DIG: UM SERVIÇO PARA PROVER CUSTOS E ESTATÍSTICAS PARA O PROCESSAMENTO DISTRIBUÍDO DE CONSULTAS

Nicolaas Ruberg

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Profª. Maria Luiza Machado Campos, Ph.D.

Prof. Fábio André Machado Porto, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

NOVEMBRO DE 2002

RUBERG, NICOLAAS

DIG: Um Serviço para Prover Custos e Estatísticas para o Processamento Distribuído de Consultas [Rio de Janeiro] 2002

X, 101 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Processamento de Consultas
2. Otimização de Consultas
3. Aquisição de Estatísticas e Custos
4. SGBDs Distribuídos e Heterogêneos

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DIG: UM SERVIÇO PARA PROVER CUSTOS E ESTATÍSTICAS PARA O PROCESSAMENTO DISTRIBUÍDO DE CONSULTAS

Nicolaas Ruberg

Novembro/2002

Orientador: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

Parâmetros de custos e estatísticas constituem a base das técnicas de otimização de consultas. Contudo, em ambientes distribuídos e heterogêneos, a aquisição e o tratamento dessas informações costumam ser abordados como tarefas do processador global de consultas, limitando tais funcionalidades a uma arquitetura de sistema específica. Além disso, nesses ambientes, o processo de aquisição de custos envolve um número grande de parâmetros e requer métodos adequados para coleta de dados em fontes específicas.

O DIG (*Distributed Information Gatherer*) consiste em um provedor de custos e estatísticas que, através de um serviço flexível e independente, visa apoiar o processo de otimização global de consultas em um ambiente distribuído, heterogêneo e com fontes de dados autônomas. A arquitetura do DIG apresenta dois tipos básicos de componentes: módulos provedores de custos e estatísticas, que realizam o tratamento e a publicação dos dados coletados; e módulos coletores responsáveis pela aquisição de dados nas diversas fontes, repassando-os para os respectivos provedores. A aquisição de dados é realizada pelo coletor DIG através da submissão de consultas ou comandos pré-estabelecidos (em um arquivo específico de configuração) para cada estatística ou parâmetro de custo a ser coletado. Os dados coletados são publicados pelo provedor DIG através de um catálogo genérico de custos e estatísticas, que suporta desde fontes de dados semi-estruturadas ou não estruturadas (por exemplo, arquivos de texto e páginas *Web*) até fontes de dados com SGBDs sofisticados.

Nós desenvolvemos um protótipo do DIG que foi avaliado com coletores específicos para um *middleware* de consulta sobre fontes de dados semi-estruturados e também para um protótipo de SGBD baseado em objetos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DIG: A SERVICE TO PROVIDE COSTS AND STATISTICS FOR DISTRIBUTED QUERY PROCESSING

Nicolaas Ruberg

November/2002

Advisor: Marta Lima de Queirós Mattoso

Department: Computer Science and Systems Engineering

Cost parameters and database statistics are the basis of query optimization techniques. However, in distributed and heterogeneous database systems, acquiring and treating information to help the optimization process are often tasks of a global query processor, tailoring these functionalities to a specific system architecture. Moreover, this acquisition process involves a large number of parameters and requires customized methods to retrieve data from specific sources.

DIG (Distributed Information Gatherer) is a cost and statistics provider that, through an independent and flexible service, aims to support global query optimization processing in distributed, heterogeneous database systems over autonomous data sources. The DIG architecture presents two basic components: a module provider of costs and statistics, that does the handling and the publication of the collected data; and module collectors responsible for the data acquisition on diverse sources, which pass the collected data to the according provider. The data acquisition is performed by the DIG collector through the submission of queries or predefined commands (in a specific configuration file) for each statistic or cost parameter being collected. The collected data are published by the DIG provider through a generic catalog of costs and statistics, which supports from semi-structured or non-structured data sources (e.g., text files and web pages) to data sources with sophisticated SGBDs.

We have developed a DIG prototype and evaluated it with specific wrappers for a query middleware on semi-structured data sources and also for an object-based DBMS.

AGRADECIMENTOS

À prof[a]. Marta Mattoso, pela orientação e pelos conhecimentos transmitidos durante o trabalho desenvolvido.

À banca examinadora, prof[a]. Maria Luiza Campos e Fabio Porto, pelas considerações e sugestões ao trabalho.

À Gabriela Ruberg esposa dedicada, editora minuciosa e consultora sagaz que me motivou e permitiu que este trabalho fosse realizado.

Ao meu pai, Wilhelmus Ruberg, inspiração em todos os momentos.

À minha mãe, Angela Ruberg, incentivo na busca do crescimento.

Ao amor das minhas irmãs, Claudia e Adriana.

À alegria e amor dos meus avós, Izolina e Arcídio.

À minha avó Joanna e tia Gerrie pelo carinho e atenção.

Ao meu tio Carlos, tia Nídia e prima Mariana.

Aos amigos, professores e colegas que direta ou indiretamente contribuíram para esta conquista.

ÍNDICE DO TEXTO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Capítulo 1

INTRODUÇÃO

O contexto, a motivação e os principais objetivos deste trabalho são apresentados neste capítulo, além de uma breve descrição sobre a organização do texto e os capítulos restantes.

1.1 MOTIVAÇÃO

O considerável progresso no armazenamento persistente de dados e na tecnologia de redes impulsionou o crescimento explosivo das bibliotecas digitais e a demanda por informações em tempo real. Dentre os novos cenários tecnológicos, o mais popular é a *World Wide Web* (*Web*), pois permite acesso a diversas fontes de dados de forma simples e independente da localização física. Porém, as interfaces disponibilizadas pela *Web* não são suficientes para garantir eficiência no acesso aos grandes volumes de dados disponíveis na rede. Neste contexto, são necessários mecanismos que auxiliem os sistemas de consulta na identificação da capacidade de processamento das diferentes fontes de dados, no intuito de definir as melhores alternativas para a execução das consultas submetidas.

Existem diversas propostas de arquiteturas para os ambientes de consulta com características da *Web*, que são distribuição, heterogeneidade e autonomia. Dentre as arquiteturas mais conhecidas, podemos citar[1]: arquitetura *multidatabase* (*MultiDataBase System* - MDBS) (BOUGUETTAYA *et al.*, 1999, ZHU e LARSON, 1998); bancos de dados federados (SHETH e LARSON, 1990); *middleware* de consulta (OBJECTDRIVER, 2002, SIMON, 2001); sistemas de consulta mediados (MQS – *Mediated Query Systems*) (DITTRICH e DOMENIG, 1999, HAAS *et al.*, 1997, NAACKE *et al.*, 1998, PIRES, 1997); e provedores de serviço de consulta (BRAUMANDL *et al.*, 2001).

A arquitetura MDBS e os bancos de dados federados estão normalmente associados à existência de Sistemas Gerenciadores de Bases de Dados (SGBDs) e de uma forte definição de esquema global. Já um *middleware* de consulta é definido por unidades que permitem consultar diferentes fontes de dados sem considerar a existência de um catálogo global, mas oferecem pouco ou nenhum suporte para a integração semântica do resultado de uma consulta. Os sistemas de consulta mediados fornecem uma interface global para consulta em um ambiente distribuído, onde o retorno de cada

[1] Cabe ressaltar que os nomes aqui utilizados para as arquiteturas não são consenso entre os autores. Os termos *Multidatabase* e Bancos de Dados Federados, por exemplo, são usados distintamente por autores como SHETH e LARSON (1990) e ÖZSU e VALDURIEZ (1999).

consulta possui uma apresentação unificada e de fácil compreensão. A implementação de um MQS é realizada por mediadores, responsáveis pela integração semântica da consulta, e por tradutores para as fontes de dados (WIEDERHOLD, 1992). Recentemente, tem sido citada uma nova tecnologia baseada em provedores de serviços, a qual facilita e agiliza a integração de aplicações através de um modelo comum para comunicação entre programas (KREGER, 2001). Este modelo é definido por padrões atuais como o HTTP (*HyperText Transport Protocol*), o XML (*eXtensible Markup Language*), o SOAP (*Simple Object Access Protocol*), a WSDL (*Web Services Description Language*) e o UDDI (*Universal Description, Discovery and Integration*), típicos da *Web* (VAUGHAN-NICHOLS, 2002). Entretanto, essa tecnologia não resolve as questões semânticas tratadas em MQSs, MDBSs e bancos de dados federados.

A motivação inicial do nosso trabalho foi contribuir para a otimização de consultas em uma arquitetura de mediadores através de um provedor de custos e estatísticas. Em um sistema de consulta mediado, cabe ao mediador determinar o melhor plano global de execução para as consultas submetidas. Esse plano de execução é montado a partir da decomposição da consulta original em subconsultas que são enviadas para os tradutores relacionados (OZSÜ e VALDURIEZ, 1999). A geração de um plano global eficiente é uma tarefa extremamente complexa, pois um grande número de fatores deve ser considerado. A escolha do plano global é, na grande maioria das vezes, auxiliada por um modelo de custo que indica a eficiência do plano de execução em análise. A acurácia do modelo de custo é função dos parâmetros utilizados, isto é, das métricas de desempenho. Portanto, é essencial que métricas de desempenho, baseadas em custos e estatísticas sobre as fontes de dados, sejam utilizadas para orientar o processo de otimização global (ROTH *et al.*, 1999). Além disso, diferentemente de uma arquitetura de banco de dados tradicional, nem sempre as fontes de dados de um MQS estão associadas a SGBDs. Esse fato possui duas implicações relevantes:

(i) o mediador precisa conhecer a real capacidade de processamento de consultas de cada fonte de dados; e

(ii) a aquisição de custos e estatísticas sobre uma fonte de dados desprovida de um SGBD requer métodos especiais para inspeção dos dados e geração das estimativas.

Neste contexto, também é importante que o mediador possa extrair parâmetros para a definição de critérios relativos à qualidade de serviço (QoS – *Quality of Service*) oferecida pelos tradutores, como disponibilidade e tempo médio de espera pelo serviço (MANI e NAGARAJAN, 2002).

1.2 OBJETIVOS

Este trabalho apresenta o *Distributed Information Gatherer* (DIG), um provedor de custos e estatísticas que visa apoiar o processo de otimização global de consultas em um ambiente distribuído, heterogêneo e com fontes de dados autônomas. A arquitetura do DIG define as unidades de *software*, as interfaces, o catálogo de serviços e os principais procedimentos necessários para a aquisição e a publicação das características gerais, custos e estatísticas sobre o processamento de consultas em diferentes fontes de dados. Em suma, o DIG oferece um serviço que modela e provê informações quanto à capacidade de processamento de consultas de fontes autônomas em ambientes distribuídos e heterogêneos, seja em um MDBSs, seja na *Web* ou em outros ambientes com distribuição e heterogeneidade.

Nas arquiteturas distribuídas e heterogêneas, os trabalhos sobre o provimento de custos e estatísticas para o processamento de consultas estão relacionados a funcionalidades embutidas nos módulos responsáveis pela otimização global das consultas e pelo acesso aos dados. Em MQSs, por exemplo, tais funcionalidades são desempenhadas pelo mediador de consulta e seus tradutores. Avaliamos diferentes arquiteturas quanto ao provimento de custos e estatísticas, dentre as quais destacamos os seguintes trabalhos: o sistema CORDS (ZHU e LARSON, 1998); as arquiteturas de mediadores *Garlic* (GARLIC, 1999, HAAS *et al.*, 1997) e DISCO (NAACKE *et al.*, 1998); e o projeto *ObjectGlobe* (BRAUMANDL *et al.*, 2001).

Na arquitetura *multidatabase* CORDS, o servidor MDBS inclui funcionalidades para realizar uma coleta de amostras de desempenho a partir da execução de consultas especiais nas fontes de dados. O servidor MDBS mantém uma base de dados sobre o desempenho médio dos algoritmos tradicionais de consulta em um SGBD, calculado a partir das amostragens realizadas. Em função das amostras de desempenho coletadas, o servidor MDBS infere quais são os algoritmos de execução utilizados e os respectivos

custos de execução. A arquitetura CORDS possui um bom suporte para fontes de dados que possuem SGBDs, mas não apresenta métodos para a aquisição de custos e estatísticas em fontes de dados desprovidas de SGBDs.

A arquitetura de mediadores *Garlic* considera que um plano de execução de uma consulta é representado por uma árvore de operadores ditos POPs (*Plan OPerators*), onde cada POP corresponde a um operador algébrico de consulta (por exemplo, *scan*). Os planos de execução das subconsultas enviadas para cada tradutor são tratados como operadores de alto nível, ditos PUSHDOWN POPs, cujo custo é calculado pelo tradutor a partir dos custos dos POPs envolvidos. A arquitetura *Garlic* pressupõe que o mediador não precisa conhecer os custos dos POPs das fontes de dados de uma consulta, limitando-se à estimativa de custo do PUSHDOWN POP informada por cada tradutor. Na nossa proposta, assim como no *Garlic*, reconhecemos a importância do tradutor auxiliar o processo de estimativa de custos. Porém, o modelo de custos do *Garlic* é limitado a tradutores que consigam fornecer as estimativas do PUSHDOWN POP.

A abordagem para otimização de consultas baseada em custos da arquitetura de mediadores DISCO (*Distributed Information Search COmponent*) combina o uso de um modelo de custo genérico e de informações de custo específicas. Nessa abordagem, os tradutores são responsáveis por exportar para o mediador os custos e estatísticas específicos de suas fontes de dados. Por sua vez, o mediador também mantém um modelo de custo genérico que é aplicado às fontes de dados cujos custos não são informados pelos respectivos tradutores. O DISCO reforça a importância do tradutor fornecer as estimativas de custo e estatísticas sobre a fonte de dados. Porém, o conjunto de estatísticas fornecidas pelo DISCO é bastante simples, restringindo os tipos de modelos de custo empregados pelo mediador.

Observa-se que, nos diversos sistemas analisados, a obtenção de custos para a otimização de consultas é realizada através de um módulo proprietário da arquitetura, incorporado ao processador global de consultas. Entretanto, a aquisição de custos e estatísticas em um módulo independente confere maior flexibilidade e generalidade a esse tipo de serviço. Um catálogo de estatísticas e custos para o processamento de consultas em um ambiente distribuído é muito sensível a mudanças como a inclusão de novas fontes de dados e a alteração dos esquemas locais publicados para a arquitetura de integração. Com a separação do processo de aquisição desses dados, obtêm-se maior

controle sobre o impacto das alterações no ambiente, isolando-se e encapsulando-se a heterogeneidade das fontes de dados quanto ao catálogo de estatísticas e custos. Deste modo, ao fornecer um modelo genérico de parâmetros de custos, estatísticas e operações típicas de um MQS, aliado a um serviço de obtenção desses parâmetros, essa funcionalidade torna-se disponível para qualquer arquitetura de mediadores, bem como para outros sistemas que necessitem conhecer a capacidade de processamento de consultas das fontes de dados em um ambiente distribuído. Esta questão é relevante em cenários como a *Web* e como em plataformas de *grid* (FOSTER *et al.*, 2001), sendo essa generalidade e flexibilidade o principal foco da nossa pesquisa.

Um trabalho próximo ao nosso é o projeto *ObjectGlobe*, que evoluiu para uma versão baseada em serviços *Web* chamada *ServiceGlobe* (KEIDL *et al.*, 2002), onde é previsto um módulo que representa um catálogo de localização de serviços e publicação de custos e estatísticas sobre provedores de dados, de funções e de processamento. Entretanto, a aquisição de dados deste serviço é passiva, isto é, depende dos demais componentes da arquitetura publicarem (repassarem) suas informações. Além disso, ainda cabe ao processador global de consultas tratar a ausência de informações sobre os provedores de serviços, mantendo para isso um modelo de custo genérico.

A arquitetura do DIG oferece um serviço flexível e independente, apto a realizar uma aquisição ativa de dados. Ou seja, os custos e estatísticas são adquiridos pela arquitetura periodicamente (aquisição programada) ou através de solicitação do usuário (aquisição por demanda). O DIG apresenta dois tipos básicos de componentes (RUBERG *et al.*, 2002b): módulos provedores de custos e estatísticas, que realizam o tratamento e a publicação dos dados coletados; e módulos coletores responsáveis pela aquisição de dados nas diversas fontes, repassando-os para os respectivos provedores.

A aquisição de dados é realizada pelo coletor DIG através da submissão de consultas ou comandos pré-estabelecidos (em um arquivo específico de configuração) para cada estatística ou parâmetro de custo a ser coletado. Os dados coletados são publicados pelo provedor DIG através de um catálogo genérico de custos e estatísticas que permite a sua utilização no processamento de consultas em ambientes distribuídos, heterogêneos e com fontes de dados autônomas. Além disso, a publicação do catálogo de serviços do DIG é adaptável, pois o catálogo de serviços suporta desde fontes de dados semi-estruturadas ou não estruturadas (por exemplo, arquivos de texto e páginas

Web) até fontes de dados com SGBDs sofisticados, oferecendo uma interface adequada para cada caso.

Os módulos do sistema DIG foram implementados em um protótipo desenvolvido na linguagem de programação C++. Foram considerados dois tipos diferentes de fontes de dados: o protótipo de SGBD baseado em objetos GOA (MAURO *et al.*, 1997, GOA, 2002) e o *middleware* de consulta *LeSelect* (SIMON, 2001). A implementação do coletor DIG utiliza um tradutor de consultas baseado na arquitetura de mediadores HIMPAR apresentada em PIRES (1997).

Para verificar a viabilidade da arquitetura proposta, o protótipo do DIG foi avaliado sobre bases de dados relativas a empréstimos financeiros, distribuídas entre fontes de dados GOA e *LeSelect*. O experimento realizado consistiu em recuperar custos, estatísticas e informações sobre estas bases, disponibilizando os dados coletados através do formato XML. Foram avaliados aspectos como a confecção do arquivo de publicação, o processo de aquisição dos custos pelos coletores e a gerência do provedor DIG sobre os dados coletados. Verificamos que a aquisição dos custos e estatísticas foi realizada com sucesso, bem como as informações coletadas mostraram-se relevantes para a otimização das possíveis consultas sobre a base disponibilizada.

1.3 ORGANIZAÇÃO DO TEXTO

Este trabalho está organizado em cinco capítulos distribuídos da seguinte forma.

O Capítulo 2 apresenta os cenários onde é necessária a integração de informação e discorre sobre as características das fontes típicas de dados nesses ambientes. São apresentadas também as principais arquiteturas propostas na literatura para a integração de informação: Bancos de Dados Federados; arquitetura *multidatabase*; *middlewares* de consulta; arquitetura de mediadores; e os sistemas mediados de consulta. Estas arquiteturas são analisadas quanto ao tratamento dos aspectos de: distribuição, heterogeneidade e autonomia.

O Capítulo 3 aborda a aquisição de custos e estatísticas para o processamento de consultas em Sistemas Mediados de Consulta. Nesse contexto, são apresentadas as duas formas tradicionais para otimização de consultas: através de heurísticas e/ou através de custos. Destacamos as características da otimização de consultas baseada em custos,

primeiramente em um ambiente centralizado e, em seguida, em um ambiente distribuído, heterogêneo e com fontes autônomas. Identificamos as diferenças entre esses dois cenários e abordamos a complexidade acrescida na otimização pelas características de distribuição e heterogeneidade. Ao final deste capítulo, apresentamos os principais trabalhos relacionados à aquisição de custos e estatísticas no processamento distribuído de consultas.

O Capítulo 4 apresenta as características e os módulos funcionais do *Distributed Information Gatherer* (DIG), a arquitetura proposta para a aquisição e publicação de informações, custos e estatísticas sobre o processamento de consultas em ambientes distribuídos, heterogêneos e com fontes de dados autônomas. Foram identificadas as informações, custos e estatísticas necessárias ao processamento de consultas tal que o DIG abrangesse uma grande variedade de fontes de dados. Esses dados foram utilizados para definir o catálogo de publicação do DIG. Além das definições sobre as informações a serem coletadas pelo DIG, também foi especificado o processo de aquisição destes dados.

O protótipo desenvolvido para verificar a viabilidade da arquitetura do DIG é apresentado no Capítulo 5. Descrevemos o ambiente utilizado no desenvolvimento do protótipo do DIG e quais foram os padrões utilizados na programação dos módulos implementados. Nesse capítulo são reportados os detalhes do experimento realizado para a validação do DIG, cujos testes de aquisição foram realizados em um Sistema Mediado de Consulta distribuído em diferentes fontes de dados. Descrevemos também as bases de dados que foram utilizadas no processo de validação.

Por fim, no Capítulo 6 são expostas as conclusões deste trabalho, destacando as contribuições apresentadas e algumas perspectivas de trabalhos futuros.

Capítulo 2

ARQUITETURAS PARA INTEGRAÇÃO DE INFORMAÇÃO

Neste capítulo são abordados aspectos relativos à integração de informação em ambientes distribuídos, heterogêneos e com fontes autônomas de dados. São apresentadas as principais arquiteturas que provêem essa integração e como essas propostas permitem a realização de consultas sobre as fontes de dados.

2.1 INTRODUÇÃO

Para melhor compreender a arquitetura do DIG, proposta nesta dissertação, é necessário conhecer o contexto em que o DIG pode ser aplicado. O DIG é uma ferramenta que apoia o processo de otimização de consultas em um ambiente distribuído, heterogêneo e com fontes autônomas de dados. Portanto, faz-se necessário conhecer as principais arquiteturas encontradas na literatura que tratam a integração de informação e permitem a realização de consultas sobre estas fontes.

A necessidade da integração de informação surgiu do considerável progresso no armazenamento persistente de dados e na tecnologia de redes, que permitiu a publicação e o acesso a grandes volumes de informação. Tal progresso no armazenamento persistente pode ser observado nas corporações modernas, cujos dados são encontrados espalhados em sistemas legados, em sistemas gerenciadores de bases de dados e em bases textuais (ABITEBOUL, 1997). Por outro lado, o avanço na tecnologia de redes permitiu que esses dados, persistidos sobre diversas tecnologias, pudessem ser processados de forma distribuída. Isto pode ser observado nos produtos oferecidos pelos principais fornecedores de sistemas de bancos de dados (IBM, *Oracle*, *Microsoft* e *Sybase*) (KOSSMANN, 2000).

Além disso, mais recentemente, um grande volume de dados passou a estar disponível em páginas *Web*, publicadas em diversos computadores que utilizam a Internet como infra-estrutura. Nas páginas *Web*, os dados estão representados tradicionalmente através do padrão HTML (*Hyper Text Markup Language*) (FLORESCU *et al.*, 1998), onde um conjunto de marcadores define as características visuais dos dados a serem apresentados.

A *Web* pode ser considerada como uma grande base de dados distribuída, cujo mecanismo típico de consulta são as ferramentas de busca por palavras. Uma consulta nestas ferramentas é submetida e processada da seguinte forma (SERGEY e LAWRENCE, 1998): palavras significativas são informadas com critérios de seleção e a ferramenta de busca retorna as páginas HTML (FLORESCU *et al.*, 1998) em que estas palavras são encontradas, respeitando os critérios especificados. Para operacionalizar este tipo de consulta, as ferramentas precisam recuperar previamente as páginas HTML

a partir dos sítios *Web* e criar índices para as diversas palavras contidas nestas páginas. Em função do respectivo marcador HTML, uma palavra ou uma frase pode ter recebido um peso diferenciado para auxiliar a busca.

Por outro lado, os Sistemas Gerenciadores de Bases de Dados (SGBDs) tradicionais utilizam linguagens de consultas *ad hoc* que permitem a formulação de consultas mais sofisticadas e com resultados mais precisos do que uma simples busca por palavras. Isto é possível porque os SGBDs utilizam um esquema definido *a priori.* Na *Web*, as ferramentas típicas de consulta reconhecem apenas palavras, ou seja, os dados são conjuntos de palavras que podem ser classificadas em função de seus marcadores visuais e de seu sítio *Web* (SERGEY e LAWRENCE, 1998). Já nos SGBDs, é mantido um catálogo de descrições sobre os tipos que definem a base de dados (ELMASRI e NAVATHE, 1994). Na abordagem de banco de dados existe um modelo de representação da informação onde são descritas as coleções de dados e seus respectivos relacionamentos. Através do esquema de um SGBD, além de consultas sobre as coleções dos dados, também é possível realizar consultas sobre os relacionamentos entre estes dados.

Para permitir consultas em ambientes cujas fontes de dados possam ser sítios *Web*, SGBDs ou outro tipo de fonte de dados, é necessária uma infra-estrutura distribuída que disponibilize as informações destas fontes de forma integrada. O processo de integração de informação deve lidar com a distribuição das fontes e com a heterogeneidade dos modelos de dados (ASHISH e KNOBLOCK, 1997). Além disso, a falta de gerência ou controle sobre as fontes de dados, relativa à autonomia destas fontes, dificulta ainda mais a realização de consultas em ambientes distribuídos e heterogêneos.

Neste capítulo são abordados aspectos relativos à integração de informação em ambientes distribuídos, heterogêneos e com fontes autônomas de dados. Além disso, são detalhadas as principais arquiteturas encontradas na literatura sobre a integração de informação. Por fim, são identificados os requisitos básicos de uma arquitetura de *software* que permita o processamento de consultas nesse contexto.

2.2 DISTRIBUIÇÃO, HETEROGENEIDADE E AUTONOMIA

O principal desafio da integração de informação é fornecer uma interface que permita um acesso uniforme ao vasto conjunto de fontes de dados disponíveis na *Web* e nas redes corporativas. Idealmente, esta interface deve permitir que as fontes de dados sejam vistas como uma única base de dados. Contudo, qualquer solução de sucesso para integrar fontes de dados preexistentes, espalhadas em uma corporação e na Internet, deve observar questões sobre distribuição, heterogeneidade e autonomia (ÖZSU e VALDURIEZ, 1999, ELMAGARMID *et al.*, 1999).

A distribuição das fontes de dados refere-se à separação e à localização física das estruturas que armazenam os dados. Em muitos ambientes, os dados podem estar distribuídos entre múltiplos bancos de dados ou em vários sítios *Web*. Além disso, podemos ter SGBDs onde os dados de um mesmo tipo estão distribuídos em subconjuntos próprios (partições verticais e/ou horizontais, conforme critérios de fragmentação previamente definidos), ou ainda, em múltiplas réplicas. As fontes de dados de um ambiente distribuído podem estar espalhadas em um mesmo sítio computacional, em uma rede local ou em uma rede de longa distância.

A heterogeneidade diz respeito às diferenças entre as fontes de dados quanto à estrutura, formato e semântica dos dados. As fontes de dados podem ser relacionais, orientadas a objeto ou ter um outro modelo de dados. Diferentes sítios podem ter abordagens distintas para implementar SGBDs, linguagens de consulta, linguagens de programação e sistemas de arquivos. Cada fonte de dados pode ter um esquema local de dados próprio, sendo que a integração de várias fontes requer o mapeamento entre os respectivos esquemas locais e um esquema global unificado. Esse mapeamento geralmente envolve inúmeros conflitos que devem ser solucionados (WIEDERHOLD, 1992).

Podemos dizer que uma fonte é autônoma quando esta possui um controle isolado e/ou independente de outras fontes de dados ou unidades externas. A autonomia pode ser observada sobre três aspectos: i) autonomia de projeto; ii) autonomia de comunicação; e iii) autonomia de execução. Na autonomia de projeto, a fonte é quem define o modelo de dados, a linguagem de consulta, a interpretação semântica dos dados e as restrições aplicadas, entre outros aspectos relacionados à representação da

informação. A autonomia de comunicação trata do livre arbítrio da fonte de dados para definir quando e como serão respondidas as consultas submetidas. Na autonomia de execução, a ordem das transações e operações executadas na fonte de dados não é controlada por nenhuma unidade externa. Neste caso, a fonte de dados não distingue se a operação é local ou global. As fontes de dados são capazes de se associar ou se desassociar do ambiente global, assim como de definir que operações e dados serão disponibilizados para o ambiente global (por exemplo, projeções, seleções, custos e estatísticas).

Integrar diversas fontes de dados é permitir que operações de consultas e/ou atualizações sejam realizadas com uma visão global, de forma transparente quanto à localização e à representação dos dados das fontes. À medida que aumentam as diferenças entre os dados das fontes, mais complexa se torna essa integração. De forma geral, o processo de integração de fontes de dados ocorre em dois passos: i) tradução do esquema local; e ii) integração do esquema global. No primeiro passo, os esquemas das fontes de dados são traduzidos para uma representação canônica (padrão) intermediária. O uso de uma representação canônica facilita o processo de tradução, simplificando os tradutores a serem escritos. A escolha do modelo canônico define o poder de expressão de uma arquitetura de integração. Em princípio, o modelo escolhido deve permitir a expressão necessária para incorporar os conceitos disponíveis em todas as fontes de dados que serão integradas.

No segundo passo da integração de informação, cada esquema intermediário é integrado a um esquema global conceitual. Em algumas metodologias, esquemas locais externos são considerados para a integração, ao invés dos esquemas locais completos, pois pode não ser conveniente incorporar todo o esquema local. Por exemplo, quando a fonte de dados possui informações incompletas e/ou sigilosas, publica-se para a arquitetura de integração um esquema local diferente do existente na fonte de dados.

A seguir, são apresentadas as principais arquiteturas encontradas na literatura sobre a integração de fontes autônomas de dados em um ambiente distribuído e heterogêneo, a partir das quais identificamos um modelo de referência.

2.3 AS ARQUITETURAS DE INTEGRAÇÃO

Existem diversas propostas de arquiteturas para integração de informação que permitem o processamento de consultas sobre fontes de dados distribuídas (OZSÜ e VALDURIEZ, 1999, SHETH e LARSON, 1990, ELMAGARMID *et al.*, 1999). Dentre as mais conhecidas, podemos citar: bancos de dados federados; arquitetura *multidatabase* (*MultiDataBase System* - MDBS); *middleware* de consulta; arquitetura de mediadores; e sistemas de consulta mediados. É importante ressaltar que não existe um consenso entre os autores sobre a terminologia das arquiteturas de integração, nem sobre a definição de tais arquiteturas.

Cada uma dessas arquiteturas aborda de forma distinta os aspectos de distribuição, heterogeneidade e autonomia. Os bancos de dados federados são baseados na integração completa de diversos SGBDs, de forma a permitir uma visão única e total dos dados integrados (esquema global) (ATTALURI *et al.*, 1995, BOUGHETTAYA *et al.*, 1999). *Multidatabase* é uma solução para integração de fontes de dados implementadas por SGBDs, com modelos de dados diferentes (por exemplo, orientado a objeto e relacional), onde a definição do esquema global é relaxada, podendo ter esquemas sobre visões dos dados das fontes. Um *multidabase* define uma linguagem única para o acesso aos dados (EVRENDILEK *et al.*, 1997).

Já os *middlewares* de consulta são unidades de *software* que permitem o acesso a fontes de dados distribuídas. Este acesso é padronizado, simplificando o desenvolvimento das aplicações. Entretanto, ao contrário de um *multidatabase*, os *middlewares* de consulta não realizam nenhuma integração relativa à definição de visões unificadas (esquemas globais) sobre as fontes de dados. Estas arquitetura se preocupam em disponibilizar uma linguagem de consulta única sobre as fontes de dados, delegando a responsabilidade de integração de informação para as aplicações que acessam os dados (SIMON, 2001, OBJECTDRIVER, 2002).

Uma arquitetura de mediadores define os componentes de *software* e seus papéis necessários para a integração de fontes de dados distribuídas, heterogêneas e autônomas, tendo como aspectos principais o alto grau de distribuição e heterogeneidade dos dados, bem como a definição do esquema integrador (não necessariamente global ou completo) dos dados (PIRES, 1997). Para esta arquitetura, o

esquema integrador consiste na interseção de esquemas locais das fontes de dados. Nas arquiteturas de mediadores, além do processamento de consultas, também são permitidas operações de atualizações sobre a fontes de dados, desde que os dados manipulados tenham visibilidade no esquema integrador. Já os sistemas de consulta mediados (*Mediated Query Systems* – MQS) adotam a especificação de *software* das arquiteturas de mediadores e definem em detalhes os requisitos para a realização de consultas em um contexto de integração de informação (DITTRICH e DOMENIG, 1999). A seguir, detalharemos cada uma dessas arquiteturas para integração de informação.

2.3.1 Bancos de Dados Federados

Um dos principais objetivos de uma federação de bancos de dados é permitir que um esquema único e global de integração seja definido sobre as fontes de dados participantes (GARDARIN *et al.*, 1996, SHETH e LARSON, 1990). Regras de atualização e apresentação dos dados são especificadas para as fontes locais, de forma a permitir um esquema único para toda a federação. A integração é determinada pelas necessidades dos usuários da federação e pela tolerância das fontes locais às regras da federação. Uma federação de bancos de dados típica define um modelo de dados comum e uma linguagem interna de comandos.

Os bancos de dados federados possuem as seguintes unidades de *software* (ELMAGARMID *et al.*, 1999): um processador de transformação, um processador de filtro e um processador de construção, vide Figura 1. O processador de transformação é responsável pela tradução do esquema local de uma fonte de dados para o modelo de dados comum à federação, sendo também responsável pela tradução da linguagem de comandos interna da federação para a linguagem de consulta local da fonte de dados.

O processador de filtragem é quem limita o conjunto de operações que podem ser submetidas a uma fonte de dados da federação, consistindo em uma unidade de controle de acesso à fonte. Esse módulo também verifica se as consultas globais obedecem às regras de integração semântica definidas pela federação.

Por fim, o processador de construção detém todo o conhecimento sobre o esquema global da federação. É esta unidade que realiza a decomposição de uma consulta global

em consultas locais. Nesta unidade é mantido um dicionário com os esquemas de exportação das fontes locais e o esquema global da federação.

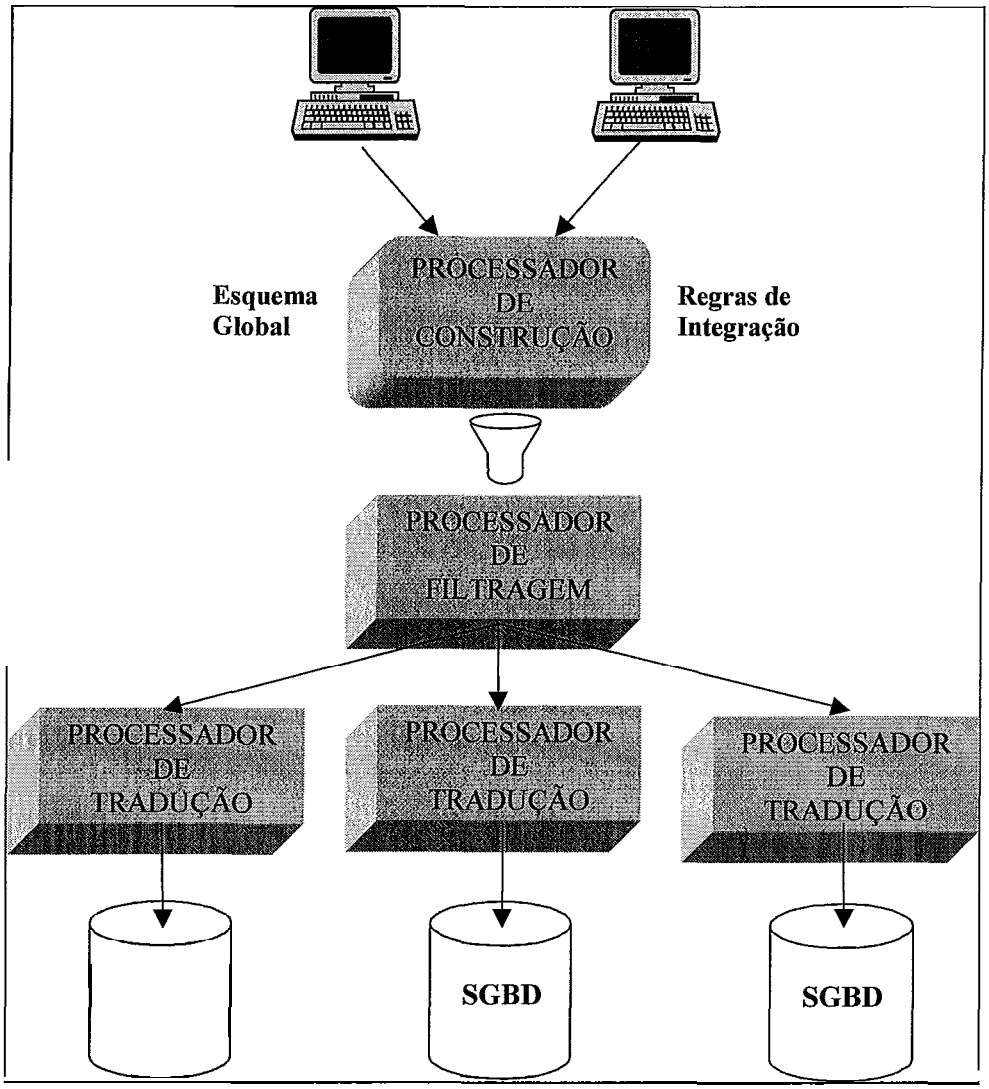

Figura 1. Bancos de dados federados

2.3.2 ARQUITETURA *MULTIDATABASE*

Multidatabases representam arquiteturas distribuídas que atendem a um ambiente de integração de informação cujas fontes de dados possuem autonomia (BOUGHETTAYA *et al.*, 1999). Em uma arquitetura *multidatabase* podem existir diversos graus de detalhamento do esquema global. Variar o detalhamento de um esquema global implica em definir o quão relaxada é a definição deste esquema nas fontes de dados da arquitetura *multidatabase*. Com o grau de detalhamento máximo, temos um *multidatabase* cujo esquema global compreende todos os objetos pertencentes

a cada fonte de dados da arquitetura. Sob esta ótica, um *multidatabase* se confunde com uma federação de bancos de dados. Por outro lado, podemos projetar um *multidatabase* que não possui um esquema global, onde as consultas são simplesmente passadas para as fontes de dados e as repostas das fontes são unidas, compondo a resposta à consulta global.

Obviamente, o processo de tradução para um modelo canônico em um *multidatabase* só é necessário se as fontes de dados forem heterogêneas e se cada esquema local (das fontes de dados) possuir um modelo próprio. O processo de tradução pode ser postergado para permitir uma maior flexibilidade (EVRENDILEK *et al.*, 1997). É o caso, por exemplo, da integração de fontes de dados relacionais e fontes orientadas a objeto. As fontes de dados relacionais são integradas, definindo um esquema de dados relacional, que posteriormente é integrada a um esquema orientado a objetos. O esquema relacional é enxergado pelo esquema orientado a objetos através de um tradutor como se fosse uma única fonte de dados.

A metodologia de desenvolvimento de uma arquitetura de *multidatabase* varia conforme as necessidades e visões do implementador. Não existe uma orientação precisa de como construir os diversos módulos da arquitetura (ELMAGARMID *et al.*, 1999). Vale ressaltar também que, como a autonomia é a principal característica desta arquitetura, cada banco de dados componente de um *multidatabase* desconhece os aspectos globais da arquitetura e pode não ter acesso aos demais bancos de dados participantes.

2.3.3 *MIDDLEWARE* DE CONSULTA

Um *middleware* de consulta é composto por módulos que facilitam a publicação de dados e serviços em um ambiente distribuído e heterogêneo, sem a definição de um esquema global (SIMON, 2001). Entre uma aplicação usuária e as fontes de dados é definida uma camada que permite realizar consultas de integração de informação, mas não são providos mecanismos para garantir transparência quanto à distribuição física das fontes. Esta camada de *software* possui dois componentes básicos: o servidor do *middleware*, que é associado a cada fonte de dados; e o cliente do *middleware*, que permite à aplicação usuária acessar as fontes de dados.

O servidor do *middleware* realiza a tradução do modelo de dados de uma determinada fonte para um modelo de dados comum. O cliente do *middleware* disponibiliza o acesso às fontes de dados para as aplicações, o qual pode ser realizado através de diferentes protocolos de comunicação (*socket*, http, ftp, JDBC, etc.). Estes dois componentes de um *middleware* de consulta oferecem um método comum de acesso às diversas fontes de dados, permitindo que as aplicações possam construir uma visão unificada e integrada dos recursos (dados e programas) disponíveis nessas fontes.

Em um *middleware* de consulta, os "donos" das fontes de dados podem publicar seletivamente seus recursos para diferentes grupos de usuários, ditos comunidades do *middleware*. Ou seja, o dono de uma fonte de dados que desejar participar de uma comunidade deve instalar o servidor do *middleware* e definir exatamente quais recursos serão compartilhados. Mesmo com a instalação de um servidor de *middleware*, não está garantida a participação da fonte de dados em uma comunidade do *middleware* de consulta, pois fica a critério da fonte a publicação de seus recursos.

A concepção de um *middleware* de consulta não inclui um serviço para o registro de quais fontes de dados fazem parte de uma determinada comunidade. Em um *middleware* de consulta não existe a definição de um esquema global, sendo cada fonte responsável pela publicação de um esquema para os dados que disponibiliza. Também não existe uma unidade central de integração, como na arquitetura de mediadores. De forma geral, uma aplicação usuária acessa uma fonte de dados através de uma API (*Application Program Interface*) que permite a consulta ao esquema publicado pela fonte. Um *middleware* de consulta não prevê tratamento para a integração do resultado de uma consulta sobre diferentes fontes de dados.

2.3.4 ARQUITETURA DE MEDIADORES

A arquitetura de mediadores é baseada na utilização de um conjunto de módulos de *software* que realizam a mediação entre as aplicações dos usuários e as fontes de dados. Estes módulos mediadores formam uma camada de *software* distinta que tornam as aplicações independentes das fontes de dados (WIEDERHOLD,1992), conforme é apresentado na Figura 2.

Na Figura 2 podemos observar a camada de mediação que separa o processamento orientado ao usuário do acesso às fontes de dados. Eventualmente, as aplicações dos

usuários necessitarão de vários mediadores para acessar as fontes de dados, de acordo com os domínios de informação.

Esta arquitetura de integração é composta por **mediadores**, responsáveis pela integração semântica das consultas, e por **tradutores** para as fontes de dados. Cada mediador acessa um ou mais tradutores, os quais acessam diretamente as fontes de dados. As interfaces dos tradutores e dos mediadores definem os pontos em que o serviço de comunicação é necessário (ou seja, entre um tradutor e um mediador e entre um mediador e uma aplicação do usuário). Este serviço de comunicação pode ser implementado por protocolos proprietários ou utilizando protocolos abertos como o HTTP ou IIOP do CORBA.

A camada de mediação provê a separação entre a aplicação e as fontes de dados, conferindo assim uma maior independência da funcionalidade das aplicações em relação às possíveis reorganizações e redistribuições das estruturas físicas de dados sobre os nós da rede. A arquitetura em três camadas da Figura 2 realiza uma troca explícita para favorecer a flexibilidade, permitindo que uma grande variedade de aplicações possam manipular dados em fontes distribuídas.

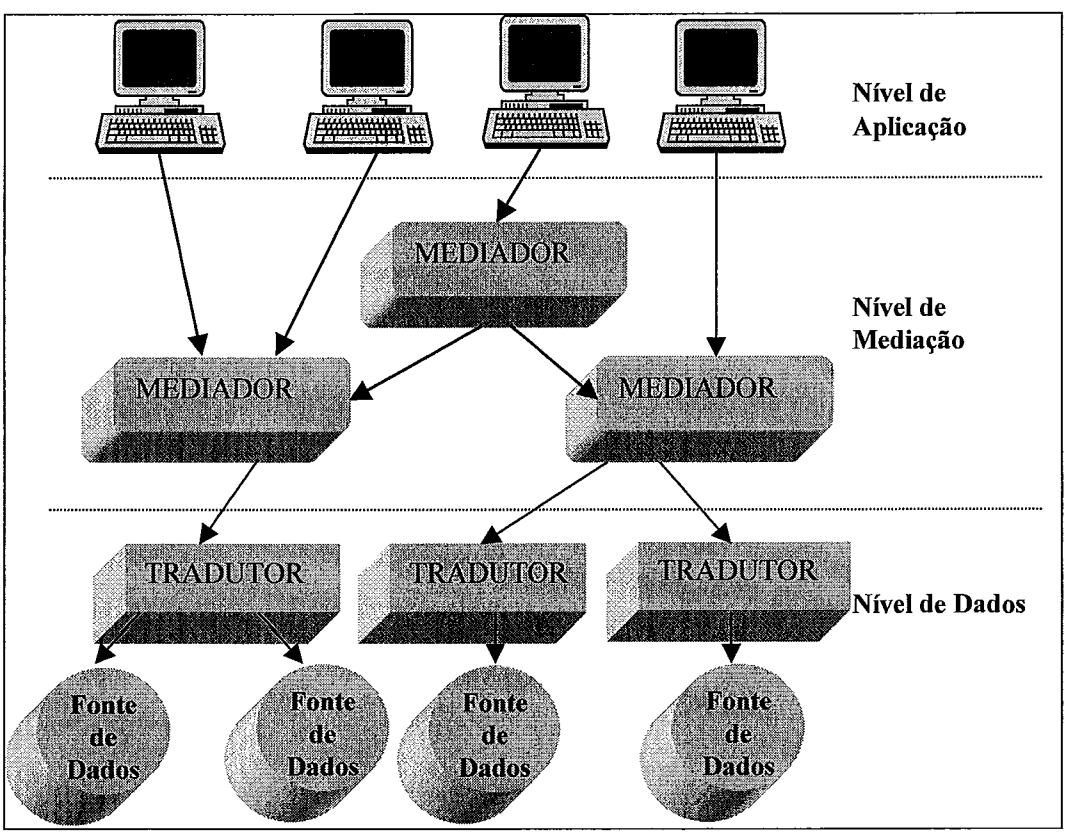

Figura 2. Arquitetura de Mediadores

2.3.4.1 Os Mediadores

Um mediador explora o conhecimento codificado sobre algum conjunto ou subconjunto de dados para uma camada de aplicação em um nível de abstração mais alto (WIDERHOLD, 1992). Alguns requisitos são desejáveis na construção de um mediador, destacando-o das outras arquiteturas de integração. Ele deve ser pequeno e simples. Além disso, é desejável que ele permita a inspeção pelos usuários, isto é, que as regras utilizadas pelo mediador estejam disponíveis para um usuário especialista ou para um outro mediador. A possibilidade de inspeção no mediador permite que os usuários possam escolher entre os diversos mediadores disponíveis em um dado momento, um diferencial importante desta arquitetura. Podem existir, também, meta-mediadores, contendo uma lista dos mediadores disponíveis e a descrição dos recursos que estes disponibilizam.

Uma das principais características de um mediador é a facilidade de acesso, conseguida através de uma interface bem definida entre as aplicações e o mediador, com conceitos genéricos, flexíveis e que permitem composições e interações (ELMAGARMID *et al.*, 1999). Uma interface de especificação estática não é capaz de lidar com a variedade de controles e fluxos de informação que são necessários nas interfaces entre mediadores e aplicações. Já as linguagens de programação e as linguagens declarativas de consulta são genéricas e atendem satisfatoriamente aos requisitos de flexibilidade, pois possuem um amplo poder de expressão semântica.

Vale ressaltar que a interface de acesso a um mediador não é direta com o usuário, mas, ao invés disso, é orientada à comunicação entre programas. Os programas de aplicação, executando nas estações dos usuários, devem prover a entrada de comandos e a exibição de resultados adequada para o usuário.

2.3.4.2 Os Tradutores

Os tradutores são as unidades de *software* que realizam a comunicação entre os mediadores e as diferentes fontes de dados. Eles traduzem um determinado modelo de dados local para a visão "global" definida pela arquitetura. Para permitir a interação entre os mediadores e os tradutores, normalmente são utilizadas as linguagens de consulta típicas de banco de dados (DITTRICH e DOMENIG, 1999), como a SQL e a OQL. Além da linguagem de consulta, o modelo de dados de um SGBD (seja o conceito

relacional de tabela ou os conceitos orientados a objetos de classe e do próprio objeto, por exemplo) também fornece uma boa referência para a definição da interface dos tradutores com os mediadores.

Uma interface simplificada entre o mediador e o tradutor pode ocorrer quando as fontes de dados utilizam um mesmo modelo de dados e uma mesma linguagem de consulta. Porém, este caso costuma ser uma exceção (ÖZSU e VALDURIEZ, 1999). As fontes de dados mais comuns são advindas de sistemas corporativos legados, SGBDs, páginas *Web* e planilhas (VICTORINO, 2001). Por isso, os tradutores são construídos para realizar a adequação dos diferentes modelos de dados e formas de acesso às fontes de dados em relação ao modelo de dados e à linguagem do mediador.

2.3.5 Sistemas de Consulta Mediados

Os sistemas de consulta mediados (MQSs – *Mediated Query Systems*) constituem um caso particular da arquitetura de mediadores, com funcionalidades específicas para o processamento de consultas. Os MQSs são sistemas que consideram fontes de dados heterogêneas, as quais disponibilizam uma interface global para consultas. O retorno das consultas possui uma apresentação unificada e sua implementação é realizada por mediadores (WIEDERHOLD,1992).

DITTRICH e DOMENIG (1999) propõem uma classificação para MQSs considerando características que vão além das definições apresentadas para as arquiteturas de mediadores. Esta classificação aponta os principais aspectos dos MQSs e como cada aspecto do sistema pode ser abordado em relação a questões sobre distribuição, autonomia e heterogeneidade. A avaliação dos MQSs é realizada conforme duas dimensões: as características funcionais e as características de implementação. A primeira dimensão representa como um MQS define sua interface para as aplicações. Já as características de implementação tratam de como é construída esta interface.

2.3.5.1 Características Funcionais

São quatro as características funcionais de um MQS: i) as características de consulta; ii) a apresentação dos resultados; iii) as propriedades das estruturas de dados; e iv) a extensibilidade.

As características de consulta incluem os aspectos da linguagem de consulta, do tipo de consultas realizadas pelo sistema e do grau de dependência das consultas em relação à definição do esquema global. A linguagem de consulta de um MQS pode ser típica de bancos de dados, onde o usuário recupera informação baseado em nomes de atributos através de uma descrição declarativa da consulta, ou como nas ferramentas de busca da *Web*, onde o usuário utiliza sentenças formadas por palavras-chaves e conectivos lógicos. Quanto ao tipo, uma consulta pode ser **exata**, quando a estrutura das fontes de dados é previamente conhecida, ou **vaga**, caso contrário. A dependência do esquema define o grau de conhecimento prévio sobre as fontes de dados que o usuário precisa ter para descrever uma consulta. Em sistemas com um esquema global, existe um mapeamento definido entre a informação disponível no mediador e a informação armazenada nas fontes de dados. Assim, o usuário não precisa conhecer detalhes sobre os esquemas locais das fontes de dados. Por outro lado, quando não existe um esquema global, é necessário que a consulta expresse a estrutura de cada uma das fontes a serem acessadas, considerando os eventuais conflitos.

A apresentação dos resultados está diretamente ligada ao à forma como um MQS constrói o resultado das consultas realizadas. Caso o MQS possua esquema global as coleções resultantes das subconsultas nas fontes de dados são agrupadas, eventuais integrações dos dados são realizadas (ex. ajustes dos tipos) e por fim as redundâncias dos dados são tratadas. No caso dos MQSs sem um esquema global de dados, duas formas podem ser aplicadas: uma lista de classificação por relevância ou o retorno incremental. O primeiro caso monta uma lista de possíveis resultados para cada consulta, os quais são ordenados pela relevância da informação retornada em relação à consulta especificada. No segundo caso, é permitido ao usuário melhorar os resultados de uma consulta através da inclusão de critérios adicionais sobre a informação pesquisada.

Quanto às propriedades das estruturas de dados, os MQSs podem ter dados estruturados, dados não-estruturados e dados semi-estruturados. O primeiro tipo descreve os dados que aderem a um esquema bem definido. O segundo tipo não define nenhuma composição determinada além de seqüências de *bytes* ou cadeias de caracteres. O terceiro e último tipo descreve dados que possuem um estrutura, mas esta não é rígida, podendo estar incompleta. Por exemplo, em uma determinada fonte de dados, a informação endereço pode ter as seguintes representações: `struct(string`

`rua, short numero, short cep, string localizacao)` ou **`struct`**`(string nome_da_casa, short cep, string localizacao).`

Por fim, a extensibilidade é uma característica altamente desejável em um MQS, tal que novas fontes de dados possam ser facilmente acrescentadas à arquitetura, bem como fontes existentes possam ser desligadas sem comprometer o sistema. Os MQSs podem ser avaliados quanto ao grau de extensibilidade oferecido.

2.3.5.2 CARACTERÍSTICAS DE IMPLEMENTAÇÃO

As implementações das interfaces de comunicação e das características funcionais de um MQS dependem do projeto do sistema. Os principais aspectos de implementação são: i) a arquitetura do MQS; ii) a representação interna dos dados; iii) o processamento de consultas; e iv) os metadados.

Os sistemas de consulta baseados em mediadores possuem uma arquitetura de *software* divida em três camadas (WIEDERHOLD, 1992). Os aspectos como centralização ou descentralização e as funcionalidades definidas para a camada de mediação definem a construção da arquitetura. Com uma arquitetura descentralizada, a mediação é realizada por uma série de componentes da rede, permitindo uma fácil extensão do MQS. Com um sistema centralizado, a vantagem está na simplificação da comunicação entre os módulos do MQS e do processo de gerência dos componentes.

As funcionalidades da camada de mediação ressaltam a questão da divisão das atividades entre os tradutores e os mediadores. Podemos ter tradutores mais sofisticados que realizam sobre as fontes de dados todas as funcionalidades específicas e adaptações necessárias para a integração do modelo global. A vantagem de um tradutor mais sofisticado consiste em uma maior velocidade do processamento de consultas na camada de mediação. Por outro lado, uma arquitetura, com tradutores, mais simples facilita a inclusão de novas fontes de dados na mesma.

A representação interna dos dados define como os dados e as consultas que podem ser realizadas por um MQS serão representados. Se o MQS é desenvolvido para selecionar apenas registros e relacionamentos **simples**, um modelo de dados com pouco poder de expressão é suficiente. Se o sistema representa dados e relacionamentos **complexos**, um modelo de dados mais rico deve ser escolhido, como por exemplo, o modelo orientado a objetos.

O processo entre a submissão de uma consulta e o envio desta para a camada das fontes de dados é implementado pelo componente de processamento de consultas de um MQS. Os fatores que definem como este componente será construído consideram que tipos de consultas serão suportados e quais fontes podem estar envolvidas em uma consulta, entre outras questões. Duas são as características que se destacam na definição do processamento de consultas em um MQS: a seleção da fonte de dados e a seqüência de divisão, otimização e interpretação da consulta. O processo de seleção da fonte de dados está relacionado aos tipos de dados armazenados nas fontes. O processador de consulta global de um MQS, ao receber uma consulta, seleciona quais as possíveis fontes para uma dada consulta. As fontes com dados semi-estruturados e não-estruturados podem ser selecionadas através das suas descrições. Na seleção das fontes que irão compor a resposta à consulta global também é considerada a disponibilidade da fonte, além do seu desempenho.

O processo de divisão, otimização e interpretação de uma consulta é o próximo passo após a seleção das fontes de dados. A interpretação de uma consulta define quais atributos locais correspondem aos atributos globais envolvidos na consulta. Em seguida, na divisão da consulta, as fontes de dados são escolhidas e as operações globais são adequadas para a execução nas fontes locais. Quando o MQS suporta buscas exatas e vagas, é na divisão da consulta que estes aspectos são tratados. Na fase de otimização das consultas, a capacidade de processamento das fontes de dados precisa ser avaliada (HAAS *et al.*, 1999), o que define a ordem de realização das operações e a eficiência do processamento global da consulta.

Os componentes responsáveis pelo processamento de consultas em um MQS requerem um amplo conhecimento sobre as fontes de dados disponíveis. Grande parte da meta-informações tem que ser coletadas e armazenadas em repositórios de metadados. Portanto, é interessante que sejam avaliadas características sobre ontologias[2] e sobre o conteúdo dos metadados, bem como sobre a aquisição destes. O conteúdo dos metadados depende dos requisitos do sistema e das necessidades internas do MQS (dados sobre as fontes de dados, estatísticas, etc.). A aquisição dos metadados pode ser

[2] Uma ontologia é "a categorização de um conceito" (BRAGA, 2000). Em particular, está relacionada ao problema que informações similares podem ser representadas utilizando vocábulos diferentes (por exemplo, "palestra" e "seminário" possuem conceitos semelhantes).

responsabilidade dos tradutores ou de outros componentes específicos. Quando ocorre a utilização de componentes específicos, uma linguagem de especificação dos metadados é necessária, tornando a definição de metadados em uma tarefa a ser realizada pelas fontes de dados (TOMASIC *et al.*, 1995). A evolução das fontes de dados está relacionada com a aquisição dos metadados. Ou seja, quando o esquema de uma fonte de dados é alterado, esta informação deve ser refletida no repositório de metadados.

2.4 ARQUITETURA DE REFERÊNCIA

A definição dos aspectos mais relevantes para o processamento de consultas em fontes de dados heterogêneas, distribuídas e autônomas faz com que os MQSs sejam uma arquitetura de referência. As arquiteturas apresentadas anteriormente podem ser enxergadas como diferentes especificações de MQSs, variando algum aspecto funcional ou de implementação. Por exemplo, um *middleware* de consulta pode ser considerado um MQS com o relaxamento da implementação do mediador. Por isso, utilizaremos a arquitetura de MQSs como cenário para integração de informação no decorrer do texto.

Vale ressaltar que todas as arquiteturas de integração apresentadas neste capítulo assumem que estão disponíveis informações para apoiar o processamento de consultas nas suas fontes de dados. Entretanto, a aquisição destas informações não é trivial. Após a compreensão do cenário em que o DIG está inserido, é necessário detalharmos aspectos sobre como é realizada a aquisição de dados para o processamento e otimização de consultas nas arquiteturas de integração de informação. , em particular nos MQSs – nossa arquitetura de referência.

No próximo capítulo, nós abordaremos as características e os principais desafios para o provimento de estatísticas, custos e demais informações necessárias para o processamento eficiente de consultas.

Capítulo 3

AQUISIÇÃO DE CUSTOS NO PROCESSAMENTO DE CONSULTAS EM MQSs

Métricas de desempenho, baseadas em custos e estatísticas sobre as fontes de dados que participam de um sistema mediado de consulta (MQS), são fundamentais para a geração de planos eficientes para a execução de consultas globais. Neste capítulo são abordadas as características e os desafios da aquisição de parâmetros de custos e estatísticas para o processamento distribuído de consultas em MQSs. Ao final, analisamos como esses dados são obtidos nos principais trabalhos sobre o processamento global de consultas, enfatizando o tratamento da heterogeneidade e da autonomia das fontes de dados.

3.1 INTRODUÇÃO

A visão clássica do processamento de consultas abrange basicamente as seguintes etapas (ELMASRI e NAVATHE, 1994, KOSSMANN, 2000): o recebimento de uma consulta, usualmente escrita em uma linguagem declarativa como a SQL ou a OQL, e sua tradução para um formato interno de representação do processador de consultas; a otimização (realizada em diversas etapas) desta consulta e a geração de um plano de execução eficiente; e, finalmente, a execução propriamente dita da consulta, conforme o plano gerado, produzindo a exibição do resultado obtido.

Em um sistema de consulta mediado (MQS – *Mediated Query System*), cabe ao processador de consultas do mediador determinar o melhor plano global de execução para as consultas submetidas. Esse plano de execução é montado a partir da decomposição da consulta original em subconsultas que são enviadas para os tradutores relacionados. A geração de um plano global eficiente é uma tarefa extremamente complexa, pois um grande número de fatores deve ser considerado. Portanto, é essencial que métricas de desempenho, baseadas em custos e estatísticas sobre as fontes de dados, sejam utilizadas para orientar o processo de otimização global (ROTH *et al.*, 1999).

Em um ambiente distribuído, o modelo clássico de processamento da consulta deve se ajustar aos requisitos da distribuição. Assim, o processamento de consultas em um ambiente distribuído é dividido em seis etapas principais (ÖZSU e VALDURIEZ, 1999): (i) decomposição da consulta; (ii) localização dos dados; (iii) otimização global da consulta, com a geração de um plano global de execução contendo subconsultas a serem enviadas às fontes de dados; (iv) otimização local das subconsultas nas fontes de dados correspondentes, gerando planos locais de execução; (v) execução local das subconsultas; e (vi) construção do resultado global. Nas três etapas iniciais, o processador global de consultas requer informações sobre as fontes de dados para poder gerar planos globais de execução.

Na etapa de otimização global de consultas, é fundamental o conhecimento sobre as fontes de dados para permitir a geração de planos globais de execução eficiente. Basicamente, a otimização de uma consulta pode ser heurística ou baseada em custos (ELMASRI e NAVATHE, 1994). Em um ambiente heterogêneo, onde o bom

desempenho depende de uma combinação complexa de fatores, as abordagens heurísticas podem induzir o processo de otimização a erros drásticos. Por outro lado, nesse tipo de ambiente, a otimização de consultas baseada em custos geralmente garante a geração de um plano de consulta "ótimo" (HAAS *et al.*, 1997).

Diferentemente de uma arquitetura tradicional de banco de dados, nem sempre as fontes de dados de um MQS estão associadas a SGBDs. Esse fato possui duas implicações relevantes: (i) o mediador precisa conhecer a real capacidade de processamento de consultas de cada fonte de dados; e (ii) a aquisição de custos e estatísticas sobre uma fonte de dados desprovida de um SGBD requer métodos especiais para inspeção dos dados e geração das estimativas. Neste contexto, também é importante que o mediador possa extrair parâmetros para a definição de critérios relativos à qualidade de serviço (QoS – *Quality of Service*) oferecida pelos tradutores, como disponibilidade e tempo médio de espera pelo serviço (MANI e NAGARAJAN, 2002).

Muitas das técnicas para processamento e otimização de consultas em sistemas distribuídos homogêneos também podem ser aplicadas em sistemas heterogêneos, como em um MQS, desde que sejam observados alguns aspectos relativos à heterogeneidade e à autonomia das fontes de dados. Podemos afirmar que a heterogeneidade torna o processamento de consultas mais complexo que em sistemas distribuídos tradicionais, pois conforme em SHETH e LARSON (1990):

⇒ As fontes de dados apresentam capacidades de processamento distintas entre si;

⇒ O custo do processamento de consultas é peculiar a cada fonte de dados, aumentando assim a complexidade das funções de custo a serem avaliadas; e

⇒ A capacidade de otimização local difere em cada fonte de dados, sendo em geral restrita.

A característica de autonomia de um MQS garante que as fontes de dados tenham independência para determinar quais funcionalidades (possivelmente apenas um subconjunto das existentes) são publicadas para o processador global de consultas. Além disso, a autonomia também permite à fonte de dados restringir a disponibilidade e a precisão das informações sobre estatísticas e custos necessários à otimização de consultas. Por esta razão, as funções de custo das fontes de dados normalmente não são

conhecidas pelo mediador. A expressão máxima da autonomia de uma fonte de dados é a possibilidade desta encerrar seus serviços sempre que julgar necessário, tornando-se indisponível temporariamente para o processador de consultas (TOMASIC *et al.*, 1995). Por isso, é importante que o processamento distribuído de consultas seja tolerante a essa indisponibilidade, e que possua mecanismos para identificar quando uma fonte de dados está inacessível e não participará do processamento de uma consulta. Por exemplo, para resolver este problema, o tempo médio de espera por um serviço pode ajudar ao mediador decidir quais fontes de dados serão utilizadas em um dado plano de execução da consulta.

Além dos desafios acima, o processamento de consultas enfrenta uma complexidade adicional advinda da arquitetura de um MQS. Em SGBDs distribuídos, os processadores de consulta têm que lidar apenas com a distribuição dos dados entre diversos sítios. Em sistemas de consulta mediados, a informação também está distribuída em diversos sítios, mas um sítio pode conter fontes de dados distintas e heterogêneas. Ou seja, cada tradutor pode acessar mais de uma fonte de dados (DITTRICH e DOMENIG, 1999).

Em geral, as abordagens baseadas em custos para a otimização de consultas enfrentam três problemas: i) a dificuldade de representar os custos do processamento de consultas através de um modelo adequado; ii) a dificuldade de obter valores (estatísticas e custos) para os parâmetros de entrada das funções que compõem o modelo de custo; e iii) a dificuldade na utilização do modelo de custo, isto é, na escolha da estratégia para a composição dos planos de execução a partir das funções de custo.

Neste capítulo são apresentadas as características da otimização de consultas baseada em custos nos MQSs, enfatizando a representação dos custos e o problema da aquisição de parâmetros determinantes para o cálculo destes custos em um ambiente heterogêneo. Finalizando o capítulo, são analisadas as características de trabalhos relevantes sobre a aquisição de custos e estatísticas para a otimização de consultas em ambientes com fontes de dados distribuídos, heterogêneos e autônomos.

3.2 OTIMIZAÇÃO DE CONSULTAS EM MQSS

A autonomia e a distribuição das fontes de dados em um MQS afetam diretamente duas etapas do processamento global de consultas: a decomposição da consulta e a otimização global. Para realizar o processo de decomposição da consulta é necessário que a unidade de mediação possua informações sobre o esquema das fontes. Além disso, é necessário também que o mediador conheça as regras de mapeamento do esquema "global" para os esquemas locais das fontes de dados Essas regras podem estar armazenadas em uma base de metadados que representa a integração semântica dos esquemas locais (BRÜGGER *et al.*, 1999).

É importante observar que a heterogeneidade de modelos de dados pode resultar em perdas de semântica na integração das fontes de dados de um MQS. Por exemplo, caso o modelo de dados do esquema global seja relacional e o modelo de uma das fontes de dados seja orientado a objetos, informações sobre o uso de herança serão perdidas.

O processo de otimização global de uma consulta é ainda mais sensível às particularidades de um MQS do que a etapa de decomposição, pois as mudanças nas informações de custos e estatísticas são mais dinâmicas do que as mudanças dos esquemas das fontes de dados. Além disso, o plano de execução de uma consulta deve refletir a capacidade de processamento das fontes envolvidas. Por exemplo, fontes de dados que representam SGBDs hierárquicos não possuem a capacidade de realizar operações de agregação de valores. Portanto, cabe à unidade de mediação realizar esta tarefa (GARDARIN *et al.*, 1996).

O processo de otimização de consultas em um MQS pode ser realizado através de heurísticas ou da aplicação de custos e estatísticas em um modelo analítico para orientar a escolha dos melhores planos de execução de cada consulta. O uso de heurísticas simplifica o mediador, já que não são necessárias informações detalhadas sobre as fontes de dados. Por outro lado, a utilização de modelos de custos permite a geração de planos de execução de consulta mais eficientes quando comparados aos gerados pela aplicação de heurísticas (HAAS *et al.*, 1997). Vale ressaltar, todavia, que a abordagem heurística pode ser utilizada como um complemento à otimização baseada em custos, visando restringir o espaço de busca (freqüentemente vasto) para a aplicação de funções de custo. Observe que o uso de heurísticas pode diminuir o número de possíveis

soluções, mas geralmente não restringe a busca a um único plano de execução. Heurísticas podem guiar, por exemplo, a definição do tipo de decomposição que será aplicada nas consultas globais.

3.2.1 OTIMIZAÇÃO BASEADA EM HEURÍSTICAS

Basicamente, existem duas alternativas para a geração heurística do plano de execução de uma consulta em MQSs (HAAS *et al.*, 1997, KOSSMANN, 2000). A primeira alternativa consiste em decompor a consulta global na "menor" subconsulta possível, onde cada subconsulta é executada por uma determinada fonte de dados. Nesse caso, uma fonte de dados pode ser responsável pela execução de várias subconsultas geradas. O processador de consulta global se encarrega de coletar os resultados parciais e montar o resultado final para o usuário. Esta abordagem possui como vantagens a simplicidade na decomposição da consultas e o maior número de possibilidades para explorar a otimização global. Entretanto, como desvantagem, é o processador global da consulta que realiza a maior parte do trabalho. Além disso, esta abordagem demanda uma grande quantidade de troca de mensagens entre o mediador e os tradutores (e, eventualmente, entre o tradutor e a fonte de dados).

A segunda alternativa para a geração heurística do plano global de execução de uma consulta é baseada na decomposição da consulta global visando a "maior" subconsulta possível, de acordo com a capacidade individual de processamento das fontes de dados envolvidas. Cada fonte de dados executa apenas uma subconsulta e os resultados parciais são coletados pelo processador global. Nesse caso, o processador global tem menos trabalho a realizar e é reduzido o número de mensagens entre os componentes da arquitetura. Porém, são necessários tradutores mais sofisticados para que realizem as tarefas de otimização/processamento das subconsultas recebidas. GARDARIN *et al.* (1997) mostram a necessidade de incorporar funcionalidades adicionais aos tradutores, de tal forma que informações sobre as capacidades de processamento de consulta das fontes de dados sejam repassadas ao otimizador global.

Basear apenas em heurísticas a geração de planos de execução para consultas pode levar a planos ineficientes. É comprovado que a otimização de consultas utilizando modelos de custos produz melhores resultados (ROTH *et al.*, 1999, ZHU *et al.*, 2000,

GARDARIN *et al.*, 1997), enfatizando a aplicação de heurísticas à definição do tipo de decomposição que irá ser aplicado na consulta.

3.2.2 OTIMIZAÇÃO BASEADA EM CUSTOS

Uma área de pesquisa que tem sido alvo de grande interesse é a que trata da definição de modelos de custo para o processamento distribuído de consultas (RUBERG, 2001) e do problema associado à obtenção de informações sobre custos das fontes de dados (ROTH *et al.*, 1999, ZHU *et al.*, 2000). Vale ressaltar que, nesse contexto, o otimizador global de consultas não é capaz de estimar o custo dos planos de execução sem a cooperação dos tradutores que acessam as fontes de dados.

As estratégias para estimativa de custo de um plano global de execução consistem em avaliar primeiramente o custo das subconsultas que serão submetidas às fontes de dados. Em seguida, determina-se os custos da composição dessas subconsultas até que o plano global de execução tenha sido totalmente avaliado. Ao final, é obtida uma estimativa do custo total para a execução da consulta global.

Para determinar o custo das subconsultas executadas nas fontes de dados, ZHU e LARSON (1998) definem como abordagens básicas em um ambiente distribuído e heterogêneo:

1) Usar conhecimento previamente informado sobre as fontes de dados, bem como suas características externas, para determinar subjetivamente os custos associados;

2) Tratar as fontes de dados como uma "caixa-preta", executando algumas consultas de teste de desempenho sobre elas e, a partir da aplicação de técnicas matemáticas especiais sobre os resultados obtidos, determinar as informações sobre custo;

3) Monitorar o comportamento da execução de consultas nas fontes de dados e, dinamicamente, coletar as informações de custo.

Essas abordagens podem ser combinadas em uma abordagem híbrida mais flexível. Deste modo, cada fonte de dados pode receber um tratamento mais adequado à sua capacidade de processamento de consultas.

3.3 CUSTOS NA OTIMIZAÇÃO DE CONSULTAS EM AMBIENTES DISTRIBUÍDOS E HETEROGÊNEOS

Os sistemas de bancos de dados costumam representar seus planos de execução da consulta através de árvores (ELMASRI e NAVATHE, 1994). Os nós de um plano de execução representam operadores de consulta, onde cada operador é responsável por uma operação em particular (ex. *join*, *group-by*, *sort*, *scan*, etc.). As arestas do plano representam a relação consumidor-produtor entre os operadores (KOSSMANN, 2000).

Deste modo, uma alternativa para compor o custo total do plano de execução de uma consulta é considerar cada nó da árvore de consulta como uma unidade que recebe fluxos de dados de entrada e produz na saída, normalmente, um outro fluxo de dados. Cada nó da árvore passa a ter um custo associado, que corresponde ao custo do processamento a ser realizado nas entradas da consulta para produzir a saída (vide Figura 3). Estes nós são chamados de operadores do plano ou POPs (*Plan Operators*), os quais podem ser classificados em função da operação que realizam, tal que podemos ter operadores de junção, ordenação, seleção, busca, varredura e criação de coleções temporárias.

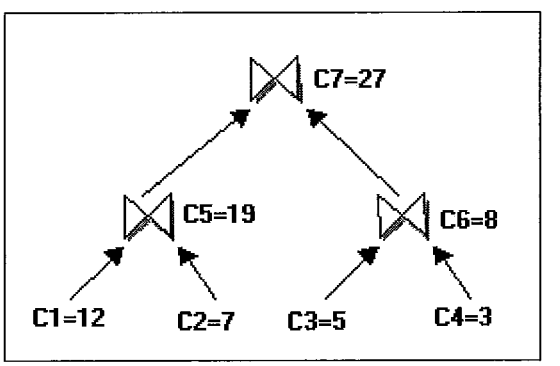

Figura 3. Custo dos operadores em um plano de execução da consulta

Portanto, o custo do plano de execução de uma consulta consiste no custo acumulado dos operadores neste plano. Como todo operador em um plano de execução é raiz de um subplano próprio, o custo total de um operador inclui o custo dos seus operadores de entrada (nós filhos), sendo representado pelo custo do operador raiz deste plano. Assim como a cardinalidade resultante de um operador do plano é derivada da cardinalidade de seus nós filhos, a cardinalidade resultante do operador raiz de um plano de execução representa a cardinalidade do resultado da consulta.

Para calcular os custos acumulados e as cardinalidades de um plano global de consulta, três estimativas básicas podem ser obtidas para cada operador do plano de execução (HAAS *et al.*, 1997):

⇒ o custo total, em tempo medido para a obtenção do resultado na primeira execução do POP;

⇒ o custo de re-execução, relativo ao custo total de uma execução sucessiva do POP; e

⇒ a cardinalidade do resultado do POP.

A diferença entre o custo total e o de re-execução corresponde ao custo das inicializações que podem ocorrer na primeira vez que um operador é executado. Por exemplo, o custo total de um POP para varrer uma coleção temporária deve incluir o custo de popular e varrer a coleção, mas o seu custo de re-execução inclui apenas o custo da varredura.

O custo total, o custo de re-execução e a cardinalidade de um POP são computados usando fórmulas de custo que modelam o comportamento de execução dos operadores de consulta. É interessante que as fórmulas de custo, tais como uso de CPU e E/S (e, em alguns sistemas, mensagens), tenham a maior precisão possível. Todavia, a aquisição de custos e estatísticas não está restrita a apenas esses três parâmetros.

O otimizador de consultas de um MQS precisa estimar o custo dos planos globais de execução das consultas considerando o custo das operações a serem realizadas nas diversas fontes de dados. É necessário, portanto, estimar o custo dos nós do plano de execução que serão aplicados às fontes de dados. Para esta estimativa, duas questões devem ser respondidas:

⇒ que informações são necessárias para compor estes custos; e

⇒ como estas informações podem ser obtidas.

A Figura 4 apresenta o fluxo de informação no processamento de consultas desde as estatísticas básicas até o custos dos POPs em um ambiente tradicional, centralizado. Podemos observar que este modelo é incompleto para o ambiente de um MQS, pois não é considerada a diversidade das fontes de dados envolvidas na consulta. Um problema em MQSs é a impossibilidade de compor as fórmulas de custos do otimizador, pois grande parte das informações sobre as fontes de dados não estão disponíveis no

mediador. Assim, para o otimizador de consultas poder calcular o custo de um plano de execução de forma precisa é necessária a cooperação dos tradutores. Vamos analisar como os papéis apresentados na Figura 4 são alterados para contemplar as características de um MQS.

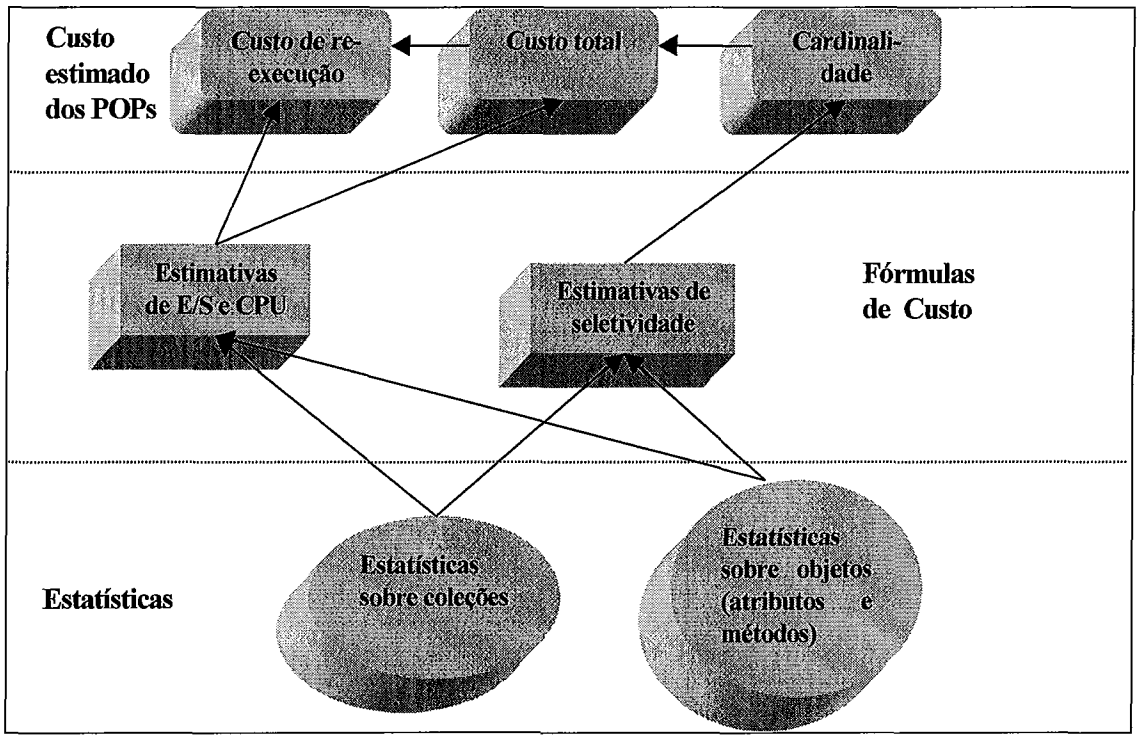

Figura 4. Fluxo de informações na otimização tradicional, baseada em custos

O modelo de custo do GARLIC (HAAS *et al.*, 1997) será utilizado para exemplificar como os custos das fontes de dados compõem o custo de um plano de execução.

O primeiro desafio para um otimizador em um MQS é a integração dos custos do processamento realizado pelas fontes de dados (NAACKE *et al.*, 1998). ROTH *et al.* (1999) apresentam uma forma bastante elegante para integrar os custos das fontes de dados na otimização de consultas. Um operador (POP), definido como `PUSHDOWN`, encapsula as partes do plano de execução de uma consulta que serão realizadas nas fontes de dados. Ou seja, estes operadores representam as folhas do plano de execução da consulta. Como resultado desta visão, o custo total, o custo de re-execução e a cardinalidade resultante constituem informações suficientes para integrar o custo de um operador `PUSHDOWN` no custo total do plano de execução de uma consulta.

A conseqüência da utilização deste modelo é que a avaliação do plano de execução de uma consulta pelo otimizador não precisa ser modificada, basta apenas integrar o custo dos POPs do tipo `PUSHDOWN` no custo global do plano de consulta. Uma outra vantagem dessa abordagem é que as fórmulas de custo global não precisam saber nada sobre a estratégia de execução local das subconsultas a serem realizadas pelas fontes de dados.

O problema passa a ser a determinação do custo do operador `PUSHDOWN`. Duas considerações podem ser feitas a esse respeito. Primeiro, o conjunto de estatísticas e fórmulas para calcular o custo do `PUSHDOWN` é determinado apenas na fonte de dados em que o operador é aplicado. Além disso, o cálculo do `PUSHDOWN` pode ser feito no tradutor ou no mediador.

É importante observar que o operador `PUSHDOWN` é uma representação para o mediador da porção da consulta submetida a uma fonte de dados. Esta consulta pode ser qualquer operação que a fonte de dados possa realizar (por exemplo, uma seleção, uma junção, um método, etc.). Portanto, é necessário que o cálculo das fórmulas de custo seja realizado pelas fontes de dados, ou seja, por seus respectivos tradutores.

Para cada fonte de dados, o tradutor deve conhecer as fórmulas de custos que permitem estimar o desempenho dos operadores de consulta. Estas fórmulas devem ser personalizadas de acordo com o modelo de execução de consultas das fontes de dados. Para fontes de dados em SGBDs relacionais, ZHU *et al.* (2000) apresentam uma avaliação dos modelos de execução mais comuns. A partir destes modelos são elaboradas fórmulas de custo para os diversos operadores relacionais. Através destas fórmulas, os tradutores das fontes de dados com SGBDs relacionais podem estimar o custo de um operador de consulta com pouca informação sobre os dados da fonte.

Caso não se possa definir previamente o modelo de execução de consultas de uma fonte de dados, pode-se inferir o comportamento dos algoritmos executados nessa fonte a partir de algoritmos típicos correspondentes às operações realizadas na fonte. Por exemplo, pode-se assumir que a execução de um operador de seleção em uma fonte de dados é obtida pela varredura seqüencial não indexada de uma coleção. Entretanto, à medida que são necessários custos e estatísticas mais precisos, deve-se conhecer mais detalhes sobre a fonte de dados. Por exemplo, se um operador de seleção é aplicado em

uma coleção da fonte, a presença de índices nessa coleção implicará no uso de uma fórmula específica de custo.

Além da definição das fórmulas de custo, são necessários os custos de recuperação das coleções, dos objetos, dos atributos e os custos de execução de métodos nas fontes de dados. Os custos destas recuperações devem ser disponibilizados pelo tradutor. Os custos de execução de métodos podem ter sua avaliação simplificada, considerando que um método possui o custo total da sua execução ou o custo da sua re-execução. BOULOS e ONO (1999) apresentam avaliações mais precisas sobre os custos da execução de métodos, considerando as assinaturas dos métodos e construindo histogramas com as estatísticas coletadas.

As estatísticas sobre os dados das fontes em um ambiente distribuído e heterogêneo constituem parâmetros essenciais para as fórmulas de custo. Por exemplo, a cardinalidade de uma coleção ajuda a estimar a cardinalidade dos resultados intermediários das consultas, isto é, dos operadores aplicados à fonte de dados. Sobre os atributos, são necessárias, por exemplo, estatísticas sobre os valores mínimos e máximos que podem ser usados para calcular a seletividade do predicado.

A Figura 5 apresenta um fluxo de informações necessárias para um otimizador global de consultas em um ambiente distribuído e heterogêneo. Com as informações apresentadas, o otimizador pode combinar os custos de `PUSHDOWN` POPs com os custos de POPs internos para calcular o custo total de um plano de execução.

Um catálogo tradicional de custos e estatísticas inclui parâmetros como a cardinalidade das coleções, o número de páginas ocupadas em disco e as características físicas da base de dados (tamanho da página de dados, políticas de armazenamento das estruturas, etc.). Em um ambiente distribuído e heterogêneo, muito dessas informações estão encapsuladas nas fontes de dados e não estão disponíveis para o otimizador global de consultas. Além disso, o otimizador global requer características adicionais, como as relativas à rede de comunicação (por exemplo, tempo de transmissão de um pacote). Vale ressaltar que o esforço para a aquisição e a manutenção de parâmetros de custos e estatísticas no processamento de consultas é proporcional à precisão com a qual estes são adquiridos ou calculados (PIATETSKY-SHAPIRO e CONNEL, 1984). Por isso, considerando que a otimização não deve pesar no custo de execução de uma consulta,

grande parte do processo de aquisição e tratamento de tais parâmetros deve ser realizada em um momento anterior ao do processamento das consultas.

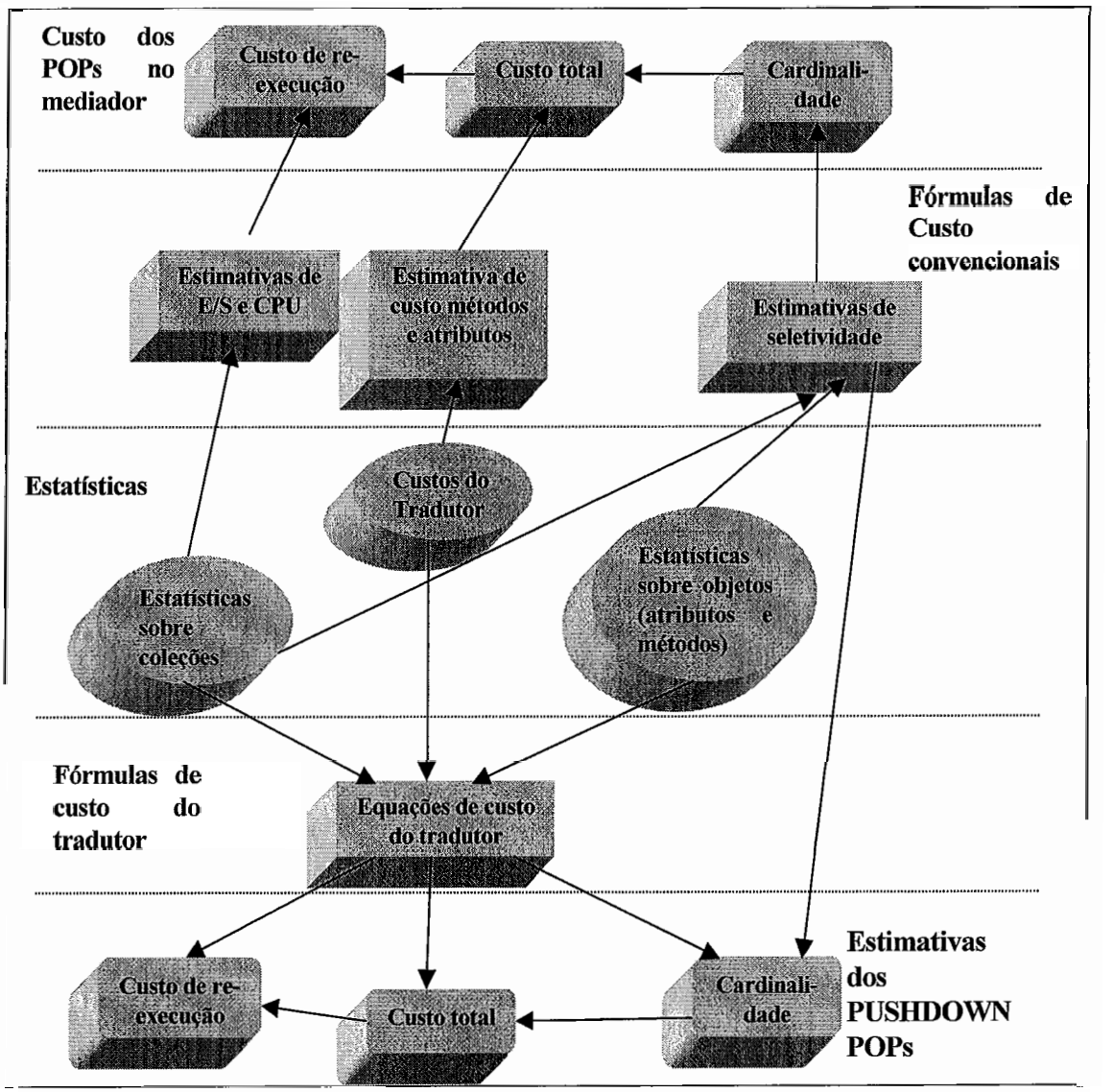

Figura 5. Fluxo de informações na otimização em um ambiente distribuído e heterogêneo

3.4 TRABALHOS RELACIONADOS

Nas arquiteturas distribuídas e heterogêneas, os trabalhos existentes sobre o provimento de custos e estatísticas para o processamento de consultas estão relacionados a funcionalidades embutidas nos módulos responsáveis pela otimização global das consultas e pelo acesso aos dados. Em MQSs, por exemplo, tais

funcionalidades geralmente são desempenhadas pelo mediador de consulta e seus tradutores.

Avaliamos diferentes arquiteturas quanto ao provimento de custos e estatísticas, dentre as quais destacamos os seguintes trabalhos: o sistema CORDS (ATTALURI *et al.*, 1995); as arquiteturas de mediadores *Garlic* (GARLIC, 1999, HAAS *et al.*, 1997) e DISCO (TOMASIC, *et al.*, 1995); e o projeto *ObjectGlobe* (BRAUMANDL *et al.*, 2001). Esses trabalhos são apresentados a seguir.

3.4.1 CORDS

Na arquitetura *multidatabase* CORDS (ATTALURI *et al.*, 1995), o servidor MDBS inclui funcionalidades para realizar uma coleta de amostras de desempenho, as quais são obtidas a partir da execução de consultas especiais nas fontes de dados. O servidor MDBS mantém uma base de dados sobre o desempenho médio dos algoritmos tradicionais de consulta em um SGBD, calculado a partir das amostragens realizadas. Em linhas gerais, de acordo com as amostras de desempenho coletadas, o servidor MDBS pode inferir quais são os algoritmos de execução utilizados e os respectivos custos de execução.

A arquitetura CORDS define quatro categorias de fontes de dados, de acordo com o provimento de custos e estatísticas. Estas categorias classificam como as fontes de dados disponibilizam os custos de processamento e o tamanho do resultado das consultas (ZHU *et al.*, 2000), sendo tais fontes divididas em:

- Fontes de dados que provêem estimativas de custo e tamanho do resultado. Neste caso, o processador global de consultas não necessita estimar os custos nem o tamanho do resultado das consultas;
- Fontes de dados que podem expor seus planos de execução. Algumas fontes de dados com SGBDs podem fornecer detalhes sobre seus planos de execução local, mas não disponibilizam os respectivos custos e estimativas de tamanho de resultado. Logo, o processador global de consultas globais deve estimar os custos e tamanhos de resultado de cada componente individual do plano disponibilizado;

- Fontes de dados com catálogos mas sem exposição de planos de execução. Este tipo de fonte dispõe de um catálogo que contém informações estruturais e algumas informações estatísticas sobre os dados armazenados. Deste modo, o processador global de consultas deve estimar os custos dos planos locais em função dessas informações; e
- Fontes de dados sem catálogo. O processador global de consultas define as funções de custo das fontes de dados.

Para as três últimas categorias, o otimizador global deve estimar as funções locais de custo do processamento de consultas nas fontes de dados. O CORDS apresenta três técnicas para realizar essa estimativa: por amostragem de consultas; por consultas de testes; e por *piggybacking*.

Na amostragem de consultas, as consultas submetidas às fontes de dados podem ser agrupadas em função do respectivo algoritmo de execução. Por exemplo, consultas que utilizam um mesmo método de acesso (como uma junção com índice) podem ser colocadas em um mesmo grupo. Após a definição dos grupos de algoritmos de execução, consultas exemplos são submetidas às fontes e o custo de processamento é observado. Um método matemático baseado em regressão múltipla é utilizado para estabelecer a estimativa da fórmula de custo para os integrantes de cada grupo.

A estimativa por consultas de testes determina alguns parâmetros de custo sobre as fontes de dados através da execução de consultas específicas. Por exemplo, pode-se determinar a cardinalidade e o volume de dados das coleções de uma determinada fonte. Para tanto, uma consulta específica pode ser submetida para estimar a cardinalidade das coleções e uma outra consulta pode recuperar objetos individuais para ajudar na estimativa dos respectivos volumes ocupados.

A técnica de estimativa de custos e estatísticas por *piggybacking* consiste em recuperar, durante a submissão de consultas a uma fonte de dados, atributos adicionais além dos solicitados em cada consulta. Sobre estes atributos adicionais, custos e estatísticas são calculados. Por exemplo, os números de valores distintos podem ser estimados para um atributo adicionado a uma consulta que acesse a coleção a qual o atributo pertença.

A arquitetura CORDS é bastante completa em sua classificação sobre as possíveis fontes de dados quanto ao provimento de custos e estatísticas. Essa arquitetura enfatiza a necessidade de conhecimento prévio sobre os métodos de acesso disponíveis nas fontes de dados e que este conhecimento pode alterar significativamente as estimativas de desempenho. Assumindo que informações sobre métodos de acesso são importantes no processo de otimização global de consultas, na arquitetura do DIG (RUBERG *et al.*, 2002b) – tema desta dissertação, isto também é registrado. A arquitetura CORDS possui um bom suporte para fontes de dados que possuem SGBDs, mas não apresenta métodos para a aquisição de custos e estatísticas em fontes de dados desprovidas de SGBDs. Nesse caso, a arquitetura assume um modelo arbitrário de custo. No DIG, nós estendemos as idéias apresentadas na arquitetura CORDS para permitir que as características do processamento de consultas em fontes limitadas possam ser descritas e seus custos e estatísticas possam ser adquiridos.

3.4.2 *GARLIC*

A arquitetura de mediadores *Garlic* (GARLIC, 1999) considera que o plano global de execução de uma consulta é representado por uma árvore de POPs. Os planos de execução das subconsultas enviadas para cada tradutor são tratados como operadores de alto nível, ditos `PUSHDOWN` POPs. Estes operadores têm seu custo calculado pelo tradutor a partir dos custos dos POPs envolvidos.

A arquitetura *Garlic* pressupõe que o mediador não precisa conhecer os custos dos operadores que irão compor um `PUSHDOWN` POP, sendo suficiente para o mediador a estimativa de custo do `PUSHDOWN` POP informada por cada tradutor. O operador `PUSHDOWN` tem como parâmetros fornecidos: o tempo de execução do operador; o tempo de re-execução do operador; e a cardinalidade resultante. Estes parâmetros simplificam a integração dos custos das fontes de dados em um plano global de consulta.

O sistema *Garlic* isola de forma adequada os operadores que devem ser calculados nas fontes de dados. Deste modo, o processo de otimização global de consultas fica simplificado tanto pelo uso do operador `PUSHDOWN` como pelos parâmetros que o operador disponibiliza. Porém, o *Garlic* pressupõe que o tradutor é capaz de estimar os custos que irão compor o operador `PUSHDOWN`.

O *Garlic* requer uma implementação de tradutor mais sofisticada para que ele possa atender aos requisitos do mediador. Isto é conseqüência do uso do operador `PUSHDOWN`, que simplifica a otimização global. Em RUBERG *et al.* (2002b), para a especificação da arquitetura do DIG, as extensões de funcionalidades relativas à aquisição de custos e estatísticas são destacadas do tradutor. Deste modo, é gerada uma unidade independente de coleta ao invés de um tradutor sobrecarregado. Com isso, favorecemos a especialização do tradutor no processamento de consultas e permitimos o compartilhamento de tais funcionalidades em diversos tradutores.

Uma limitação do *Garlic* ocorre quando o tradutor não consegue fornecer ou estimar o custo de um operador `PUSHDOWN`, pois essa tarefa não é prevista para o mediador. O provimento de custos e estatísticas do *Garlic* é bastante específico da arquitetura na qual foi concebido, por isso requer muitas alterações caso se deseje utilizar esta funcionalidade em uma outra arquitetura. Além disso, as estatísticas e custos adquiridos são orientados à representação dos operadores `PUSHDOWN`. Ou seja, não há flexibilidade para a representação de informações sobre as estruturas de dados (coleções, atributos, relacionamentos, etc.) ou mesmo sobre as características gerais das fontes de dados.

3.4.3 DISCO

A abordagem para otimização de consultas baseada em custos da arquitetura de mediadores DISCO (*Distributed Information Search COmponent*) (TOMASIC et al., 1998) combina o uso de um modelo de custo genérico e de informações de custo específicas fornecidas por tradutores. Nesta abordagem, os tradutores são responsáveis por exportar para o mediador os custos e estatísticas específicas de suas fontes de dados. Por sua vez, o mediador também mantém um modelo de custo genérico que é aplicado às fontes de dados cujos custos não são informados pelos respectivos tradutores.

Para permitir um modelo de custo extensível, a arquitetura DISCO prevê uma ferramenta específica para o implementador do tradutor. Esta ferramenta permite que sejam exportadas estatísticas, tamanhos de coleções e regras de custo de processamento. Para estatísticas sobre as coleções, o tradutor disponibiliza uma tripla com o número de objetos, o tamanho total e o tamanho médio das coleções. Para estatísticas sobre os atributos, a tripla se refere aos valores máximos e mínimos dos atributos, além do

número de valores distintos. Para regras de custo, têm-se no tradutor fórmulas para estimar o tempo de reposta para a recuperação da primeira tupla de uma consulta, o tempo de resposta para a obtenção da tupla seguinte e o tempo total (em milisegundos) de execução de uma consulta.

O DISCO foi um dos primeiros trabalhos a apontar o papel do tradutor como fornecedor dos custos e estatísticas para o processamento de consultas no mediador. Contudo, as informações de custos do DISCO são muito simples e, dependendo do modelo de custos adotado pelo mediador, podem ser insuficientes. Assim como o DISCO, o nosso trabalho assume que informações de custos e estatísticas devem ser coletadas junto à fonte de dados. Porém, o modelo de representação de custos e estatísticas do DISCO permite atender a somente uma pequena variedade de modelos de custos. No DIG, nós apresentamos um conjunto mais abrangente de informações a serem coletadas, como por exemplo o tempo de execução de um método, informações sobre as relações entre as coleções, etc. Além disso, o DISCO não separa os papéis de tradução de consultas e de coleta de dados nas fontes, nem os papéis de processamento de consultas e de tratamento das informações coletadas no mediador.

3.4.4 *OBJECTGLOBE*

O *ObjectGlobe* (BRAUMANDL *et al.*, 2001) consiste em um processador distribuído de consultas através da Internet. Sua arquitetura define três tipos de provedores de serviços: os provedores de dados, que disponibilizam as coleções a serem consultadas; os provedores de função, que oferecem os operadores para processar os dados consultados; e os provedores de ciclo (processamento), os quais são os servidores que irão executar os operadores de consulta.

O processamento de consultas no *ObjectGlobe* possui quatro etapas: pesquisa, otimização, conexão e execução. O serviço de pesquisa permite que sejam encontrados os provedores de dados (fontes de dados), os provedores de ciclo e os provedores de função que participarão da consulta. Na etapa de otimização, através das informações disponibilizadas pelo servidor de pesquisa, é determinado o plano de execução de menor custo para as consultas submetidas. Na etapa de conexão, os operadores do plano gerado são enviados para os provedores de ciclo. Por fim, na etapa de execução, o plano de uma consulta é processado seguindo a seqüência especificada na otimização.

O serviço de pesquisa mantém diversos tipos de informações sobre os provedores. Tais como: informações descritivas sobre os provedores de dados e funções; descrição da capacidade de processamento dos provedores de ciclo; estatísticas para a otimização da consulta; e informações de autorização. O serviço de pesquisa do *ObjectGlobe* acumula as funções de base de metadados dos provedores de dados e funções, de catálogo de informações físicas dos provedores de ciclo, de catálogo de estatísticas para a otimização da consulta e de diretório de autenticação.

Os custos disponíveis pelo serviço de pesquisa têm suas grandezas medidas em valor monetário ou tempo de resposta. No serviço de pesquisa são encontradas informações tais como histogramas da seletividade dos predicados simples, latência e largura de banda que conecta dois provedores de ciclo, a carga típica dos provedores de ciclo e URLs de funções que possam ser utilizadas para estimar o custo dos operadores de consultas.

Por ter uma estrutura bem relaxada, com diversos provedores de serviço, não se consegue identificar os papéis de mediação e tradução no *ObjectGlobe*. Uma característica que pode representar um fator limitante para o *ObjectGlobe* é este projeto ser baseado na tecnologia JINI (SUN, 1999), da *Sun Microsystems,* isto implica que os provedores de dados e funções necessariamente devem ter uma máquina virtual *Java* o que nem sempre é encontrado. Esse problema é solucionado no *ServiceGlobe* (KEIDL *et al.*, 2002), uma evolução do projeto *ObjectGlobe* onde as funcionalidades para o processamento distribuído de consultas foram reprojetadas dentro dos padrões de serviços *Web*.

O projeto *ObjectGlobe,* assim como, o seu sucessor – o *ServiceGlobe*, pode ser considerado um trabalho próximo ao nosso, pois é previsto um módulo específico para representar o catálogo de localização de serviços e publicação de custos e estatísticas sobre provedores de dados, de funções e de ciclo. Entretanto, a aquisição de dados deste serviço é passiva, isto é, depende dos demais componentes da arquitetura repassarem suas informações. Além disso, ainda cabe ao processador global de consultas tratar a ausência de informações sobre os provedores de serviços, mantendo para tanto um modelo de custo genérico.

3.5 CONSIDERAÇÕES FINAIS

Em todos os sistemas analisados, fundamental importância é atribuída às estimativas de custos e informações sobre o processamento de consultas nas fontes de dados, para que o mediador possa realizar uma otimização global das consultas. Alguns sistemas incluem funcionalidades no tradutor de consultas para atender às necessidades de estimativas e informações sobre as fontes de dados. Esta idéia é utilizada no nosso trabalho, que especializa essa atividade em coletores de custos e estatísticas associados ao tradutor. Observamos também que a obtenção de custos e estatísticas para a otimização de consultas é sempre realizada através de um módulo proprietário da arquitetura.

A aquisição de custos e estatísticas e o provimento destas informações em um módulo independente conferem maior flexibilidade e generalidade a esse tipo de serviço. Deste modo, essa funcionalidade torna-se disponível para diferentes arquiteturas de mediadores, bem como para outros sistemas que necessitem conhecer a capacidade de processamento de consultas das fontes de dados em um ambiente distribuído. Esta questão é relevante em cenários como a *Web* e em plataformas de *grid* (FOSTER *et al.*, 2001), sendo essa generalidade um dos principais focos da nossa pesquisa.

Esta flexibilidade e generalidade, abordada no nosso trabalho, tem como desafio produzir uma arquitetura de *software* que seja aderente aos requisitos de distribuição e heterogeneidade encontrados nos ambiente dos MQSs. Além disso, cada sistema costuma realizar a aquisição de custos e estatísticas de acordo com um modelo específico de custo. Isto não se aplica em uma arquitetura genérica para o provimento de custos e estatísticas. Um outro desafio consiste em disponibilizar um conjunto abrangente de informações que atendam aos possíveis modelos de custos encontrados nos diversos MQSs.

Capítulo 4

DIG: UM SERVIÇO PARA PROVER CUSTOS E ESTATÍSTICAS PARA O PROCESSAMENTO DISTRIBUÍDO DE CONSULTAS

O processamento distribuído de consultas requer diversas informações sobre as fontes de dados envolvidas, incluindo estatísticas e parâmetros de custo. Neste capítulo são apresentadas as características do Distributed Information Gatherer (DIG), uma arquitetura de serviço distribuído para a aquisição e publicação de informações que visa auxiliar o processamento de consultas em um ambiente distribuído, heterogêneo e com fontes de dados autônomas.

4.1 INTRODUÇÃO

Praticamente todas as técnicas para o processamento eficiente de consultas assumem que estão disponíveis informações sobre a base de dados, como estatísticas de distribuição de valores e parâmetros de custo de processamento (ELMASRI e NAVATHE, 1994, OZSÜ e BLAKELEY, 1995, OZSÜ e VALDURIEZ, 1999, FREIRE *et al.*, 2002). Em um SGBD centralizado, estas informações são tradicionalmente providas pelo próprio sistema de banco de dados através de um catálogo específico. Já em um ambiente distribuído e heterogêneo, a aquisição e a publicação dessas informações não é uma tarefa trivial (HAAS *et al.*, 1997), a qual geralmente é atribuída ao próprio processador global de consultas.

Entretanto, ao separarmos as atividades do processador de consultas global e o processo de aquisição de custos e estatísticas, o MQS se torna mais flexível. Isto pode ser verificado quando observamos que os algoritmos de otimização de consultas são indiferentes à inserção ou à remoção de fontes de dados no ambiente. Porém, o catálogo de custos e estatísticas é sensivelmente alterado quando ocorrem tais mudanças, pois deve refletir os novos parâmetros. Com a separação do processo de aquisição de custos e estatísticas, é possível fornecer estimativas para o processamento de consultas de diferentes arquiteturas, tais como os sistemas mediados e os *grids* computacionais (FOSTER *et al.*, 2001).

O *Distributed Information Gatherer* (DIG) (RUBERG *et al.*, 2002b) representa uma arquitetura para o provimento de informações, custos e estatísticas sobre fontes de dados autônomas em um ambiente distribuído e heterogêneo. O sistema DIG visa auxiliar o processamento distribuído de consultas, apresentando de modo uniforme e integrado a capacidade das fontes de dados através de informações relativas aos dados armazenados e à capacidade de execução de consultas. Estas informações são representadas em um modelo de classes definidas sobre os grupos de custos e estatísticas, adquiridos no DIG por unidades de coleta que estão associadas aos tradutores das fontes de dados. Após a coleta, estas informações são disponibilizadas pelo DIG (possivelmente com algum tratamento específico) para o processador global de consultas através de um provedor global de custos e estatísticas.

Assim, um ponto chave na arquitetura do DIG é o catálogo de informações disponibilizadas. Este catálogo permite que os diversos processadores globais de consultas tenham uma visão unificada dos custos e estatística coletados pelo DIG. Estas informações podem ser obtidas através da inspeção dos dados ou derivadas de funções de custo.

Vale ressaltar que o DIG não tem como objetivo processar consultas, mas oferecer uma vasta gama de informações típicas e necessárias para realizar essa tarefa em um ambiente distribuído, heterogêneo e com fontes autônomas. Em uma arquitetura de mediadores, por exemplo, o DIG pode atuar como apoio ao mediador no processo de otimização de consultas. Além disso, nesse caso, o DIG pode utilizar os serviços do tradutor para realizar a aquisição de custos e estatísticas nas diversas fontes de dados. A Figura 6 apresenta como o sistema DIG se relaciona com os demais componentes de uma arquitetura de mediadores.

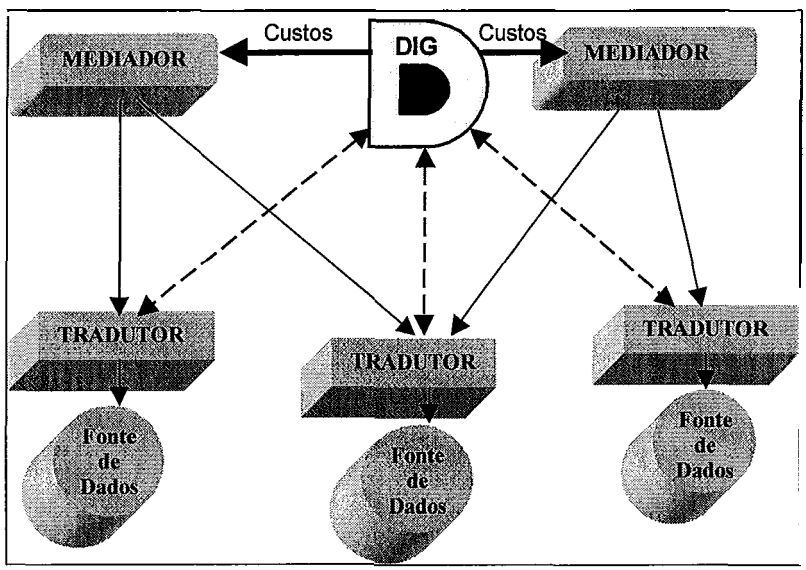

Figura 6. Arquitetura de Mediadores e o DIG

Neste capítulo são apresentadas a arquitetura do DIG e suas principais unidades, com destaque para o módulo provedor de custos e estatísticas e para a unidade de coleta de dados. Apresentamos também como as fontes de dados podem ser registradas na arquitetura do DIG, através de um arquivo de publicação. Por fim, detalhamos o catálogo de custos e estatísticas, que representa de modo unificado as informações disponibilizadas na interface do DIG com o processador global de consultas.

4.2 ARQUITETURA DO DIG

A arquitetura do DIG é representada por um sistema distribuído composto por dois módulos básicos: um serviço **provedor DIG** de informações sobre custos, estatísticas e capacidades de processamento de consultas, responsável pela interface final do sistema; e um **coletor DIG** de custos e estatísticas, que interage com as fontes de dados. A Figura 7 apresenta uma visão geral da arquitetura do sistema DIG, realçando também as interfaces internas e externas de cada módulo componente.

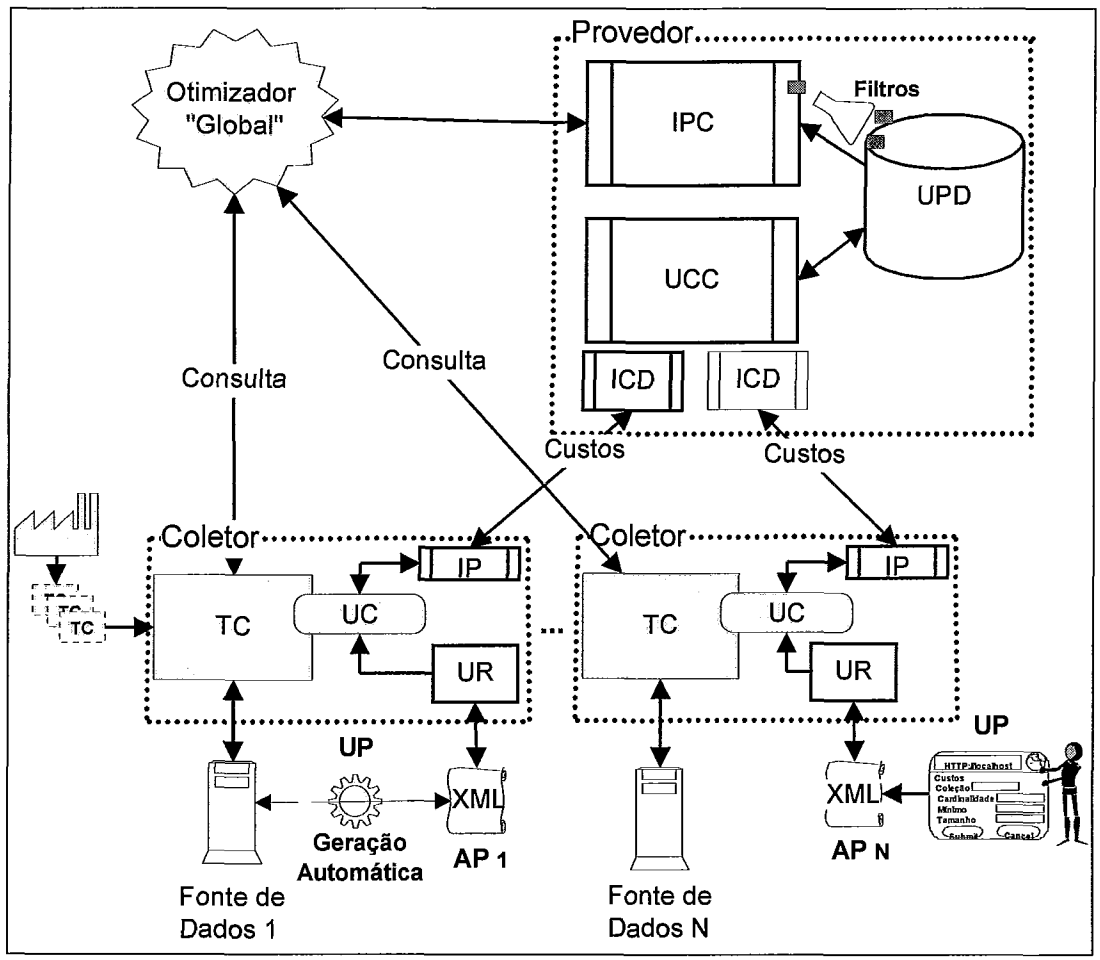

Figura 7. Arquitetura do sistema DIG

Alguns aspectos foram fundamentais para nortear a concepção do DIG. Em primeiro lugar, consideramos a heterogeneidade do ambiente, provendo mecanismos adequados para lidar com fontes de dados de diferentes capacidades (possivelmente restritas) para o processamento de consultas. Assumimos também que podem existir fontes de dados desprovidas de um SGBD.

Um outro aspecto importante no projeto da arquitetura do DIG foi a elaboração do catálogo de serviços. Para oferecer uma interface uniforme e extensível, o modelo do catálogo de serviços do DIG é provido independentemente do modelo de dados das fontes. Podem ser representadas informações sobre fontes semi-estruturadas, relacionais e/ou baseadas em objetos. O modelo do catálogo foi elaborado visando também a abrangência do serviço oferecido ao processador global de consultas, provendo um vasto número de informações de custo. Além disso, a arquitetura do DIG dispõe de mecanismos para aquisição e armazenamento de informações sobre a execução de métodos e programas. Essa característica é importante em MDBSs para a execução de programas armazenados nos bancos de dados (*stored procedures*), sendo também essencial para serviços de consulta na *Web* e em plataformas de *grid*.

Aspectos que dizem respeito à interação entre as fontes de dados estão fora do escopo da arquitetura do DIG. Visto que o sistema DIG não é responsável pelo processamento das consultas, não cabe ao provedor DIG tratar questões como a redundância de informações entre as fontes de dados ou a extração de um esquema global. Consideramos que o custo de operadores que atuam sobre duas ou mais fontes de dados é estimado a partir dos custos obtidos em cada fonte de dados. Assumimos que essa composição de custos é realizada pelo otimizador global de consultas, de acordo com sua estratégia para percorrer o espaço de possíveis planos de execução.

Os dois principais pilares de sustentação da arquitetura proposta são o provedor e o coletor DIG, tendo como infra-estrutura básica o catálogo de custos e estatísticas, conforme veremos em detalhes a seguir.

4.2.1 O PROVEDOR DIG

O provedor DIG é o módulo responsável pela gerência das informações coletadas nas diversas fontes de dados e pela publicação destas informações em uma interface padronizada, dita catálogo de serviços. É este módulo que coordena as unidades de coleta de custos e estatísticas. Além disso, cabe ao provedor DIG alimentar a base dos dados coletados da arquitetura, permitindo que estes sejam tratados posteriormente para gerar novos parâmetros de custo.

O provedor DIG possui os seguintes componentes: uma unidade de controle dos coletores de dados; uma interface com os coletores de dados; uma interface com o processador global de consultas; e uma unidade de persistência dos dados coletados.

4.2.1.1 UNIDADE DE CONTROLE DOS COLETORES (UCC)

A **Unidade de Controle dos Coletores** registra quais coletores devem ser acionados para a aquisição de custos e estatísticas. Esta unidade dispara o processo de aquisição das informações nos coletores, permitindo que essa aquisição seja específica de um determinado coletor ou geral a todos os coletores.

A UCC mantém um cadastro dos coletores registrados para o provedor DIG. Este cadastrado é representado por um arquivo com entradas da forma **estação:tradutor**, onde **estação** é o endereço IP, ou nome da estação, e **tradutor** identifica qual tradutor será acessado na estação. Cada entrada **estação:tradutor** identifica um coletor DIG. Ao iniciar um provedor DIG, esse arquivo é lido e são determinados quais coletores serão consultados, criando na UCC uma lista de coletores ativos. Ou seja, essa lista contém os coletores que efetivamente serão utilizados no processo de aquisição de custos e estatísticas.

O processo de aquisição de informações do DIG é disparado a partir da execução de um método da UCC, chamado ***acquireCosts***. Este método itera a aquisição de custos de estatísticas (utilizando a Interface com Coletores de Dados) para cada entrada da lista de coletores ativos. O processo de aquisição pode ser realizado a qualquer tempo, porém o processador global de consultas poderá solicitar que esta aquisição seja realizada periodicamente, para que a base de informações do DIG mantenha-se atualizada.

4.2.1.2 INTERFACE COM COLETORES DE DADOS (ICD)

Toda a comunicação entre o provedor e o coletor de dados do DIG é realizada através da **Interface com Coletores de Dados**. Para o provedor DIG, a ICD disponibiliza o acesso individual a cada coletor de dados. Esta interface possui métodos e objetos referentes aos dados publicados pelos coletores e à administração das unidades de coleta. O principal método desta interface permite disparar o processo de aquisição de dados no coletor. Já os objetos acessíveis por esta interface representam a última aquisição realizada.

Basicamente, a ICD consiste em um conjunto de objetos distribuídos que permitem o acesso aos dados do coletor. Esta interface pode ser implementada através de tecnologias de objetos distribuídos padrão de mercado, como por exemplo CORBA ou serviços *Web*.

4.2.1.3 INTERFACE COM PROCESSADOR GLOBAL DE CONSULTAS (IPC)

A **Interface com Processador global de Consultas** representa o catálogo de serviços publicado pelo provedor DIG para o ambiente distribuído. Esta interface pode ser acessada por um processador global de consultas ou por qualquer outro sistema interessado na descrição da capacidade de processamento de consultas das fontes de dados disponíveis no ambiente.

Filtros para tratamentos especiais sobre os dados adquiridos (como redução de distorções, mineração de dados, etc.) podem estar associados à IPC. Ou seja, podem ser gerados parâmetros de custos derivados a partir dos dados coletados. Nesse caso, os dados derivados podem ser materializados ou representados por visões sobre a base de dados coletados.

As estatísticas e custos básicos desta interface são detalhados no catálogo de serviços do DIG (vide Seção 4.2.4). Podem ser utilizados diferentes critérios de consulta à IPC, determinando assim a granulosidade das informações obtidas – de acordo com o interesse do processador global de consultas. É possível recuperar todas informações de um determinado sítio ou apenas de uma dada coleção, ou ainda um custo ou estatística específica (como, por exemplo, a cardinalidade de uma coleção).

4.2.1.4 UNIDADE DE PERSISTÊNCIA DE DADOS COLETADOS (UPD)

Por fim, a **Unidade de Persistência de Dados coletados** armazena todas as informações obtidas por um determinado provedor. Esta base de custos e estatísticas permite que os dados obtidos possam ser manipulados e submetidos a tratamentos posteriores. Desta forma, regras podem ser aplicadas para otimizar o fornecimento e a qualidade dos valores apresentados pela IPC ao processador global de consulta.

Esta unidade pode ser representada por qualquer tipo de serviço de persistência, desde arquivos texto até SGBDs. Obviamente, é interessante que o acesso aos dados armazenados seja extremamente eficiente. Independente da tecnologia utilizada para

implementar a UPD, é importante também que seja possível publicar diferentes visões sobre os dados armazenados.

4.2.2 O COLETOR DIG

Um coletor da arquitetura do DIG tem a responsabilidade de realizar a aquisição de custos e estatísticas em uma determinada fonte de dados (SGBD, arquivo, programa, etc.) e informar ao provedor o que foi coletado. Os coletores DIG funcionam como adaptadores para as diversas fontes de dados de um ambiente distribuído.

O coletor DIG requer que sejam descritos os métodos de aquisição de dados nas fontes integradas à arquitetura, possuindo as seguintes unidades funcionais: uma unidade de registro; uma unidade coletora de custos para a fonte de dados; e uma interface com o provedor DIG.

4.2.2.1 UNIDADE DE REGISTRO (UR)

A **Unidade de Registro** é responsável pela inscrição das fontes de dados no coletor DIG. Para tanto, a UR recebe os arquivos de publicação de custos e estatísticas das fontes de dados, contendo um *parser* XML e funções para extração de *tags*. Essa unidade lê as informações sobre os métodos de aquisição de custos e estatísticas descritos para cada fonte de dados do sistema.

Observe que cada coletor DIG pode estar associado a uma ou mais fontes de dados. Para cada fonte de dados, a UR gera o mapeamento adequado entre os dados publicados pela fonte e a interface de custos e estatísticas disponibilizada pelo coletor para o provedor DIG.

4.2.2.2 UNIDADE COLETORA DE CUSTOS (UC)

A **Unidade Coletora de custos** pode ser vista como um "cartucho" de funcionalidades a ser adicionado ao tradutor de consultas. Isto significa que o coletor DIG aproveita a infra-estrutura oferecida pelo **Tradutor de Consultas** (TC) para acessar a respectiva fonte de dados. Vale ressaltar que o contexto típico de aplicação do DIG é um sistema mediado de consulta, onde já existem tradutores para as fontes de dados. Contudo, nada impede que um tradutor exclusivo seja construído e utilizado para atender às necessidades do DIG.

A unidade coletora emprega os métodos informados pela UR para adquirir os custos e estatísticas da fonte de dados. Estes métodos podem ser representados em uma linguagem canônica ou através de uma linguagem conhecida pelo tradutor. Observe também que um método de aquisição pode estar implementado na fonte de dados (por exemplo, através de comandos para consulta em SQL) ou no próprio coletor. Caso a UC não consiga executar um método de aquisição ou o retorno desse método não seja válido, o coletor pode associar valores padrões aos respectivos custos ou estatísticas.

É importante ressaltar que a adição de uma nova fonte de dados no sistema não representa uma tarefa complexa, pois o conjunto de informações a serem adquiridas em uma fonte varia de acordo com a necessidade/capacidade de cada coletor. Deste modo, a atividade de coleta não inviabiliza a disponibilidade do serviço. Além disso, podem ser utilizados geradores de tradutores específicos a partir de um conjunto mínimo de funcionalidades, como proposto em SAHUGUET e AZAVANT (1999).

Os principais métodos para aquisição de custos e estatísticas do DIG são descritos em detalhes na Seção 4.3.

4.2.2.3 INTERFACE COM PROVEDOR DE SERVIÇO (IP)

A **Interface com Provedor de serviço** é a contrapartida da Interface com Coletores de Dados existente no provedor DIG, representando um conjunto de serviços para acionar a aquisição das informações nas fontes de dados. Cada categoria de custos e estatísticas é representada por um serviço específico na IP.

Através desta interface, o coletor DIG publica as informações coletadas, além de poder requisitar alguns procedimentos ao provedor DIG.

4.2.3 UNIDADE DE PUBLICAÇÃO

A **Unidade de Publicação** (UP) é responsável pela geração do arquivo de publicação das fontes de dados do sistema DIG, ou seja, pelo registro dos métodos de aquisição de custos e estatísticas de cada fonte de dados que é monitorada pelo DIG. Este arquivo tem um formato bem definido e, portanto, pode ser construído manualmente para as fontes de dados com um pequeno número de coleções.

Uma UP mais sofisticada, ainda no caso de fontes de dados com poucas coleções, pode utilizar formulários HTML para acionar *scripts* que geram os arquivos de publicação do DIG no formato XML. Em fontes de dados contendo um maior número de coleções, como ocorre nas fontes de dados representadas por SGBDs, o arquivo de publicação tende a ser extenso e a entrada manual de dados pode ser exaustiva. Nesse caso, a UP pode realizar a geração automática das descrições sobre a aquisição de custos e estatísticas através da leitura do catálogo interno do SGBD. Esta tarefa não é complexa, pois atualmente vários bancos de dados oferecem ferramentas para a extração de dados no formato XML.

4.2.3.1 ARQUIVO DE PUBLICAÇÃO (AP)

O **Arquivo de Publicação** registra para o coletor DIG qual fonte de dados está sendo tratada, onde um coletor pode estar associado a uma ou mais fontes de dados. O objetivo do arquivo de publicação é permitir ao coletor identificar as particularidades de cada fonte de dados. Por exemplo, neste arquivo pode estar descrito como um objeto é recuperado, como um determinado método é executado, entre outras operações. Cada fonte de dados que participa do sistema DIG deve conter um arquivo de publicação com suas devidas informações.

```
<Site name="Porto">
  <Collection name="Alunos">
    <SimpleAttribute name="orientador"> <Reference>True</Reference>
      <Count>"select count(a.orientador) from a in Alunos"</Count>
    </SimpleAttribute>
  </Collection>
  <Collection name="Professores">
    <Cardinality>"select count(p) from p in Professores"
</Cardinality>
  </Collection>
  <Method name="getStudent">
    <Execution>"select a from a in Alunos"</Execution> </Method>
  <Relationship name="orientado por">
    <FirstCollection>Alunos</FirstCollection>
    <SecondCollection>Professores</SecondCollection>
    <RelationAttribute>Orientador</RelationAttribute>
  </Relationship>
  <Type name="Aluno">
    <CollectionType>Aluno</CollectionType>
    <Extent>Alunos</Extent>
    <Schema>nome;cr;orientador</Schema>
  </Type>
  <AbleToProject>True</AbleToProject>
  <NetworkTime>10</NetworkTime>
  <PageSize>1024</PageSize>
</Site>
```

Figura 8. Exemplo de arquivo de publicação do coletor DIG

O arquivo de publicação é descrito no formato XML de acordo com um DTD específico. Os métodos para aquisição de custos e estatísticas podem ser descritos na linguagem canônica de consulta da arquitetura de mediação, pois o DIG não impõe um modelo canônico de dados ou uma linguagem específica de consulta. Nesse caso, cabe aos tradutores de consultas a conversão (se necessária) da linguagem canônica escolhida para a linguagem das fontes de dados específicas e vice-versa. No exemplo da Figura 8, a linguagem canônica para descrição dos métodos de aquisição de custos e estatísticas é a SQL3 (EISENBERG E MELTON, 1999). Estes métodos são submetidos à fonte de dados pelo respectivo coletor DIG.

No arquivo de publicação de uma fonte de dados podem ser descritos métodos para a aquisição de custos e estatísticas sobre coleções, atributos, métodos, tipos e relacionamentos (vide exemplo na Figura 8). A capacidade de realizar operações como seleção, junção, projeção, etc., também pode constar no arquivo de publicação. Caso essas operações não sejam informadas, assumimos que a fonte de dados não possui a capacidade de realizá-las.

4.2.4 Esquema de Classes para Representar Custos

Basicamente, a especificação do catálogo de informações disponibilizadas pelo provedor DIG aborda dois aspectos:

(i) quais informações são relevantes para o processamento de consultas em um ambiente distribuído; e

(ii) como tais informações podem ser extraídas das fontes de dados envolvidas.

O catálogo do DIG foi concebido para ser flexível, suportando o registro de custos e estatísticas a partir de fontes de dados com diferentes capacidades de processamento. Além disso, as informações do catálogo são publicadas em uma interface padronizada, definindo um formato que facilita o uso das informações coletadas pelo DIG para o otimizador global de consultas.

Com o objetivo de definir quais informações são relevantes para o processamento distribuído de consultas, na especificação do catálogo de serviços do DIG consideramos que o processador global de consultas necessita das informações de custo referentes às subconsultas executadas em cada fonte de dados. Por isso, assumimos que, idealmente,

devemos ter no catálogo de serviços os custos e estatísticas referentes aos operadores e operandos lógicos do plano global de execução da consulta, aplicados às fontes de dados. Vale ressaltar que características físicas (tamanho de página, memória, tempo de CPU de uma operação, etc.) não possuem destaque no contexto da arquitetura do DIG porque geralmente estão encapsuladas nas fontes de dados, possivelmente até não existindo devido à heterogeneidade do ambiente. Todavia, o DIG também permite que essas informações sejam coletadas e publicadas no catálogo de serviços.

As informações disponíveis no catálogo do DIG consistem em um conjunto "básico" de operadores comuns às diversas linguagens de consulta. Assim, o DIG não está limitado a uma linguagem de consulta específica, podendo fornecer custos e estatísticas sobre diversos operadores de diversas álgebras. Estas informações são descritas pelo diagrama de classes UML apresentado na Figura 9.

O catálogo de serviços do DIG classifica custos, estatísticas e informações relevantes ao processamento de consultas conforme as seguintes categorias: sítio, coleção, atributos simples, atributos do tipo coleção, métodos, tipos, relacionamentos e operadores (vide Figura 9). WANG (2001) apresenta um modelo de custo para o processamento de consultas em bases de objetos. Em tal modelo são detalhados os custos e estatísticas sobre coleções, atributos, métodos e tipos que utilizamos para compor o catálogo do DIG. Em RUBERG (2001) é apresentado um modelo de custos para a avaliação de expressões de caminho onde é detalhado um conjunto de estatísticas necessárias para a estimativa de custos sobre relacionamentos entre coleções de dados. Também são detalhadas funções de custo para a estimativa de fatores de seletividade de junções. As estatísticas sobre relacionamentos e as estimativas sobre fatores de seletividade do operador de junção apresentadas em RUBERG (2001) foram utilizadas na definição das informações do catálogo DIG.

A entidade principal no catálogo do DIG é o **sítio** lógico (*site*), que representa uma fonte de dados e tem como atributos simples as informações físicas de uma determinada fonte de dados, além da capacidade de processamento de consultas desta fonte. Além disso, um sítio possui estruturas que representam outros aspectos da aquisição de custos e estatísticas na fonte de dados, em diferentes níveis de abstração, referentes aos tipos e às coleções de dados, aos atributos, aos métodos e aos relacionamentos.

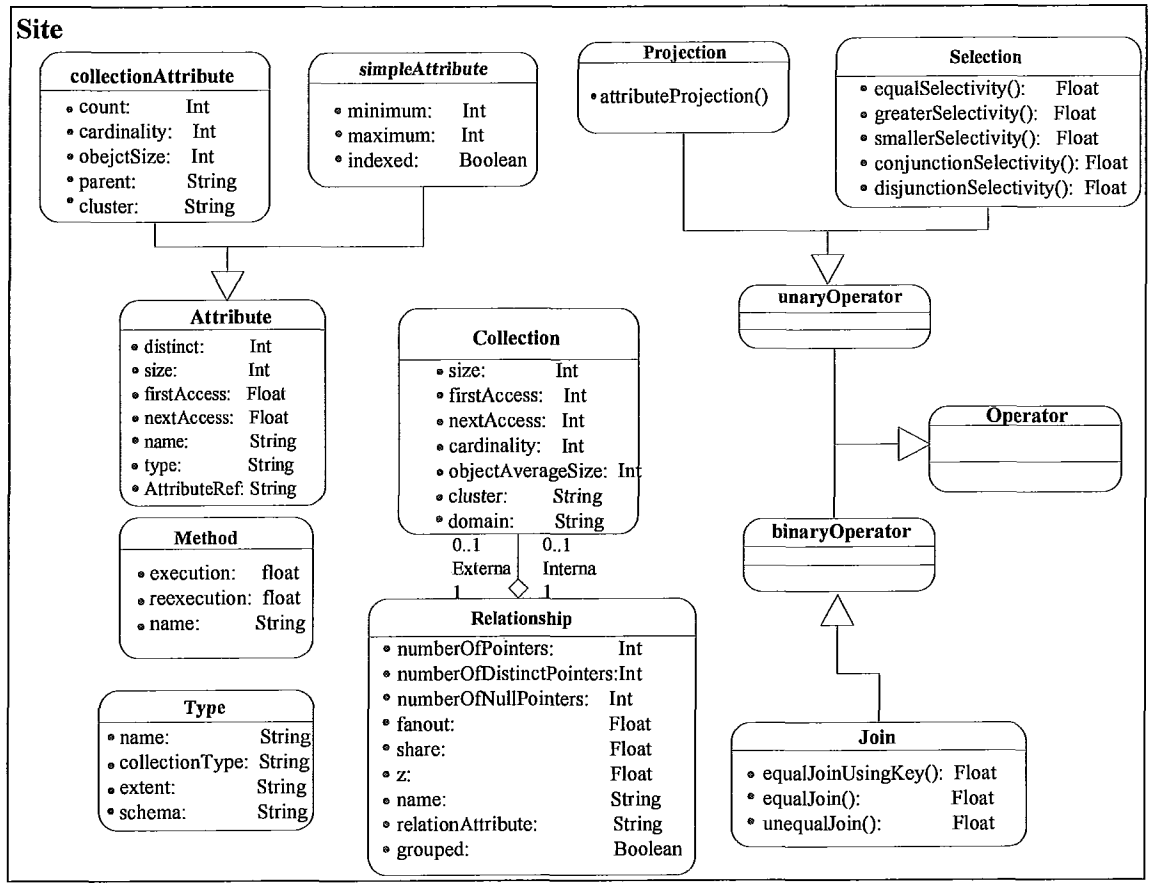

Figura 9. Diagrama de classes UML do catálogo de serviços do DIG

Parâmetros de custos relativos aos operadores de consulta, como projeção, seleção e junção, podem ser calculados a partir de estatísticas previamente coletadas em um sítio. Observe que o esquema da Figura 9 pode ser estendido para suportar características adicionais como, por exemplo, operadores de agregação de consulta. O trabalho de BOUGANIM *et al.* (2001) apresenta um modelo de custos para lidar com funções que possuem um alto custo de processamento, e como tais funções devem ter um tratamento diferenciado no processo de otimização de consultas. Este tratamento particular para funções é adotado no catálogo DIG, pois consideramos que a execução de funções e programas deve ser tratada como operadores especiais para fins de levantamento de custos e estatísticas. No catálogo DIG, a classe "`operator`" representa esse conceito, como pode ser visto na Figura 9.

Em BOULOS e ONO (1999) são definidas as estratégias para estimativa de custos na execução de métodos em um sistema gerenciador de bases de dados relacional-objeto. A estimativa de tempo de execução de métodos foram observadas no catálogo do DIG na classe "`method`", como visto na Figura 9.

4.3 MÉTODOS PARA AQUISIÇÃO DE CUSTOS E ESTATÍSTICAS

O catálogo de custos e estatísticas do DIG possui duas visões diferentes:

- a **visão de aquisição**; e
- a **visão de publicação.**

Na primeira visão, é expresso como os coletores DIG adquirem os parâmetros de custos e estatísticas. Por exemplo, para coletar a cardinalidade de uma determinada coleção (`Collection`), o respectivo parâmetro *cardinality* corresponde ao resultado da seguinte consulta: "`select count(*) from Collection`". Ou seja, nesse nível de abstração, o parâmetro *cardinality* descreve o comando que deve ser submetido ao tradutor de uma fonte de dados para que o valor deste parâmetro possa ser atribuído.

Na segunda visão, os parâmetros de custos e estatísticas já possuem valores atribuídos, os quais podem ser disponibilizados para o processador global de consultas. Por exemplo, o parâmetro *cardinality* poderia ter o valor "1046" indicando que a coleção possui 1046 instâncias. É importante observar que alguns parâmetros podem estar presentes na visão de publicação e não constarem na visão de aquisição. Isto acontece porque esses parâmetros podem ser calculados em função de outros parâmetros adquiridos.

Nós classificamos os dados coletados pelo DIG em: **informações**, que representam características gerais sobre o elemento descrito, como a capacidade de processamento de consultas da fonte de dados e a existência de índices em uma estrutura de dados; **estatísticas**, relativas às características obtidas por amostragem nas fontes de dados ou no acesso a essas fontes, em geral parâmetros internos de desempenho (isto é, utilizados para compor funções de custos); e **custos**, que geralmente representam tempo gasto ou número de operações realizadas no processamento de consultas. Para cada dado coletado pelo DIG, nós apresentamos uma descrição, um método típico de aquisição e a unidade de medida utilizada. Vale ressaltar que o método de aquisição indicado representa uma alternativa de coleta, visto que esta pode ser realizada por diferentes procedimentos.

As estatísticas e custos coletados pelo DIG são divididos em diferentes grupos de afinidade: sítios lógicos; tipos de dados; coleções; objetos, abrangendo características de

atributos (mono e/ou multivalorados) e métodos; relacionamentos; e operadores de consulta, onde destacamos os operadores de seleção, junção e projeção.

O conjunto de métodos para aquisição e estimativa de custos e estatísticas que são propostos no catálogo do DIG é o mais abrangente possível, sem comprometer a complexidade de construção dos coletores de custo. Porém, os parâmetros disponibilizados não são exaustivos. Caso seja necessário disponibilizar algum novo parâmetro, a arquitetura do DIG e a especificação do catálogo são suficientemente flexíveis para incorporar as novas definições. A seguir, apresentaremos os métodos para a aquisição de dados em cada grupo de afinidade.

4.3.1 DADOS SOBRE SÍTIOS LÓGICOS

Este grupo possui informações que detalham características específicas das fontes de dados da arquitetura do DIG. Basicamente, essas informações são relativas à máquina servidora e aos programas que hospedam a fonte de dados. Os principais dados coletados sobre sítios lógicos são exibidos na Tabela 1. Observe que existe uma informação sobre o sítio para indicar se este representa um SGBD, visto que nesse caso a complexidade e a capacidade de processamento de consultas da fonte são potencialmente significativas, sendo portanto interessante destacar essa característica.

Tabela 1. Dados sobre Sítios Lógicos

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	networkTime	Tempo de envio de um pacote de dados na rede.	medido ou informado	ms
Informações:				
	pageSize	Tamanho da página de dados no sítio.	informado	bytes
	ableToProj	Indica se o sítio é capaz de realizar projeções.	informado	lógico
	ableToJoin	Indica se o sítio é capaz de realizar junções.	informado	lógico
	isDBMS	Indica se o sítio representa um SGBD.	informado	lógico

Vale ressaltar também que, em um ambiente distribuído, não é trivial determinar o método de aquisição e publicação de certas informações. Por exemplo, o tempo de envio de um pacote de dados na rede pode ser muito influenciado pela carga de comunicação no instante da medição. Uma alternativa para melhorar a qualidade dos valores publicados para essa estatística é manter um registro histórico das medições.

Nesse caso, a visão de publicação desta estatística pode ser determinada em função da hora de consulta ao provedor DIG, por exemplo.

4.3.2 DADOS SOBRE TIPOS

As características sobre os tipos de dados não representam estatísticas ou custos propriamente ditos, mas sim informações de catálogo necessárias para o processamento de consultas. Essas informações (vide Tabela 2) auxiliam a enriquecer a semântica dos dados publicados pelo DIG. Por exemplo, a lista de coleções associadas a um tipo serve para apoiar a identificação das fontes de dados envolvidas em uma consulta. O esquema de um tipo T (*schema*(T)) retorna o tipo dos atributos de T. Já a extensão de um tipo T (*extent*(T)) apresenta o nome da coleção padrão onde são armazenados as instâncias desse tipo.

Tabela 2. Dados sobre Tipos

DADO		DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Informações:				
	typeCollections(T)	Coleções do tipo T.	informado	vetor de nomes de coleções
	extent(T)	Extensão do tipo T.	informado	nome da coleção
	schema(T)	Esquema do tipo T.	informado	vetor de nomes de tipos

4.3.3 DADOS SOBRE COLEÇÕES

Talvez o principal aspecto capturado pelo catálogo do DIG seja o conceito de uma coleção de objetos. Dados como a cardinalidade, o volume ocupado em disco e o tamanho médio dos objetos de uma coleção são extremamente necessários para a predição de desempenho do processamento de consultas (OZSÜ e VALDURIEZ, 1999). Sobre este aspecto, diversas informações, estatísticas e custos podem ser adquiridos e posteriormente disponibilizados no catálogo do DIG para o processador global de consultas.

Dentre os dados coletados sobre coleções, podemos destacar os custos de primeiro acesso e acesso subsequente aos objetos de uma coleção. Estes custos são importantes para determinar, por exemplo, o custo de leitura da coleção completa (HAAS *et al.*, 1997). Observe que o primeiro acesso a uma coleção corresponde a uma leitura de

acesso aleatório, enquanto que um acesso subsequente corresponde à leitura seqüencial, sendo o custo deste último inferior ao primeiro. Entretanto, esses custos podem se igualar caso a lista de identificadores das instâncias de uma coleção possua uma ordem diferente da que determina o armazenamento dessas instâncias.

Tabela 3. Dados sobre Coleções

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	card(C)	Cardinalidade da coleção C.	*count*(select * from C)	objetos
	size(C)	Tamanho total da coleção C.	*size*(select * from C)	bytes
	size(C.oid)	Tamanho médio de um objeto da coleção C.	*size*(C) / *card*(C)	bytes
Custos:				
	firstAccess(C)	Custo do primeiro acesso à coleção C.	*firstAccess*(select C.oid from C)	ms
	nextAccess(C)	Custo de um acesso sucessivo à coleção C.	*nextAccess*(select C.oid from C)	ms
Informações:				
	type(C)	Tipo da coleção C.	catálogo (informado)	nome do tipo
	isClustered(C)	Indica se a coleção C está agrupada fisicamente com outra coleção.	informado	nome da coleção

4.3.4 DADOS SOBRE OBJETOS

Este grupo de estatísticas e custos abrange métricas referentes à estrutura e ao comportamento dos objetos armazenados em uma fonte de dados. Neste tema, identificamos três subgrupos: estatísticas e custos sobre atributos monovalorados, sobre atributos multivalorados (do tipo coleção) e sobre métodos.

4.3.4.1 ATRIBUTOS MONOVALORADOS

Estes dados são essenciais à maioria dos métodos para estimativa de fatores de seletividade, dentre outras métricas derivadas. O custo de acesso a um atributo simples normalmente é confundido com o custo médio de acesso a um objeto na coleção, já que, de forma geral, para recuperar um atributo é necessário recuperar o objeto inteiro. Contudo, podemos encontrar fontes de dados que armazenam separadamente os atributos dos objetos. Por exemplo, isso ocorre caso coleção esteja fragmentada verticalmente. Estes dados são descritos a seguir na Tabela 4.

Tabela 4. Dados sobre Atributos Monovalorados

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	distinct(C.a)	Número de valores distintos do atributo "a" da coleção C.	*count*(select distinct(C.a) from C)	objetos
	min(C.a)	Valor mínimo do atributo "a" da coleção C.	select min(C.a) from C	valor
	max(C.a)	Valor máximo do atributo "a" da coleção C.	select max(C.a) from C	valor
	size(C.a)	Tamanho médio do atributo "a" da coleção C.	*size*(select C.a from C) / *count*(select C.a from C)	valor
Custos:				
	firstAccess(C.a)	Custo do primeiro acesso ao atributo "a" da coleção C.	*firstAccess*(select C.a from C)	ms
	nextAccess(C.a)	Custo de um acesso sucessivo ao atributo "a" da coleção C.	*nextAccess*(select C.a from C)	ms
	firstIndexed-Access(C.a)	Custo do primeiro acesso ao atributo "a" da coleção C, o qual está indexado.	*firstAccess* (select C.a from C)	ms
	nextIndexed-Access(C.a)	Custo do próximo acesso ao atributo "a" da coleção c, o qual está indexado.	*nextAccess* (select C.a from C)	ms
Informações:				
	isIndexed(C.a)	Indica a existência de índice no atributo "a" da coleção C.	informado	lógico
	type(C.a)	Tipo do atributo "a" da coleção C.	informado	bytes
	isReference(C.a)	Identifica se o atributo "a" é do tipo referência (ponteiro).	informado	lógico

A informação sobre a existência de índice em um atributo serve para auxiliar a estimativa de custos das operações que utilizam este atributo como base de acesso aos objetos da coleção. É importante observar que nos limitamos a informar a existência do índice e não detalhamos qual o tipo do mesmo. Isto porque assumimos que esta informação é suficiente, já que a capacidade de processamento de consultas da fonte de dados pode ser bastante limitada.

4.3.4.2 ATRIBUTOS MULTIVALORADOS

Este subgrupo de dados do DIG é relativo aos atributos multivalorados, também conhecido como sendo do tipo coleção (*Collection Value Attribute* – CVA) (WANG,

2000). Esse tipo de atributo permite armazenar múltiplas ocorrências, possivelmente replicadas. Por isso, é necessário termos dados que reflitam tanto as estatísticas e custos por objeto que contém esse atributo como todas ocorrências do atributo em uma coleção, conforme podemos ver na Tabela 5.

Tabela 5. Dados sobre Atributos Multivalorados

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	count(C.a)	Nº total de ocorrências do atributo "a" da coleção C, considerando as réplicas.	Para toda instância de C, totalize "select count(C.a)"	objetos
	distinct(C.a)	Número de objetos distintos do atributo "a" partindo-se da coleção C.	*count* (select distinct (a) from C in colection, a in C.a)	objetos
	card(C.a)	Cardinalidade média do atributo-coleção.	Para toda a instância de C, determine o número médio de atributos a em C.	objetos
	size(C.a)	Tamanho total do atributo-coleção.	select C.a from C	bytes
	sizeObject(C.a)	Tamanho médio de um elemento do atributo-coleção.	*size*(C.a)/*count*(C.a)	bytes
Custos:				
	firstAccess(C.a)	Custo médio do primeiro acesso ao atributo "a".	*firstAccess*(select C.a from C)	ms
	nextAccess(C.a)	Custo médio de um acesso sucessivo ao atributo "a".	*nextAccess*(select C.a from C)	ms
Informações:				
	parent(a)	Coleção que possui o atributo "a".	informado	nome da coleção
	type(C.a)	Tipo do atributo "a".	informado	nome do tipo
	cluster(C.a)	Padrão de agrupamento pelo atributo-coleção (por exemplo, agrupado pelo pai).	informado	nome
	isReference(C.a)	Identifica se o atributo "a" é do tipo referência (ponteiro).	informado	lógico

É importante observar que o projetista de uma fonte de dados pode não ter nomeado o CVA, cabendo à própria fonte ou ao integrador de esquemas (quando registrar esta informação no arquivo de publicação) realizar esta identificação.

4.3.4.3 MÉTODOS

O último subgrupo de dados sobre objetos refere-se a custos e estatísticas sobre métodos. No arquivo de publicação devem ser disponibilizadas as assinaturas dos métodos de uma fonte de dados, para que sejam medidos os respectivos tempos de primeira e próxima execução. O provedor DIG pode manter o histórico dos tempos medidos e disponibiliza o caso médio para o processador global de consultas. Este tratamento é bastante simplificado, mas a discussão sobre a execução de métodos ainda é muito recente.

É importante deixar claro o que consideramos como métodos: são métodos as operações que possuem o seu comportamento aplicado apenas ao objeto de uma coleção. Como exemplo temos o método *idade*() que, quando aplicado a um objeto (aluno.*idade*()), calcula a idade em função da data atual e do atributo que representa a data de nascimento. Esse conceito é diferente da execução de operações nas coleções e entre coleções. As operações nas coleções podem ser modeladas como operadores de consulta aplicados às coleções. Já as operações entre coleções devem ter seus custos estimados no processador global da consulta, como é apresentado em BOUGANIM *et al.* (2001).

Tabela 6. Dados sobre Métodos

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Custos:				
	exec(*method*)	Tempo de execução do método *method*.	*firstAccess*(assinatura informada)	ms
	rexec(*method*)	Tempo de reexecução do método *method*.	*nextAccess*(assinatura informada)	ms

4.3.5 DADOS SOBRE RELACIONAMENTOS

O conjunto de informações sobre os relacionamentos existentes entre as coleções é fundamental para determinar o custo de operadores como junções ou navegações de uma expressão de caminho. Os métodos de aquisição e estimativa destes dados são apresentados na Tabela 7.

Tabela 7. Dados sobre Relacionamentos

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	pointer(R.a,C)	Número total de ponteiros do atributo "a" da coleção R para objetos da coleção C (incluídas as réplicas).	*count* (select a from R)	ponteiros
	distinct-Pointer(R.a,C)	Número total de ponteiros distintos do atributo "a" da coleção R para objetos da coleção C.	*count* (select distinct (a) from R)	ponteiros
	null-Pointer(R.a,C)	Número total de objetos da coleção R cujos ponteiros (no atributo "a") para objetos da coleção C são todos nulos.	*count* (select a from R where a = NULL)	objetos
	fanout(R.a,C)	Número médio de ponteiros do atributo "a" da coleção R para objetos da coleção C (incluídas as réplicas).	*pointer*(R.a,C)/*card*(R)	valor
	share(R.a,C)	Número ponteiros do atributo "a" da coleção R que apontam para o mesmo objeto da coleção C (incluídas as réplicas).	*pointer*(R.a,C)/*distinct-Pointer*(R.a,C)	valor
	$Z_{R,C}$	Número médio de ponteiros distintos do atributo "a" de um objeto da coleção R para objetos da coleção C, considerando os objetos de R que possuem pelo menos uma referência não nula.	*distinctPointer*(R.a,C)/ ((*card*(C) – *null-Pointer*(R.a,C))	valor
Informações:				
	isClustered (R.a,C)	Indica se a coleção C está agrupada pelas referências de R através do atributo "a".	informado	lógico

Utilizamos o modelo de custo apresentado em RUBERG (2001) para identificarmos as principais estatísticas deste grupo, as quais são: o número total de ponteiros a partir de objetos de uma coleção para objetos de outra coleção; o número de ponteiros distintos a partir de objetos de uma coleção para outra; o número de objetos de uma coleção que não referenciam nenhum objeto; o número médio por objeto dos ponteiros de uma coleção para outra; e o número médio de ponteiros distintos a partir de

um objeto da coleção para outra, contando apenas os objetos da coleção de origem que possuem um ou mais referências.

4.3.6 DADOS SOBRE OPERADORES DE CONSULTA

Os operadores de consulta são as unidades básicas de um plano de execução de consulta, dentre os quais podemos destacar os operadores de seleção, de junção e de projeção (ELMASRI e NAVATHE, 1994). Em ambientes distribuídos, heterogêneos e com fontes autônomas, podemos encontrar os mais diferentes tipos operadores de consultas, representando programas, funções ou procedimentos.

De maneira geral, quatro informações devem ser adquiridas sobre um operador de consulta: i) se a fonte de dados possui a capacidade de realizá-lo; ii) quanto custa a execução deste operador; iii) qual a cardinalidade resultante; e iv) qual o volume de dados retornado. Observe que as três últimas informações dependem dos parâmetros utilizados para a execução do operador de consulta (coleções de entrada, predicados aplicados, etc.).

As estatísticas e custos sobre um operador de consulta podem ser obtidos por inspeção (medindo o resultado de cada execução do operador) ou pela aplicação de funções de estimativa. Essa última alternativa é a mais flexível e possui menor custo de aquisição. A cardinalidade resultante e o volume de dados retornado por um operador de consulta podem ser obtidos pelo fator de seletividade deste operador, sendo esse parâmetro a base da estimativa de custos de um plano de execução de consulta. Nós iremos apresentar alguns métodos que utilizamos para estimar os fatores de seletividade de operadores de seleção e de junção, considerando tanto a aplicação de funções matemáticas como também o uso de valores padrões. Estes valores padrões serão aplicados caso a fonte de dados não disponibilize as informações necessárias para calcular a seletividade do operador.

Na Tabela 8 são apresentados as funções utilizadas para estimar os fatores de seletividade do operador de **seleção** para predicados de igualdade e de desigualdade, considerando uma distribuição uniforme de valores.

Tabela 8. Fatores de Seletividade do Operador de Seleção

DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	PADRÃO	UNIDADE
$S_{sel}(a$=valor)	Seletividade da igualdade.	1/(*card*(C))	1/10	valor
S_{sel} (a >valor)	Seletividade da desigualdade.	(*max*(a) – valor)/ (*max*(a) – *min*(a))	1/3	valor
S_{sel} (a <valor)	Seletividade da desigualdade.	(valor – *min*(a))/ (*max*(a) – *min*(a))	1/3	valor

Os operadores de **junção** são os mais complexos em relação à estimativa de custos, pois as fontes de dados podem processá-los de diferentes maneiras. O cálculo dos fatores de seletividade do operador de junção é apresentado na Tabela 9.

Tabela 9. Fatores de Seletividade do Operador de Junção

DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	PADRÃO	UNIDADE
$S_{join}(a = \underline{b})$	Junção de duas coleções onde o atributo "a" é igual ao atributo "b" e "b" é chave da coleção C.	*card*(R_a) * *fanout*(R_a,C_b)	1/10	valor
S_{join} ($\underline{a} = b$)	Junção de duas coleções onde o atributo "a" é igual ao atributo "b" e "a" é chave da coleção R.	*card*(R_a) * *share*(C_b, R_a)	1/10	valor
S_{join} (a=b)	Junção de duas coleções onde o atributo "a" é igual ao atributo "b", onde nem "a" nem "b" são chaves.	*card*(R_a)**card*(S_b)	1/5	valor
S_{join} (a>b)	Seletividade da Junção com uma desigualdade.	(2**max*(a)-*max*(b)-*min*(b) – 1) / (2*(*max*(a)-*min*(a))	1/3	valor

A aplicação do operador de **projeção** pode diminuir sensivelmente o volume das coleções intermediárias geradas no processamento de uma consulta. Neste caso, o volume estimado do resultado intermediário corresponde ao produto do tamanho médio do atributo pela cardinalidade da coleção, conforme mostra a Tabela 10. Caso a fonte de dados não possua capacidade de realizar projeções, esse operador pode ser aplicado no mediador, onde as possibilidades de redução nos custos do processamento de consultas são bem mais restritas.

Tabela 10. Custos do Operador de Projeção

	DADO	DESCRIÇÃO	FORMA DE AQUISIÇÃO	UNIDADE
Estatísticas:				
	sizeProjection(C.a)	Volume resultante da projeção do atributo "a" da coleção C.	*size*(a)**card*(C)	bytes
	cardProjection(C.a)	Cardinalidade resultante da projeção do atributo "a" da coleção C.	*card*(C.a)	objetos
Custos:				
	timeProjection(C.a)	Tempo para projetar o atributo "a" da coleção C.	medido	ms

4.4 CONSIDERAÇÕES FINAIS

A qualidade das informações disponibilizadas pelo DIG está diretamente ligada à qualidade das descrições presentes no arquivo de publicação das fontes de dados. Isto exige que as consultas especificadas para a aquisição sejam as mais representativas possíveis. Assim, faz-se necessário na confecção do arquivo de publicação um bom conhecimento sobre a fonte de dados.

Os métodos *firstAccess*() e *nextAccess*() estão presentes em vários grupos de custos e estatísticas do DIG. Eles permitem reconhecer se uma fonte de dados possui um *cache* que torne mais eficiente a execução "quente" de consultas ou métodos. O método *firstAccess*() irá medir o tempo da primeira execução de uma consulta ou método, quando o *cache* de dados está vazio (execução "fria"). O método *nextAccess*() irá refletir o tempo que a consulta ou método gasta após uma execução prévia, isto é, em uma execução "quente". Assumimos que a medição destes parâmetros é realizada em tempo total gasto desde a solicitação da operação até o retorno do resultado, em uma abordagem de "caixa-preta".

Para a realização das aquisições de custos e estatísticas, o DIG espera que o tradutor ou a fonte de dados disponibilize algumas métodos especiais. Alguns destes métodos são tradicionalmente encontrados em fontes de dados, como o *count*(), o *size*(), o *distinct*(), o *min*() e o *max*(). Porém, caso isso não ocorra, esses métodos (e os mais específicos que sejam necessários) poderão ser implementados no tradutor de consultas, ou ainda no próprio coletor DIG .

A confecção do catálogo do DIG foi bastante detalhada, o que permite que mediadores com modelos de custo mais completos sejam atendidos. A abrangência do catálogo do DIG decorre do estudo de diversos modelos de custos, como: WANG (2001), onde os aspectos dos SGBDOOs e SGBDs relacionais foram incorporados; RUBERG (2001), onde os aspectos sobre os relacionamentos entre as coleções, estimativa de expressões de caminho e fator de seletividade de junções foram observados; BOULOS e ONO (1999), que fundamentaram o processo de aquisição de custos sobre métodos; e BOUGANIM *et al.* (2001), que destacam a necessidade de estimativas sobre funções para processo de otimização de consultas.

O estudo desses modelos de custo teve o intuito de produzir um catálogo bastante completo. Porém, novos custos ou estatísticas podem ser necessários para um mediador específico. Para tanto, a arquitetura DIG foi concebida de forma a incorporar facilmente as mudanças que venham a ser necessárias no catálogo de custos. É importante observar que, apesar da notação do catálogo utilizar termos de objetos (como ponteiros, coleções, métodos, etc.), estas estruturas representam conceitos que são encontrados e podem ser mapeados nos diversos modelos de dados de MQSs, sejam eles relacionais, orientados a objetos ou até mesmo semi-estruturados.

Capítulo 5

O PROTÓTIPO DO DIG

Para implementar, analisar e validar a arquitetura proposta, desenvolvemos um protótipo do sistema DIG. No processo de validação foi utilizado um sistema mediado de consultas para uma aplicação bancária real, baseado na arquitetura HIMPAR, com dois tipos de fontes de dados: um SGBD baseado em objetos (GOA) e um middleware de consulta sobre dados relacionais (LeSelect).

5.1 INTRODUÇÃO

As funcionalidades do sistema DIG foram implementadas em um protótipo que visa verificar a viabilidade da arquitetura proposta através da análise de aspectos como: a facilidade de integração dos coletores DIG aos tradutores de consultas; a adaptabilidade dos métodos de aquisição de estatísticas e custos às diversas fontes de dados; e a integração do sistema DIG com um processador global de consultas.

Antes mesmo da codificação e validação experimental da arquitetura proposta, foi necessário definir, configurar e adequar um ambiente para o processamento distribuído de consultas onde o funcionamento do sistema DIG pôde ser analisado. Para tanto, consideramos a especificação da arquitetura de integração HIMPAR (PIRES, 1999) aplicada a fontes representadas pelo SGBD GOA (MAURO *et al.*, 1999, MATTOSO *et al.*, 2000, GOA, 2002) e pelo *middleware* de consulta *LeSelect* (SIMON, 2001).

O protótipo desenvolvido para o sistema DIG possui uma infra-estrutura de comunicação flexível, baseada em objetos CORBA, para a definição das interfaces entre o provedor do catálogo de serviços e os coletores de custos e estatísticas. Para minimizar os efeitos da heterogeneidade na evolução do DIG, adotamos o uso de padrões que simplificaram a construção dos coletores e facilita a portabilidade destes componentes para um maior número de plataformas. Um exemplo desses padrões no protótipo do DIG é a STL (*Standard Template Library*) (STL, 2002), uma biblioteca pública da linguagem de programação C++ que foi utilizada para a construção das estruturas de dados.

Considerando a complexidade e o grande número de funcionalidades da arquitetura proposta, concentramos nosso trabalho de validação das características mais significativas da especificação do sistema DIG. O escopo dos testes de validação foi orientado pela identificação de um conjunto mínimo de funcionalidades da arquitetura do DIG, que abrange:

- ✓ o registro das diversas fontes de dados que participam do sistema;
- ✓ a identificação dos métodos de aquisição a serem aplicados em cada fonte;
- ✓ a coleta propriamente dita dos custos e estatísticas;

- ✓ o controle do processo de aquisição de dados, reunindo-os em uma base persistente devidamente organizada; e
- ✓ a publicação das informações coletadas pelo sistema DIG.

Estas funcionalidades estão distribuídas entre os dois componentes básicos da arquitetura do DIG: o provedor e o coletor. Ao coletor DIG cabe o registro das fontes de dados que são monitoradas e a interpretação do arquivo de publicação dessas fontes, além da implementação das funções de aquisição dos respectivos custos e estatísticas. As funcionalidades esperadas no provedor DIG são relativas ao controle dos coletores, ao armazenamento e à publicação dos dados adquiridos. Através do provedor DIG, o processo de coleta pode ser disparado por um único comando.

Em relação à interface entre o coletor e o provedor, no protótipo do DIG foram implementados objetos CORBA que correspondem às informações modeladas no catálogo de serviços do DIG. Verificamos também que esta interface suporta alterações em sua estrutura, como o acréscimo de novas estatísticas, de forma bastante simples e sem grande impacto no restante da arquitetura.

A partir da definição do conjunto básico de funcionalidades do sistema DIG, nós exploramos um cenário de teste com diversos aspectos relativos à distribuição, heterogeneidade e autonomia. O restante deste capítulo detalha o projeto e a validação do sistema DIG.

5.2 CENÁRIO DE VALIDAÇÃO

A validação da arquitetura proposta neste trabalho requer a definição de um ambiente de teste que abrange um sistema mediado de consulta e diferentes fontes de dados distribuídos. Neste contexto, adotamos a arquitetura HIMPAR (*Heterogeneous Interoperable Mediators and Parallel Architecture*), um sistema mediado de consulta cujo protótipo vem sendo desenvolvido na COPPE, como a plataforma de integração de informação onde o protótipo DIG poderá ser avaliado quanto às suas funcionalidades. As fontes de dados utilizadas no cenário de validação foram o SGBD GOA e *middleware* de consulta *LeSelect*.

O sistema mediado de consultas HIMPAR consiste em uma arquitetura aberta e extensível que visa a interoperabilidade de fontes de dados em ambientes distribuídos e

heterogêneos. A arquitetura HIMPAR é baseada em padrões existentes na área de tecnologia de objetos. A integração semântica das fontes de dados utiliza um modelo baseado em objetos como modelo canônico de representação dos dados. Deste modo, as consultas podem ser escritas na linguagem OQL. Uma característica interessante desta arquitetura é a utilização de objetos CORBA como mecanismo de comunicação entre o mediador e os tradutores. O principal objetivo da utilização da plataforma CORBA na arquitetura HIMPAR é simplificar a confecção de novos tradutores e, consequentemente, facilitar a integração de diversas fontes de dados. A Figura 10 exibe a arquitetura HIMPAR em uma configuração com tradutores para o SGBD GOA e para o *middleware LeSelect*.

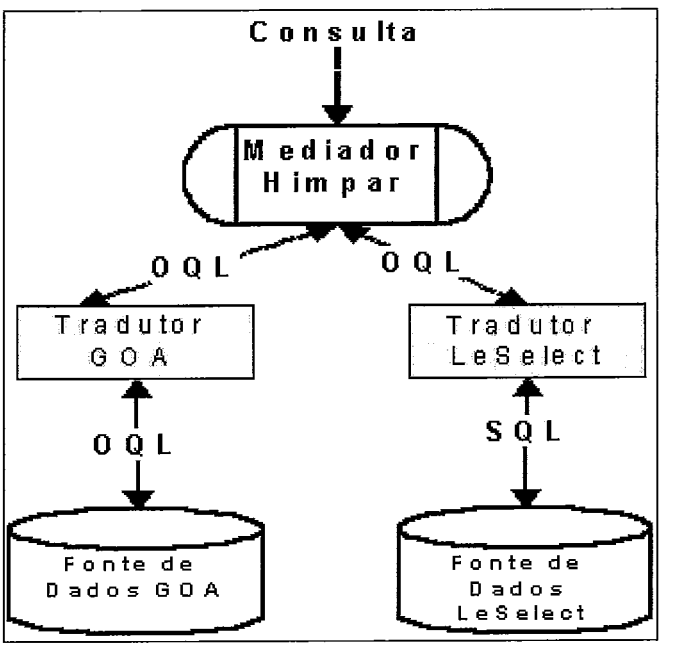

Figura 10. Arquitetura HIMPAR com dois tipos de fontes de dados

Implementamos no protótipo da arquitetura HIMPAR um tradutor específico para fontes de dados representadas pelo GOA. O SGBD GOA fornece serviços de persistência e de gerência sobre coleções de objetos. Esta gerência compreende funcionalidades como o processamento consultas, a definição de políticas para agrupamento de objetos em disco e a gerência de *cache*. O GOA utiliza a linguagem OQL para o processamento de consultas e a ODL para a definição dos esquemas de dados, ambas de acordo com o padrão ODMG (CATTEL *et al.*, 2000).

O nosso objetivo é avaliar o comportamento da arquitetura do DIG em um ambiente distribuído e heterogêneo. Por isso, para permitir que outras fontes de dados

estivessem disponíveis na arquitetura HIMPAR, nós construímos um tradutor específico para o *middleware* de consulta *LeSelect*. O *LeSelect* permite um acesso uniforme a dados heterogêneos, abrangendo diversos tipos de fontes de dados, como arquivos textos, arquivos XML e fontes JDBC. Além disso, o *LeSelect* oferece recursos para a execução de programas sobre as fontes de dados distribuídos. Os dados são disponibilizados no modelo relacional e a linguagem de consulta é SQL.

5.3 IMPLEMENTAÇÃO DO DIG

O ambiente utilizado para o desenvolvimento do protótipo do DIG foi o compilador *Borland* C++ *Builder* versão 5.0 e o ORB 4.0 da *VisiBroker*. A projeto de implementação de um protótipo para a arquitetura do DIG envolveu quatro atividades principais:

1) extensão e atualização da arquitetura HIMPAR;
2) adequação do tradutor do SGBD GOA para uma versão distribuída;
3) construção de um tradutor para o *middleware LeSelect*; e
4) codificação do protótipo do DIG.

Estas atividades são descritas e analisadas a seguir.

5.3.1 INFRA-ESTRUTURA

As três etapas iniciais do projeto do DIG foram requisitos para garantir uma infra-estrutura mínima para a validação do protótipo do DIG em um ambiente de processamento de consultas com fontes distribuídas, heterogêneas e autônomas. Nosso primeiro passo, visando utilizar a arquitetura HIMPAR na geração do protótipo do DIG, foi a atualização da versão do ORB do protótipo da HIMPAR, que se encontrava em uma versão incompatível com a plataforma de desenvolvimento disponível. Deste modo, realizamos uma migração de versão da HIMPAR para o ORB 4.0 da *Visiobroker*, efetuando os ajustes necessários nessa adequação.

O segundo passo do nosso projeto consistiu em definir e implementar o tradutor para o SGBD GOA. A versão do GOA utilizada na implementação da HIMPAR era antiga e acoplada ao tradutor. Ou seja, o tradutor deveria ser compilado com o GOA.

Nós atualizamos a versão do GOA, separamos as funcionalidades referentes à tradução e substituímos as chamadas ao fonte do GOA no tradutor por chamadas a uma API. Esta API permite o acesso via *socket* a uma base GOA. Desta forma, o tradutor GOA da arquitetura HIMPAR ficou mais isolado das atualizações realizadas no SGBD GOA.

Tendo em vista a validação da arquitetura do DIG em um ambiente distribuído e heterogêneo, construímos um tradutor para o *middleware LeSelect* na arquitetura HIMPAR. A facilidade com que o *LeSelect* permite o acesso a dados em formato texto foi o grande atrativo na escolha deste *middleware* para construção de um segundo tradutor para a HIMPAR. O *LeSelect* possui diversas interfaces de comunicação, sendo escolhida a interface de comunicação via *socket* com o tradutor, mantendo o acesso uniforme aos tradutores da HIMPAR.

A linguagem de consulta do *LeSelect* é a SQL, por isso foi necessária a construção de um *parser* de conversão entre a SQL e a OQL, que é a linguagem canônica da arquitetura HIMPAR. Implementamos este *parser* de conversão utilizando as ferramentas FLEX e BISON. A decisão de utilizar estas ferramentas foi motivada por várias razões, dentre as quais destacamos as seguintes: estas ferramentas são altamente difundidas na construção de analisadores léxicos e sintáticos; o GOA utiliza o FLEX e o BISON na especificação do seu *parser* de tradução da linguagem OQL para as estruturas internas do SGBD; e a facilidade de expansão que estas ferramentas oferecem. Um outro motivo relevante deve-se ao fato de estas ferramentas também terem sido utilizadas para a especificação de outros componentes do protótipo do DIG, conforme veremos a seguir.

5.3.2 CODIFICAÇÃO DO PROTÓTIPO

Com uma infra-estrutura básica disponível, a etapa seguinte do projeto consistiu na implementação do protótipo do DIG. Acompanhando o desenvolvimento da HIMPAR e do GOA, o protótipo do DIG foi implementado na linguagem de programação C++. A adoção da biblioteca padrão STL na construção das estruturas de dados intermediárias da arquitetura e a utilização da plataforma CORBA visaram propiciar a rápida evolução do protótipo do DIG e favorecer a interoperabilidade do sistema.

A construção dos módulos componentes da arquitetura do DIG em um protótipo teve a seguinte seqüência de atividades: implementação do coletor DIG; especificação da interface entre coletor e provedor; e implementação do provedor DIG. Primeiramente, codificamos a Unidade de Registro do coletor, cuja responsabilidade é ler o arquivo de publicação das fontes de dados e registrar os métodos de aquisição das estatísticas e custos publicados. Para esta unidade, nós construímos um *parser* XML que verifica e decompõe os elementos do arquivo de publicação, permitindo extrair as descrições dos métodos de aquisição. De maneira análoga ao GOA e à HIMPAR, esse *parser* foi implementado através das ferramentas FLEX e BISON, permitindo que novas definições para o arquivo de publicação fossem facilmente incorporadas ao protótipo do DIG.

Ainda na implementação do coletor DIG, nós mapeamos em classes de objetos a visão de aquisição dos dados que foi especificada no catálogo de serviços do DIG. Assim, para cada objeto de custo definido na especificação do DIG, obtivemos uma classe implementada no coletor. Estas classes possuem, além dos atributos especificados inicialmente, métodos para a carga das descrições sobre os métodos de aquisição de dados que são fornecidos no arquivo de publicação e métodos para a aquisição propriamente dita das estatísticas e custos.

Na Figura 11 temos a especificação da classe **`collection`** no coletor DIG. As seguintes observações podem ser feitas no código apresentado. Primeiro, adotamos o uso da STL para a definição dos tipos "cadeia de caracteres" (*string*) e vetor (*vector <attribute*>*) na lista de atributos da classe. Esses tipos facilitam a codificação da interface das classes. Segundo, vale ressaltar o método **`load()`**, responsável pela leitura do arquivo de publicação das fontes de dados que foram cadastradas na arquitetura. Destacamos também o objeto CORBA `TCollection` que é retornado após a execução do método **`acquireCosts()`** e representa uma estrutura a ser preenchida com os dados coletados. Observe que este método tem como parâmetro de entrada um objeto CORBA correspondente ao tradutor da fonte de dados ao qual o coletor DIG está associado.

```
class collection
{
  private:
  protected:
    // Collection name
    string name;
    // Acquisition
    string acquireCardinality;
    string acquireSize;
    string acquireDomain;
    string acquireFirstAccess;
    string acquireNextAccess;
    // Specific Methods and properties
    int cardinality;
    int size;
    int objectAverageSize;
    string domain;
    int firstAccess;
    int nextAccess;
    string cluster;
    vector <attribute*> attributeBag;
  public:
    collection();
    collection(char* s);
    ~collection();
    char *getName();
....// Main acquisition methods
    void load (string xml);
    Translator::TCollection* acquireCosts (GoaTranslator*
ptr_translator);
};
```

Figura 11. Classe `collection` no coletor DIG

Um aspecto interessante da interface entre o coletor e o provedor do sistema DIG é que cada objeto especificado na visão de aquisição do coletor corresponde (quase diretamente) a um objeto CORBA na interface. Portanto, para a especificação da entidade *Collection* temos um objeto CORBA **`TCollection`**, cuja definição em IDL é apresentada na Figura 12.

```
struct TCollection
  {
    string name;
    short cardinality;
    long size;
    short objectAverageSize;
    string domain;
    short firstAccess;
    short nextAccess;
    string cluster;
    TattributeList attributeBag;
  };
```

Figura 12. Classe *TCollection* da interface entre provedor e coletor DIG

É importante observar que a especificação em C++ da classe `collection` e da IDL `TCollection` são equivalentes, tendo apenas como diferença a forma de expressar os atributos.

Na codificação do provedor DIG foram desenvolvidos: a Unidade de Controle dos Coletores (UCC), que inclui um módulo de registro dos coletores; e a Unidade de Persistência de Dados coletados (UPD). No módulo de registro da UCC é possível definir quais coletores serão acessados no processo de aquisição dos custos, gerando uma lista de coletores ativos. Através dessa lista, a UCC solicita que cada coletor dispare o processo de aquisição de custos e estatísticas, cuja ativação é realizada através de um método chamado **`acquireCosts()`**. Este método pode ser entendido como um correspondente global do método homônimo no coletor DIG. As informações coletadas pelo sistema DIG são reunidas pelo provedor e armazenadas em arquivos de texto formatado em XML através de funções que gerenciam a UPD. Todas estas operações podem ser solicitadas através de uma interface de comandos.

5.4 O EXPERIMENTO

Para validar o protótipo do DIG, montamos um estudo de caso utilizando bases reais de informações sobre empréstimos de um banco de desenvolvimento. Estas bases foram distribuídas na arquitetura HIMPAR através dos tradutores construídos para o SGBD GOA e o *middleware LeSelect*. Um objetivo suplementar deste experimento é exemplificar como ocorrem a configuração, a ativação da coleta de informações e o acesso aos dados da arquitetura do DIG, conforme veremos nas próximas seções.

5.4.1 BASE DE TESTE

As bases de informações utilizadas no nosso experimento possuem um esquema de dados baseado no Sistemas de Controle de Contratos (SCC) do Banco Nacional de Desenvolvimento Econômico e Social – BNDES. Dentre as entidades da base de empréstimos do SCC, nós utilizamos as seguintes: "Empresa", "Contrato" e "Parâmetro" (vide Figura 13).

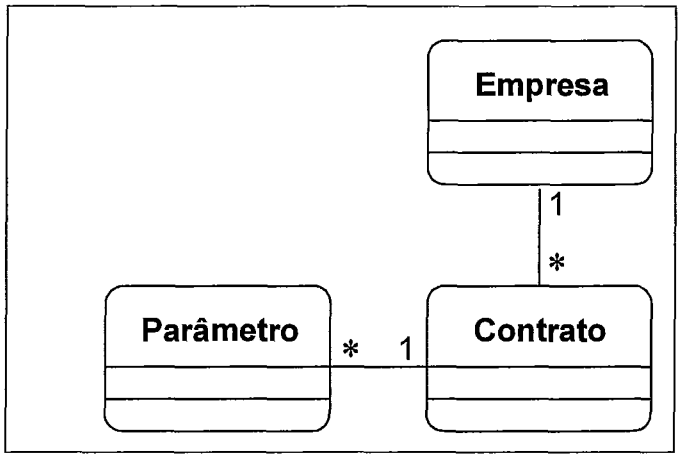

Figura 13. Diagrama UML simplificado da base de empréstimos

A entidade "Empresa" representa uma instituição financeira que adquiriu um empréstimo junto ao BNDES. A entidade "Contrato" identifica um dado empréstimo feito a uma empresa, para o qual são registradas informações como: a data da contratação, a que linha de financiamento pertence o contrato, quem é o mutuário final, etc. Por fim, a entidade "Parâmetro" identifica o fluxo financeiro realizado para o pagamento do empréstimo. Um parâmetro possui uma data de início e término do fluxo financeiro, a periodicidade do pagamento das prestações, qual a fórmula para o cálculo de cada prestação e o código do evento financeiro ao qual o parâmetro está associado.

A base de empréstimos utilizada no nosso experimento está distribuída em dois tipos de fontes de dados: no SGBD GOA e em arquivos de texto com campos separados por vírgulas. Estes arquivos de texto são integrados pelo *middleware LeSelect*. A distribuição das fontes de dados é exibida na Figura 14. Observe que não cabe ao DIG realizar a integração das informações entre as fontes de dados, resolvendo conflitos como os encontrados em relação à entidade "Contratos". Esta tarefa de integração é realizada pelo mediador da HIMPAR. Entretanto, o DIG pode coletar, além dos custos e estatísticas de cada fonte, a estrutura dos dados armazenados, facilitando assim a tarefa de integração.

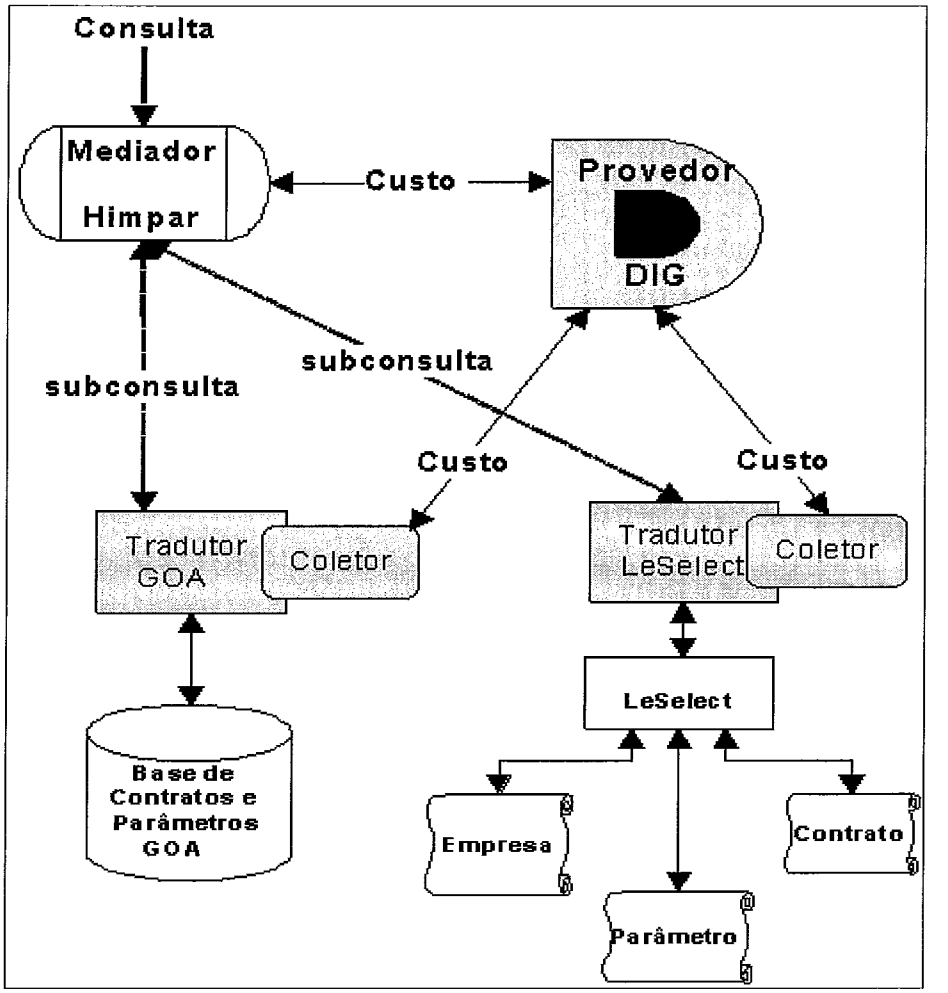

Figura 14. Cenário para validação do protótipo do DIG

Note que a base de teste possui um esquema simplificado. Entretanto, a configuração do ambiente experimental que utilizamos é bastante rica de fatores relacionados à: distribuição dos dados; heterogeneidade das estruturas de dados, das ferramentas e das plataformas computacionais, como os gerentes de dados, as linguagens de consulta e os modelos de representação; e à autonomia das fontes de dados. A simplificação do esquema utilizado na base de teste permite que a nossa análise experimental seja focalizada na complexidade do ambiente tratado e, por conseqüência, na interoperabilidade da arquitetura proposta.

5.4.2 ADESÃO DAS FONTES DE DADOS

A primeira configuração necessária para a realização dos testes de validação do protótipo é a adesão das fontes de dados à arquitetura do DIG. Esta adesão é representada pela confecção de arquivos de publicação contendo os métodos de coleta

para os parâmetros a serem adquiridos em cada fonte de dados. Contudo, o registro efetivo das fontes de dados que publicam suas estatísticas e custos para a arquitetura do DIG é feito através do cadastro dos respectivos tradutores em um arquivo específico, a ser lido pela Unidade de Controle dos Coletores (UCC), conforme é exibido na Figura 15. Cada entrada desse arquivo representa uma porta de acesso (via *socket*) ao coletor DIG de uma fonte de dados.

```
Localhost:base1.goa
Localhost:base2.leselect
```

Figura 15. Arquivo de registro dos coletores que participam do sistema DIG

No exemplo da Figura 15 são registrados os coletores para duas fontes de dados. Todavia, nada impede que seja registrado um número maior de fontes de diferentes tipos (por exemplo, `site1:base3.leselect`, `site2:base4.goa`, e outras).

```
<Site name="porto">
  <Collection name="Parametros">
    <SimpleAttribute name="periodicidade">
      <Distinct>"select count(distinct p.periodicidade) from p in
Parametros"</Distinct>
      <Minimum>"select min(p.periodicidade) from p in
Parametros"</Minimum>
      <Maximum>"select max(p.periodicidade) from p in
Parametros"</Maximum>
    </SimpleAttribute>
    <Cardinality>"select count(p) from p in Parametros"</Cardinality>
    <Size>"size(select p from p in Parametros)"</Size>
    <Averagesize>"div(size(select p from p in Parametros),
                  select count(p) from p in
Parametros)"</Averagesize>
    <FirstAccess>"select first p from p in Parametros"</FirstAccess>
    <NextAccess>"select first p from p in Parametros"</NextAccess>
 </Collection>
...
 <Relationship name="fluxo">
    <FirstCollection>Parametros</FirstCollection>
    <SecondCollection>Contratos</SecondCollection>
    <RelationAttribute>fluxo</RelationAttribute>
    <NumberOfPointers>number_of_pointers(fluxo)</NumberOfPointers>
    <NumberOfDistinctPointers>number_of_distinct_pointers(fluxo)
</NumberOfDistinctPointers>
    <NumberOfNullPointers>number_of_null_pointers(fluxo)
</NumberOfNullPointers>
 </Relationship>

 <NetworkTime>10</NetworkTime>
 <PageSize>1024</PageSize>
 <AbleToProject>True</AbleToProject>
</Site>
```

Figura 16. Parte do arquivo de publicação do SGBD GOA

Partes dos arquivos gerados para o ambiente de validação do DIG, abrangendo respectivamente fontes GOA e *LeSelect*, são mostradas na Figura 16 e na Figura 17. Esses arquivos de publicação foram gerados conforme um DTD específico. Na Figura 16 nós podemos verificar a presença de algumas funções externas à linguagem de consulta que são utilizadas para a estimativa de estatísticas e custos, como é o caso da função `number_of_pointers` na descrição de um parâmetro do relacionamento "fluxo". Estas funções são implementadas no próprio coletor DIG.

```
<Site name="miami">
  <Collection name="Empresas">
    <SimpleAttribute name="sequencial">
      <FirstAccess>"select first e.sequencial from e in Empresas"
</FirstAccess>
      <NextAccess>"select first e.sequencial from e in Empresas"
</NextAccess>
    </SimpleAttribute>
    <Cardinality>"select count(e) from e in Empresas"</Cardinality>
    <Size>"size(select e from e in Empresas)"</Size>
    <Averagesize>"div(size(select e from e in Empresas),
                  select count(e) from e in Empresas)"</Averagesize>
    <Key>cgc</Key>
    <FirstAccess>"select first e from e in Empresas"</FirstAccess>
    <NextAccess>"select first e from e in Empresas"</NextAccess>
  </Collection>

  <Collection name="Contratos">
    <SimpleAttribute name="tipo_contrato">
    <Size>"size(select c from c in Contratos)"</Size>
    <Averagesize>"div(size(select c from c in Contratos),
                  select count(c) from c in Contratos)"</Averagesize>
...
 </Collection>
...
</Site>
```

Figura 17. Parte do arquivo de publicação do arquivo texto (*LeSelect*)

Note que, na Figura 17, os métodos para aquisição de estatísticas e custos são implementados pelo tradutor de consultas, uma vez que a fonte de dados representa um arquivo de texto. Além disso, a linguagem de consulta do tradutor no *LeSelect* é a SQL, que possui sintaxe e modelo de dados diferentes da linguagem de consulta da HIMPAR (OQL). Para que os métodos de aquisição a serem submetidos pelos coletores DIG fossem descritos através da linguagem canônica de consulta da arquitetura de mediação HIMPAR, foi necessário o desenvolvimento de um conversor OQL→SQL, conforme mencionado anteriormente. Deste modo, obtivemos uma maior uniformidade na declaração dos arquivos de publicação do DIG.

5.4.3 AQUISIÇÃO DE DADOS

Uma vez cadastradas as fontes de dados da arquitetura do DIG, destacamos dois eventos principais no processo de aquisição de estatísticas e custos, os quais são disparados a partir do provedor DIG: (i) o **registro das fontes de dados**, através da interpretação dos arquivos de publicação para cada coletor declarado para a arquitetura; e (ii) a **coleta de estatísticas e custos**, com povoamento da Unidade de Persistência de Dados. O primeiro evento é disparado através do provedor DIG chamando-se o método `load()` para cada coletor cadastrado. Esse método pode ser invocado, por exemplo, por um processo monitor dedicado que seja capaz de reconhecer quando uma nova fonte é adicionada à arquitetura. No nosso experimento, o registro das fontes de dados ocorre sempre que o provedor DIG é iniciado.

O evento (ii) é acionado no provedor DIG pelo método `acquireCosts()`. Observe que o processo de aquisição de estatísticas e custos do sistema DIG é realizado de maneira independente do processador global de consultas, sendo interessante que essa aquisição ocorra em um momento anterior ao da otimização de consultas, para que já existam dados disponíveis no catálogo do DIG. No experimento realizado, cada aquisição gerou um conjunto de dados formatados em XML, os quais foram arquivados em documentos do tipo texto.

A aquisição de custos e estatísticas pode ser realizada por inspeção dos dados ou por estimativa através de funções matemáticas aplicadas a estatísticas previamente adquiridas. Quando a aquisição é realizada por inspeção dos dados, um método ou consulta é realizado na fonte de dados e os valores medidos correspondem às estimativas dos parâmetros. Por exemplo, a aquisição do parâmetro "`Cardinality`" da coleção "`Parametros`" no SGBD GOA é feita por inspeção, conforme pode ser observado no método de aquisição descrito na Figura 16. Para esse parâmetro, o coletor DIG executa uma consulta que varre todos os objetos da coleção para contabilizar a respectiva cardinalidade. Para adquirir parâmetros relacionados a operadores de consulta, pode ser empregado o método da "caixa-preta", onde consultas são submetidas para medir volume, cardinalidade e tempo de processamento do operador.

Alguns parâmetros, ao invés de ter sua estimativa realizada através de inspeção, podem ser calculados matematicamente. Através de funções de custo, podemos estimar

um parâmetro utilizando estatísticas e custos medidos previamente. Por exemplo, na Figura 17, o tamanho médio do atributo ("`AverageSize`") da coleção "`Empresas`" pode ser obtido pela divisão do tamanho da coleção pela cardinalidade da mesma. No desenvolvimento do protótipo do DIG as estimativas dos operadores foram calculadas conforme apresentado no Capítulo 4.

5.4.4 PUBLICAÇÃO DOS DADOS COLETADOS

A publicação das estatísticas e custos coletados pelo DIG é uma questão chave para a arquitetura proposta. Por isso, considerando que o DIG visa suportar diferentes arquiteturas para o processamento distribuído de consultas, o catálogo de informações deve ser implementado em tecnologias que favoreçam a interoperabilidade. O formato XML oferece consideráveis vantagens para essa tarefa, dentre as quais destacamos a flexibilidade para representação e a portabilidade deste formato (especificado em texto puro).

A Figura 18 apresenta algumas das medições realizadas sobre a base armazenada no SGBD GOA, conforme os métodos de aquisição mostrados no respectivo arquivo de publicação. Podemos destacar o elemento *Relationship*, que traz informações sobre as referências existentes entre as coleções "Parâmetros" e "Contratos" através do relacionamento "fluxo". Além disso, observe que diferentes coleções podem ter diferentes parâmetros de estatísticas e custos a serem coletados.

No protótipo do DIG, a Unidade de Persistência de Dados foi implementada como um gerador de documentos XML. O acesso aos dados coletados é realizado preferencialmente através de funções implementadas no provedor DIG, que podem realizar algumas transformações sobre esses dados. Entretanto, é possível realizar o acesso direto (somente para leitura) aos documentos gerados pela aquisição de dados.

```
<Site name="porto">
  <Collection name="Parametros">
    <SimpleAttribute name="periodicidade">
      <Distinct>5</Distinct>
      <Minimum>0</Minimum>
      <Maximum>11</Maximum>
    </SimpleAttribute>
    <Cardinality>335</Cardinality>
    <Size>24792</Size>
    <AverageSize>74</AverageSize>
    <FirstAccess>24</FirstAccess>
    <NextAccess>20</NextAccess>
  </Collection>

  <Collection name="Contratos">
    <SimpleAttribute name="tipo_contrato">
      <Distinct>6</Distinct>
      <Minimum>0</Minimum>
      <Maximum>9</Maximum>
    </SimpleAttribute>
    <Key>num_contrato</Key>
    <Cardinality>572</Cardinality>
    <Size>62486</Size>
    <AverageSize>109</AverageSize>
    <FirstAccess>18</FirstAccess>
    <NextAccess>15</NextAccess>
  </Collection>

  <Relationship name="fluxo">
    <FirstCollection>Parametros</FirstCollection>
    <SecondCollection>Contratos</SecondCollection>
    <RelationAttribute>fluxo</RelationAttribute>
    <NumberOfPointers>572</NumberOfPointers>
    <NumberOfDistinctPointers>572</NumberOfDistinctPointers>
    <NumberOfNullPointers>54</NumberOfNullPointers>
  </Relationship>

  <NetworkTime>10</NetworkTime>
  <PageSize>1024</PageSize>
  <AbleToProject>TRUE</AbleToProject>
</Site>
```

Figura 18. UPD (formato XML) de um SGBD baseado em objetos

Na Figura 19, as medições são relativas às aquisições sobre o *middleware* de consulta *LeSelect*, cuja base está armazenada em arquivos de texto. Neste caso, por tratar-se de uma fonte de dados sem os recursos de um SGBD, não existem relacionamentos explícitos entre as coleções. É interessante ressaltar também os tempos de acesso medidos nesta fonte de dados. Os tempos do primeiro acesso e do próximo acesso são idênticos, evidenciando a falta de um *cache* de dados.

```
<Site name="miami">
  <Collection name="Empresas">
    <SimpleAttribute name="sequecial">
      <FirstAccess>8</FirstAccess>
      <NextAccess>8</NextAccess>
    </SimpleAttribute>
    <Cardinality>499</Cardinality>
    <Size>42826</Size>
    <AverageSize>85</AverageSize>
    <Key>cgc</Key>
    <FirstAccess>15</FirstAccess>
    <NextAccess>15</NextAccess>
  </Collection>

  <Collection name="Contratos">
    <SimpleAttribute name="tipo_contrato">
      <Distinct>6</Distinct>
      <Size>1</Size>
      <FirstAccess>10</FirstAccess>
      <NextAccess>10</NextAccess>
      <Minimum>0</Minimum>
      <Maximum>9</Maximum>
    </SimpleAttribute>
    <Cardinality>572</Cardinality>
    <Size>62486</Size>
    <AverageSize>109</AverageSize>
    <FirstAccess>18</FirstAccess>
    <NextAccess>18</NextAccess>
  </Collection>

  <NetworkTime>10</NetworkTime>
  <AbleToProject>FALSE</AbleToProject>
</Site>
```

Figura 19. UPD (formato XML) de texto

5.4.5 ADAPTABILIDADE DO DIG

Um aspecto importante da arquitetura do DIG é a adaptabilidade do sistema às mudanças ocorridas nas fontes de dados, que em geral são freqüentes em ambientes distribuídos, heterogêneos e com fontes autônomas. Tal adaptabilidade reflete o nível de transparência que o sistema DIG oferece para o processador global de consultas em relação à heterogeneidade do ambiente. Podemos identificar dois tipos básicos de mudanças nas fontes de dados de um sistema mediado de consulta: inserção ou remoção de uma fonte de dados no ambiente; e alteração do esquema de dados ou dos operadores de consulta suportados em uma determinada fonte. No primeiro caso, a configuração do DIG é bastante simples, pois é necessário apenas que seja incluído ou removido o registro da fonte de dados em um arquivo de configuração. Para inserir uma fonte, é necessário também criar o respectivo arquivo de publicação.

Podem ocorrer mudanças nas fontes de dados, tais como uma alteração de esquema (por exemplo, a criação de uma coleção). Para refletir estas mudanças, é necessário apenas atualizar no arquivo de publicação a estrutura alterada. Eventualmente, para aquisição de um dado parâmetro, a fonte de dados pode não ter a capacidade de realizar a coleta deste, fazendo-se necessário acrescentar esta funcionalidade ao tradutor. Por exemplo, para realizar a aquisição da cardinalidade de uma coleção espera-se que a fonte de dados implemente o método *count*(). Caso a fonte de dados não disponha deste método, é necessário que o tradutor implemente esta funcionalidade.

Além das mudanças ocorridas nas fontes de dados, também sofrem alterações os requisitos de informação do processador global de consultas. Ou seja, são necessários (no mediador) parâmetros não previstos no catálogo do DIG. Ou ainda, um novo processador global de consultas (com um modelo de custos próprio) pode ser instalado no sistema. Para atender a novos requisitos desse tipo, deve-se adicionar a *tag* dos novos parâmetros no DTD do arquivo de publicação. Assim, é preciso acrescentar a *tag* no dicionário do *parser* de importação do arquivo de publicação e nas classes de interface (tanto entre coletores e provedores DIG, quanto no catálogo de serviços). Essa mudança não é complexa, devido à modularidade da arquitetura e ao baixo acoplamento entre os módulos do DIG. Entretanto, esta alteração exige que sejam recompilados os coletores e os provedores DIG. Uma alternativa para minimizar o impacto dessas mudanças seria definir *tags* genéricas que pudessem ser especificados pelo usuário em um dicionário configurável (arquivo externo ao *parser*).

5.5 ANÁLISE EXPERIMENTAL

Através de uma análise experimental, pudemos verificar que o protótipo do DIG realizou adequadamente a aquisição de custos, estatísticas e informações, apesar das várias diferenças que foram exploradas nas fontes de dados. Desta forma, evidenciamos o potencial da arquitetura proposta no contexto da otimização de consultas em ambientes distribuídos, heterogêneos e com fontes autônomas. Além disso, verificamos que o acréscimo de novos parâmetros para aquisição pelo coletor DIG consiste em uma tarefa simples na maioria dos casos, fato que demonstra a flexibilidade da arquitetura do

DIG para a adesão de novas fontes de dados, bem como para a incorporação de futuras extensões ao catálogo do DIG.

Todas as funcionalidades de aquisição de custos e estatísticas do DIG estão encapsuladas em uma biblioteca específica, representada pelos arquivos `costs.h` e `costs.c`. Isolar estas funcionalidades em uma biblioteca separada tem o objetivo de simplificar o processo de integração dos coletores DIG aos tradutores de consulta. Assim, esse processo fica restrito à compilação da biblioteca do coletor junto com o tradutor e à publicação (na interface do tradutor) dos objetos CORBA correspondentes às classes de estatísticas e custos.

Destacamos que as informações coletadas pelo DIG contribuem fortemente para a geração de planos de execução de consultas eficientes. Por exemplo, suponha uma consulta envolvendo uma junção entre duas coleções armazenadas em nós distintos. Mesmo em uma abordagem de otimização heurística, o processador de consultas precisa dispor de informações básicas sobre as coleções envolvidas, como a cardinalidade dessas coleções, para decidir corretamente (em termos de eficiência) como a consulta global será decomposta e enviada para as fontes de dados. Em uma otimização baseada em custos, essas informações constituem uma condição *sine qua non* para a geração dos planos de execução de consulta.

Se o otimizador global de consulta não dispusesse do DIG, seria necessário implementar todas as funcionalidades para a aquisição dessas informações no mediador da HIMPAR. Uma implementação particular não teria a flexibilidade do DIG, e eventualmente viria a restringir a evolução do modelo de custos no otimizador global. Essa evolução é esperada, pois novas fontes autônomas e heterogêneas geralmente são agregadas à arquitetura durante a vida útil do sistema. Uma outra mudança, menos provável, que poderia acontecer no otimizador global de consultas seria a alteração do modelo canônico da arquitetura. Caso isto ocorra em um ambiente sem o DIG, todos os métodos de aquisição deverão ser reescritos. Através do DIG, bastaria alterar o arquivo de publicação das fontes de dados.

Capítulo 6

CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são registradas as considerações finais deste trabalho, com destaque para as contribuições apresentadas e algumas perspectivas para trabalhos futuros.

6.1 CONSIDERAÇÕES FINAIS

No contexto das arquiteturas modernas de banco de dados, é extremamente importante o suporte ao processamento eficiente de consultas em ambientes distribuídos, heterogêneos e com fontes de dados autônomas. Particularmente nesses ambientes, o quesito desempenho representa uma considerável barreira para a viabilização dos serviços de consulta distribuída.

Técnicas de otimização de consultas, em geral, utilizam parâmetros de custos e estatísticas para orientar o processador de consultas na escolha dos melhores planos de execução. Contudo, o processo de aquisição e tratamento destes parâmetros costuma ser assumido na literatura como uma tarefa implícita nas atribuições do processador global de consultas, limitando o catálogo de custos e estatísticas a uma arquitetura de sistema específica. Essa abordagem é adotada em sistemas distribuídos como o CORDS (ATTALURI *et al.*, 1995), o *Garlic* (GARLIC, 1999, HAAS *et al.*, 1997) e o DISCO (TOMASIC *et al.*, 1995). Um problema dessa abordagem consiste em conciliar a carga de trabalho do processador de consultas e a qualidade das informações de custo adquiridas, pois o tempo de otimização deve ser irrisório e a precisão dos custos e estatísticas é proporcional ao esforço empregado no processo de aquisição e tratamento destes dados. Isso se agrava especialmente em ambientes como a *Web*, onde o processo de aquisição de custos envolve um número maior de parâmetros e requer métodos específicos para coleta de dados em fontes restritas.

A nossa proposta é baseada na idéia de que um sistema independente para a aquisição de custos e estatísticas confere maior flexibilidade e generalidade a esse tipo de serviço. Assim, é possível atender a qualquer arquitetura de mediadores, bem como a outros sistemas que precisem de informações sobre a capacidade de processamento de consultas das fontes de dados em um ambiente distribuído. Na linha da nossa proposta, o projeto *ObjectGlobe* (BRAUMANDL *et al.*, 2001) apresenta um módulo que atua como um catálogo de localização de serviços e publicação de custos e estatísticas sobre provedores de dados, de funções e de processamento. Contudo, este módulo consiste simplesmente em um serviço de diretório e depende que os demais componentes da arquitetura repassem suas informações. Além disso, cabe ao processador global de consultas tratar adequadamente as informações registradas no catálogo.

Basicamente, a aquisição de custos e estatísticas pode ser realizada por inspeção dos dados ou por estimativa através de funções matemáticas aplicadas a estatísticas previamente adquiridas (como, por exemplo, a cardinalidade de uma coleção). No primeiro caso, geralmente é empregado o método da "caixa-preta" para medir volume, cardinalidade e tempo de processamento. A principal desvantagem dessa abordagem é que nem sempre se pode varrer todas as bases de dados para adquirir as estatísticas, sendo bastante custosa essa inspeção. Por isso, muitas vezes é necessário o uso de técnicas de amostragem para minimizar o custo da inspeção. Por outro lado, esse problema não ocorre no uso de funções matemáticas para estimar estatísticas e custos. Contudo, essa estratégia depende da qualidade do modelo de estimativa para que as margens de erro não sejam significativas. A aquisição de estatísticas e custos para o processamento de consultas distribuídas é tratada em trabalhos como ZHU *et al.* (2000), BOUGANIM *et al.* (2001), ROTH *et al.* (1999), ZHU e LARSON (1998) e em NAACKE *et al.* (1998). Já modelos de custos para o processamento de consultas podem ser encontrados em BOULOS e ONO (1999), WANG (2001) e em RUBERG *et al.* (2002a). A arquitetura do DIG permite que sejam empregados métodos de aquisição por inspeção de dados ou por estimativa, ou qualquer combinação destas abordagens. Nossa proposta também facilita a aplicação de técnicas para o refinamento das informações coletadas, como apresentado em GRUSER *et al.* (1999).

Como conclusão, destacamos que estatísticas e custos são informações essenciais para o processamento eficiente de consultas, principalmente em ambientes distribuídos e heterogêneos, sendo condição *sine qua non* para a aplicação de técnicas de otimização baseada em custos. Mesmo em uma abordagem de otimização heurística, o processador de consultas requer informações básicas sobre as coleções envolvidas, como a cardinalidade dessas coleções, para decidir corretamente (em termos de eficiência) como a consulta global será decomposta e enviada para as fontes de dados.

6.2 CONTRIBUIÇÕES

Neste trabalho nós apresentamos o *Distributed Information Gatherer* (DIG), um sistema que visa apoiar o processo de otimização global de consultas em ambientes distribuídos, heterogêneos e com fontes autônomas de dados. A arquitetura do DIG (RUBERG *et al.*, 2002b) define as unidades de *software*, as interfaces, o catálogo de

serviços e os principais procedimentos necessários para a aquisição e a publicação das características gerais, custos e estatísticas sobre o processamento de consultas em diferentes fontes de dados.

A originalidade da arquitetura do DIG consiste na especialização do serviço de descrição da capacidade de consulta das fontes de dados, separando-o em um sistema independente e acessível a diferentes arquiteturas para processamento distribuído de consultas. A arquitetura do DIG é bastante flexível, permitindo que as características de heterogeneidade e autonomia das fontes de dados não sejam impeditivas para a coleta de custos e estatísticas destas fontes. Além disso, o DIG permite realizar uma aquisição ativa de dados. Ou seja, os custos e estatísticas são adquiridos pela arquitetura, periodicamente (aquisição programada) ou através de solicitação do usuário (aquisição por demanda).

Os dois principais componentes da arquitetura do DIG são: o provedor de custos e estatísticas, que realiza o tratamento e a publicação dos dados coletados; e o módulo coletor, responsável pela aquisição de dados nas diversas fontes, repassando-os para os respectivos provedores. A aquisição de dados é realizada pelo coletor DIG através da submissão de consultas ou comandos pré-estabelecidos para cada estatística ou parâmetro de custo a ser coletado. Os dados coletados são publicados pelo provedor DIG através de um catálogo genérico de custos e estatísticas que permite a sua utilização no processamento de consultas em ambientes distribuídos, heterogêneos e com fontes de dados autônomas. Além disso, a publicação do catálogo de serviços do DIG é adaptável, pois o catálogo de serviços suporta desde fontes de dados semi-estruturadas ou não estruturadas (por exemplo, arquivos de texto e páginas *Web*) até fontes de dados com SGBDs sofisticados, oferecendo uma interface adequada para cada caso.

O processo de aquisição de consultas e estatísticas a partir de fontes autônomas em um ambiente distribuído e heterogêneo é bastante complexo. Uma contribuição do nosso trabalho foi destacar a importância do "publicador" nesse processo. Tal papel já havia sido evidenciado no contexto da integração de informações em ELMAGARMID *et al.* (1999), TOMASIC *et al.* (1995) e SIMON (2001). Um "publicador" conhece as particularidades de uma determinada fonte de dados e deve descrever como são os métodos de aquisição de estatísticas e custos nessa fonte, o que determina a qualidade

das informações adquiridas. Assim, é fundamental a acurácia dos dados informados pelo "publicador".

Visando validar a arquitetura proposta, desenvolvemos um protótipo do sistema DIG, através do qual foi possível analisar a coleta e a publicação de informações para o otimizador global da consulta. Através do catálogo de custos e estatísticas publicados pela arquitetura do DIG, o otimizador global de consultas pode estimar previamente os custos de processamentos das subconsultas que serão submetidas às fontes de dados. O otimizador global pode utilizar as informações disponibilizadas pelo protótipo do DIG para estimar o custo de uma subconsulta de acordo com seu modelo de custos, ou solicitar ao provedor DIG que forneça essa estimativa.

Verificamos em uma análise experimental que o protótipo do DIG realizou adequadamente a aquisição de custos, estatísticas e informações, apesar das várias diferenças que foram exploradas nas fontes de dados. Desta forma, evidenciamos o potencial da arquitetura proposta no contexto da otimização de consultas em ambientes distribuídos, heterogêneos e com fontes autônomas. Observamos também que a incorporação de novos parâmetros para aquisição pelo coletor DIG consiste em uma tarefa simples na maioria dos casos, demonstrando a flexibilidade da arquitetura proposta para a adesão de novas fontes de dados, bem como para futuras extensões ao catálogo do DIG.

6.3 TRABALHOS FUTUROS

Atualmente, estamos complementando a implementação do provedor DIG, permitindo que as informações adquiridas nas fontes de dados sejam persistidas em um SGBD. Um SGBD oferece maior agilidade nas respostas do provedor DIG, assim como permite manter um histórico das informações coletadas ao longo do tempo. Esse histórico pode agregar valor às informações publicadas pelo DIG, já que alguns dados possuem uma estimativa mais precisa quando avaliada ao longo do tempo (CHAUDHURI e NARASAYYA, 2001). Neste sentido, uma continuação interessante do nosso trabalho seria definir quais estatísticas e custos poderiam se beneficiar de uma avaliação histórica.

Um outro trabalho em andamento é a implementação do provedor DIG como um serviço *Web*. Para tanto, são necessárias definições na linguagem WSDL (*Web Service Description Language*) para os serviços oferecidos pelo provedor DIG. Além disso, as informações de estatísticas e custos devem ser disponibilizadas através de mensagens SOAP (*Simple Object Access Protocol*). As tecnologias relacionadas a serviços *Web* também podem ser aplicadas na comunicação do provedor DIG com os coletores. Apesar da interface CORBA, utilizada no protótipo do DIG, permitir uma comunicação distribuída entre o provedor e os coletores, o ambiente é dependente do ORB para permitir esta comunicação. O serviço *Web* é mais desacoplado e independente de uma unidade central para gerência da comunicação, o que daria uma maior flexibilidade à arquitetura DIG.

Por fim, uma perspectiva interessante para a aplicação da arquitetura do DIG seria explorar as facilidades oferecidas pelo provedor DIG para a aquisição e a publicação de dados sobre a qualidade de serviços de consulta e sobre o processamento de consultas em XML.

REFERÊNCIAS BIBLIOGRÁFICAS

ABITEBOUL, S., 1997, "Querying semi-structured data". In: *Proceedings of the 6th Int. Conf. on Database Theory*, pp. 1-18, Greece, Jan.

ASHISH, N., KNOBLOCK, C.A., 1997, "Wrapper Generation for Semi-structured Internet Sources", *ACM SIGMOD Record*, v. 26, n. 4 (Mar), pp. 8-15.

ATTALURI, G.K., BRADSHAW, D.P., COBURN, N., et al., 1995, "CORDS Multidatabase Project", *IBM Systems Journal*, v. 34, n. 1, pp. 39-62.

BOUGANIM, L., FABRET, F., PORTO, F., et al., 2001, "Processing Queries with Expensive Functions and Large Objects in Distributed Mediator Systems". In: *Proceedings of 17th Int. Conf. On Data Engineering (ICDE 2001)*, pp. 91-98, Heildelberg, Germany, Apr.

BOUGUETTAYA, A., BENATALLAH, B., ELMAGARMID, A., 1999, "An Overview of Multidatabase Systems: Past and Present. Management of Heterogeneous and Autonomous Database Systems". In: *Management of Heterogeneous and Autonomous Database Systems*, Morgan Kaufmann Publishers, Inc., ISBN 1-55860-216-X, pp. 1-32.

BOULOS, J., ONO, K., 1999, "Cost Estimation of User-Defined Methods in Object-Relational Database Systems", *ACM SIGMOD Record*, v. 28, n. 3 (Sep), pp. 22-28.

BRAGA, R.M.M., MATTOSO, M., WERNER, C.M.L, 2000, "Using Ontologies for Domain Information Retrieval". In: *Proceedings of 11th Int. Conf. and Workshop on Database and Expert Systems Applications (DEXA'00), Workshop on Domain Engineering*, pp. 831-835, Greenwich, UK, Sep.

BRAUMANDL, R., KEIDL, M., KEMPER, A., et al., 2001, "ObjectGlobe: Ubiquitous query processing on the Internet", *The VLDB Journal*, v. 10, n. 1, pp. 48-71.

BRÜGGER, T.S., PIRES, P.F., MATTOSO, M., 1999, "Mediators Metadata Management Services: An implementation using the GOA++ System", *Electronic Journal of SADIO (Argentine Society for Informatics and Operations Research), http://www.dc.uba.ar/sadio/ejs*, v. 2, n. 1 (Jun), pp. 30-47.

CATTEL, R., BARRY, D., BERLER, M., *et al.*, 2000, *The Object Data Standard: ODMG 3.0*. 1st ed. San Francisco, Morgan Kaufmann Publishers Inc.

CHAUDHURI, S., NARASAYYA, V., 2001, "Automating Statistics Management for Query Optimizers", *IEEE Transactions on Knowledge and Data Engineering*, v. 13, n. 1 (Jan/Feb), pp. 7-20.

DITTRICH, K.R., DOMENIG, R., 1999, An Overview and Classification of Mediated Query Systems, *ACM SIGMOD Record*, v. 28, n. 3 (Sep), pp. 63-72.

EISENBERG, A., MELTON, J., 1999, "SQL:1999, *formerly known as* SQL3", *ACM SIGMOD Record*, v. 28, n.1 (Mar), pp. 131-138.

ELMAGARMID, A., RUSINKIEWICZ, M., SHETH, A., 1999, *Management of Heterogeneous and Autonomous Database Systems*. 1st ed. São Francisco, Morgan Kaufmann Publishers, Inc.

ELMASRI, R., NAVATHE, S.B., 1994, *Fundamentals of Database Systems*. 2nd ed. Melo Park, Addison-Wesley Publishing Company.

EVRENDILEK, C., DOGAG, A., NURAL, S., OZCAN, F., 1997, "Multidatabase Query Otimization", *Distributed and Parallel Database*, v. 5, n. 1 (Jan), pp. 1-39.

FLORESCU, D., LEVY, A., MENDELZON, A., 1998, "Database Techniques for the World-Wide Web", *ACM SIGMOD Record*, v. 27, n. 3 (Sep), pp. 59-74.

FOSTER, I., KESSELMAN, C., TUECKE, S., 2001, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *Int. Journal of High Performance Computing Applications*, v. 15, n. 3 (Aug), pp. 200-222.

FREIRE, J., HARITSA, J., RAMANATH, M., et al., 2002, "*StatiX: Making XML Count*". In: Proceedings of ACM Int. Conf. on Management of Data, pp. 181-192, Wisconsin, USA, Jun.

GARDARIN, G., NAAKE, H., TOMASIC, A., 1997, *Leveraging Mediator Cost Models with Heterogeneous Data Sources*, Relatório Técnico n° 3143, INRIA, França, Mar.

GARDARIN, G., SHA, F., TANG, Z., 1996, "Calibrating the Query Optimizer Cost Model of IRO-DB, an Object-Oriented Federated Database System". In: *Proceedings of 22nd Int. Conf. On Very Large Data Bases (VLDB 1996)*, pp. 378-389, Bombay, India, Sep.

GARLIC, 1999, "The Garlic Project". In: *IBM Almadem Research Center, available in* http://www.almaden.ibm.com/cs/garlic/, Sep.

GOA, 2002, "GOA++ Gerente de Objetos Armazenados". *Available in* http://www.cos.ufrj.br/~goa, Oct.

HAAS, L., KOSSMANN, D., WIMMERS, E., et al., 1997, "Optimzating Queries across Diverse Data Sources". In: *Proceedings of 23rd Int. Conf. On Very Large Data Bases (VLDB 1997)*, pp. 276-285, Athens, Greece, Aug.

KEIDL, M., SELTZSAM, S., STOCKER, K., KEMPER, A., 2002, "ServiceGlobe: Distributing E-services Across the Internet". In: *Proceedings of 28th Int. Conf. On Very Large Data Bases (VLDB 2002)*, Hong Kong, China, Aug.

KOSSMANN, D., 2000, "The State of the Art in Distributed Query Processing", *ACM Computing Surveys*, v. 32, n.4 (Dec), pp. 422-469.

KREGER, H, 2001, "Web Services Conceptual Architecture (WSCA 1.0)". In: *IBM Software Group, available in* http://www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf , May.

MANI, A., NAGARAJAN, A., 2002, "Understanding Quality of Service for Web Services", *IBM DevelporWorks Eletronic Magazine, available in* http://www-106.ibm.com/developerworks/library/ws-quality.html, Jan.

MATTOSO, M., WERNER, C., ROCHA, R., PINHEIRO, R., MURTA, L., ALMEIDA, V., COSTA, M., BEZERRA, E., SOARES, E., RUBERG, N., 2000, "Persistência de Componentes num Ambiente de Reuso". In: *XIV Simpósio Brasileiro de Engenharia de Software* (SBES'2000), Sessão de Ferramentas, João Pessoa, Brasil, Oct.

MAURO, R., ZIMBRÃO, G., BRUGGER, T., *et al.*, 1997, "GOA++: Tecnologia, Implementação e Extensões aos Serviços de Gerência de Objetos". In: *XII Simpósio Brasileiro de Banco de Dados*, pp. 272-286, Fortaleza, Brasil, Oct.

NAACKE, H., GARDARIN, G., TOMASIC, A., 1998, "Leveraging Mediator Cost Models with Heterogeneous Data Sources". In: *Proceedings of 14th Int. Conf. On Data Engineering (ICDE 1998)*, pp. 351-360, Florida, USA, Feb.

OBJECTDRIVER, 2002, "An open Object Wrapper dedicated to Relational Databases Reusing". In: *INFOBJECTS, available in* http://www.infobjects.com/.

ÖSZU, M.T., BLAKELY, J.A., 1995, "Query Processing in Object-Oriented Database Systems". In:*Modern Database Systems*, ACM Press and Addison Wesley, ISBN 0-201-59098-0, pp. 146-174.

ÖZSU, M.T., VALDURIEZ, P., 1999, *Principles of Distributed Database Systems*. 2nd ed., New Jersey, USA, Prentice Hall.

PIATETSKY-SHAPIRO, G., CONNEL, C., 1984, "*Accurate Estimation of the Number of Tuples Satisfying a Condition*". In: Proceedings of ACM Int. Conf, on Management of Data, pp. 256-276, Boston, USA, Jun.

PIRES, P.F., 1997, *HIMPAR, uma Arquitetura para Interoperabilidade de Objetos Distribuídos*. Tese de M.Sc., PESC-COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

ROTH, M. T., OZCAN, F., HAAS, L. M., 1999, "Cost Models Do Matter: Providing Cost Information For Diverse Data Sources in a Federated System". In: *Proceedings of 25th Int. Conf. On Very Large Data Bases (VLDB 1999)*, pp. 599-610, Edinburgh, Scotland, Sep.

RUBERG, G., 2001, *Um Modelo de Custo para o Processamento de Consultas em Bases de Objetos Distribuídos*. Tese de M.Sc., PESC-COPPE, UFRJ, Rio de Janeiro, RJ, Brasil.

RUBERG, G., BAIAO, F., MATTOSO, M., 2002a, "Estimating Costs of Path Expression Evaluation in Distributed Object Databases". In: *Proceedings of 13th Int. Conf. of Database and Expert System Applications (DEXA 2002)*, pp. 351-360, Aix-en-Provence, France, Sep.

RUBERG, N., RUBERG, G., MATTOSO, M., 2002b, "DIG: Um Serviço de Custos e Estatísticas para o Processamento Distribuído de Consultas". In: *XVII Simpósio Brasileiro de Banco de Dados*, pp. 121-135, Gramado, Brasil, Oct.

SAHUGUET, A., AZAVANT, F., 1999, "Building Light-Weight Wrapper for Legacy Web Data-Sources using W4F". In: *Proceeding of 25th Int. Conf. On Very Large Data Bases (VLDB 1999)*, pp. 738-741, Edingurgh, Scotland, Sep.

SERGEY, B., LAWRENCE, P., 1998, *"The Anatomy of a Large-Scale Hypertextual Web Search Engine"*. In: Proceedings of 6th WWW7/Computer Networks, v. 30, n.1-7, pp. 107-117, California, USA, Apr.

SHETH, A. P., LARSON, J. A., 1990, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, v. 22, n.3 (Sep), pp. 183-236.

SIMON, E., 2001, "Le Select, a Middleware System that Eases the Publication of Scientific Data Sets and Programs". In: *Workshop on Information Integration on the Web*, pp. 2, Rio de Janeiro, Brasil, Apr.

STL, 2002, "STL: C++ *Standard Template Library*". In: *SGI – Silicon Graphics, available in* http://www.sgi.com/tech/stl/index.html.

SUN, 1999, *Jini Architectural Overview, Technical White Paper available in* http://www.sun.com , *Sun Microsystems*.

TOMASIC, A., RASCHID, L., VALDURIEZ, P., 1995, *Scalling Heterogeneus Databases and the Design of Disco*, Relatório Técnico n° 2704, INRIA, França, Nov.

TOMASIC, A., RASCHID, L., VALDURIEZ, P., 1998, "Scalling Access to Heterogeneus Data Sources with Disco", *IEEE Transactions on Knowledge and Data Engineering*, v.10, n. 5 (Sep/Oct), pp 808-823.

VAUGHAN-NICHOLS, S.J, 2002, "Web Services: Beyond the Hype". *IEEE Computer Magazine*, v. 35, n. 2, pp. 18-21

VICTORINO, M., MOURA, A., 2001, "Using Mediator and Datawarehouse Technologies for Developing an Environmental Decision Support System". In: Workshop on Information Integration on the Web, pp. 141-147, Rio de Janeiro, Brasil, Apr.

WANG, Q., 2000, *Cost-Based Object Query Optimization.* Ph.D. Thesis, Oregon Graduate Institute of Science and Technology, Beaverton, Oregon, Estados Unidos.

WIEDERHOLD, G. 1992, "Mediators in the Architecture of Future Information Systems", *IEEE Computer Magazine*, v.25, n.3 (Mar), pp 38-49.

ZHU, Q., LARSON, P., 1998, "Solving Local Cost Estimation Problem for Global Query Optimization in Multidatabase Systems", *Distributed and Parallel Databases*, v.6, n.4, pp. 373-421, Jan.

ZHU, Q., SUN,Y., MOTHERAMGARI, S., 2000, "Developing Cost Models with Qualitative Variables for Dynamic Multidatabase Environments". In: *Proceedings of 16th IEEE Int. Conf. On Data Eng. (ICDE2000)*, pp. 413-424, San Diego, Estados Unidos, Feb/Mar.