respectiva tabela. O arquivo deve estar dentro do repositório e ter o nome idêntico ao recurso com uma extensão *map*, considerando o exemplo que está sendo utilizado, o arquivo será nomeado como *teste.map*.

```
&table=tabela
table.*=tabela.colunaA+colunaB+colunaC
table.column1=tabela.colunaA
table.column2=tabela.colunaB
table.column3=tabela.colunaC
```

Figura 5.13 – Arquivo de configuração para o mapeamento de esquema

- **Mapeamento** – este módulo recebe a consulta fragmentada, verificando quais são os valores correspondentes ao nome do recurso, aos nomes das tabelas, aos nomes das colunas no Repositório de Esquema para o banco de dados de destino retornando ao módulo Conversor o resultado do mapeamento e os demais fragmentos da consulta. A Figura 5.14 ilustra o mapeamento realizado baseado no arquivo de configuração *teste.map*.

Figura 5.14 – Decomposição de uma consulta pelo QCON

Ao final de todo o processo, a partir dos fragmentos enviado pelo módulo de mapeamento, o conversor constrói uma nova consulta, conforme ilustrado na Figura 5.15, em um formato que o banco de dados exposto no recurso *teste* possa entender. O esquema global, não é um esquema estático definido neste trabalho. O esquema deve possuir um formato que atenda a necessidade dos participantes do sistema de integração e deve ser um dos requisitos do projeto de integração.

```
select colunaA,colunaB,colunaC from tabela where colunaA = 100
```

Figura 5.15 – Nova consulta gerada pelo QCON

O QCON é um componente independente, desenvolvido para ser utilizado como uma biblioteca Java, permitindo que aplicações desenvolvidas nesta linguagem possam utilizar este componente para problemas similares. Assim sua utilidade não fica limitada ao QCON-G.

6. TESTES REALIZADOS

Neste capítulo será descrita a metodologia utilizada para os testes de um sistema de banco de dados construído utilizando o DIGE, descrevendo o ambiente onde os testes foram realizados, as configurações de cada um dos bancos de dados que foram integrados, um esquema global definido para os testes, bem como os resultados obtidos e uma análise a respeito dos resultados.

6.1. Metodologia utilizada nos testes

O ambiente de teste foi elaborado conforme a Figura 6.1, utilizando dois computadores interligados em uma rede local através de um *switch ethernet*. A aplicação cliente foi executada em um *notebook* que está interligado a rede local através de um roteador *wireless*.

Figura 6.1 – Configuração Utilizada nos Testes

Para a realização dos testes foi desenvolvido um conjunto de aplicações testes para aferir o tempo de execução para cada situação. Para que haja um parâmetro de comparação, os testes foram realizados utilizando o ambiente construído pelo DIGE e outro construído pelo OGSA-DAI para comparar os valores entre eles. O conjunto de aplicações testes foram executadas na máquina cliente.

Estes testes foram divididos em três conjuntos, conforme descrito a seguir:

Primeiro Conjunto

O primeiro conjunto de testes constituiu em executar uma consulta do tipo *select* para um banco de dados. Para realizar este teste, foram construídas duas aplicações para realizar a consulta, uma para cada ambiente. Para as execuções

com o OGSA-DAI, a aplicação recebia três alterações de parâmetro, modificando o local do SA, o nome do recurso e a consulta a ser submetida, conforme havia a modificação do banco a ser utilizado. Para as execuções utilizando o DIGE, a aplicação recebia duas alterações, como a localização do SA e o nome do recurso. A consulta não era alterada, pois o DIGE possui suporte a conversão de consultas.

Os parâmetros modificados tanto para o DIGE como o OGSA-DAI eram:

- Local do SA – indica qual o SA que seria utilizado para receber a requisição de consulta, os quais variaram entre a máquina GAS e a Abutre.
- Nome do recurso – para cada recurso descrito na Tabela 6.2 que permite o acesso ao respectivo banco.

Para o OGSA-DAI tem outro parâmetro que sofria modificação, que é a consulta, a qual cada consulta submetida a cada banco tinha seus atributos alterados conforme as características de cada banco descrito na Tabela 6.2.

Este conjunto de teste foi realizado com o intuito de verificar a sobrecarga gerada pela conversão de consulta.

Segundo Conjunto

O segundo conjunto de testes constituiu em realizar uma consulta do tipo *select* para mais de um banco de dados. Para as execuções utilizando o OGSA-DAI, foram construídas três aplicações distintas, para cada uma acessar dois, três e quatro bancos de dados. Para cada uma das aplicações, houve um tratamento distinto para acessar cada banco de dados. A forma de realizar o controle para executar consultas em mais de um banco de dados é criando um *workflow* específico para cada recurso que irá utilizar na consulta. Assim para a aplicação que acessa dois bancos, necessita de dois *workflows* para a execução, e assim sucessivamente para as outras aplicações.

Para o DIGE, foi construída somente uma aplicação que realizava a consulta coletiva de forma simultânea. O único parâmetro modificado durante os testes era o nome do recurso. Conforme o Apêndice D – Arquivos de configuração de recursos é possível agrupar um conjunto de recursos em um recurso. Para tanto foram criados novos recursos os quais permitiam o acesso a dois, três e quatro bancos de dados. Não é possível utilizar o agrupamento de recursos com o OGSA-DAI, pois a consulta enviada a todos os bancos devem ser idênticas.

Este conjunto de teste foi realizado com intuito de verificar a escalabilidade da ferramenta.

Terceiro Conjunto

O terceiro conjunto de testes constituiu em realizar consultas do tipo *insert*, *update* e *delete* para um banco de dados. Neste conjunto de testes foram criadas duas aplicações para realizar as consultas, uma para o DIGE e outra para o OGSA-DAI, onde foram alterados dois parâmetros para cada tipo de teste:

- A consulta – as consultas enviadas alternavam entre *insert*, *update* e *delete*, onde as consultas utilizadas no ambiente DIGE estão descritos na Tabela 6.4, e para o OGSA-DAI as consultas tinham que ter seus atributos alterados conforme as características do banco utilizado.
- O recurso – tanto para o DIGE como para o OGSA-DAI, este parâmetro foi modificado entre os recursos SP1 e SP2, conforme a Tabela 6.2, utilizando os SGBD's Postgres e MySQL.

Durante este conjunto de testes utilizou-se somente um SA, a máquina GAS, e a finalidade destes testes foi verificar a sobrecarga causada pela conversão de consulta para atualizar um banco de dados.

Os equipamentos e a infraestrutura necessária para a realização dos testes foram concedidos pelo LAHPC-EPUSP. As configurações de hardware e software de cada computador utilizado nos testes possuem suas características descritas na Tabela 6.1. Em cada nó da grade computacional foram criados dois bancos de dados, os quais um utilizou o SGBD MySQL e o outro utilizou o PostgreSQL.

A escolha dos SGBD's e a opção de como estruturar os bancos de dados, foi uma maneira encontrada para proporcionar heterogeneidade ao ambiente de testes. Os bancos de dados foram construídos conforme a descrição na Tabela 6.2. Para cada um deles foi criada uma tabela com quatro colunas e povoadas com mil registros. Para permitir uma heterogeneidade entre os bancos de dados, cada um deles possui nome de banco, nome de tabela e o nome de coluna totalmente distinto entre si.

Tabela 6.1 – Máquinas e Ferramentas utilizadas

	GAS	Abutre	Anakin
Processador	Intel Xeon 3.2GHz (dois)	AMD Athlon 1.8GHz	Intel Core 2 Duo 2GHz
Memória	4GB	1GB	4GB
Sistema Operacional	OpenSuse Linux 11.1	OpenSuse Linux 11.1	Windows Vista Home Premium
Middlewares	Globus Toolkit 4.0.8 OGSA-DAI 3.0	Globus Toolkit 4.0.8 OGSA-DAI 3.0	OGSA-DAI 3.0 Client Toolkit
Banco de dados	MySQL 5.0.67 PostgreSQL 8.3.5	MySQL 5.0.67 PostgreSQL 8.3.5	

Tabela 6.2 - Características de cada recurso

Máquina	SGBD	Nome do Recurso	Nome do Banco	Nome da Tabela	Nome dos Atributos
Gas	PostgreSQL	SP1	spaulo1	cadastro	num_cli nom_cli end_cli tel_cli
Gas	MySQL	SP2	spaulo2	register	identifier full_name location phone_num
Abutre	PostgreSQL	CAM1	campinas1	pessoa	num nome residencia fone
Abutre	MySQL	CAM2	campinas2	client	id name address phone

Para que um conjunto de bancos de dados integrados possam se comportar como um único banco, deve haver algum tipo de padronização de forma que todos os usuários do sistema acessem e o utilizem da mesma forma. Uma das características do QCON-G para realizar a conversão de consultas entre bancos de dados, é a adoção de um modelo padrão ou global de esquema de dados. Para os testes realizados, foi adotado o modelo descrito na Tabela 6.3.

As informações contidas no esquema global devem ser mapeadas para o esquema local de cada banco de dados para que a conversão de consulta possa ser realizada. A Figura 6.2 ilustra como o modelo global foi aplicado, utilizando o recurso

SP1 como exemplo. Este arquivo de configuração, denominado *sp1.map*, está localizado no repositório de esquema para que o QCON realize a conversão de forma que uma consulta possa ser executada no recurso SP1.

Tabela 6.3 – Modelo global adotado

Nome da Tabela	Nome das Colunas
tabela	identificador paciente endereço telefone

```
&tabela=cadastro
tabela.*=cadastro.num_cli+nom_cli+end_cli+tel_cli
tabela.identificador=cadastro.num_cli
tabela.paciente=cadastro.nom_cli
tabela.endereco=cadastro.end_cli
tabela.telefone=cadastro.tel_cli
```

Figura 6.2 – Arquivo de configuração para mapeamento do recurso sp1

Como o DIGE adota um esquema global para o acesso aos dados, as consultas para serem realizadas neste ambiente não necessitam ser personalizada conforme o banco acessado. Para tanto, basta formular a consulta baseada no esquema global para que todos os bancos pertencentes ao sistema compreenda o que foi requisitado. Conforme a configuração de cada recurso, a consulta é convertida para o formato que será aceito pelo banco de dados de destino. Desta forma, a Tabela 6.4 ilustra as consultas utilizadas nos testes para o ambiente DIGE, as quais possuem somente dados descritos no esquema global.

Tabela 6.4 – Consultas baseadas no esquema global utilizado pelo DIGE

Descrição da consulta	
select	SELECT identificador, paciente, endereço, telefone FROM tabela WHERE identificador < 10
insert	INSERT INTO tabela (identificador, paciente, endereço, telefone) VALUES (1, 'JoaoPaulo0002 BD_SPaulo1', 'Avenida_0001', '0800-0001');
update	UPDATE tabela SET paciente = 'Fernando Ryoji Kakugawa12' WHERE identificador = 1;
delete	DELETE FROM tabela WHERE identificador = 1;

Para a consulta do tipo *select*, cada banco de dados retorna nove registros que atendem a restrição da consulta. As consultas do tipo *insert*, *update* e *delete*,

foram realizadas para somente um registro em cada um dos bancos de dados. Cada teste foi realizado dez vezes para aferir o tempo de execução de cada tipo de consulta. Com a sequência de valores obtida para cada tipo de teste, realizou-se uma média aritmética com os valores e em seguida o cálculo do desvio padrão.

6.2. Análise dos Resultados

Uma forma de avaliar os resultados obtidos foi comparar os tempos de execução entre o OGSA-DAI e o DIGE, verificando o quanto que o tratamento da heterogeneidade entre esquemas sobrecarrega o sistema. Desta forma os testes foram realizados enviando consultas da DML (*select, insert, update, delete*) em um ambiente criado pelo OGSA-DAI nativo, sem alterações, e para um ambiente criado pelo DIGE e em seguida comparando os valores obtidos.

Nos resultados apresentados, existem alguns aspectos os quais merecem comentários antes de iniciar a análise sobre eles. O primeiro aspecto está relacionado à diferença nos tempos aferidos utilizando a máquina GAS e Abutre, as quais mesmo utilizando a mesma infraestrutura de software, possuíam diferenças as quais são atribuídas à diferença de hardware entre as máquinas. Outro aspecto está relacionado à diferença de tempo verificadas na execução dos SGBD's, a qual é atribuída à característica de cada fabricante onde este trabalho não irá entrar no mérito de qual possui melhor desempenho, atribuindo a isso como heterogeneidade do sistema.

O OGSA-DAI, a partir da versão 3.2, disponibiliza um componente denominado DQP (Alpdemir, Mukherjee, *et al.*, 2003) que oferece suporte para trabalhar com bancos que possuam esquemas diferentes, realizando uma integração vertical, integrando colunas de diferentes tabelas para montar uma nova tabela.

O DIGE adota uma abordagem diferente no que diz respeito a forma como integrar bancos de dados com esquemas diferentes, realizando uma integração horizontal, como se os registros estivessem fragmentados em todos bancos integrados. Como o OGSA-DAI não oferece suporte a este tipo de integração, o DIGE adicionou mecanismos que permitem manipular bancos de dados com esquema diferente através de um conversor de consulta.

A Figura 6.3, ilustra os tempos obtidos para os testes do Primeiro Conjunto, e a Tabela 6.5 mostra o desvio padrão para os valores obtidos, para realizar um *select* aos bancos de forma direta. As diferenças entre as médias de tempo de execução para o DIGE e o OGSA-DAI não foram maiores que 2,79%, demonstrando que o DIGE não gera uma sobrecarga que inviabilize o seu uso.

Figura 6.3 – Tempo de execução *select* para um banco utilizando acesso direto

Tabela 6.5 – Desvio padrão para os tempos da Figura 6.3

	DIGE	OGSA-DAI
PostgreSQL GAS	65,28	57,93
MySQL GAS	47,47	97,63
Postgres Abutre	91,77	64,63
MySQL Abutre	41,26	46,81

A Figura 6.4 ilustra o tempo de execução de uma consulta que é delegada a outro SA conforme exemplo ilustrado na Figura 5.8, e o desvio padrão expresso na Tabela 6.6. O teste utilizado aqui continua sendo do Primeiro Conjunto, onde a consulta é enviada para um SA, o qual verifica que o recurso desejado encontra-se em outro SA, delegando a execução do *workflow* ao segundo SA.

Novamente foi constatado o mesmo comportamento dos testes anterior, onde a diferença de execução ficou em torno de 0,09% para o banco que utiliza o Postgres, e 0,97% para o acesso ao banco que utiliza MySQL.

Figura 6.4 – Tempo de execução *select* para um banco delegando a consulta para outro SA

Tabela 6.6 – Desvio padrão para os tempos da Figura 6.4

	DIGE	OGSA-DAI
Postgres	134,48	185,71
MySQL	127,46	111,87

Comparando os tempos de execução para os testes do Primeiro Conjunto, é constatado um aumento no tempo em todas as situações, utilizando o DIGE ou o OGSA-DAI. Para efeito de ilustração, serão tomados como análise os tempos obtidos com o DIGE e o OGSA-DAI para a execução do banco que utiliza o Postgres na máquina Abutre. Para o acesso direto, a execução da consulta ficaram respectivamente em torno de 731,8 e 728 ms conforme ilustrado na Figura 6.3, mas para a Figura 6.4, utilizando o mesmo conjunto de testes, o Primeiro Conjunto, ficaram em torno de 1149,5 e 1150,5 ms para a execução da consulta.

Este aumento no tempo de execução está relacionado ao encaminhamento da consulta para um segundo SA, para que este possa realizar o acesso ao banco de dados, gerando uma sobrecarga que não é verificada no acesso direto. Neste teste, uma requisição contendo a consulta foi enviada a máquina GAS, que verificou que o banco requisitado está exposto na máquina Abutre delegando a execução da consulta.

É importante comentar que o fato de delegar a consulta a outro SA, apesar de um aumento no tempo de execução, tem se a vantagem do administrador do sistema não necessitar configurar todos os bancos os quais serão integrados, bastando indicar ao primeiro SA que a aplicação acessa onde estão os demais SA's. Esta característica permite que o banco seja configurado uma vez, somente no SA

onde ele está exposto, eliminando o trabalho de configurar todos os SA's com todos os bancos. Outro aspecto atraente na questão de delegar a consulta está na distribuição da execução e conversão da consulta, realizando este processo em paralelo e eliminando um ponto único centralizado para o processamento.

A partir dos testes, onde foi executado o Segundo Conjunto, uma consulta *select* envolvendo mais de um banco de dados de forma simultânea, conforme os tempos ilustrados na Figura 6.5, foi verificado que o DIGE possui um tempo de execução mais eficiente. Este fato é atribuído ao OGSA-DAI não possuir mecanismos que permitam interagir entre bancos com esquemas distintos, deixando o tratamento da heterogeneidade a cargo dos desenvolvedores da aplicação final. Neste contexto, a aplicação deve considerar as diferenças entre os bancos de dados, tratar as diferenças enviando as requisições para cada banco e manipular todas as respostas de forma que o resultado seja unificado e compreensível a aplicação. Estas atribuições geram uma sobrecarga a cada banco adicionado. Assim, um aumento do número de banco de dados acarreta em um aumento no tempo de execução, e considerando também que a aplicação cliente centraliza todo o início de execução de cada *workflow* abrindo conexões para todos os bancos de dados desejados. Para as aferições realizadas, as diferenças ficaram em torno de 2,79% para a consulta utilizando dois bancos, 17,10% utilizando três bancos e 33,14% utilizando quatro bancos. A Tabela 6.7 o desvio padrão para os valores aferidos em cada situação.

Figura 6.5 – Tempo de execução de um *select* em mais de um banco utilizando acesso direto

Tabela 6.7 – Desvio padrão para os tempos da Figura 6.5

	DIGE	OGSA-DAI
2	114,24	52,62
3	123,69	138,19
4	47,23	149,46

Utilizando o ambiente construído pelo DIGE, não foi verificado este aumento no tempo de execução da consulta com o aumento no número de bancos, pois o *workflow* para a execução da consulta é dividido entre os SA's envolvidos na operação, realizando este processo de maneira distribuída. Foi observado que existe um pequeno aumento no tempo de execução da consulta quando ela é realizada para mais de um banco de dados, mas este tempo ficou estabilizado em torno de 800ms para até quatro bancos de dados. Um aspecto que merece comentário está no detalhe que para o acesso direto, conforme os testes realizados na Figura 6.5, todos os bancos devem estar expostos em um mesmo SA, causando uma sobrecarga devido ao SA ter que iniciar um processo para realizar as múltiplas consultas para cada banco e converter cada uma destas consultas.

A degradação no tempo de execução para um sistema que utiliza o OGSA-DAI também é verificada na Figura 6.6, onde o acesso aos bancos de dados é realizado através do encaminhamento da execução da consulta a outro SA. Já para o ambiente utilizando o DIGE o tempo de execução fica estabilizado em torno de no máximo 1200ms para qualquer uma das situações, enquanto que o OGSA-DAI comporta-se da mesma maneira conforme ilustrado no acesso direto, elevando o tempo de execução da consulta em 15,50% utilizando dois bancos, 18,57% utilizando três bancos e 30,83% utilizando quatro bancos.

Os testes do Segundo Conjunto, os quais executam consulta *select* de forma simultânea em mais de um banco de dados, foram realizados também com o intuito de verificar o quanto o DIGE permite escalabilidade sem degradar o tempo de execução da consulta. A estabilidade atingida no tempo de execução para consultas em um ou mais bancos de dados é reflexo da abordagem adotada em delegar a execução do *workflow* de uma consulta para o SA onde o banco está exposto, distribuindo o processamento de tarefas como o acesso e busca no banco de dados, assim como a conversão de consulta.

Figura 6.6 – Tempo de execução de um *select* para mais de um banco delegando a consulta para outro SA

Tabela 6.8 – Desvio padrão para os tempos da Figura 6.6

	DIGE	OGSA-DAI
2	42,56	59,23
3	102,15	74,56
4	54,29	136,64

O comportamento do tempo de execução dos dois ambientes fica bem nítido se agrupar o tempo de execução de cada um, conforme a Figura 6.7, é possível verificar que o tempo de execução do ambiente utilizando OGSA-DAI possui uma relação direta com a quantidade de bancos envolvidos na consulta, enquanto que para o ambiente utilizando DIGE, existe uma pequena sobrecarga quando aumenta a quantidade de bancos de um para dois, mas o tempo de execução fica estabilizado em torno de 800ms independente do número de bancos de dados utilizados para realizar uma consulta simultânea.

Figura 6.7 – Comportamento dos tempos de execução utilizando acesso direto

Este mesmo fenômeno é verificado na Figura 6.8, quando é delegada a execução da consulta a outro SA. O ambiente utilizando OGSA-DAI tem um

aumento no tempo de execução conforme adiciona-se bancos a consulta simultânea, enquanto que para o ambiente utilizando DIGE o tempo de execução fica estável independente do número de bancos. Um aspecto que merece comentário é o aumento de tempo verificado para o DIGE na Figura 6.7, onde existe uma sobrecarga quando aumenta à quantidade de banco de um para dois. Este fenômeno não ocorre quando a execução é delegada a outro SA conforme a Figura 6.8, onde os tempos de execução, independentemente da quantidade de bancos, se mantém constante. Como comentado anteriormente, esta estabilidade se deve ao fato das consultas serem distribuídos para a execução, onde cada *workflow* é executado no SA onde o banco está exposto.

Figura 6.8 - Comportamento dos tempos de execução delegando a consulta a outro SA

Os testes envolvendo o Terceiro Conjunto têm seus tempos de execução ilustrados na Figura 6.9, os quais foram realizados utilizando o recurso SP1 que é um banco de dados baseado em Postgres hospedado na máquina GAS. Os tempos de execução entre o ambiente utilizando o DIGE e o OGSA-DAI, não tiveram diferença de tempos maiores que 2,34% entre ambos, o que mais uma vez demonstra que a sobrecarga gerada pelo DIGE não o inviabiliza em relação ao OGSA-DAI. A Tabela 6.9 apresenta o desvio padrão para cada uma das consultas.

O mesmo conjunto de testes foi aplicado ao recurso SP2 que está hospedado na máquina GAS, e os valores de tempo de execução estão ilustrados na Figura 6.10. O mesmo comportamento verificado na Figura 6.9 ocorreu nestes testes, onde as diferenças entre DIGE e OGSA-DAI não foram superiores a 1,10%.

Figura 6.9 – Tempo de execução para *delete*, *insert* e *update*, utilizando banco Postgres na máquina GAS

Tabela 6.9 – Desvio padrão para os tempos da Figura 6.9

	DIGE	OGSA-DAI
delete	37,48	42,69
insert	39,86	32,50
update	42,76	60,75

Figura 6.10 - Tempo de execução para *delete*, *insert* e *update*, utilizando banco MySQL na máquina GAS

Tabela 6.10 – Desvio padrão para os tempos da Figura 6.10

	DIGE	OGSA-DAI
delete	22,21	31,38
insert	25,05	26,94
update	37,30	38,91

6.3. Protótipo de uma Aplicação

Como forma de validar a ferramenta DIGE, foi desenvolvido um protótipo de aplicação baseado em um projeto de integração de bancos de dados para o atendimento e gestão de um grupo específico de pacientes para a rede de saúde pública (Brito, Kakugawa, *et al.*, 2009), específico para hemofílicos, devido à complexidade envolvida em seu tratamento, assim como o alto custo dos medicamentos envolvidos. Através de um estudo de caso, foi realizado um levantamento detalhado do processo de atendimento aos pacientes hemofílicos assim como o processo de gestão dos medicamentos.

6.3.1. Estudo de Caso

Foi realizado um estudo no intuito de desenvolver um sistema para o compartilhamento de dados entre os hemocentros de São Paulo e Campinas. Na atual infraestrutura, a maioria das atividades é realizada de forma manual em papel. Para o hemocentro de São Paulo, existe um pequeno banco de dados com algumas informações do paciente e sobre o consumo de medicamento para o auxílio no processo manual de geração de relatório. No hemocentro de Campinas, o medicamento é controlado pelo almoxarife local que possui um sistema informatizado. Além disso, existem dois sistemas que mantêm informações referentes ao paciente, onde ambos não interoperam entre si e nem com o controle de medicamento.

As informações a respeito de um paciente hemofílico são muito importantes devido à complexidade envolvida no seu atendimento. Para isso existe a necessidade dos hemocentros compartilharem informações de seus pacientes para um melhor atendimento sem ferir as políticas locais de acesso aos dados. Outro item relevante no estudo está no controle de medicamentos devido ao seu elevado custo financeiro.

Para exemplificar uma situação, pacientes que possuem cadastro, domicílio e uma rotina de atendimento em São Paulo, e devido a alguma eventualidade, forem atendidos em Campinas, os seus dados poderão ser acessados. Para o usuário do sistema fica a impressão de que as informações estão registradas no banco de dados local, facilitando as atividades operacionais. Para o paciente, não haverá a

necessidade de um novo cadastro, devido ao fato de estar fora de seu domicílio, e o médico poderá fazer um diagnóstico mais preciso pela disponibilidade das informações de tratamento do paciente.

6.3.2. A Aplicação Desenvolvida

Com base nestes dados foi desenvolvido um protótipo de aplicação para o atendimento aos pacientes hemofílicos baseado em uma plataforma composta de bancos de dados que envolvem os Hemocentros de São Paulo e Campinas. O protótipo foi desenvolvido como uma aplicação web baseado em JSP e foi denominado Hemosystem.

Este protótipo é uma alternativa para o esforço do Ministério da Saúde que está desenvolvendo um projeto denominado Hemovida Coagulopatias Web (Hemovida Web Coagulopatias, 2010), onde o objetivo é monitorar doenças para um melhor planejamento do programa nacional de coagulopatias. Apesar de o projeto ter como objetivo integrar todos os centros de hemofilia do país, a abordagem atualmente utilizada para a integração, está baseada no modelo cliente/servidor onde os dados estão armazenados em um datacenter do Datasus.

Após o levantamento sobre os requisitos que o sistema deveria contemplar, foram construídos dois bancos de dados de forma hipotética, representando os bancos de dados de São Paulo e Campinas, respectivamente baseados nos SGBD's Postgres e MySQL, apresentados no Apêndice B – Bancos de dados integrados pelo Hemosystem deste trabalho. Após a confecção dos bancos, definiu-se um esquema global para o acesso aos bancos, conforme o Apêndice A – Esquema global utilizado para o Hemosystem, realizando o mapeamento de cada banco para o esquema global, conforme os arquivos de configuração do Repositório de Esquema que ficam localizados em `$DAI_HOME/qcong/mapping` de cada máquina.

Para este protótipo do Hemosystem, ficou definida a seguinte topologia conforme mostra a Figura 6.11, onde cada hemocentro possui seu SA para expor seu banco. O usuário final acessaria os dados dos bancos através de um *browser* utilizando um computador conectado a rede local ou a internet. O gestor do departamento de hemofilia, por definição para este protótipo, acessa os bancos de

dados através do SA do hemocentro de São Paulo, mas este acesso poderia ser realizado também através do servidor web de Campinas.

Figura 6.11 – Topologia da Aplicação

O SA de São Paulo possui três recursos, os quais possuem seus arquivos de configuração apresentados no Apêndice D – Arquivos de configuração de recursos. O primeiro recurso foi denominado *saopaulo*, e permite o acesso ao banco de dados de São Paulo. O segundo recurso foi denominado *campinasR* e indica que o banco de dados desejado está exposto em um recurso denominado *campinas* no SA de Campinas. Estes dois recursos estão ilustrados na Figura 6.12 e devem ser utilizados individualmente.

Figura 6.12 – Funcionamento recursos sao paulo e campinas

O terceiro recurso, denominado *ambos*, permite utilizar os bancos de dados de São Paulo e Campinas de forma simultânea para realizar consultas. Este terceiro recurso está indicado na Figura 6.13, o qual permite utilizar um conjunto de recursos, e para esta situação permite utilizar os recursos *saopaulo* e *campinasR*.

Figura 6.13 – Funcionamento recurso *ambos*

O mesmo procedimento foi realizado no SA de Campinas para permitir utilizar os recursos do SA de São Paulo da mesma forma como os de Campinas são utilizados. Esta abordagem permite que o acesso aos dados não fique centralizado sempre em um mesmo SA.

Com os bancos de dados de cada instituição integrados e trabalhando como se fossem um único banco, construiu-se uma interface para que os usuários finais pudessem utilizar os dados armazenados em ambas as instituições. Para tal foi desenvolvido um protótipo de uma aplicação web para utilizar este sistema integrado construído pelo DIGE, e sua tela de *login* está ilustrada na Figura 6.14. ilustra a tela de *login* do sistema.

Figura 6.14 – Tela de login do sistema

Para este protótipo, o HemoSystem permite o acesso compartilhado aos dados do hemocentro de São Paulo e Campinas, permitindo descentralizar o armazenamento de dados evitando o ponto único sujeito a falhas e gargalo dentro do sistema. Com o acesso compartilhado, pacientes com cadastro em São Paulo podem ser atendidos normalmente em Campinas como se suas informações estivessem armazenadas localmente. Um aspecto interessante está na possibilidade do gestor geral poder verificar os dados de um hemocentro em particular, assim como verificar os dados de todos os hemocentros para o gerenciamento das políticas de saúde pública.

A Figura 6.15, ilustra o resultado de uma consulta em ambos os bancos de forma simultânea para localizar as informações a respeito de um paciente, o qual para efeito de ilustração, não havia a informação de onde estaria cadastrado.

The screenshot displays the HemoSystem web interface. At the top, a red header contains the text "HEMO SYSTEM". Below this, a black navigation bar includes links for "Home", "Procurar", and "Logout". The main content area is divided into two sections: "MEDICO" on the left and "DADOS DO PACIENTE" on the right. The "MEDICO" section features a link labeled "Procurar Paciente". The "DADOS DO PACIENTE" section contains a form with the following fields and values: Identificador (1), Nome (Fernando Ryoji Kakugawa), Nascimento (1981-01-01), Mãe (Rosa Kakugawa), Estado Civil (casado), CPF (111111111), RG (11000000), Endereço (Rua Escola Politecnica 355), Complemento (93a), Bairro (Jaguare), and Municipio (Campinas).

DADOS DO PACIENTE	
Identificador	1
Nome	Fernando Ryoji Kakugawa
Nascimento	1981-01-01
Mãe	Rosa Kakugawa
Estado Civil	casado
CPF	111111111
RG	11000000
Endereço	Rua Escola Politecnica 355
Complemento	93a
Bairro	Jaguare
Municipio	Campinas

Figura 6.15 – Resultado de uma Consulta

7. CONCLUSÕES e TRABALHOS FUTUROS

A partir dos resultados e o conhecimento obtido durante as pesquisas e a implementação do DIGE e do Hemosystem, é possível considerar algumas reflexões a respeito do trabalho realizado, assim como sugestões para trabalhos futuros e aprimoramento da ferramenta.

7.1. Conclusões

A principal contribuição deste trabalho é a proposta de uma arquitetura que realiza a integração de bancos de dados relacionais e heterogêneos utilizando conceitos de grade computacional. Esta arquitetura permite ao usuário o acesso compartilhado aos dados distribuídos utilizando um esquema global e uma consulta em SQL convencional, ocultando a localização e o esquema local de cada banco.

Comparando a implementação de um sistema de bancos de dados que integra um conjunto de bancos de dados heterogêneos utilizando diretamente o OGSA-DAI ter-se-á um adicional de linhas de códigos de aproximadamente 40% em relação a uma que utiliza a ferramenta DIGE. Este adicional é decorrente da necessidade do programador de controlar as diferenças entre os bancos de dados manipulando requisições e unificando os resultados, gerando uma sobrecarga que degrada o tempo de execução das consultas conforme aumenta o número de bancos de dados envolvidos.

A contribuição do DIGE está em oferecer uma ferramenta que facilita esta implementação através de configurações acerca da localização do banco de dados e da informação de seu esquema lógico. Um sistema de bancos de dados construído utilizando esta ferramenta pode acessar dados de diversos bancos através de uma consulta SQL no formato do esquema global. A complexidade envolvida no acesso aos diferentes bancos de dados fica transparente na construção deste sistema.

É importante ressaltar que esta transparência também é refletida na facilidade de adicionar ou remover bancos de dados ao sistema, sem a necessidade de alterar o código fonte da aplicação.

O DIGE efetivamente manipulou as consulta de maneira que cada banco de dados envolvido no sistema de integração pudesse compreender o que estava

sendo solicitado. A sobrecarga causada neste processo não causou tempos maiores que 3,85% se comparado aos acessos utilizando um banco de dados pelo OGSA-DAI. O tempo de execução se manteve estabilizado para múltiplos bancos de dados, o que demonstra a sua escalabilidade.

7.2. Trabalhos Futuros

A principal melhoria a ser realizada está no refinamento do sistema com o propósito de disponibiliza-lo para uso baseado em uma versão estável. Entre as possibilidades de estudos e desenvolvimentos futuros estão:

- O uso de ontologia na conversão de esquema para que o QCON realize conversões de consulta mais complexas. Este tipo de abordagem permitiria identificar entidades que são visualizadas como atributos e garantir um tratamento adequado, oferecendo uma possibilidade maior no quesito integração de esquemas.
- O aperfeiçoamento do *framework* DIGE, como forma de facilitar o uso do DIGE em outras infraestruturas compatíveis com o OGSA-DAI, além do GT4.
- Construção de *activities* que permitam a compatibilidade com o componente DQP do OGSA-DAI, permitindo realizar integração vertical de dados com a conversão de consultas.
- Mecanismos de tolerância a falhas, pois em situação de consulta simultânea, caso um *node* da grade não esteja no ar ou durante a execução da consulta uma das máquinas apresente algum problema, o sistema seja capaz de detectar o encerramento inesperado e tomar alguma providência.
- A integração do DIGE com a arquitetura com suporte a replicação proposta por Brito (Brito, 2009). Este processo resultaria em uma ferramenta mais robusta, permitindo que cada instituição pertencente ao sistema utilize seu banco de dados original, garantindo alguns mecanismos de tolerância a falhas como:
 - Replicar dados como forma de garantir disponibilidade, caso algum banco fique fora do ar.
 - O uso de réplicas como cópia de segurança.

REFERÊNCIAS

- ALLCOCK, W. et al. **The globus striped GridFTP framework and server.** Proceedings of the 2005 ACM/IEEE conference on Supercomputing. 2005.
- ALPDEMIR, M. N. et al. **OGSA-DQP: A Service-Based Distributed Query Processor.** UK e-Science All Hands Meeting Nottingham. 2003. p. 501-508.
- ANTONIOLETTI, M. et al. **The design and implementation of grid database services in OGSA-DAI.** Concurrency and Computation: Practice and Experience, n. 17, p. 357–376, 2005.
- APACHE Tomcat. **The Apache Software Foundation**, 2010. Disponível em: <<http://tomcat.apache.org/>>. Acesso em: 05 mar. 2010.
- BARBOSA, A. C. P. **Middleware para Integração de Dados Heterogêneos Baseado em Composição de Frameworks.** 2001. 154p. Tese (Doutorado). Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2001.
- BARBOSA, A. C. P.; PORTO, F.; MELO, R. N. **Configurable data integration middleware.** Journal of the Brazilian Computer Society 8, 2002. 12-19.
- BARBOSA, D. C. M. **Sistemas de informação em saúde; a percepção e a avaliação dos profissionais diretamente envolvidos na atenção básica de Ribeirão Preto / SP.** 2006. 111p. Dissertação (Mestrado). Faculdade de Medicina de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2006.
- BIANCARDI, C. **Distribuição e Execução de Wrappers em Ambiente de Grid para o CoDIMS.** 100 p. Dissertação (Mestrado). Centro Tecnológico, Departamento de Informática, Universidade Federal do Espírito Santo, 2005.
- BOOTH, D.; HAAS, H.; MCCABE, F. **Web Services Architecture.** World Wide Web Consortium (W3C), 2004. Disponível em: <<http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis>>. Acesso em: 22 Janeiro 2010.

BRITO, M. S. D. **Arquitetura de um sistema para integração de bancos de dados com suporte a replicação utilizando tecnologia de grades computacionais.** 115p. Dissertação (Mestrado). Escola Politécnica, Universidade de São Paulo, São Paulo, 2009.

BRITO, M. S. et al. **An Architecture for Integrating Databases with Replication Support Based on the OGSA-DAI Middleware.** 12th IEEE International Conference on Computational Science and Engineering (CSE 2009). Vancouver: 2009. p. 298-305.

BRITO, M. S.; SATO, L. M. **Extending ogsa-dai possibilities with a jdbc driver.** 11th International Conference Computational Science and Engineering, CSE'08. São Paulo: 2008. p. 155-162.

CHINNICI, R. et al. **Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.** World Wide Web Consortium (W3C), 2007. Disponível em: <<http://www.w3.org/TR/2007/REC-wsdl20-20070626/>>. Acesso em: 22 Janeiro 2010.

COMITO, C. et al. **A service-oriented system for distributed data querying and integration on Grids.** Future Generation Computer Systems, v. 25, n. 5, p. 511 - 524, 2009. ISSN 0167-739X.

COMITO, C.; TALIA, D. **GDIS: A Service-Based Architecture for Data Integration on Grids.** In: MEERSMAN, R.; TARI, Z.; CORSARO, A. **On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops.** Springer Berlin / Heidelberg, 2004. p. 88-98.

ELMAGARMID, A.; RUSINKIEWICZ, M.; SHETH, A. **Management of Heterogeneous and Autonomous Database Systems.** San Francisco: Morgan Kaufmann, 1999.

FIGUEIREDO, G.; BRAGANHOLLO, V.; MATTOSO, M. **Um Mediador Para o Processamento de Consultas sobre Bases XML Distribuídas.** Sessão de Demos do Simpósio Brasileiro de Banco de Dados (SBBD), João Pessoa, 2007. 21-26.

FONTES, V. et al. **CoDIMS-G: a data and program integration service for the grid**. Proceedings of the 2nd workshop on Middleware for grid computing. Toronto. 2004. p. 29 - 34.

FOSTER, I. **Globus toolkit version 4: Software for Service-Oriented systems**. IFIP International Conference on Network and Parallel Computing. Springer-Verlag LNCS 3779. 2005. p. pp 2-13.

FOSTER, I. et al. **Grid Services for Distributed System Integration**. Computer Journal, Los Alamitos, v. 35, n. 6, p. 37-46, 2002.

FOSTER, I. et al. **The Physiology of the Grid: An Open Grid Services architecture for Distributed Systems Integration**. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.137.5856>>. Acesso em: 20 jun. 2009.

FOSTER, I.; KESSELMAN, C. **The Grid: Blueprint for a New Computing Infrastructure**. San Francisco: Elsevier, 2004.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. **The Anatomy of the Grid - Enabling Scalable Virtual Organizations**. International Journal of High Performance Computing Applications, 2001. 200–222.

GALLO, E. et al. **GISE: A Data Access and Integration Service of Epidemiological Data for a Grid-Based Monitoring and Simulation System**. 40th Annual Simulation Symposium. Norfolk. 2007. p. 267-274.

GUDGIN, M. et al. **SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)**. World Wide Web Consortium (W3C), 2007. Disponível em: <<http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>>. Acesso em: 22 Janeiro 2010.

HEMOVIDA Web Coagulopatias. **Hemovida**, 2010. Disponível em: <<http://200.214.44.188/>>. Acesso em: 10 jun. 2010.

HOLANDA, A. J. **Proposta de uma Arquitetura Interoperável para um Sistema de Informação em Saúde**. 2005. 87p. Tese (Doutorado). Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FFCLRP), Ribeirão Preto, 2005.

HONG, C. et al. **OGSA-DAI 3.0 - The What's and Whys**. UK e-Science All Hands Meeting. 2007.

LAURE, E. et al. **Programming the Grid with gLite**. Computational Methods in Science and Technology, 2006. 33-45.

LEÃO, B. F. et al. CBIS - Congresso Brasileiro de Informática em Saúde. **O Desafio de Integrar Sistemas de Informação em Saúde**, 2004. Disponível em: <<http://www.sbis.org.br/cbis9/arquivos/919.pdf>>. Acesso em: 24 nov. 2009.

LYNDEN, S. et al. **The design and implementation of OGSA-DQP: A service-based distributed query processor**. Future Generation Computer Systems, v. 25, n. 3, p. 224 - 236, 2009. ISSN 0167-739X.

MACKENZIE, C. M. et al. **Reference model for service oriented architecture 1.0**. OASIS Standart, 2006.

NAVATHE, S.; ELMASRI, R.; LARSON, J. **Integrating User Views in Database Design**. IEEE Computer Society Press, Los Alamitos, CA, USA, p. 50-62, 1986.

OGF. Open Grid Forum. **Open Grid Forum**. Disponível em: <<http://www.gridforum.org/>>. Acesso em: 03 maio 2010.

OGSA-DAI. **OGSA-DAI Project**, 2010. Disponível em: <<http://www.ogsadai.org.uk/>>. Acesso em: 15 jul. 2009.

OMII-UK. **OMII-UK Software Solutions for e-Research**, 2010. Disponível em: <<http://www.omii.ac.uk>>. Acesso em: 15 ago. 2009.

OZSU, M. T.; VALDURIEZ, P. **Principles of Distributed Database Systems**. 2nd edition. ed. Englewood Cliffs: Prentice-Hall, 1999.

PINTO, H. L. M.; BRAGA, J. L. **Sistemas legados e as novas tecnologias: técnicas de integração e estudo de caso**. Informática Pública. Belo Horizonte. 2005. p. 47-70.

REIS, M. G. C. et al. **Diretrizes para apresentação de dissertações e teses/ Divisão de Biblioteca da Epusp**. São Paulo: Universidade de São Paulo. Escola Politécnica. Divisão de Biblioteca, v. III, 2006. 103 p.

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

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos - Principios e Paradigmas**. 2a. ed. São Paulo: Pearson Education, 2008.

TUECKE, S. et al. OGSI-SpecificationV110.pdf. **The Cover Pages**, 2003. Disponível em: <<http://xml.coverpages.org/OGSI-SpecificationV110.pdf>>. Acesso em: 26 Janeiro 2010.

W3C. World Wide Web Consortium (W3C). **World Wide Web Consortium (W3C)**, 2010. Disponível em: <<http://www.w3.org/>>. Acesso em: Janeiro 2010.

Apêndice A – Esquema global utilizado para o Hemosystem

O diagrama a seguir descreve o esquema global adotado pelo Hemosystem para manipular um sistema de bancos de dados integrados utilizando o DIGE.

Apêndice B – Bancos de dados integrados pelo Hemosystem

O Hemosystem utiliza de forma hipotética dois bancos de dados que simulam o banco do hemocentro de São Paulo, o qual foi construído baseado no SGBD Postgres e o banco do hemocentro de Campinas baseado no MySQL. A seguir está a descrição de cada banco de dados.

Banco de dados de São Paulo

Tabela "public.atendimentos"		
Coluna	Tipo	Modificadores
cod_atendimentos	integer	not null default nextval
cod_prontuario	integer	
desc_sintoma	character varying(300)	
desc_diagnostico	character varying(300)	
remedio_receitado	character varying(300)	(FIELD, TYPE, KEY)
ndices:	date	
"atendimentos_pkey" PRIMARY KEY, btree (cod_atendimentos)		

Tabela "public.cadastro"		
Coluna	Tipo	Modificadores
cod_cadastro	integer	not null default nextval
nome	character varying(64)	
nascimento	date	
nome_mae	character varying(64)	
estado_civil	character varying(12)	
cpf	character varying(14)	
ndices:	character varying(12)	
"cadastro_pkey" PRIMARY KEY, btree (cod_cadastro)		

Tabela "public.exame"		
Coluna	Tipo	Modificadores
cod_exame	integer	not null default nextval
cod_prontuario	integer	
desc_exame	character varying(30)	
arquivo	character varying(100)	
ndices:	date	
"exame_pkey" PRIMARY KEY, btree (cod_exame)		

Tabela "public.informa_estoque"		
Coluna	Tipo	Modificadores
cod_informa	integer	not null default nextval
cod_remedio	integer	
data	date	
ndices: qtd	integer	
"informa_estoque_pkey" PRIMARY KEY, btree (cod_informa)		

Tabela "public.medicacao"		
Coluna	Tipo	Modificadores
cod_medicacao	integer	not null default nextval
cod_prontuario	integer	
cod_remedio	integer	
data	date	
qtd_medicacao	integer	
ndices: licado	character varying(1)	
"medicacao_pkey" PRIMARY KEY, btree (cod_medicacao)		

Tabela "public.prontuario"		
Coluna	Tipo	Modificadores
cod_prontuario	integer	not null default nextval
cod_cadastro	integer	
ndices:enca	character varying(20)	
"prontuario_pkey" PRIMARY KEY, btree (cod_prontuario)		

Tabela "public.remedio"		
Coluna	Tipo	Modificadores
cod_remedio	integer	not null default nextval
desc_remedio	character varying(30)	
lote	character varying(30)	
data	date	
qtd_total	integer	
ndices:al idade	date	
"remedio_pkey" PRIMARY KEY, btree (cod_remedio)		

Banco de dados de Campinas

describe atualiza_estoque;						
Field	Type	Null	Key	Default	Extra	
idt_atualiza	int(11)	NO	PRI	NULL	auto_increment	
idt_medicamento	int(11)	YES		NULL		
dat_atualiza	date	YES		NULL		
num_qtd	int(11)	YES		NULL		

describe consultas;						
Field	Type	Null	Key	Default	Extra	
idt_consultas	int(11)	NO	PRI	NULL	auto_increment	
idt_prontuario	int(11)	YES		NULL		
txt_sintoma	varchar(300)	YES		NULL		
txt_diagnostico	varchar(300)	YES		NULL		
txt_medicamento	varchar(300)	YES		NULL		
dta_consulta	date	YES		NULL		

describe endereco;						
Field	Type	Null	Key	Default	Extra	
idt_end	int(11)	NO	PRI	NULL	auto_increment	
idt_pessoa	int(11)	YES		NULL		
nom_end	varchar(100)	YES		NULL		
num_comple	varchar(5)	YES		NULL		
nom_bairro	varchar(30)	YES		NULL		
nom_cidade	varchar(30)	YES		NULL		
nom_federacao	varchar(30)	YES		NULL		
num_cod_post	varchar(10)	YES		NULL		
num_tel	varchar(10)	YES		NULL		
num_cel	varchar(10)	YES		NULL		
txt_email	varchar(30)	YES		NULL		

describe exame;						
Field	Type	Null	Key	Default	Extra	
idt_exame	int(11)	NO	PRI	NULL	auto_increment	
idt_prontuario	int(11)	YES		NULL		
tip_exame	varchar(30)	YES		NULL		
arquivo	varchar(100)	YES		NULL		
dat_exame	date	YES		NULL		

```
describe medicacao;
```

Field	Type	Null	Key	Default	Extra
idt_medicacao	int(11)	NO	PRI	NULL	auto_increment
idt_prontuario	int(11)	YES		NULL	
idt_medicamento	int(11)	YES		NULL	
dat_medicacao	date	YES		NULL	
num_qtd	int(11)	YES		NULL	
bol_aplicacao	varchar(1)	YES		NULL	

```
describe pessoa;
```

Field	Type	Null	Key	Default	Extra
idt_pessoa	int(11)	NO	PRI	NULL	auto_increment
nom_nome	varchar(64)	YES		NULL	
dta_nasc	date	YES		NULL	
nom_mae	varchar(64)	YES		NULL	
nom_est_civil	varchar(12)	YES		NULL	
num_cpf	varchar(14)	YES		NULL	
num_rg	varchar(12)	YES		NULL	

```
describe medicamento;
```

Field	Type	Null	Key	Default	Extra
idt_medicamento	int(11)	NO	PRI	NULL	auto_increment
nom_medicamento	varchar(30)	YES		NULL	
num_lote	varchar(30)	YES		NULL	
dat_entrada	date	YES		NULL	
num_qtd_recebido	int(11)	YES		NULL	
dat_validade	date	YES		NULL	

```
describe prontuario;
```

Field	Type	Null	Key	Default	Extra
idt_prontuario	int(11)	NO	PRI	NULL	auto_increment
idt_pessoa	int(11)	YES		NULL	
tip_hemofilia	varchar(20)	YES		NULL	

Apêndice C – Arquivos de configuração para Hemosystem

A seguir são apresentados os arquivos para mapeamento de esquema utilizado pelo no Hemosystem para realizar a conversão de consulta. Este arquivo indica os dados do esquema global e qual o equivalente no esquema local. Uma das características deste arquivo, é que eles devem ser nomeados com o nome do recurso com a extensão *map*. O arquivo de configuração para o mapeamento deve estar localizado no repositório de esquema localizado no seguinte local \$DAI_HOME/qcong/mapping. Aqui estão apresentados o mapeamento para os dois bancos utilizados, o saopaulo.map que está localizado no SA de São Paulo e o campinas.map que está no SA de Campinas.

saopaulo.map

```
#-----
# Mapeamento para recurso saopaulo
#-----

&paciente=cadastro
paciente.*=cod_cadastro+nome+nascimento+nome_mae+estado_civ+cpf+rg
paciente.id_paciente=cadastro.cod_cadastro
paciente.nome=cadastro.nome
paciente.data_nascimento=cadastro.nascimento
paciente.nome_mae=cadastro.nome_mae
paciente.estado_civil=cadastro.estado_civ
paciente.cpf=cadastro.cpf
paciente.rg=cadastro.rg

&endereco=residencia
endereco.*=cod_residencia+cod_cadastro+residencia+complemento+bairro+munic
pio+estado+cep+tel+cel+email
endereco.id_endereco=residencia.cod_residencia
endereco.id_paciente=residencia.cod_cadastro
endereco.desc_endereco=residencia.residencia
endereco.complemento=residencia.complemento
endereco.bairro=residencia.bairro
endereco.municipio=residencia.municipio
endereco.estado=residencia.estado
endereco.cep=residencia.cep
endereco.telefone=residencia.tel
endereco.celular=residencia.cel
endereco.email=residencia.email

&prontuario=prontuario
prontuario.*=cod_prontuario+cod_cadastro+tipo_doenca
prontuario.id_prontuario=prontuario.cod_prontuario
prontuario.id_paciente=prontuario.cod_cadastro
prontuario.tipo_hemofilia=prontuario.tipo_doenca

&historico=atendimentos
```

```

historico.*=cod_atendimentos+cod_prontuario+desc_sintoma+desc_diagnostico+r
emedio_receitado+data
historico.id_historico=atendimentos.cod_atendimentos
historico.id_prontuario=atendimentos.cod_prontuario
historico.sintoma=atendimentos.desc_sintoma
historico.diagnostico=atendimentos.desc_diagnostico
historico.medicamento_receitado=atendimentos.remedio_receitado
historico.data=atendimentos.data

&exame=exame
exame.*=cod_exame+cod_prontuario+desc_exame+arquivo+data
exame.id_exame=exame.cod_exame
exame.id_prontuario=exame.cod_prontuario
exame.exame=exame.desc_exame
exame.arquivo=exame.arquivo
exame.data=exame.data

&medicacao=medicacao
medicacao.*=cod_medicacao+cod_prontuario+cod_remedio+data+qtd_medicacao+fla
g_aplicado
medicacao.id_medicacao=medicacao.cod_medicacao
medicacao.id_prontuario=medicacao.cod_prontuario
medicacao.id_medicamento=medicacao.cod_remedio
medicacao.data=medicacao.data
medicacao.quantidade=medicacao.qtd_medicacao
medicacao.aplicado=medicacao.flag_aplicado

&medicamento=remedio
medicamento.*=cod_remedio+desc_remedio+lote+data+qtd_total+prazo_validade
medicamento.id_medicamento=remedio.cod_remedio
medicamento.medicamento=remedio.desc_remedio
medicamento.lote=remedio.lote
medicamento.data_entrada=remedio.data
medicamento.quantidade_total=remedio.qtd_total
medicamento.validade=remedio.prazo_validade

&atualiza_medicamento=informa_estoque
atualiza_medicamento.*=cod_informa+cod_remedio+data+informa_qtd
atualiza_medicamento.id_atualizacao=informa_estoque.cod_informa
atualiza_medicamento.id_medicamento=informa_estoque.cod_remedio
atualiza_medicamento.data=informa_estoque.data
atualiza_medicamento.quantidade_atual=informa_estoque.informa_qtd

```

campinas.map

```

#-----
# Mapeamento para recurso campinas
#-----

&paciente=pessoa
paciente.*=idt_pessoa+nom_nome+dta_nasc+nom_mae+nom_est_civil+num_cpf+num_r
g
paciente.id_paciente=pessoa.idt_pessoa
paciente.nome=pessoa.nom_nome
paciente.data_nascimento=pessoa.dta_nasc
paciente.nome_mae=pessoa.nom_mae
paciente.estado_civil=pessoa.nom_est_civil
paciente.cpf=pessoa.num_cpf
paciente.rg=pessoa.num_rg

```

```

&endereco=endereco
endereco.*=idt_end+idt_pessoa+nom_end+num_comple+nom_bairro+nom_cidade+nom_
federacao+num_cod_post+num_tel+num_cel+txt_email
endereco.id_endereco=endereco.idt_end
endereco.id_paciente=endereco.idt_pessoa
endereco.desc_endereco=endereco.nom_end
endereco.complemento=endereco.num_comple
endereco.bairro=endereco.nom_bairro
endereco.municipio=endereco.nom_cidade
endereco.estado=residencia.nom_federacao
endereco.cep=endereco.num_cod_post
endereco.telefone=endereco.num_tel
endereco.celular=residencia.num_cel
endereco.email=endereco.txt_email

```

```

&prontuario=prontuario
prontuario.*=idt_prontuario+idt_pessoa+tip_hemofilia
prontuario.id_prontuario=prontuario.idt_prontuario
prontuario.id_paciente=prontuario.idt_pessoa
prontuario.tipo_hemofilia=prontuario.tip_hemofilia

```

```

&historico=consultas
historico.*=idt_consultas+idt_prontuario+txt_sintoma+txt_diagnostico+txt_me
dicamento+dta_consulta
historico.id_historico=consultas.idt_consultas
historico.id_prontuario=consultas.idt_prontuario
historico.sintoma=consultas.txt_sintoma
historico.diagnostico=consultas.txt_diagnostico
historico.medicamento_receitado=consultas.txt_medicamento
historico.data=consultas.dta_consulta

```

```

&exame=exame
exame.*=idt_exame+idt_prontuario+tip_exame+arquivo+dat_exame
exame.id_exame=exame.idt_exame
exame.id_prontuario=exame.idt_prontuario
exame.exame=exame.tip_exame
exame.arquivo=exame.arquivo
exame.data=exame.dat_exame

```

```

&medicacao=medicacao
medicacao.*=idt_medicacao+idt_prontuario+idt_medicamento+dat_medicacao+num_
qtd+bol_aplicacao
medicacao.id_medicacao=medicacao.idt_medicacao
medicacao.id_prontuario=medicacao.idt_prontuario
medicacao.id_medicamento=medicacao.idt_medicamento
medicacao.data=medicacao.dat_medicacao
medicacao.quantidade=medicacao.num_qtd
medicacao.aplicado=medicacao.bol_aplicacao

```

```

&medicamento=medicamento
medicamento.*=idt_medicamento+nom_medicamento+num_lote+dat_entrada+num_qtd_
recebido+dat_validade
medicamento.id_medicamento=medicamento.idt_medicamento
medicamento.medicamento=medicamento.nom_medicamento
medicamento.lote=medicamento.num_lote
medicamento.data_entrada=medicamento.dat_entrada
medicamento.quantidade_total=medicamento.num_qtd_recebido
medicamento.validade=medicamento.dat_validade

```

```

&atualiza_medicamento=atualiza_estoque
atualiza_medicamento.*=idt_atualiza+idt_medicamento+dat_atualiza+num_qtd

```

```
atualiza_medicamento.id_atualizacao=atualiza_estoque.idt_atualiza  
atualiza_medicamento.id_medicamento=atualiza_estoque.idt_medicamento  
atualiza_medicamento.data=atualiza_estoque.dat_atualiza  
atualiza_medicamento.quantidade_atual=atualiza_estoque.num_qtd
```

Apêndice D – Arquivos de configuração de recursos

O arquivo a seguir descreve um banco exposto como um recurso computacional em um SA. Para este primeiro arquivo, informações como o identificador do recurso (nome do recurso), o tipo de recurso, características do SGBD utilizado, características do banco, como o local do *driver* e a localização do banco, e as *activities* que estão habilitadas para utilizar este recurso.

Arquivo de configuração para o recurso saopaulo

```
id=saopaulo
type=uk.org.ogsadai.DATA_RESOURCE
creationTime=null
terminationTime=null
PROPERTIES
uk.org.ogsadai.resource.dataresource.product=Postgresql
uk.org.ogsadai.resource.dataresource.vendor=Postgresql
uk.org.ogsadai.resource.dataresource.version=8.3.9
END
CONFIG
dai.driver.class=org.postgresql.Driver
dai.data.resource.uri=jdbc:postgresql://localhost/hemosystem1
dai.login.provider=uk.org.ogsadai.LOGIN_PROVIDER
END
ACTIVITIES
uk.org.ogsadai.SQLQuery=uk.org.ogsadai.SQLQuery
uk.org.ogsadai.SQLUpdate=uk.org.ogsadai.SQLUpdate
org.lahpc.qcong.server.SQLQueryQCONG=org.lahpc.qcong.server.SQLQueryQCONG
org.lahpc.qcong.server.SQLUpdateQCONG=org.lahpc.qcong.server.SQLUpdateQCONG
END
dataResourceClass=uk.org.ogsadai.resource.dataresource.jdbc.JDBCDataResource
```

Este segundo arquivo é para expor um recurso que “aponta” para outro recurso em outro SA. As informações mais relevantes aqui são a localização do recurso e o nome do recurso no outro SA. Este tipo de recurso é utilizado para delegar a execução de uma consulta para outro SA.

Arquivo de configuração para o recurso campinasR

```
id=campinasR
type=uk.org.ogsadai.DATA_RESOURCE
creationTime=null
terminationTime=null
PROPERTIES
END
CONFIG
dai.data.resource.uri=http://143.107.164.250:8080/wsrf/services/dai/
dai.data.resource.id=campinas
END
ACTIVITIES
uk.org.ogsadai.SQLQuery=uk.org.ogsadai.RemoteAsynchSQLQuery
```

```
END
dataResourceClass=uk.org.ogsadai.resource.dataresource.remote.RemoteResourc
e
```

Por fim, o recurso que expõe um conjunto de recursos. Aqui a informação relevante é a lista de recursos que fazem parte deste recurso. É esta configuração que permite o acesso a um conjunto de recursos a partir de um recurso.

Arquivo de configuração para o recurso ambos

```
id=ambos
type=uk.org.ogsadai.DATA_RESOURCE
creationTime=null
terminationTime=null
PROPERTIES
END
CONFIG
    saopaulo=saopaulo
    campinasR=campinasR
END
ACTIVITIES
uk.org.ogsadai.SQLBag=uk.org.ogsadai.SQLBag
org.lahpc.qcong.server.SQLBagQCONG= org.lahpc.qcong.server.SQLBagQCONG
END
dataResourceClass=uk.org.ogsadai.resource.dataresource.group.ResourceGroupD
ataResource
```