Pós-Graduação em Ciência da Computação

"OTIMIZAÇÃO DE ACESSO EM UM SISTEMA DE INTEGRAÇÃO DE DADOS ATRAVÉS DO USO DE CACHING E MATERIALIZAÇÃO DE DADOS"

Por

Maria da Conceição Moraes Batista

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, Abril/2003

Agradecimentos

À minha orientadora Profa. Ana Carolina, pela clareza, precisão, segurança e presença constante com que me orientou, e mais ainda, por ter sido também minha amiga.

A Bernadette Lóscio, amiga, companheira especialmente brilhante e fonte de ajuda inesgotável, que compartilhou comigo e muitas vezes me levou pela mão nessa jornada longa e difícil, mas que certamente valeu a pena.

Aos membros da banca, Prof. Fernando Fonseca e Profa. Marta Mattoso pela disposição e interesse de vir e contribuir com suas experiências e idéias valiosas.

À equipe formada por Guilherme Amorim, Haroldo Amaral e Luciano Galvão, meu muito obrigada pelo esforço e dedicação sem os quais eu não teria conseguido concluir o trabalho.

Aos companheiros de mestrado, em especial ao amigo Hélio Lopes dos Santos por ter sido co-responsável por todas as experiências enriquecedoras que compartilhamos ao longo do curso.

Aos colegas de trabalho, em especial a Gustavo Guedes pela paciência e tolerância nas minhas ausências.

Aos meus irmãos. À minha mãe, pela doçura constante com que me incentivou. Obrigada ao meu pai, que no início do trabalho se foi, mas que certamente participou dos momentos e torceu por mim.

A todos os amigos, que sempre ficaram do meu lado sem eu poder ficar do lado deles. Agradeço a Catarina, Fernanda e Luciana pelo apoio, incentivo e paciência. Agradeço a Marconi, por sempre estar disponível com sua alegria. Um obrigada ainda mais carinhoso às amigas Silvana Gondim e Fabiana Nader, pelo carinho, ombro amigo e por não me deixarem esmorecer. Obrigada também a Ana Paula e Lúcio, pela vibração e carinho.

Por fim, o agradecimento maior e com todo amor, vai para o amigo-irmão Ricardo Rodrigues, por ter sido tudo de bom nas horas mais difíceis e certamente sem sua participação, esse trabalho não teria terminado.

Resumo

Sistemas de integração de dados oferecem acesso uniforme sobre fontes de dados heterogêneas e distribuídas. Para fornecer um acesso integrado a diversas fontes de dados, duas abordagens clássicas foram propostas na literatura atual: abordagem materializada e abordagem virtual. Na abordagem materializada, os dados são previamente acessados, integrados e armazenados em um *data warehouse* e as consultas submetidas ao sistema de integração são processadas nesse repositório sem haver acesso direto às fontes de dados. Na abordagem virtual, as consultas submetidas ao sistema de integração são decompostas em subconsultas endereçadas diretamente às fontes de dados. Os dados obtidos das fontes como resposta a essas subconsultas são integrados e retornados ao usuário solicitante.

O nosso trabalho, consiste em criar um ambiente de integração de dados provenientes de múltiplas fontes no ambiente Web o qual combina recursos de ambas as abordagens suportando o processamento de consultas virtuais e materializadas. Um outro recurso de nossa proposta é a inserção de um subsistema de gerenciamento de uma *cache* para armazenar os resultados das consultas mais freqüentemente submetidas pelo usuário.

O ambiente tem recursos de materialização de dados em um *data warehouse*, e o processo de materialização é feito seletivamente com base na análise e classificação de critérios de qualidade e custo associados aos dados das fontes. Essa seleção criteriosa visa equilibrar melhorias no tempo de resposta das consultas com taxas de custo de manutenção do *data warehouse* aceitáveis.

A partir de uma arquitetura de integração de dados baseada na abordagem virtual, foram incluídos módulos para gerenciamento do *data warehouse*, gerenciamento da *cache* e módulos de processamento de consultas sob três formas: virtuais com acesso às fontes de dados, materializadas com acesso ao *data warehouse* e consultas acessando diretamente a *cache*. Todos

esses recursos são colocados em conjunto visando obter ganhos no desempenho do processamento das consultas no sistema de integração.

Abstract

Data integration systems are planned to offer uniform access to data from heterogeneous and distributed sources. Two basic approaches have been proposed in the literature to provide integrated access to multiple data sources. In the materialized approach, data are previously accessed, cleaned, integrated and stored in the data warehouse and the queries submitted to the integration system are evaluated in this repository without direct access to data sources.

In the virtual approach, the queries posed to the integration system are decomposed into queries addressed directly to the sources. The data obtained from the sources are integrated and returned to the user.

In this work we present a data integration environment to integrate data distributed on multiple web data sources which combines features of both approaches supporting the execution of virtual and materialized queries. Other distinguished feature of our environment is that we also propose the use of a cache system in order to answer the most frequently asked queries.

The environment has resources for selectively materialize data in a data warehouse. This materialization process implies in analyzing a set of quality and cost criteria associated with the data in order to determine if the materialization will improve time response gains and minimize maintenance cost of the data warehouse.

In an existing virtual-based architecture for data integration we have inserted software modules for management of the data warehouse and cache, and queries processing under three ways: accessing the data sources (virtuals), accessing the data warehouse (materialized) and accessing the cache contents directly. All these resources are put together with the goal of optimizing the overall query response time.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

No ambiente Web, é muito comum os dados serem disponibilizados em múltiplas e distintas fontes de dados, tais como: bancos de dados relacionais, bancos de dados orientados a objetos, bases de conhecimento, sistemas de arquivos, documentos digitais, sistemas legados, mensagens de correio eletrônico, planilhas de cálculo, páginas Web, entre outros. Esse cenário muitas vezes dificulta atividades de consultas a um domínio específico, devido a dificuldades de acessar, reunir e integrar informações originárias das múltiplas fontes heterogêneas em um período de tempo aceitável, e também devido ao fato de que as informações obtidas podem apresentar inconsistências.

Como ilustração, consideremos um exemplo dentro de um domínio específico para uma aplicação de filmes em cartaz: O *Internet Movie Database*[1] é um website, ou fonte de dados Web, que contém dados sobre filmes, seus elencos, gêneros e diretores. Em outras fontes Web é possível encontrarmos críticas sobre os filmes, por exemplo, em páginas de jornais locais, e algumas outras fontes podem conter horários de exibição desses filmes. Combinando dados dessas fontes, um sistema de integração deveria estar apto a responder consultas do tipo *"Que filmes, em que horários e cinemas, com o ator Brad Pitt no elenco estarão sendo exibidos em Paris hoje à noite?"* sem que o usuário tenha de acessar diretamente as fontes de informação para combinar esses dados.

Um outro ponto que deve ser ressaltado é que devido ao constante crescimento da Internet, a Web vem se tornando uma fonte poderosa de informações e como tal deve ser explorada para fornecer essas informações de forma confiável, rápida e uniforme. Muito comumente, as fontes

[1] http://www.imdb.com

de dados distribuídas no ambiente Web e em outros ambientes dinâmicos, são fontes autônomas, heterogêneas e sem muita estruturação de seus dados. Muitas aplicações processando em ambientes dinâmicos podem necessitar de ter acesso integrado a dados distribuídos em diferentes fontes de dados.

Neste contexto, sistemas de integração de dados vêm surgindo como ferramentas para oferecer acesso uniforme a dados distribuídos em fontes heterogêneas e autônomas [Ambite98, Ashish00, Baru99, Chawathe94, Draper01a, Erdmann00, Hammer95, Labio97, Naumann99]. A autonomia das fontes de dados vem do fato que muito comumente, existem aplicações locais processando seus dados sem interagir com o sistema de integração. O acesso uniforme pode ser oferecido através da resolução das heterogeneidades presentes nos dados e criação de uma visão integrada das fontes de dados subjacentes que reflita os requisitos e necessidades do usuário. Os usuários de um sistema de integração de dados submetem consultas sobre essa visão integrada sem necessidade de perder tempo em buscas e pesquisas no ambiente dinâmico. Basicamente, a tarefa de um sistema de integração é responder a consultas que podem requerer extração e combinação de dados originários de múltiplas fontes.

A presença de heterogeneidades nos dados é um fator principal para criar dificuldades em sistemas de integração de dados e, é possível observar essas heterogeneidades sob uma outra classificação, a qual pode ocorrer em vários níveis [Hasselbring00]:

- Nível técnico ⇒ Heterogeneidade resultante de diferentes plataformas de hardware, sistemas operacionais, sistemas de gerenciamento de bancos de dados e linguagens de programação;
- Nível Conceitual ⇒ Nesse nível, a heterogeneidade é resultado de interpretações distintas do significado de certos termos, de diferentes modelos de dados (ex.: relacional, orientado a objetos, etc.), e também de diferentes processos de modelagem para os mesmos conceitos do mundo real. De processos de modelagem desse tipo são derivadas as discrepâncias esquemáticas [Krishnamurthy91], problema típico da área de banco de dados onde um dado em uma base corresponde, por exemplo, a um metadado de uma outra base de dados. A tarefa de conciliar esse tipo de discrepância não é trivial, levando em consideração que cada modelo contém uma semântica dos objetos do mundo real expressa de forma diferente.

Sistemas de integração de dados normalmente seguem duas abordagens clássicas [Abiteboul99], onde cada uma segue uma arquitetura de implementação:

- Abordagem Virtual – Nessa abordagem, os dados permanecem nas fontes e as consultas submetidas ao sistema de integração de dados são decompostas em tempo de execução em subconsultas direcionadas às fontes de dados. Em sistemas de integração de dados baseados na abordagem virtual [Chawathe94, Baru99], um componente de software – o mediador – recebe as consultas de usuário e faz a decomposição em subconsultas sobre as fontes de dados. Os resultados das subconsultas enviados pelas fontes de dados são traduzidos, filtrados e combinados e a resposta final é retornada ao usuário. Esse tipo de sistema é mais adequado quando há necessidade de entregar dados a partir de fontes altamente dinâmicas com garantias que os dados estão sempre atualizados. Por outro lado, existem desvantagens com relação aos seguintes fatores: (i) a possibilidade das fontes de dados tornarem-se indisponíveis ao sistema de integração e; (ii) o tempo de resposta das consultas que pode ser muito grande devido ao grande número de fontes que devem ser acessadas;

- Abordagem Materializada – Nessa abordagem os dados são previamente acessados, recuperados, integrados e armazenados em um repositório central, o *data warehouse* [Widom95, Gupta95], e as consultas submetidas ao sistema de integração são avaliadas nesse repositório sem haver acesso às fontes de dados. A abordagem materializada possui vantagens relacionadas com facilidade e efetividade no acesso, uma vez que os dados são disponibilizados em um repositório local ao sistema de integração. Entretanto, existem desvantagens no problema de manter os dados do *data warehouse* consistentes com relação aos dados nas fontes e com relação ao problema de manter dados replicados no *data warehouse*.

O nosso trabalho consiste em construir um ambiente de integração de dados que combina recursos de ambas as abordagens suportando a execução de consultas virtuais e materializadas. O principal objetivo de reunir tais recursos é minimizar os impactos dos problemas mais comuns das abordagens mencionadas. Algumas porções de dados mais intensivamente indisponíveis e estáticos serão materializados em um *data warehouse* e dados mais dinâmicos serão acessados por consultas virtuais. Essa seleção é feita com base em uma análise de critérios de qualidade e custo que são associados às fontes de dados para garantir que a materialização de dados tanto poderá ajudar na melhoria do desempenho das consultas como também não acarretará em um alto custo de manutenção do *data warehouse*. O sistema é baseado em uma arquitetura híbrida para integração de dados.

Outro fator de contribuição do ambiente é o uso de uma *cache* local ao sistema de integração. Ou seja, um outro repositório para armazenar resultados "prontos" para as

consultas mais freqüentemente submetidas ao sistema de integração. A *cache* foi concebida com o intuito de retornar resultados de algumas consultas de usuário com o mínimo tempo de processamento.

Em resumo, a contribuição chave e principal objetivo de nosso trabalho consiste na criação de um ambiente de integração de dados que processa três tipos de consultas:

- Consultas virtuais a dados que serão obtidos sob demanda, ou seja, acessados diretamente das fontes de dados;
- Consultas a dados materializados que serão obtidos a partir do *data warehouse*;
- Consultas que terão seus resultados previamente armazenados em uma *cache* e serão recuperados a partir dessa *cache* local ao sistema de integração.

1.2 Estrutura da Dissertação

O nosso trabalho será estruturado como se segue:

- **Capítulo 2 – Integração de Dados**: serão descritos aspectos de integração de dados, problemas relacionados e as principais arquiteturas utilizadas e sistemas desse tipo. São abordados nesse capítulo, alguns sistemas de integração de dados construídos com base nas arquiteturas clássicas. Também será reservada uma seção para discorrer sobre problemas na integração de dados;
- **Capítulo 3 – Arquiteturas Híbridas**: esse capítulo abordará os mais recentes sistemas de integração de dados que utilizam arquiteturas híbridas. Serão feitas críticas, pontuação de vantagens e comparações entre os sistemas descritos;
- **Capítulo 4 – Integração de Dados com Critérios de Qualidade**: o capítulo 4 dedica-se a mostrar como podem ser feitas a integração e materialização de dados com o uso de critérios de qualidade como por exemplo: disponibilidade da fonte de dados, tempo de resposta, freqüência de acessos e atualidade, entre outros. As técnicas aqui discutidas servirão de base para o nosso processo de seleção de dados para materialização no *data warehouse*;
- **Capítulo 5 – Consultas com Uso de Cache e Materialização de Dados**: aqui será detalhada toda a arquitetura do sistema proposto, como ela será inserida em uma arquitetura de mediadores já existente, seus módulos, a comunicação entre eles, a base de dados materializados e cache. Será mostrado como as consultas de usuário deverão

ser processadas pelo sistema e como será o processo de integração de instâncias de dados;

- **Capítulo 6 – Protótipo do Ambiente de Integração de Dados**: o capítulo 6 vem para destacar as tecnologias utilizadas juntamente com as tarefas que foram necessárias para a implementação de um protótipo. Também serão detalhados todos os processos, experiências e resultados associados com o protótipo implementado;
- **Capítulo 7 – Conclusões e Trabalhos Futuros**: esse capítulo finalizará a dissertação mostrando as conclusões obtidas com o trabalho, contribuições do mesmo, algumas sugestões para trabalhos futuros e possíveis melhorias no sistema desenvolvido até então.

Devemos ressaltar que as principais contribuições de nosso trabalho de pesquisa serão abordadas nos capítulos 5 e 6 dessa dissertação.

Capítulo 2

Integração de Dados

2.1 Introdução

A principal tarefa de um sistema de integração de dados é fornecer uma interface uniforme para responder consultas que normalmente requerem extração e combinação de dados originários de múltiplas fontes distintas, heterogêneas, muitas vezes distribuídas e autônomas [Levy99, Florescu98]. Em ambientes dinâmicos como a Web, aplicações de integração de dados enfrentam desafios maiores, pois devem lidar com fatores ainda mais críticos: um grande número de fontes de dados altamente evolutivas, pouca representação de metadados e características das fontes de dados e um alto grau de autonomia das mesmas.

Normalmente, uma ocupação importante de sistemas de integração de dados é administrar heterogeneidades que suas fontes de dados podem apresentar. Algumas delas são listadas abaixo:

- Modelos de dados ⇒ Fontes de dados podem apresentar modelos de dados distintos: relacional, orientado a objetos, multidimensional, semi-estruturado;
- Esquema ⇒ Um esquema determina relações, atributos, restrições sobre os dados das fontes. Fontes de dados podem ter esquemas altamente estruturados e restritivos como podem ter esquemas apenas indicativos;
- Codificação dos valores ⇒ Valores e itens de dados podem estar codificados de maneira distinta. Por exemplo, um valor de pedido está em US$ em uma fonte e em R$ em outra fonte; O item endereço pode ser um string em uma fonte e uma tupla relacional em outra;
- Linguagem de consulta ⇒ Algumas fontes podem apresentar linguagens de consultas nativas (SQL, OQL, etc.) e outras não possuem esse recurso;

- SGBD ou sistema de arquivo ⇒ Fontes que estão sendo gerenciadas por SGBDs ou sistemas de arquivos e fontes sem gerenciamento;
- Sistema operacional e hardware;
- Semântica ⇒ Heterogeneidades semânticas estão relacionadas com o significado da informação [Florescu98].

Além de tratar heterogeneidades, o sistema de integração de dados deve abordar aspectos relativos à autonomia e distribuição das fontes de dados. A autonomia provém do fato de que as fontes geralmente operam de forma independente ou semi-independente e, conseqüentemente, podem modificar seus dados e esquemas a qualquer momento sem notificar o sistema de integração. A principal questão da distribuição das fontes de dados diz respeito aos desafios impostos pela localização dessas fontes em um ambiente globalizado e heterogêneo que é a Internet.

Nesse capítulo serão mostrados aspectos de integração de dados e o restante do capítulo está organizado da seguinte forma: A seção 2.1 discute as abordagens de integração de dados e as arquiteturas usadas para implementar as abordagens, na seção 2.2 são abordados sistemas e pesquisas recentes na área de integração de dados e na seção 2.3 são discutidos os principais problemas de integração de dados.

2.2 Abordagens de Integração de Dados

Algumas pesquisas [Baru99, Erdmann00, Ambite98, Lóscio01, Ashish99a] tratam do problema de integração de dados de fontes heterogêneas. Normalmente, sistemas de integração de dados seguem duas abordagens principais: a abordagem virtual e a abordagem materializada [Abiteboul99].

Na abordagem virtual, as informações são extraídas diretamente das fontes apenas quando solicitadas em uma consulta. Na abordagem materializada, as informações relevantes são recuperadas, integradas e armazenadas em um repositório central. É nesse repositório que são processadas as consultas feitas ao sistema sem haver acessos diretos às fontes de dados.

Cada uma das abordagens possui algumas características, vantagens e desvantagens e se aplicam em situações diferentes. Uma das maiores vantagens da abordagem virtual é que as informações recuperadas estão sempre atualizadas. Por outro lado, a abordagem virtual não é adequada quando os passos de tradução e integração das informações são extensos e ficam muito caros ou quando há a possibilidade de as fontes de dados freqüentemente ficarem inacessíveis.

A abordagem materializada possui uma grande vantagem que é a rapidez com que as informações integradas ficam disponíveis, pois não existe acesso direto às fontes de dados. Um problema desse tipo de abordagem é manter a consistência entre os dados das fontes e os dados do repositório. A abordagem materializada geralmente é indicada para algumas situações, quando:

- Porções específicas das informações e consultas previsíveis são requisitadas;
- É necessário para os usuários que o tempo de resposta das consultas seja curto, sem necessidade das informações estarem em seu estado mais atualizado;
- É necessário para usuários do sistema obterem informações que não constam diretamente nas fontes de dados, tais como, informações históricas.

A abordagem materializada não deve ser utilizada quando as consultas devem fornecer informações sempre atualizadas.

A questão de uma abordagem ser ou não adequada varia de acordo com a aplicação que irá utilizá-la. Entretanto, para aplicações complexas e de grande escala, a aplicação de recursos de ambas as abordagens pode ser necessária. Em alguns casos, um sistema de integração de dados deve possuir algumas informações recuperadas, processadas e integradas previamente enquanto outras informações devem ser recuperadas apenas quando requisitadas nas consultas.

Sistemas de integração de dados em geral apresentam dois tipos de arquitetura clássicos:

- Arquitetura de mediadores que implementa a abordagem virtual e possui um módulo de software – o mediador – que recebe e trata as consultas submetidas ao sistema de integração e é responsável pela decomposição dessas consultas em subconsultas que serão enviadas às fontes de dados;
- Arquitetura de data warehouse que implementa a abordagem materializada. Os dados são recuperados, integrados e armazenados em um repositório central – o *data warehouse* – onde as consultas requisitadas ao sistema de integração são processadas sem haver acesso às fontes de dados.

As próximas seções são dedicadas a detalhar cada uma das arquiteturas de sistemas de integração de dados.

2.2.1 Arquitetura de Data Warehouse

A arquitetura de integração de dados com *Data Warehouse* implementa a abordagem materializada, onde os dados são previamente recuperados, integrados e armazenados em um

repositório central, o ***Data Warehouse*** [Widom95, Gupta95]. O *data warehouse* é a visão materializada dos dados e quaisquer consultas submetidas ao sistema de integração serão avaliadas no próprio *data warehouse*. Uma visão é uma relação derivada a partir de outras relações base e a visão pode ser materializada através do armazenamento de tuplas da mesma em um banco de dados.

Em [Rundensteiner00] vemos que a abordagem de construção do repositório de dados, comumente referenciada como *data warehousing*, é caracterizada pelas seguintes propriedades:

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

A Figura 2.1 mostra a arquitetura de integração de dados com *data warehouse*:

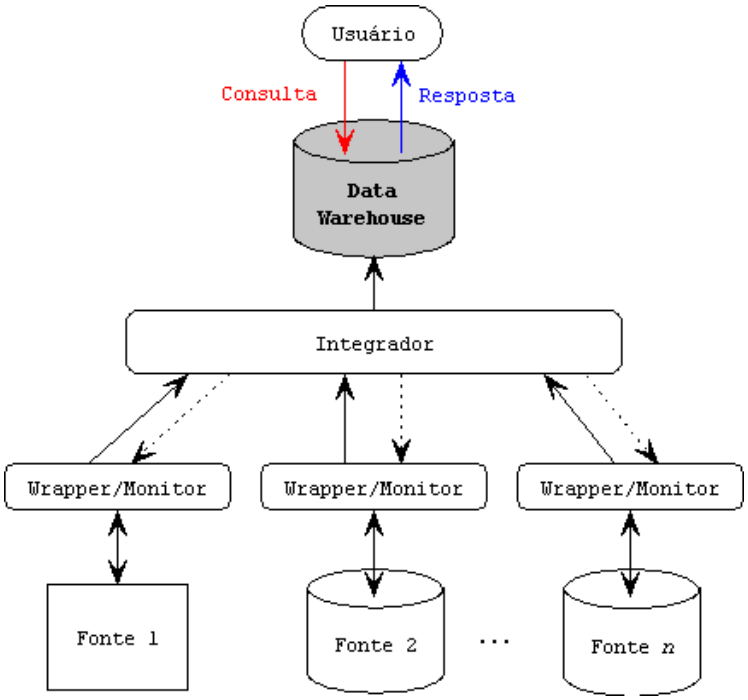


Figura 2.1 – Arquitetura de integração de dados com Data Warehouse

As fontes de dados podem ser sistemas de arquivos, bancos de dados, documentos HTML, sistemas legados, bases de conhecimento, entre outras. Conectado com cada fonte está um módulo de software *wrapper/monitor*. O componente *wrapper* desse módulo é responsável pela

tradução da informação do formato da fonte de dados para o formato e modelo de dados utilizados pelo sistema de *data warehousing*. O componente *monitor* faz a detecção automática de modificações nas fontes de dados que sejam relevantes para o sistema e repassam essas alterações ao módulo integrador. O integrador, por sua vez, é responsável por "instalar" as informações no *warehouse*, o que inclui: filtragem, sumarização e integração das informações com outras provenientes de outras fontes. A fim de integrar propriamente informações no *data warehouse*, pode ser necessário ao integrador obter informações adicionais de uma ou mais fontes de dados.

O grande problema em sistemas de integração de dados com arquitetura de *data warehouse* diz respeito à manutenção da consistência dos dados entre o repositório e as fontes [Agrawal97, Zhuge95]. Devido à independência das fontes em relação ao *data warehouse*, os dados das mesmas podem sofrer atualizações que não irão refletir no repositório. Normalmente, existem duas abordagens para a manutenção da consistência:

- **Rematerialização de visão** ⇒ Todo o conteúdo do *data warehouse* é descartado e a visão é novamente materializada com os dados das fontes. Esse tipo de procedimento é bastante custoso uma vez que elimina todo o repositório central e cria um novo com os dados alterados;
- **Manutenção incremental** ⇒ As alterações nos dados das fontes são propagadas incrementalmente para o *data warehouse*.

Para determinar o tipo de manutenção da visão do *data warehouse* deve-se levar em conta as capacidades que as fontes de dados locais possuem de enviar suas modificações para que a atualização das visões materializadas seja feita de forma adequada. Fontes de dados podem ser classificadas de acordo com seu nível de atividade relacionado com a forma de notificação de mudanças para o sistema de integração de dados:

- **Fontes de dados com atividade satisfatória** ⇒ A fonte de dados tem condições de enviar ao sistema de integração de dados as modificações que foram efetuadas nos seus dados;
- **Fontes de dados com atividade restrita** ⇒ A fonte de dados notifica modificações apenas através do envio de mensagens simples;
- **Fontes sem atividade** ⇒ A fonte de dados não é capaz de comunicar qualquer notificação de modificações. Nesse caso apenas a rematerialização da visão é possível.

A arquitetura de *data warehouse* apresenta algumas vantagens e desvantagens para sistemas de integração de dados: a ineficiência da arquitetura de *data warehouse* é clara quando as

consultas requerem que as informações estejam sempre atualizadas. Por outro lado, essa arquitetura é mais adequada para as seguintes situações:

- As consultas que serão feitas ao sistema de integração de dados podem ser previsíveis e não muito variantes dentro dessa previsão;
- Existe uma necessidade de alto desempenho nas consultas em detrimento de ter o estado mais atualizado das informações;
- Algumas consultas requerem informações típicas de um data warehouse como, por exemplo, informações históricas e temporais e valores agregados.

2.2.2 Arquitetura de Mediadores

De acordo com Wiederhold [Wiederhold92], por definição, mediadores são módulos de software que exploram o conhecimento representado em um conjunto ou subconjunto de dados para gerar informações para aplicações residentes em uma camada superior. A arquitetura de mediadores, em sistemas de integração de dados implementa a abordagem virtual. A Figura 2.2 apresenta um esquema para a arquitetura de mediadores:

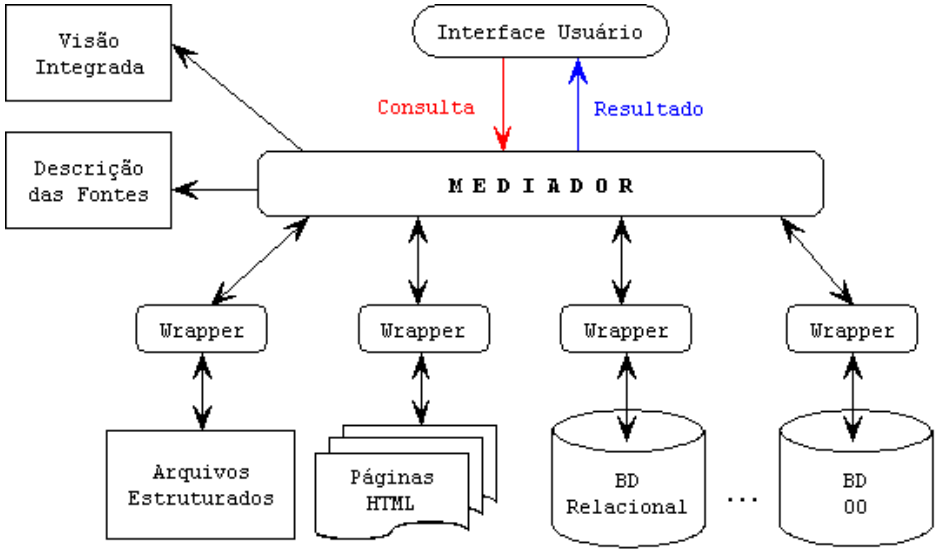


Figura 2.2 – Arquitetura de mediadores

Os mediadores oferecem uma visão integrada sobre os dados distribuídos em múltiplas fontes de dados e disponibilizam um esquema para essa visão, bem como descrições das fontes de dados.

O acesso aos dados das fontes é feito através de consultas que são submetidas ao sistema de integração de dados pelo mediador sobre o esquema integrado. O mediador recebe uma

consulta e decompõe a mesma em várias subconsultas menores direcionadas às fontes de dados. Essas subconsultas deverão ser traduzidas para os formatos/linguagens de consultas suportados pelas fontes individuais. As fontes de dados, por sua vez, recebem as subconsultas traduzidas, processam e retornam o seu resultado para o mediador. O trabalho de tradução mediador/fonte/mediador é feito pelo módulo de software *wrapper*. Programas *wrapper*, ou tradutores, convertem os dados originais das fontes para um modelo de dados comum e também traduzem consultas do mediador em consultas específicas da fonte de dados.

A arquitetura de mediadores em sistemas de integração de dados permite que usuários tenham acesso às informações sempre em seu estado mais atualizado. Essa arquitetura se mostra adequada nas seguintes situações:

- Quando existe a necessidade de recuperação em tempo real de informações com um alto grau de modificações, ou seja, altamente dinâmicas;
- Em consultas freqüentes a uma grande quantidade de fontes de dados autônomas e distribuídas.

Por outro lado, a implementação em questão é ineficiente quando:

- Fontes de dados freqüentemente ficam inacessíveis;
- Os passos de tradução e integração em tempo real são lentos e muito caros.

2.3 Sistemas de Integração de Dados

Essa subseção dedica-se a mostrar algumas pesquisas feitas na área de integração de dados que resultaram na criação de sistemas implementados com uso da arquitetura de mediadores e da arquitetura de *data warehouse*.

Dos sistemas abordados aqui, o sistema WHIPS implementa arquitetura de *data warehouse*, e os sistemas TSIMMIS e MIX implementam arquitetura de mediadores.

2.3.1 O Projeto WHIPS

O projeto WHIPS [Wiener96, Labio97, Hammer95], concebido na Universidade de Stanford é um exemplo de arquitetura de *data warehouse* para integração de dados.

O projeto WHIPS apresenta uma arquitetura de *data warehouse* voltada principalmente para a construção de dois componentes que interagem com o *data warehouse*: o componente de análise de dados e consultas e o componente de integração de dados. O sistema que coleta dados das fontes, transforma e sumariza esses dados de acordo com as especificações do *data*

warehouse e integra-os no mesmo. A arquitetura específica do sistema WHIPS pode ser vista na Figura 2.3:

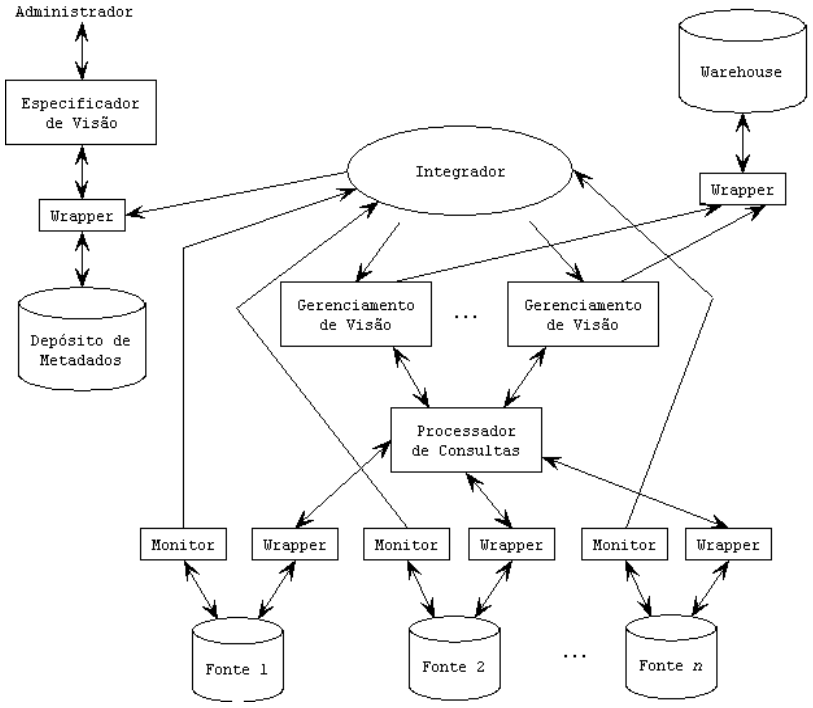


Figura 2.3 – Arquitetura do sistema WHIPS [Wiener96]

Nesse projeto foi utilizado o modelo relacional para persistir os dados no *data warehouse*: visões são definidas no modelo relacional e o *warehouse* armazena relações. Os dados das fontes subjacentes são convertidos para o modelo relacional pelos programas monitor e *wrapper* de cada fonte.

Na Figura 2.3 as setas indicam fluxo de mensagens entre os módulos e existem três operações distintas cada qual com o seu fluxo de mensagens particular:

- Primeiramente, na inicialização, os módulos devem identificar-se uns aos outros. Procedimentos semelhantes ocorrem quando uma nova fonte de dados é adicionada ao sistema;
- Em segundo lugar, vem a definição e inicialização de uma visão. Sempre que uma visão é definida pelo administrador, o especificador de visões faz uma verificação de tipos junto ao depósito de metadados e passa a definição da visão ao integrador que vai gerar um módulo de gerenciamento para a visão e instruir os monitores de todas as fontes que participam dessa visão para enviarem alterações relevantes. O módulo de gerenciamento da visão emite para o processador de consultas uma consulta correspondente à definição da visão. O processador de consultas reúne os resultados

entregues pelos *wrappers* e repassa o resultado global da consulta de volta ao gerente da visão, que por sua vez, envia esses dados ao *wrapper* do *data warehouse* para que a visão seja inicializada no DW e em seguida, o sistema é instruído a fazer a manutenção da visão;

- Terceiro, cada visão definida é atualizada em resposta a modificações que possam afetá-la. Cada monitor deve detectar as modificações que ocorrem na fonte e repassar essas modificações ao integrador. O integrador, por sua vez, as entrega aos módulos de gerenciamento das visões envolvidas e cada um irá utilizar alguns algoritmos de consistência para efetuar no DW as mudanças correspondentes das visões.

Abaixo podemos ver as principais características dos módulos do sistema WHIPS exibidos na Figura 2.3:

- **Especificador de visões** ⇒ Visões são definidas segundo o modelo relacional. Quando uma visão é definida, o especificador de visões efetua um "*parsing*" e transforma a visão em uma estrutura interna de representação relacional denominada "árvore da visão" (similar à álgebra relacional), adiciona informações relevantes originárias do depósito de metadados (ex.: atributos de chave, tipos de atributos) e envia a árvore da visão para o integrador;
- **Depósito de metadados** ⇒ O depósito de metadados mantém catalogadas informações sobre as fontes e como acessá-las; relações armazenadas em cada fonte de dados e um esquema para cada uma dessas relações. Também mantém informações a respeito de todas as definições de visões;
- **Integrador** ⇒ O integrador coordena tanto a inicialização do sistema como procedimentos para adicionar novas fontes de dados e inicialização de visões. A principal tarefa do integrador é facilitar a manutenção de visões determinando quais modificações devem ser propagadas para quais visões;
- **Gerenciamento de visões** ⇒ O módulo de gerenciamento de visão responsável pela manutenção de cada uma das visões. O gerenciamento de visão utiliza algoritmos específicos para manter a consistência das visões. Uma vantagem no uso de um módulo de gerenciamento de visões para cada visão é que o trabalho de manutenção de cada visão pode ser feito em paralelo e em máquinas diferentes;
- **Processador de consultas** ⇒ O processador de consultas tem como tarefa principal o processamento distribuído das consultas submetidas aos *wrappers*. As principais vantagens de separar processamento de consultas do gerenciamento de visões são: o

gerenciamento de visões pode criar consultas globais independentes das fontes distribuídas e um único processador de consultas pode manusear consultas para vários gerenciadores de visão. Devido ao fato dos *wrappers* encapsularem aspectos de sintaxe de consultas específicos das fontes, o processador de consultas gera consultas relacionais. Após o envio da consulta à fonte de dados, o processador de consultas espera pelo retorno do resultado da mesma para então construir um resultado global;

- **Wrappers** ⇒ Cada *wrapper* é responsável por traduzir consultas sobre as fontes de dados da representação relacional interna do sistema para consultas em uma linguagem nativa da fonte de dados subjacente. A utilização de um programa *wrapper* para cada fonte de dados encapsula do processador de consultas e de todos os outros módulos do sistema detalhes de consulta específicos das fontes. Todos os *wrappers* suportam a mesma interface embora seus códigos internos sejam específicos e dependentes das fontes de dados;

- **Fontes e monitores** ⇒ Cada fonte pode ser completamente independente do *warehouse* e do sistema WHIPS. Entretanto, existe uma larga vantagem em operar com fontes cooperativas, ou seja, fontes que notificam adequadamente suas modificações ao sistema através de seus módulos monitores específicos. Como os *wrappers*, os monitores também suportam uma interface uniforme, embora seus códigos sejam exclusivos das fontes as quais eles atendem. Cada monitor é responsável por detectar modificações externas ao sistema WHIPS que ocorrem nos dados de sua fonte e essas modificações são enviadas ao integrador;

- ***Data warehouse* e *wrapper*** ⇒ O *data warehouse* na arquitetura do WHIPS pode ser implementado por qualquer banco de dados relacional. Existe um módulo *wrapper* exclusivo para o *data warehouse* que recebe todas as definições de visões e todas as modificações nos dados da visão em um formato interno e faz a tradução para a sintaxe específica dos dados do *warehouse*. Esse *wrapper* isola as particularidades do *data warehouse* de todos os outros módulos do sistema, possibilitando o uso de qualquer banco de dados relacional na implementação do DW. As modificações recebida pelo programa *wrapper* em uma única mensagem são aplicadas no *data warehouse* em uma transação efetuada com o subsistema de consultas em funcionamento e permitindo que sistema permaneça em operação ininterruptamente.

2.3.2 O Sistema de Mediação MIX

O SDSC (San Diego Supercomputer Center) e a Universidade da Califórnia em San Diego desenvolveram um sistema de mediadores com uso de XML [Bray00] – o sistema mediador MIX [Baru99, MIX02]. A arquitetura do MIX é baseada em XML para modelagem de metadados, utiliza uma linguagem de consultas própria para XML e um mediador para processar consultas em fontes de dados distribuídas.

As principais características do sistema são:

- Integração e troca de dados são feitas unicamente através da linguagem XML. Informações de esquema e de instâncias de dados são representadas por DTDs e documentos XML respectivamente. Consultas são expressas em *XMAS (XML Matching and Structuring)* [Ludascher99], que é uma linguagem de consulta para XML bastante semelhante a XML-QL [Deutsch99] com alguns recursos adicionais para efetuar agrupamentos e ordenação de estruturas para dar origem a novos objetos XML a partir de outros objetos já existentes;
- O sistema dispõe de uma interface gráfica para navegação denominada *BBQ (Blended Browsing and Query)* [Munroe00]. Essa interface é derivada da DTD da visão do mediador e incorpora recursos de navegação e consulta sobre os dados XML. Consultas complexas podem ser construídas de forma intuitiva graças às características de aninhamento próprias de XML. A BBQ provê meios gráficos de especificar aninhamento e agrupamento dos resultados das consultas;
- O processamento de consultas pode ser efetuado sob demanda, ou seja, de acordo com as preferências do usuário estabelecidas durante a navegação na visão do mediador.

Os programas *wrapper* traduzem consultas expressas em XMAS para consultas ou comandos que as fontes de dados possam entender e também traduzem resultados de consultas emitidos pelas fontes para XML. As visões XML exportadas pelos programas *wrappers* também especificam esquemas, metadados e descrições de consultas admitidas por suas respectivas fontes de dados.

A arquitetura completa do sistema pode ser observada na Figura 2.4.

O mediador mantém uma visão integrada – a visão do mediador – que é expressa na linguagem XMAS.

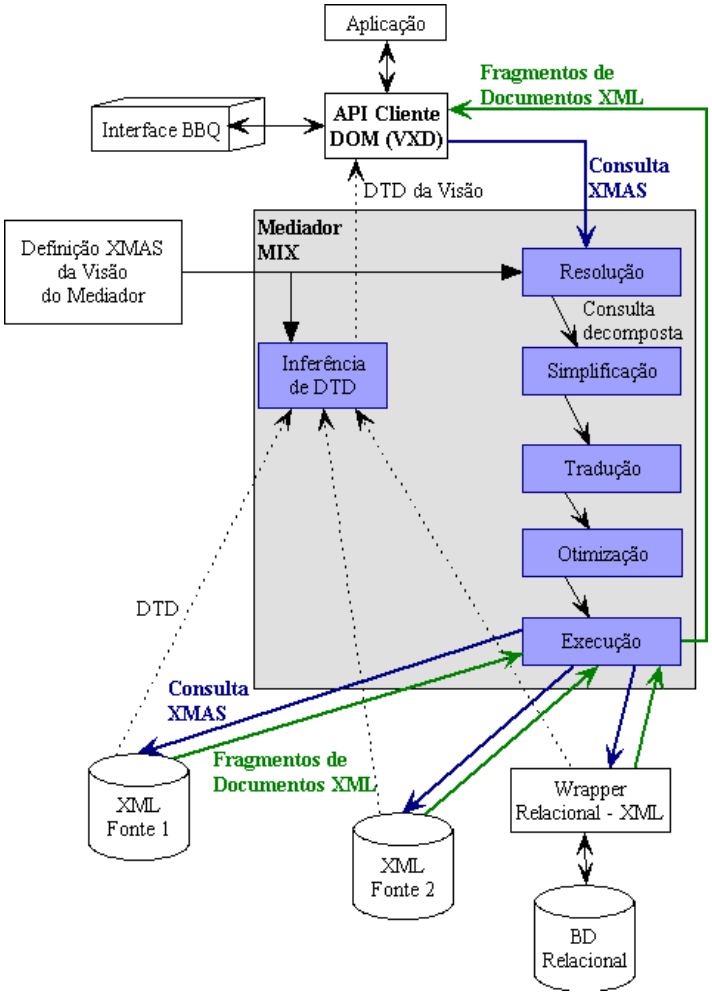


Figura 2.4 – Arquitetura do sistema MIX [Baru99]

O módulo principal do sistema é o processador de consultas que é subdividido em cinco outros módulos: resolução, simplificação, tradução, otimização e execução. O módulo de resolução é responsável por decompor a consulta submetida ao sistema em uma série de consultas menores que referenciam diretamente as visões das fontes de dados. Consultas XMAS podem ser simplificadas pelo módulo de simplificação, o qual utiliza DTDs próprias das fontes ou inferidas pelo sistema. Tarefas como avaliação e otimização de consultas podem ser aperfeiçoadas através da conversão dessas consultas para a álgebra XMAS, o que é feito pelo módulo de tradução. O módulo de otimização tem como objetivo determinar um plano de consultas eficiente. O módulo de execução, por sua vez, executa as seguintes tarefas: (i) emite consultas XMAS sobre as visões das fontes; (ii) integra as respostas emitidas pelos *wrappers* de acordo com a visão do mediador; (iii) retorna o resultado final em XML para o usuário solicitante.

O MIX também tem um módulo de inferência de DTDs para derivar DTDs integradas a partir das DTDs das fontes de dados. É sobre uma DTD integrada que a BBQ atua para fazer navegação e preparar consultas XMAS.

A interface gráfica BBQ permite que usuários formulem consultas XMAS utilizando uma GUI. A BBQ permite a construção de consultas de forma intuitiva baseadas nas DTDs da visão do mediador. O documento XML de resposta pode ser explorado através de uma API baseada em DOM [Wood00], a *DOM-VXD (DOM for Virtual XML Documents)* [Munroe00], trata-se de uma versão de DOM onde o processamento da consulta é dirigido pela navegação do cliente na visão XML virtual.

2.4 Problemas na Integração de Dados

Sistemas de integração de dados devem ter de lidar com uma série de problemas e desafios que são alvo de pesquisas na área. Nessa seção serão vistos os problemas de integração de dados mais comuns e que mais interferem no desenvolvimento desse tipo de sistema [Levy99]. A Figura 2.5 ilustra os diferentes estágios no processamento de consultas em um sistema de integração de dados:

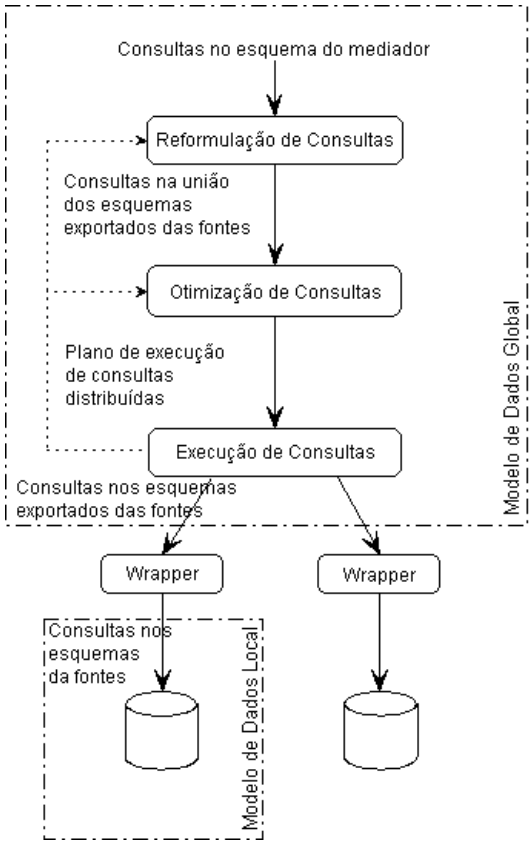


Figura 2.5 – Estágios no processamento de consultas [Levy99]

Para um melhor entendimento dos desafios intrínsecos de integração de dados, será mostrada uma análise de problemas que surgem nesse contexto relacionados com os estágios do processamento de consultas descritos na Figura 2.5.

2.4.1 Modelagem de Dados

O sistema de integração de dados deve fornecer um esquema da visão integrada de maneira que esse esquema esteja de acordo com os dados que são relevantes para os usuários. Além do esquema integrado, também devem ser disponibilizadas descrições consistentes das fontes de dados (metadados) que podem ser: a identificação da fonte de dados (ex.: nome, tipo, localização), descrição do conteúdo, restrições de conteúdo, capacidades de processamento de consultas e informações sobre o esquema da fonte. Quando descrevendo e modelando fontes de dados, o sistema deve ser capaz de lidar com as seguintes situações:

- Dados redundantes e contraditórios;
- Diferenças semânticas entre as fontes de dados, ou uma mesma informação modelada de forma diferente;
- Terminologias diferentes para os mesmos objetos.

Portanto, aspectos importantes e que devem estar definidos no sistema são as correspondências entre elementos ou objetos no esquema integrado e elementos ou objetos dos esquemas locais. Identificar correspondências entre diversos conceitos inseridos em esquemas de diferentes fontes de dados não é simples, e, naturalmente, a complexidade dessa tarefa aumenta na mesma proporção em que aumenta o número de fontes de dados. Existem algumas ferramentas que servem para auxiliar no processo de descoberta de correspondências: assertivas de correspondências [Vidal01], dicionários para identificação de sinônimos [Vidal01], ou até mesmo dicionários mais sofisticados e bases de conhecimento de conceitos e relacionamentos entre conceitos (ontologias) [Wache01] para determinar termos do domínio da aplicação e como estes se relacionam entre si. Assertivas de correspondência são tipos especiais de restrições de integridade usadas para especificar como parte(s) de um esquema estão semanticamente ligadas com outra(s) parte(s) de outro(s) esquema(s).

Um outro problema de modelagem de dados está relacionado com o grau de heterogeneidade mantido entre as fontes de dados que precisam ser integradas. Como visto anteriormente, heterogeneidades podem surgir no nível físico (ex.: plataformas de hardware e software), nível lógico (ex.: modelos de dados) e no nível conceitual (ex.: esquemas e conceitos). Por exemplo, um mesmo conceito pode ser representado de formas diferentes e para

sobrepor heterogeneidades estruturais das fontes de dados é necessário utilizar um modelo de dados comum para representação de conteúdo e estruturas de fontes de dados. A linguagem XML possui algumas características naturais para ser usada como modelo de dados comum em sistemas de integração de dados, são elas: possibilidade de definição de estruturas [Cowan01], metadados e ontologias (RDF [Lassila01] e RDF Schema [Brickley02]); APIs de acesso e manipulação de dados (ex.: DOM [Wood00]); modelo de dados, linguagens de consulta e definição de visões [Fernandez02, Boag02]; mecanismos de definição de tipos (ex.: XML Schema [Biron01]).

2.4.2 Reformulação de Consultas

Em um sistema de integração de dados, consultas são efetuadas sobre o esquema integrado e não sobre os esquemas das fontes locais. Quando a abordagem virtual é adotada, consultas sobre o esquema integrado devem ser decompostas em subconsultas direcionadas às fontes de dados. Portanto, o sistema deve possuir mecanismos para reformulação de consultas feitas sobre o esquema integrado em consultas dirigidas aos esquemas das fontes.

A reformulação de consultas pode ser feita baseando-se nas descrições das fontes de dados, estatísticas de consultas anteriormente processadas e nas correspondências mantidas entre o esquema da visão integrada e os esquemas das fontes de dados. Além de garantir que a reformulação esteja semanticamente correta, ou seja, as respostas e resultados obtidos das fontes de dados correspondem à resposta correta para uma determinada consulta, o sistema também deve garantir que fontes de dados irrelevantes para a consulta não sejam acessadas e assim minimizar problemas de desempenho.

2.4.3 Construção de Programas Wrappers (Tradutores)

A outra camada de um sistema de integração de dados é a camada de *wrappers*. Na abordagem virtual, por exemplo, diferentemente de um processador de consultas tradicional que se comunica com um gerenciamento de armazenamento local para buscar dados, o plano de execução de consultas em um sistema de integração de dados deve obter dados de fontes remotas.

Como visto anteriormente, um programa tradutor (*wrapper*) é um módulo de software que converte os dados das fontes para um modelo comum e converte consultas de aplicações em consultas específicas da fonte de dados a qual ele atende. Grandes dificuldades na construção de tradutores são provenientes do fato de que esses programas são exclusivos da fonte de dados e efetuar traduções entre modelos de dados não é uma tarefa simples. Existem algumas pesquisas

que tratam de buscar técnicas e ferramentas para automatizar a construção de tradutores [Salgado01].

2.4.4 Dados Semi-estruturados

Existem problemas que atingem várias camadas do processamento de consultas e que surgem quando há necessidade de tratar com dados semi-estruturados [Abiteboul97, Abiteboul99, Buneman97]. O termo semi-estruturado é usado para referenciar dados que não obedecem necessariamente a algum esquema rígido e pré-definido, como requerido em sistemas de bancos de dados tradicionais. Isso acontece porque os dados são muito irregulares. Por exemplo, objetos de mesmo tipo podem apresentar conjuntos de atributos variantes e por isso, o esquema que os descreve é extremamente grande. Em outros casos o esquema pode ser altamente evolutivo ou até nem mesmo declarado, ou seja, implícito nos dados. A comunidade de banco de dados vem desenvolvendo uma série de pesquisas na área de modelagem e consultas de dados semi-estruturados [Abiteboul97, Buneman97].

Dados semi-estruturados é um aspecto importante a ser verificado em sistema de integração de dados por duas razões: primeiro, em muitos casos, principalmente na Web, podemos ter fontes de dados semi-estruturados e em segundo lugar, quando integrando dados de fontes variadas de modelos diferentes, é conveniente considerar um modelo de dados que seja uma espécie de denominador comum a esses modelos. Modelos de dados para dados semi-estruturados baseados em grafos dirigidos tendem a apresentar essa propriedade. É importante notarmos que XML apresenta muitos fatores em comum com dados semi-estruturados, podendo ser usado como modelo comum.

2.5 Considerações Finais

Foram vistos aqui aspectos importantes da área de integração de dados como um todo. Os sistemas de integração de dados devem tratar heterogeneidades provenientes de diversas fontes de dados e fornecer um mecanismo de consultas uniforme para os usuários do sistema.

Existem duas abordagens clássicas para sistemas de integração de dados: abordagem virtual e abordagem materializada. A abordagem virtual é implementada pela arquitetura de mediadores e fornece consultas aos dados sob demanda. A arquitetura de *data warehouse* implementa a abordagem materializada onde os dados das fontes são previamente integrados em uma base de dados e nessa base de dados são processadas as consultas. Foram também mostrados alguns exemplos de sistemas de integração de dados de ambas as arquiteturas como o TSIMMIS, o MIX e o WHIPS.

Finalmente foram discutidas as principais dificuldades que normalmente acompanham o desenvolvimento de um sistema de integração de dados: com respeito à heterogeneidade dos dados, os problemas de modelagem que surgem; as necessidades de consultas que são processadas pelo sistema; aspectos importantes de construção de programas tradutores e necessidades de tratamento de dados semi-estruturados. Cada um desses tópicos vem sendo alvo de pesquisas dentro da área de integração de dados, uma vez que representam detalhes fundamentais a serem explorados quando da construção de um sistema de integração de dados.

Capítulo 3

Arquiteturas Híbridas

3.1 Introdução

Fontes de dados em ambientes dinâmicos naturalmente apresentam características de distribuição e autonomia bastante acentuadas. À primeira vista, pode-se pensar que um sistema de integração de dados para funcionar nesse tipo de ambiente, deveria ser projetado dentro da abordagem virtual. Isso garantiria acesso aos dados integrados sempre atualizados. Entretanto, alguns problemas podem surgir, como indisponibilidade da fonte de dados para o sistema de integração, e principalmente, o alto tempo gasto no processamento das consultas virtuais. A materialização de dados das fontes pode ser utilizada para minimizar esses problemas podendo ser viabilizada para ser implementada em conjunto com a arquitetura virtual, compondo uma arquitetura de integração de dados híbrida.

Em aplicações de materialização de dados, os dados podem ser materializados em termos de visões sobre as fontes de dados. Uma visão é uma relação derivada em termos de outras relações base. Assim, uma visão define uma função que é aplicada sobre um conjunto de tabelas base para gerar uma outra tabela derivada, e essa função é recomputada toda vez que a visão é referenciada [Gupta95].

A visão pode ser materializada através do armazenamento em um banco de dados local das tuplas que compõem a mesma, e, conseqüentemente o acesso a esses dados fica mais eficiente do que seria no caso de recomputar a visão a cada consulta. Com relação ao volume de dados que pode ser alto, deve-se ter cuidado em realizar um bom projeto físico do banco de dados que comporta as visões materializadas incluindo mecanismos de fragmentação.

O uso de visões materializadas pode incrementar a rapidez de acesso aos dados, e essa diferença de velocidade pode ser crítica em aplicações onde taxas de processamento de

consultas são altas e que possuem visões tão complexas que o fato de ter de recomputá-las a cada vez que são referenciadas inviabiliza as mesmas.

Devido à possibilidade de poder funcionar como um mecanismo otimizador de consultas, as visões materializadas podem ser um instrumento bastante útil e poderoso em sistemas de integração de dados. O uso de visões tem sido reconhecido como uma técnica importante para a integração de informações heterogêneas e distribuídas através do conceito de *visões de integração*. Visões de integração são visões definidas sobre os dados distribuídos em múltiplas fontes de dados. Como vimos no capítulo anterior, existem duas abordagens de suporte a visões de integração: a abordagem virtual e a abordagem materializada. Na abordagem virtual, as informações são extraídas das fontes de dados somente quando requisitadas em consultas e, como os dados permanecem armazenados nas fontes, diz-se que a visão de integração é uma visão *virtual*. Na abordagem materializada são definidas visões sobre dados mais críticos e essas visões são materializadas em um repositório [Pequeno00]. O enfoque híbrido pode ser usado em aplicações de larga escala onde uma parte das informações acessadas pelo sistema de integração é integrada e armazenada em um repositório (*warehouse*) e a outra parte é recuperada sob demanda diretamente das fontes de dados.

Nessa seção serão discutidos alguns sistemas de integração de dados que utilizam materialização de visões compondo uma arquitetura híbrida para integração de dados utilizando recursos da arquitetura de *data warehouse* e de *caching* em conjunto com recursos de recuperação sob demanda através do uso de mediadores.

O restante do capítulo está organizado da seguinte forma: A seção 3.2 mostra o Sistema de Materialização Seletiva de Dados [Ashish00], na seção 3.3 será mostrado o projeto IWiz [Hammer99], a seção 3.4 discute os principais aspectos do sistema Nimble [Draper01b] e a seção 3.5 traz as considerações finais a respeito do que foi discutido nas seções anteriores.

3.2 Materialização Seletiva de Dados

Uma decisão importante a ser tomada no processo de materialização de dados é selecionar visões de dados que devem ser materializadas de forma a obter a melhor combinação entre o bom desempenho no processamento das consultas e o baixo custo na manutenção dessas visões.

Este enfoque descrito em [Ashish98, Ashish99a, Ashish99b, Ashish00] trata de um *framework* para otimização de consultas efetuadas por mediadores através da materialização seletiva de dados. Materialização seletiva é um termo que está relacionado com o processo de determinar porções de dados das fontes, que podem ser materializados de forma a otimizar ao

máximo o trabalho dos mediadores. O framework criado foi inicialmente projetado para funcionar no sistema de mediadores Ariadne [Ambite98] e deu origem a um sistema de materialização seletiva de dados (SMD).

A abordagem descrita em [Ashish00] procura analisar a combinação de alguns fatores inerentes ao sistema de mediadores, mais especificamente, distribuição das consultas de usuários, estruturação de fontes de dados e custos de atualização dessas fontes para determinar porções de dados a serem materializadas, e assim, obter ganhos no tempo de resposta das consultas do mediador.

O objetivo básico é materializar localmente dados selecionados e definí-los como uma nova fonte de dados do mediador. A representação dos dados materializados é feita através de descrições semânticas e o sistema usa essa representação para determinar as partes de uma consulta que podem ser processadas e recuperadas a partir dos dados materializados [Ashish99a].

O SMD apresenta uma visão integrada das fontes de dados que é denominada *modelo de domínio*. As fontes de informação individuais são descritas em termos do modelo de domínio. Um exemplo do modelo é mostrado na Figura 3.1:

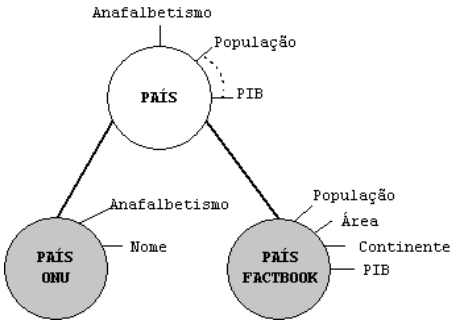


Figura 3.1 – Modelagem das fontes de dados [Ashish00]

O círculo definido como `PAÍS` representa um conceito do domínio e os círculos abaixo dele representam as fontes de dados originais. Tanto os conceitos como as fontes apresentam atributos (ex.: `População`, `Área`, `Analfabetismo`, etc.). No exemplo, o conceito de domínio `PAÍS` define uma visão integrada de duas fontes de informações de países: `PAÍS-FACTBOOK` e `PAÍS-ONU`. As consultas são efetuadas sobre o domínio e o processador de consultas gera os planos para recuperar dados das fontes envolvidas.

O sistema seleciona conceitos de domínio para materialização. Por exemplo, em um sistema de informações a respeito de países e supondo que seja detectada pelo SMD a necessidade de materializar a classe de informação *"população e produto interno bruto dos países europeus"*, a materialização será efetuada em uma nova fonte de informações (`PAÍS-EUROPEU-CACHE`)

como mostra a Figura 3.2 e o mediador deverá recuperar essas informações a partir dos dados materializados.

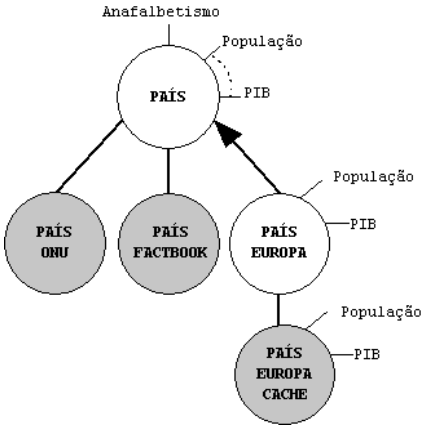


Figura 3.2 – Conceito materializado [Ashish00]

Ashish em [Ashish00] defende a hipótese de que ganhos significativos de desempenho no mediador podem ser obtidos através da materialização de uma pequena fração dos dados acessados pelo mediador e isso se dá devido a duas razões principais: primeiro, em muitas aplicações de mediadores, usuários estariam mais interessados em consultar algumas porções de dados do que outras, e assim, o SMD poderia identificar e materializar dados mais freqüentemente acessados. Em segundo lugar, algumas consultas a fontes Web podem ser extremamente dispendiosas e o sistema poderia também identificar dados que se materializados, aumentariam a velocidade de processamento dessas consultas.

Para identificar dados que são mais apropriados para serem materializados, alguns fatores são considerados e analisados:

- Classes de dados mais freqüentemente consultadas ⇒ Para identificar tais classes são levadas em conta variáveis relacionadas com a distribuição de consultas submetidas ao mediador;
- Estrutura das fontes de dados ⇒ Existem fontes na Web que não foram originalmente projetadas para serem consultadas de forma estruturada. Tais consultas podem ocorrer devido à intervenção de programas *wrappers* que fazem a tradução de consultas e formatos entre a fonte de dados e o mediador. Assim, é muito comum que certas consultas sejam muito caras para processar e seja gasto muito tempo na construção dos resultados. Devido ao excesso de processamento dos *wrappers*, a materialização dos dados subjacentes pode ajudar a melhorar sensivelmente o desempenho dessas consultas;

- Atualização das fontes de dados ⇒ Como é desejável que os usuários tenham acesso a alguns dados na sua forma mais atual, o sistema incorpora recursos para avaliar o custo de manutenção dos dados materializados quando da seleção de classes para materialização.

A próxima seção fará uma breve descrição dos módulos da arquitetura do sistema.

3.2.1 Arquitetura do Sistema de Mediação

Com base nas idéias descritas anteriormente, foi desenvolvido um sistema que executa tarefas direcionadas a otimizar consultas em sistema de mediadores através da materialização de dados [Ashish00]. O SMD inclui subsistemas para seleção de dados baseada em distribuição de consultas, estrutura das fontes de dados e aspectos de atualização, uma base local para armazenamento dos dados materializados e um módulo de manutenção da consistência desses dados.

O sistema de materialização foi projetado dentro de uma arquitetura modular onde cada um dos módulos é responsável por uma tarefa, como mostrado na Figura 3.3 e detalhado em seguida:

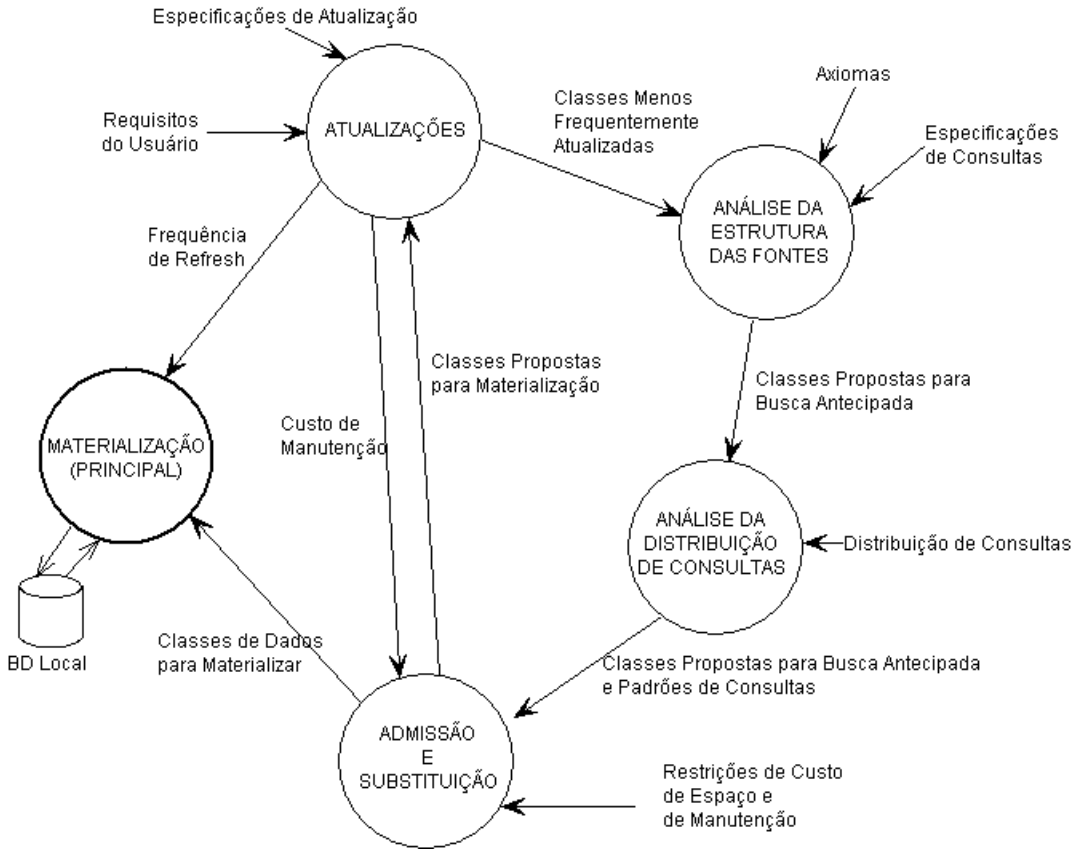


Figura 3.3 – Módulos da arquitetura do sistema [Ashish00]

O uso de uma arquitetura modular fornece meios de atribuir tarefas aos módulos dentro de um modelo conceitual de sistema mais compreensível e torna a manutenção do mesmo mais simples. Os módulos do sistema são:

- Análise da distribuição de consultas;
- Análise da estrutura das fontes de dados;
- Atualizações;
- Admissão e substituição;
- Materialização.

O módulo de atualizações inicialmente usa as especificações de atualizações para determinar que classes de dados necessitariam de atualizações muito freqüentes caso fossem materializadas, e removem essas classes da lista de classes "candidatas" à materialização. Para classes de dados que não são removidas dessa lista, o módulo de atualizações calcula o custo de manutenção das mesmas para poder computar o parâmetro de benefícios de cada uma.

O próximo módulo a entrar em execução é o módulo de análise da estrutura das fontes. Sua tarefa é identificar consultas que são dispendiosas e através do uso do conhecimento das estruturas das fontes de dados Web, identificar dados que, caso sejam previamente recuperados e materializados irão contribuir no aumento do desempenho na construção dos resultados dessas consultas. O módulo de análise da estrutura das fontes utiliza a especificação da interface gráfica do mediador para determinar quais são as consultas que podem ser processadas em uma aplicação em particular e utiliza estimativas calculadas no mediador para determinar quais dessas consultas são mais ou menos dispendiosas. Finalmente, utiliza alguns axiomas de processamento de consultas para identificar os dados que irão contribuir com melhorias no tempo de resposta das mais caras. Os axiomas são mapeamentos que informam que os dados de uma determinada classe do domínio do mediador são obtidos a partir de uma ou mais fontes de dados e, essencialmente, codificam operações (*select*, *join*, etc.) que serão executadas sobre as fontes de dados e quais são as limitações de consultas das mesmas.

A próxima etapa é o processamento do módulo de análise de distribuição de consultas, cuja tarefa é extrair padrões das classes mais acessadas dentro de um conjunto de consultas de usuários.

Os módulos de análise de estrutura das fontes e distribuição de consultas fornecem uma relação de classes "candidatas" à materialização. Entretanto, pode não ser possível materializar todas porque existem sempre limitações de espaço disponível e de recursos de manutenção de

consistência dos dados no lado do mediador. A tarefa do módulo de admissão e substituição é selecionar otimamente classes de dados para materializar, levando em conta essas restrições do mediador, e tomar decisões relacionadas com a substituição de classes já materializadas. É bastante provável que, em uma aplicação de mediador, os padrões na distribuição de consultas sofram mudanças ao longo do tempo. Em casos como esses, novas classes passam a ser consultadas com mais freqüência e o sistema de materialização deve materializá-las e eliminar classes existentes que não são mais consultadas tão freqüentemente.

O último módulo a processar na seqüência é o módulo de materialização, que é o componente principal do sistema. Ele utiliza os serviços dos módulos anteriores e interfaces para a base de armazenamento local a fim de executar as tarefas finais do processo de materialização. O módulo de admissão e substituição disponibiliza para o módulo de materialização um conjunto de classes que devem ser materializadas e este, por sua vez, recupera os dados dessas classes nas fontes e efetua a materialização dos mesmos. O módulo de materialização também interage com o mediador para informá-lo dos dados materializados. Uma outra tarefa desse módulo é, periodicamente, efetuar a manutenção dos dados materializados dentro dos intervalos apropriados de freqüência de *refresh*.

3.3 O Projeto IWiz

O trabalho descrito em [Hammer99] mostra um sistema de integração que utiliza uma abordagem híbrida de *warehousing* e mediadores. O sistema proposto fornece acesso integrado a dados heterogêneos através de uma interface comum e através de visões sobre os dados integrados definidas pelo usuário. O componente *warehouse* é usado para armazenar dados freqüentemente acessados para recuperação mais rápida. O componente de mediação suporta consultas sob demanda efetuadas diretamente nas fontes de dados quando os dados necessários não estão presentes no *warehouse*. Esse sistema denomina-se *IWiz* (*Information Wizard*) e utiliza XML como modelo de dados comum.

Algumas questões tiveram uma maior atenção e foram prioritariamente endereçadas com a criação do sistema *IWiz*:

- Modelo de dados comum $\Rightarrow$ foi escolhida a linguagem XML para representação e modelo de dados;
- Heterogeneidades semânticas $\Rightarrow$ foi desenvolvida uma classificação exaustiva para todas as heterogeneidades semânticas encontradas, e devido à característica do sistema

de integração de dados semi-estruturados a classificação foi inteiramente voltada para esse tipo de dados;

- Transformação das informações ⇒ A fim de superar as heterogeneidades semânticas, é necessária a construção de mapeamentos para transformar os dados de representações específicas das fontes para a visão global do sistema de integração. Foram criados dois processos para geração de mapeamentos, onde cada um toma como entrada os metadados do passo anterior. Cada processo gera como saída um conjunto de mapeamentos que descrevem como o esquema de entrada é transformado no esquema de saída. O primeiro processo é chamado *reestruturação de esquema* e elimina inconsistências sintáticas e semânticas do esquema da fonte com relação ao esquema global. A semântica das informações contidas no modelo de dados da fonte torna-se mais explícita e são processadas conversões de unidades usando parâmetros de medidas presentes na visão integrada. A partir desse ponto todos os esquemas das fontes de dados participantes estarão empregando a terminologia especificada no esquema global. O segundo processo é denominado *integração de esquemas* e combina os esquemas alinhados provenientes do processo anterior em um esquema da visão integrada;
- Representação de conhecimento ⇒ Foi usada uma ontologia como padrão de representação dos conceitos do domínio e dos relacionamentos entre eles.

As próximas seções irão detalhar a arquitetura do sistema e como é processada a integração dos dados.

3.3.1 Arquitetura do Sistema

O sistema *IWiz* é um *framework* onde os usuários podem projetar e gerar um esquema global para um ambiente de aplicação e, em seguida, popular uma base de dados aderente ao esquema criado com dados provenientes de uma ou mais fontes de dados através do uso de regras geradas previamente.

Nem todos os passos necessários são automatizados. A proposta é interagir diretamente com usuários para efetivar as seguintes tarefas: (1) Especificação da semântica da visão global através de uma ontologia; (2) Verificação e atualização das especificações necessárias para o processo de reestruturação que são automaticamente geradas por um algoritmo específico; (3) Verificação e atualização das especificações necessárias para o processo de integração que são automaticamente geradas por um algoritmo específico. O cenário da arquitetura do IWiz pode ser visto na Figura 3.4:

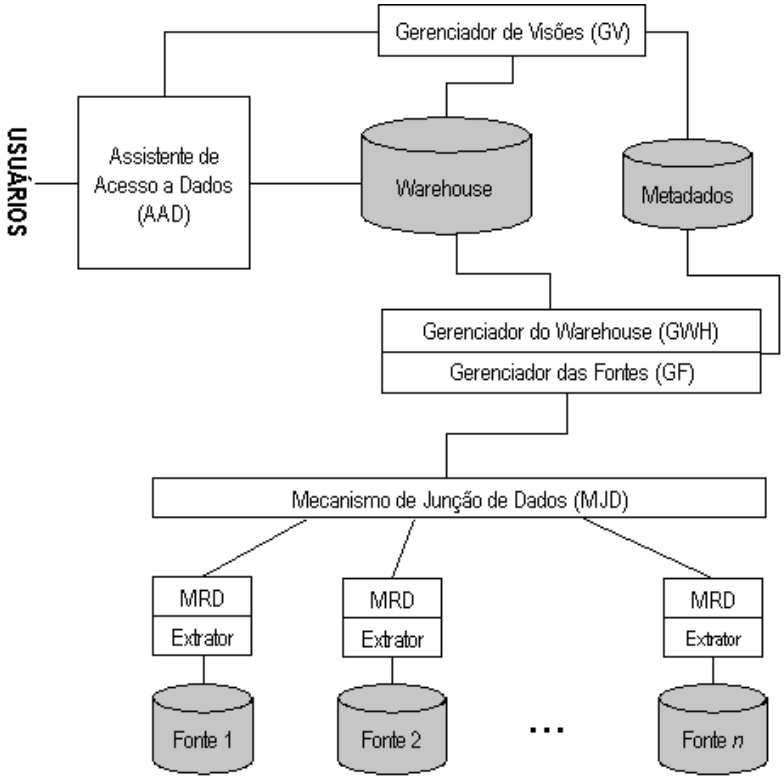


Figura 3.4 – Componentes da arquitetura do sistema IWiz [Hammer99]

As fontes são de dados semi-estruturados, e foi assumido que todos os dados serão representados em XML com DTDs correspondentes.

O mecanismo de reestruturação de dados ou MRD tem como tarefa principal transformar a estrutura e sintaxe dos documentos XML originais em documentos semanticamente equivalentes que estejam em conformidade com o esquema do *warehouse*.

Um outro componente da arquitetura é o mecanismo de junção de dados – o MJD – que faz a integração dos dados reestruturados em uma representação descrita pelo esquema global. Uma tarefa importante desse componente é o tratamento de conflitos que ocorrem tipicamente em processos desse tipo.

Os módulos gerenciadores das fontes (GF) e do *warehouse* (GWH) interagem largamente entre si. O GF deverá ser evoluído para um mediador a fim de processar consultas diretamente nas fontes de dados, e sua tarefa principal é manter um catálogo de metadados das fontes o qual é necessário para localizar dados que contribuem para o esquema global. O GWH basicamente deverá manter o conteúdo armazenado no *data warehouse*. A estratégia adotada para tal processo foi a de rematerialização completa da base de dados sendo efetuada periodicamente. O GWH e o GF interagem entre si em tarefas de manutenção e monitoração das fontes e seus

MRD's associados que devem fornecer dados para uma porção em particular do DW. Algumas interações comuns entre esses módulos são:

- Repassar a DTD global ao MRD em casos de inclusão de uma nova fonte de dados ou mesmo modificações de esquema de uma fonte existente;
- Notificações de mudanças em dados ou esquemas nas fontes de dados.

O *data warehouse* armazena dados para os vários esquemas globais (um para cada domínio de aplicação), ontologias para descrição de termos dos esquemas, dados e metadados das visões dos usuários.

Um outro componente, o gerenciador de visões (GV), tem como tarefa construir e gerenciar as visões definidas pelos usuários.

O componente AAD (Assistente de Acessos a Dados) dá facilidade aos usuários na tarefa de manipulação dos dados, o que inclui: navegar em um esquema global definido para cada domínio de aplicação, navegar na ontologia correspondente definindo termos e conceitos usados no esquema, definir visões baseadas no esquema global, navegar e submeter consultas sobre visões.

Foram estabelecidas duas fases de processamento distintas no sistema: a fase de construção onde são geradas regras de reestruturação e integração de maneira semi-automática e a fase de execução onde os dados são transportados das fontes para o *warehouse*. Na segunda fase, os usuários podem navegar e acessar os dados integrados no *data warehouse*.

Para esse sistema foram elaboradas duas versões de arquiteturas com componentes e capacidades diferentes: A versão um, como protótipo inicial, fornece apenas consultas ao *warehouse* e a versão dois, mais sofisticada, deverá fornecer suporte ao processamento de consultas distribuídas sobre as fontes de dados e *warehouse* através do uso de mediadores. O diagrama com os componentes da versão um é descrito na Figura 3.5.

Visões e consultas são elaboradas através do Assistente de Acesso a Dados (AAD) que provê uma interface gráfica para o *warehouse*. Na versão um, as operações de junção dos dados são retardadas para serem efetuadas somente quando há solicitação desses dados por parte do usuário. Em tempo de execução, quando o usuário solicita dados, o AAD e o GWH devem determinar se esses dados estão armazenados no *warehouse*, e se estiverem, serão entregues diretamente. Caso contrário, o GWH deverá invocar os módulos MRD e MJD para reestruturar e agregar apenas os itens de dados das fontes que são relevantes para a consulta.

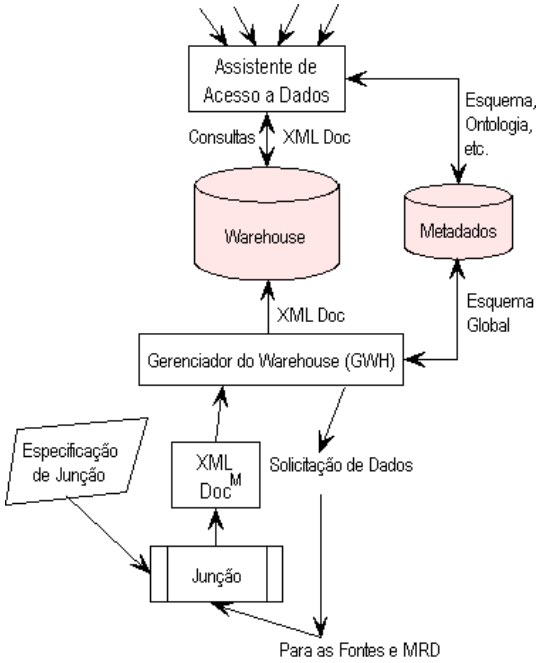


Figura 3.5 – Acesso aos dados com DW da versão um [Hammer99]

O esquema previsto para a versão dois está na Figura 3.6. A proposta é armazenar consultas mais freqüentes no *warehouse* e dados que não estão presentes no mesmo são recuperados diretamente das fontes. O módulo integrador se transformará em um módulo mediador responsável pela recuperação das informações sob a forma de consultas distribuídas. O GWH deverá direcionar as consultas (ou parte delas) que não puderem ser processadas no *warehouse* para o mediador.

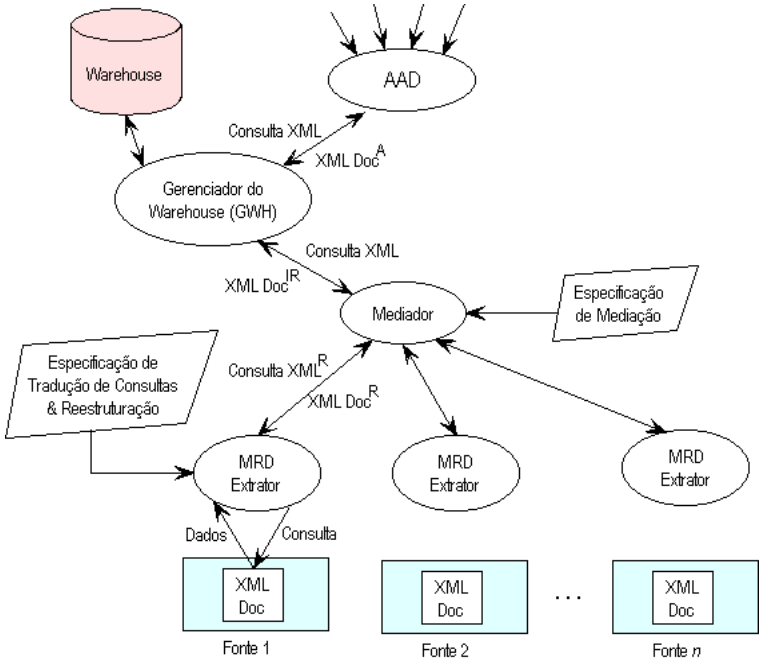


Figura 3.6 – Acesso aos dados com DW/Mediadores da versão dois [Hammer99]

3.4 O Sistema de Integração NIMBLE

O sistema de integração de dados Nimble [Draper01a, Draper01b, Nimble02] foi criado na universidade de Washington, tendo como principal objetivo, o projeto de um sistema de integração com uma arquitetura cliente-servidor tendo XML como representação dos dados.

A Figura 3.7 mostra a arquitetura do sistema Nimble:

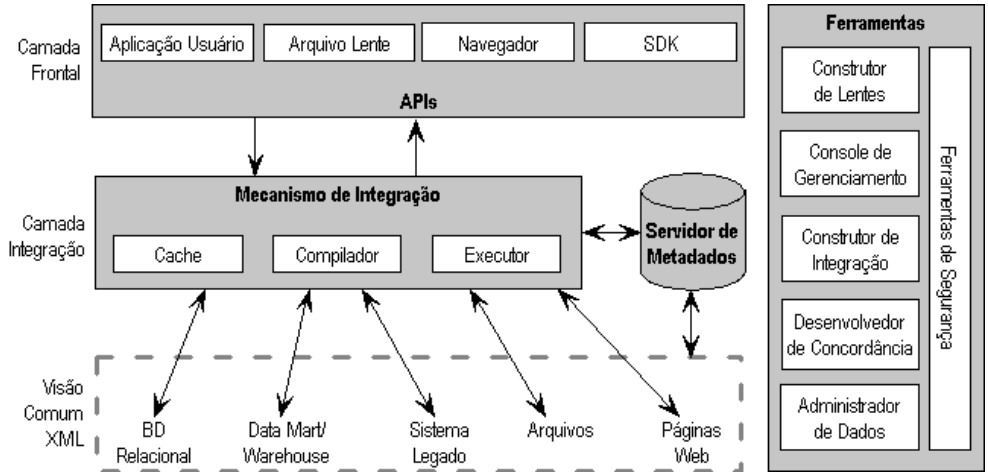


Figura 3.7 – Arquitetura do Sistema Nimble [Draper01b]

O sistema Nimble utiliza um novo tipo de conceito que é chamado arquivo *lente* [Draper01a, Nimble02]. Um arquivo lente é um formato de dados que encapsula consultas reutilizáveis que podem ser embutidas nas aplicações, ou seja, uma lente é uma consulta pré-definida combinada com informações de formatação e serve como interface primária entre usuário e os dados integrados gerados pelo mecanismo de integração. As lentes fornecem visões customizadas através das quais os usuários podem explorar e navegar nos dados subjacentes e podem ser parametrizadas para aumentar ainda mais as capacidades de customização através da possibilidade de criação de filtros nos resultados das consultas. Os arquivos lentes são criados através do módulo Construtor de Lentes e, de um ponto de vista programático, são documentos XML que contêm as seguintes informações:

- Uma consulta que pode ser parametrizada;
- Uma ou mais descrições de formatação escritas em XSLT [Clark99] para apresentação dos resultados da consulta;
- Algumas propriedades adicionais para serem utilizadas com programas que traduzem o conteúdo das lentes para a interface de usuário.

A linguagem de consulta utilizada no sistema é XML-QL [Deutsch99]. Usuários e aplicações interagem com o sistema através do esquema de mediação que é composto de definições de

visões sobre os esquemas das fontes de dados. Esses esquemas podem ser construídos de maneira hierárquica, ou seja, é possível definir sucessivos esquemas como visões sobre outros esquemas subjacentes [Draper01a].

Uma consulta XML-QL submetida ao mecanismo de integração é decomposta em fragmentos determinados de acordo com as fontes de dados envolvidas. O módulo Compilador traduz cada fragmento em um formato apropriado para endereçar cada uma das fontes de dados. O servidor de metadados contém informações de mapeamento que possibilitam a decomposição e tradução das consultas XML-QL. As informações de mapeamento são mantidas através das ferramentas de gerenciamento.

Os principais recursos que o sistema Nimble fornece são:

- Alto desempenho, processamento de consultas escalável sobre múltiplas fontes de dados;
- Mapeamento e limpeza de dados dinâmicos;
- Definição de novas consultas e visões pelos usuários;
- Uso de uma cache para armazenar localmente resultados das consultas mais freqüentes;
- XML como modelo unificador na camada interna do sistema.

Nas seções a seguir serão discutidos os principais módulos da arquitetura e os recursos mais importantes do sistema Nimble.

3.4.1 Mecanismo de Integração e Componentes

O Mecanismo de Integração Nimble (MIN) é o componente de mediação do sistema e quando recebe uma consulta encapsulada em um arquivo lente, decompõe a consulta em subconsultas direcionadas às fontes de dados envolvidas. Esse módulo determina as fontes de dados necessárias, quais dados de *cache* podem ser usados e como otimizar a consulta para selecionar o caminho mais eficiente de recuperação dos dados.

O MIN emite consultas em paralelo para as fontes de dados codificadas nas linguagens de acesso apropriadas. (ex.: uma consulta SQL é gerada se a fonte de dados é um BD relacional). Depois que as fontes retornam seus resultados, o MIN executa funções para integrar as informações e envia o resultado para o módulo Construtor de Lentes que irá tratar da formatação e apresentação final ao usuário ou aplicação.

O sistema Nimble pode acessar diferentes tipos de sistemas de armazenamento dos dados das fontes através de um protocolo próprio – o Protocolo Adaptador Nimble (PAN) – que

estabelece um padrão aberto para comunicação com uma larga variedade de tipos de armazenamentos físicos. O PAN especifica uma linguagem de consulta muito simples e de fácil implementação para a maioria das fontes de dados. Quando existe a necessidade de integrar um novo tipo de fonte de dados ao sistema, um Adaptador Nimble é criado, e esse adaptador é um componente de software que faz a tradução entre o PAN e a fonte de dados física nativa, ou seja, o adaptador executa funções de *wrapper*.

3.4.2 Ferramentas de Gerenciamento

Existem algumas ferramentas de apoio ao funcionamento geral do sistema de integração, são elas: o Construtor de Integração, Console de Gerenciamento, Servidor de Metadados, Ferramentas de Segurança e Desenvolvedor de Concordância [Nimble02].

O Construtor de Integração é a ferramenta primária usada para criar visões e mapeamentos XML. Trata-se de uma interface de usuário sofisticada que permite o mapeamento de armazenamentos físicos para visões XML e possui a habilidade de testar novas visões através da execução das mesmas e visualização dos resultados, ou mesmo exportação desses resultados para arquivo.

O módulo Console de Gerenciamento controla e monitora o funcionamento geral do sistema Nimble.

Existe também um módulo Visualizador de Metadados que gerencia todos os metadados necessários ao sistema e que estão armazenados no Servidor de Metadados. Alguns tipos de metadados são mantidos pelo sistema:

- Metadados descrevendo as características de armazenamento físico dos dados das fontes;
- Metadados a respeito de mapeamentos e visões XML;
- Metadados de objetos de concordância (correspondências) que definem processos para criação de bancos de dados de concordância;
- Metadados a respeito do conteúdo da cache.

O conjunto de módulos que engloba as Ferramentas de Segurança gerencia segurança nas conexões entre o Nimble e outros sistemas, define usuários e grupos de usuários e efetua autenticação desses últimos.

A ferramenta denominada Desenvolvedor de Concordância é responsável pela manutenção de um Banco de Dados de Concordância. Trata-se de um repositório criado para servir de apoio

ao processo de combinação de itens de dados originários de duas ou mais fontes e armazenar correspondências entre objetos do esquema do mediador e os objetos das fontes de dados.

3.4.3 Ferramentas do Usuário Final

Algumas ferramentas são fornecidas para favorecer o desenvolvimento de aplicações de integração de dados com o uso do Nimble: O Navegador de Informação, o Construtor de Lentes e as *SDKs* [Nimble02].

O Navegador de Informação pode ser usado para implementar rapidamente o conjunto de ferramentas do sistema Nimble. Trata-se de uma aplicação Web para arquivos lentes e que pode ser configurada para carregar e exibir quaisquer das lentes às quais um usuário tem acesso dentro do sistema. Cada lente é exibida em uma página Web própria onde é possível atribuir valores a parâmetros, navegar e classificar resultados de consultas fornecidos pela mesma.

O Construtor de Lentes é uma parte opcional do componente Navegador e é uma interface para construir arquivos lentes. Adicionalmente, as lentes podem apresentar resultados em formatos apropriados para dispositivos nos quais serão vistos ou para a aplicação nas quais serão disponibilizados. Por exemplo, um arquivo lente que será colocado em uma página Web, pode ser projetado para apresentar as informações em um formato compatível com o *layout* da página em questão e o mesmo arquivo de lente pode apresentar as informações de maneira diferente ao ser inserido em uma planilha Excel. Essa flexibilidade existe devido à inserção de *templates* XSLT no arquivo lente. O uso de XSLT também proporciona a possibilidade de parametrização de consultas.

As ferramentas mais importantes direcionadas para o usuário final são os *SDKs* ou *Software Development Kits*: O SDK Nimble Java e o SDK Nimble COM. Cada um deles é uma biblioteca instalada no cliente que pode ser ligada ao código da aplicação do mesmo e fornecem funcionalidades de alto nível para manipulação de lentes.

3.4.4 Cache

O sistema obedece a uma arquitetura híbrida, onde pode ser configurado para efetuar consultas sob demanda bem como consultar dados materializados localmente.

As ferramentas de gerenciamento permitem a especificação de quais fontes de dados ou consultas sobre fontes de dados devem ser materializadas em um repositório local e os dados materializados devem ser atualizados sob demanda. O processador de consulta tem conhecimento de quando deve fazer uso das cópias de dados armazenados no repositório do sistema. A estratégia de materialização aqui adotada difere de sistemas de *warehousing* no

sentido de que não existe um esquema de *data warehouse* e sim, a materialização de visões sobre o esquema de mediação. Essa capacidade do sistema fornece meios para melhoria de desempenho ao longo do tempo através da inserção de estratégias adequadas de gerenciamento dos dados materializados. Existe ainda, no estado atual em que se encontra o sistema, uma necessidade de algoritmos para decidir quais os dados de quais fontes devem ser materializados.

O processo de gerenciamento da *cache* do sistema Nimble é controlado pelo usuário administrador e se aplica a visões XML. Por exemplo, o usuário pode escolher a pré-execução de uma visão XML, o que irá gerar um documento XML, e armazenar o resultado em um BD. Assim, consultas posteriores sobre essa visão XML poderão ser direcionadas para a versão pré-computada armazenada no BD [Nimble02].

Em muitas aplicações, é muito difícil ter ao mesmo tempo todas as fontes de dados disponíveis – estas podem estar *offline* ou a conexão de rede pode não estar funcionando. Na pior das hipóteses podem existir tantas fontes de dados que a probabilidade de elas estarem todas disponíveis é quase zero. Simplesmente não é aceitável que nessas situações o sistema retorne um erro ou resultado de consulta vazio.

O sistema Nimble vem sendo adaptado para atuar inteligentemente nessas situações, além da possibilidade de o usuário administrador poder materializar dados dessas fontes, através do fornecimento de resultados parciais de consultas e indicações aos usuários de que aqueles resultados não estão completos.

3.5 Considerações Finais

Nesse capítulo foram mostrados alguns sistemas que utilizam recursos de *caching* e *data warehouse* em sistemas de mediadores para obter ganhos de desempenho no processamento das consultas submetidas ao sistema de integração de uma forma geral.

O trabalho descrito em [Ashish00] leva em consideração vários fatores do sistema, tais como distribuição de consultas do usuário, estrutura das fontes de dados, taxa de atualização das fontes de dados e espaço de armazenamento para selecionar dados a serem materializados localmente num *data warehouse*. Embora esse enfoque ofereça uma solução eficiente, os dados selecionados para materialização são inseridos no sistema de integração de dados como uma nova fonte de dados local.

Hammer [Hammer99] descreve um sistema de integração que usa XML como modelo de dados e pretende desenvolver uma arquitetura híbrida onde dados mais freqüentemente consultados são armazenados em um *data warehouse* e também acessados virtualmente por um

mediador. Um problema que esse sistema apresenta é que os passos de reestruturação e integração de dados dependem de muita interação com o usuário.

Em [Draper01b] foi desenvolvido um sistema de integração de dados também com XML como modelo de dados e recursos de configuração do sistema para efetuar consultas sob demanda bem como materializar dados localmente. O sistema faz uso de uma *cache* para armazenamento de resultados das consultas mais freqüentes. Entretanto, o sistema de *caching* é controlado pelo usuário administrador e existe a necessidade de algoritmos que definam melhor os dados (e de que fontes) devem ser materializados.

Todos esses estudos são relevantes para o nosso trabalho uma vez que, a nossa proposta de integração de dados consiste em um ambiente construído dentro dos conceitos de arquitetura híbrida onde o componente mediador pode consultar fontes de dados virtuais, dados materializados em um *data warehouse* e resultados compostos das consultas mais freqüentes armazenados em uma *cache*. Particularmente, destacamos o estudo feito em [Ashish00] que trouxe como contribuição para o nosso ambiente a concepção de selecionar dados para materializar em um *data warehouse*, ou seja, processar uma materialização de dados seletiva.

Capítulo 4

Integração de Dados com Critérios de Qualidade

4.1 Introdução

Qualidade da Informação (QI) tem se tornado um aspecto crítico na área de sistemas de informações. A popularidade e crescimento do uso de sistemas de *data warehousing* e de acesso direto a informações publicadas por várias fontes de dados, incrementou necessidades de tratamento da qualidade da informação. Entretanto, alguns problemas ainda existem e nem todas as informações são adequadas: ferramentas de verificação podem auxiliar na validação de um conjunto de dados, e ainda assim uma parte do conjunto permanecer inválida. Em geral, os dados podem apresentar baixa qualidade tanto por não refletirem a realidade, como por serem mal utilizados e mal entendidos pelo usuários. Normalmente, o custo de dados com baixa qualidade pode ser medido em termos dos requisitos de usuários. Mesmo dados precisos, se não estiverem disponíveis e dispostos de maneira em que sua interpretação seja simples, serão de pouco valor [Wang93].

Nos últimos anos, a qualidade da informação (QI) tem sido explorada em vários aspectos. Em sistemas de integração de dados, alguns estudos são voltados para a criação de metodologias de estimativas de QI em projetos de *data warehousing* [Lee02]. Outros são referentes à seleção de fontes de dados para fornecer respostas a consultas, [Naumann98a] ou para materializar dados [Zhu02].

Richard Wang em [Wang98] definiu uma lista de critérios provenientes de fontes diversas voltada para a avaliação da qualidade da informação da fonte de dados individualmente. Esse trabalho serviu de base para outros subseqüentes direcionados para integração de dados [Naumann99, Zhu02].

Nesse capítulo serão mostrados e classificados critérios de qualidade, seus métodos de avaliação e como podem ser utilizados em seleções de fontes de dados e seleção de planos de consultas em sistemas de integração e materialização de dados.

O restante do capítulo está organizado da seguinte maneira: na seção 4.2, será abordado um trabalho que classifica e define os critérios de QI; a seção 4.3 discute os métodos de tomada de decisão para tratamento de critérios na seleção de fontes de dados e na seção 4.4 será discutido um trabalho que utiliza critérios de qualidade e alguns métodos de tomada de decisão na seleção de planos de consulta em um sistema de integração.

4.2 Critérios de Qualidade da Informação (QI)

Alguns trabalhos de pesquisa consideram a qualidade da informação (QI) como sendo um dos aspectos mais importantes para integração de informações na Web, uma vez que a baixa qualidade da informação acarreta problemas para os usuários das informações distribuídas entre as fontes de dados autônomas. Muitos projetos são direcionados a esse fato através da coleta e classificação de critérios de qualidade. Naumann [Naumann00] revisa classificações existentes para critérios de QI e introduz uma nova classificação direcionada para a seleção de fontes e integração de dados, que sugere métodos e possibilidades de avaliação dos mesmos.

A qualidade da informação depende de três fatores maiores: a percepção do usuário, a informação por si só e o acesso às informações. Os três fatores são classificados como o sujeito, objeto e predicado de uma consulta e cada um deles é uma fonte para os metadados de QI ou escores dos critérios de QI. Um escore de QI é um valor associado a um determinado critério de QI. As três fontes de metadados correspondem a três classes de critérios de QI como mostrado na Figura 4.1:

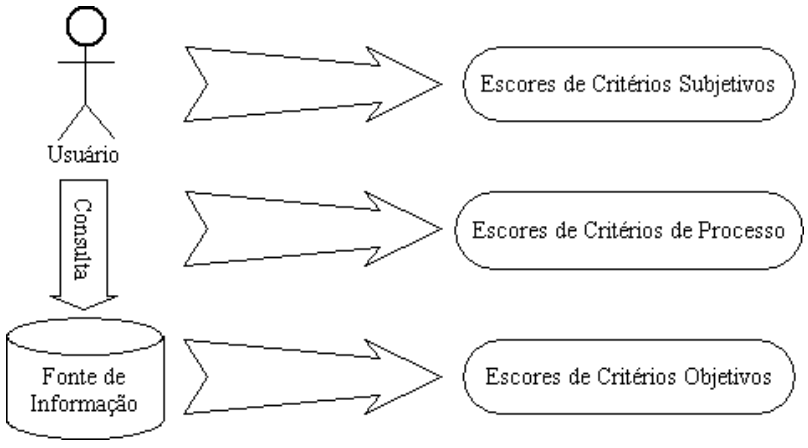


Figura 4.1 – Fontes de escores de critérios de QI [Naumann00]

As três classes estão definidas como a seguir:

- *Critérios Subjetivos*: critérios de Qualidade da Informação são classificados como critérios de sujeito ou critérios subjetivos, quando só podem ser determinados pelos usuários com base em suas visões pessoais e experiências. Um exemplo de critério subjetivo seria *interpretabilidade*;
- *Critérios de Objeto*: os escores de critérios de objeto ou critérios objetivos podem ser determinados por uma análise cuidadosa da fonte de informação, por exemplo, *completude*;
- *Critérios de Processo*: *escores* de critérios de processo apenas podem ser determinados através do processo de consultas, e assim variam de consulta para consulta e são representativos, porém temporários. Um exemplo de critério de processo seria *tempo de resposta*.

A Tabela 4.1 lista o conjunto de critérios de QI definidos em [Naumann00] dentro de cada classe e especifica o método de avaliação que deve ser aplicado para a obtenção dos escores de cada um deles.

Tabela 4.1 – Classificação de critérios de QI

Classe	Critério de QI	Método de Avaliação
Critérios de Sujeito	Possibilidade	Experiência do Usuário
	Representação Concisa	Amostragem do Usuário
	Interpretabilidade	Amostragem do Usuário
	Relevância	Avaliação Contínua do Usuário
	Reputação	Experiência do Usuário
	Compreensibilidade	Amostragem do Usuário
	Valor Agregado	Avaliação Contínua do Usuário
Critérios de Objeto	Completude	Parsing e Amostragem
	Suporte ao Usuário	Parsing e Contratação
	Documentação	Parsing
	Objetividade	Entradas de Usuário
	Preço	Contratação
	Confiabilidade	Avaliação Contínua
	Segurança	Parsing
	Atualidade	Parsing
	Rastreabilidade	Entradas de Usuário Especialista
Critérios de Processo	Precisão	Amostragem, técnicas de limpeza de dados
	Volume de Dados	Avaliação Contínua
	Disponibilidade	Avaliação Contínua
	Representação Consistente	Parsing
	Latência	Avaliação Contínua
	Tempo de Resposta	Avaliação Contínua

Critérios subjetivos devem ser fixados pelo usuário através de métodos de experiência, amostragem e avaliação contínua. Mais especificamente, os três métodos de avaliação de critérios subjetivos são:

- Experiência do Usuário: implica apenas no uso de experiências e conhecimento do usuário com relação às fontes de dados, por exemplo: relatórios e sistemas existentes;

- Amostragem do Usuário: para aplicar esse método, o usuário deve utilizar resultados de amostragens previamente efetuadas sobre a fonte de informações. Amostragens desse tipo normalmente devem ser executadas apenas uma vez e somente quando a fonte de dados sofre alterações relevantes;
- Avaliação Contínua do Usuário: trata-se de fazer uma verificação contínua das informações que o usuário recebe da fonte de informações ao longo do tempo. Esse método é o mais trabalhoso e menos compensador dos três, e deve ser aplicado quando é muito difícil conseguir extrair amostras representativas da fonte de informação.

Critérios objetivos podem ser avaliados automaticamente e apenas ocasionalmente entradas de usuário são necessárias. Escores de critérios objetivos não mudam com freqüência e em uma fonte de informação Web normalmente podem ser obtidos através de *parsing* na página principal da fonte. Os métodos de avaliação para critérios objetivos são:

- Contratação: para alguns critérios (ex: *preço* e *suporte*), os escores podem ser estimados considerando termos contratuais entre a fonte de informações e usuários consumidores;
- *Parsing*: efetuar operações de *parsing* em uma fonte de dados implica em processar os metadados da mesma. Naumann em [Naumann00] faz uma distinção entre *parsing* estrutural e *parsing* de conteúdo. *Parsing* de conteúdo considera a informação atual e o conteúdo proveniente da fonte. Aspectos como presença de *documentação* e *suporte* podem ser verificados através de uma busca por *links* de ajuda. Aspectos como *segurança* podem ser estimados por uma análise do protocolo através do qual a informação é disponibilizada. O *parsing* estrutural será discutido mais adiante juntamente com os critérios de processo;
- Amostragem: para avaliar o escore correto de alguns critérios objetivos, o conteúdo inteiro da fonte deve ser considerado. A fim de evitar essa tarefa custosa e extensa, técnicas de amostragem são aplicadas [Naumann00] para selecionar um subconjunto representativo das informações e considerar apenas esse subconjunto na determinação do escore do critério de qualidade;
- Entrada de Usuário Especialista: o conhecimento de usuários especialistas é necessário para estimar alguns critérios, por exemplo, o critério *rastreabilidade* de uma fonte de dados. Pode-se considerar um usuário especialista como sendo um administrador de sistema, por exemplo;

- Avaliação Contínua: trata-se de uma verificação contínua de como a fonte de informação se comporta para determinados critérios. Isso ocorre para o critério objetivo *confiabilidade* e também para alguns critérios de processo.

Para critérios de processo foram estipulados os seguintes métodos de avaliação:

- Escore de *precisão* deve ser obtido com o uso de técnicas de limpeza de dados muito utilizadas em *data warehouses* e em métodos de *data mining* para identificar e eliminar erros nos dados [Galhardas00];
- Avaliação contínua: alguns critérios são passíveis de sofrer mudanças constantes. Por exemplo, *latência* depende da carga da rede em determinados períodos do dia; *disponibilidade* depende de detalhes de software e hardware da fonte de informação. Portanto para tais casos, utiliza-se avaliação contínua para medir escores em intervalos regulares de tempo, construir um histórico e calcular o valor do escore com base nesse histórico;
- *Parsing*: para critérios de processo utiliza-se *parsing* estrutural, que considera estruturas da informação, tais como, posicionamento de campos em tabelas, presença de gráficos, tipos de tabelas.

4.2.1 Definição e Avaliação dos Critérios

Os critérios de QI originalmente identificados e definidos por Wang e Strong em [Wang96] são agregados em quatro categorias e quinze dimensões onde cada dimensão corresponde a um critério, conforme a Tabela 4.2:

Tabela 4.2 – Categorias e dimensões de QI

Categorias de QI	Dimensões de QI
Intrínseca	Precisão, Objetividade, Credibilidade, Reputação
Acessibilidade	Acesso, Segurança
Contextual	Relevância, Valor Agregado, Atualidade, Completude, Volume de Dados
Representacional	Simplicidade, Facilidade de Entendimento, Representação Concisa, Representação Consistente

Critérios da categoria de QI Intrínseca capturam o fato de que a informação por si só possui qualidade; a categoria de QI Contextual enfatiza requisitos que a informação deve apresentar dentro do contexto da aplicação em questão; as categorias de QI Representacional e QI de Acessibilidade enfatizam características básicas do sistema de informação.

Como visto no início dessa seção, Naumann em [Naumann00] compilou uma lista de critérios de qualidade extraídos de projetos diferentes, principalmente de Wang [Wang96], e adaptou essa lista para classificação das informações das fontes em sistemas de integração de dados. A cada um deles está associada uma função de cálculo ou um escore mensurável.

As definições para cada critério são listadas a seguir:

Disponibilidade: medida que indica a probabilidade de uma consulta ser respondida em um período de tempo. Está relacionada com aspectos de hardware, software e de conexão entre a fonte de dados e o mediador. Pode ser estimada com ajuda de estatísticas de consultas prévias. A *disponibilidade* é medida pela percentagem de tempo em que a fonte está accessível.

Precisão: indica o quanto os dados estão corretos, confiáveis e seguramente livres de erro. É sugerida a estimativa desse critério através de amostragem, onde um número de erros genéricos é obtido de uma pequena porção dos dados. É medido pelo quociente do número de valores corretos na fonte de dados e o número total de valores analisados.

Volume de Dados: originalmente definido como o tamanho do resultado de uma consulta. Para integração de dados, mede-se o volume pelo número de atributos desnecessários no resultado de uma consulta, ou seja, atributos não selecionados na consulta original do usuário.

Completude: a extensão em que os dados são completos e dentro das necessidades do usuário. Para medir a completude de uma fonte de dados do sistema de integração, é sugerido que o usuário dê um peso a cada atributo de uma consulta como um valor de 1 a 100 (muito importante) ou pode-se calcular um número que é o quociente entre o número de itens de um resultado de consulta e o número de itens do mundo real.

Representação Consistente: indica se os dados são sempre apresentados da mesma maneira e são compatíveis com dados previamente consultados. Também pode ser vista como sendo o grau em que a estrutura da informação de uma fonte de dados está em conformidade com informações similares de outras fontes. Os programas tradutores devem disponibilizar para o sistema de integração um esquema que seja consistente com o esquema global. A *representação consistente* indica o volume de trabalho do tradutor que é necessário para analisar arquivos, transformar unidades e escalas e traduzir identificadores. Esse critério é medido pelo tempo médio consumido para executar essas tarefas.

Preço: valor a ser pago pelo usuário para uma determinada consulta. É definido pelo provedor de informações, normalmente é medido por consulta e em moeda corrente.

Relevância: grau no qual as informações satisfazem as necessidades do usuário com redundância mínima. É medido como a percentagem de objetos do mundo real que a fonte de informações armazena.

Confiabilidade: representa o grau no qual o usuário pode confiar na informação. Normalmente é um valor entre 1 e 10. Um *escore* baixo de confiabilidade indica a existência de discrepâncias freqüentes nos dados originais.

Reputação: representa o renome e boa popularidade da fonte de dados. É um critério altamente subjetivo, e sugere-se que seja fornecido ao mediador de acordo com as preferências do usuário como um valor entre 1 (má reputação) e 10 (reputação muito boa).

Tempo de Resposta: tempo decorrente entre a submissão de uma consulta pelo usuário até o recebimento dos resultados. Normalmente é medido em segundos.

Atualidade: é considerado a "idade" da informação, ou seja, a extensão em que a atualidade dos dados é apropriada para a aplicação em questão. Para definir um escore de atualidade recorre-se a informações de atualizações fornecidas pelas fontes de dados.

Facilidade de Entendimento: ou *compreensibilidade*, representa o grau em que a informação pode ser compreendida pelo usuário claramente e sem ambigüidades. É um critério subjetivo que pode ser obtido com o auxílio de questionários e medido como um valor entre 1 (não entendível) e 10 (muito facilmente entendível).

Nessa classificação alguns critérios de Wang [Wang96] foram omitidos ou absorvidos em outros critérios por não serem tão relevantes para sistemas de integração de dados [Naumann99].

4.3 Seleção das Fontes de Dados

Para efetuar coleta e combinação de informações originárias das fontes de dados a qualidade da informação pode e deve ser levada em conta na construção da seleção das melhores fontes. Entretanto, a qualidade da informação possui múltiplas dimensões o que acarreta dificuldades em comparar diretamente fontes de dados com respeito a um conjunto de critérios e construir um *ranking* das mesmas [Naumann98a].

A seleção de fontes de dados em sistemas de integração pode ser efetuada com objetivos de materialização de dados ou mesmo utilização das fontes de maior qualidade no processamento de consultas submetidas ao sistema.

Selecionar as melhores fontes de dados de acordo com um conjunto de critérios é um problema de decisão com múltiplos atributos. Existem algumas técnicas para resolução de problemas desse tipo que podem ser aplicadas na seleção de fontes de dados voltada para a qualidade da informação [Naumann98b].

Dada uma consulta e um conjunto de fontes de dados capazes de responder à parte da consulta, o problema consiste em decidir quais as fontes de dados mais adequadas que devem ser acessadas. A qualidade da fonte poderá ser medida com a avaliação de escores dos critérios.

A subseção a seguir irá mostrar o uso de quatro métodos de tomada de decisão aplicados na solução do problema de seleção de fontes de dados e, em seguida, fará uma comparação entre eles.

4.3.1 Métodos para Tomada de Decisão

Para exemplificar os métodos de tomada de decisão serão utilizados três critérios de qualidade (positivos): *compreensibilidade*, *precisão*, *disponibilidade*; e dois critérios de custo (negativos): *tempo de resposta* e *preço*. Quanto mais altos os escores de critérios de qualidade e mais baixos os escores de custo, melhor classificada estará a fonte de dados associada.

Cinco fontes de dados hipotéticas (S_1,S_2,...S_5) [Naumann98b] serão analisadas e os escores de critérios para cada uma são representados por uma matriz de decisão $D = (d_{ij})_{i,j=1,...5}$ com valores também hipotéticos como mostrado na Figura 4.2. A maioria dos métodos de tomada de decisão também requer um vetor de pesos $W = (w_1,w_2,...,w_5)$ que reflete a importância do critério individual. Os pesos w_j são definidos pelo usuário e podem receber qualquer valor, havendo apenas a obrigatoriedade da propriedade $\sum_{j=1}^{5} w_j = 1$. Assim, teremos:

$$\begin{array}{r} \\ S_1 \\ S_2 \\ D = S_3 \\ S_4 \\ S_5 \end{array} \begin{array}{c} \begin{array}{ccccc} C & P & D & T & Pr \end{array} \\ \begin{pmatrix} 5 & 22 & 20 & 5 & 0{,}5 \\ 3 & 18 & 99 & 180 & 10 \\ 10 & 10 & 50 & 10 & 0 \\ 3 & 12 & 55 & 3 & 1 \\ 10 & 10 & 35 & 10 & 0{,}1 \end{pmatrix} \end{array} \qquad W = \begin{pmatrix} 18/66 \\ 9/66 \\ 6/66 \\ 22/66 \\ 11/66 \end{pmatrix}$$

onde: C é o critério compreensibilidade;
P é o critério precisão;
D é o critério Disponibilidade;
T é o critério tempo de resposta e
Pr é o critério preço.

Figura 4.2 – Matrizes de critérios e pesos [Naumann98b]

Os quatro métodos que serão abordados são os mais populares na resolução de problemas desse tipo [Naumann98b, Zhu02]: *Ponderação Aditiva Simples*, *Técnica para Ordenar Preferências por Similaridade com a Solução Ideal*, *Processo de Análise Hierárquica* e *Análise por Envoltória de Dados*. As seções a seguir dedicam-se ao detalhamento de cada um deles.

4.3.1.1 Ponderação Aditiva Simples

O método de Ponderação Aditiva Simples (*Simple Additive Weighting – SAW*) [Naumann98b, Zhu02] é um dos mais populares e bem aceitos métodos e embora sendo o mais simples, seus resultados normalmente são bem próximos de outros obtidos com métodos mais sofisticados.

Para solucionar um problema com SAW deve-se proceder com três passos: uniformizar os *escores* para torná-los comparáveis, aplicar os pesos e somar os valores dos escores de cada fonte de dados. As equações aplicadas na uniformização dos *escores* são:

$$vij = \frac{c_{ij} - c_j^{min}}{c_j^{max} - c_j^{min}} \quad \text{para critérios de qualidade (positivos)} \quad (4.1)$$

$$vij = \frac{c_j^{max} - c_{ij}}{c_j^{max} - c_j^{min}} \quad \text{para critérios de custo (negativos)} \quad (4.2)$$

onde: c_{ij} é o valor do escore do critério j para a fonte de dados i,
c_j^{max} é o valor máximo do critério j e c_j^{min} é o valor mínimo do critério j.

Para uma determinada fonte de dados S_i o seu *escore* global é calculado pela a soma ponderada definida na equação 4.3:

$$V(S_i) = \sum_{j=1}^{5} w_j \times v_{ij} \quad \text{onde i = 1,2,...5} \quad (4.3)$$

Com a aplicação das equações de uniformização, todos os valores de escores estarão dentro do intervalo [0,1], sendo os melhores *escores* de um atributo (critério) com valor próximo de 1 e os piores com valor próximo de 0. Essa propriedade assegura a possibilidade de comparação entre os escores. Assim os cálculos para determinar o ranking das fontes de dados, seriam:

$V(S_1) = v_{11} * 18/66 + v_{12} * 9/66 + v_{13} * 6/66 + v_{14} * 22/66 + v_{15} * 11/66 = \mathbf{0,7022}$
$V(S_2) = v_{21} * 18/66 + v_{22} * 9/66 + v_{23} * 6/66 + v_{24} * 22/66 + v_{25} * 11/66 = \mathbf{0,1818}$
$V(S_3) = v_{31} * 18/66 + v_{32} * 9/66 + v_{33} * 6/66 + v_{34} * 22/66 + v_{35} * 11/66 = \mathbf{0,7941}$
$V(S_4) = v_{41} * 18/66 + v_{42} * 9/66 + v_{43} * 6/66 + v_{44} * 22/66 + v_{45} * 11/66 = \mathbf{0,5463}$
$V(S_5) = v_{51} * 18/66 + v_{52} * 9/66 + v_{53} * 6/66 + v_{54} * 22/66 + v_{55} * 11/66 = \mathbf{0,7751}$

Pela análise obtida com o uso do método SAW, o *ranking* para seleção das fontes ilustradas na matriz da Figura 4.2 seria: 1. S_3; 2. S_5; 3. S_1; 4. S_4; 5. S_2.

4.3.1.2 Técnica para Ordenar Preferências por Similaridade com a Solução Ideal

O método Técnica para Ordenar Preferências por Similaridade com a Solução Ideal (*Technique for Order Preference by Similarity to Ideal Solution – TOPSIS*) é uma abordagem voltada para encontrar uma alternativa que seja a mais próxima da solução ideal e mais afastada da solução ideal-negativa em um espaço de computação multidimensional [Zhu02]. Esse espaço de computação multidimensional é determinado pelo conjunto de critérios como sendo as dimensões. A solução ideal representa uma alternativa virtual com um conjunto dos melhores escores para cada critério e a solução ideal-negativa é uma alternativa virtual com os piores escores.

Para demonstrar o método TOPSIS utilizaremos a matriz de decisão *D* da Figura 4.2. Para aplicar o método é necessária a execução dos seguintes passos:

1. Normalização da matriz de decisão D: será feito de acordo com a equação 4.4:

$$y_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{M} x_{ij}^2}} \quad \text{onde: M é o número de fontes de dados;} \qquad (4.4)$$

x_{ij} representa o escore do j-ésimo critério para a i-ésima fonte de dados.

2. Construção da matriz normalizada ponderada WY:

$$WY = v_{ij} = w_r y_{ij} \quad \text{onde wj é o peso definido para o critério (ver Figura 4.2)}$$

Determinar as soluções ideal e ideal-negativa através das equações 4.5:

$$S^+ = \left\{(max(v_{ij}) \mid j \in J),(min(v_{ij}) \mid j \in J')\right\} \qquad (4.5)$$
$$S^- = \left\{(min(v_{ij}) \mid j \in J),(max(v_{ij}) \mid j \in J')\right\}$$

onde i = 1,2,...,M;
j ∈ J associado com critérios de qualidade e j ∈ J' associado com critérios de custo

Para o nosso exemplo teremos os seguintes valores:

S^+ = (0,175; 0,088; 0,069; 0,006; 0)
S^- = (0,052; 0,040; 0,014; 0,332; 0,166)

3. Encontrar as distâncias euclideanas de cada alternativa: nesse passo, as distâncias euclideanas entre cada alternativa e as soluções ideal e ideal-negativa podem ser calculadas da seguinte forma:

$$D_i^+ = \sqrt{\sum_{j=1}^{N}(v_{ij} - s_j^+)^2}, \text{ i = 1,2,...M} \quad (4.6)$$

$$D_i^- = \sqrt{\sum_{j=1}^{N}(v_{ij} - s_j^-)^2}, \text{ i = 1,2,...M}$$

Para o nosso exemplo teremos os seguintes valores:

$$D^+ = (0{,}194;\ 0{,}081;0{,}085;0{,}639;0{,}318)$$

$$D^- = (0{,}177;\ 0{,}058;0{,}065;0{,}327;0{,}167)$$

4. Calcular a aproximação relativa com a solução ideal: a aproximação relativa da fonte de dados S_i com o ideal é calculada como determina a equação 4.7:

$$A_i = \frac{D_i^-}{D_i^+ + D_i^-}, \text{ onde } 0 \leq \text{Ai} \leq 1, \text{ i = 1,2,3,...M} \quad (4.7)$$

Todas as alternativas são comparadas com a solução ideal e com a solução ideal-negativa. Se uma alternativa apresenta $A_i = 1$, então ela mesma é a solução ideal, em contrapartida se $A_i = 0$, essa alternativa é a solução ideal-negativa. Quanto maior o valor da aproximação relativa A_i, mais próxima da solução ideal e mais afastada da solução ideal-negativa está a fonte de dados S_i. No nosso exemplo, os valores de aproximação relativa e a classificação das fontes de dados ficaram da seguinte maneira: 1. $S_3 = 0{,}861$; 2. $S_5 = 0{,}848$ 3. $S_1 = 0{,}778$; 4. $S_4 = 0{,}729$; 5. $S_2 = 0{,}142$.

4.3.1.3 Processo de Análise Hierárquica

O método Processo de Análise Hierárquica (*Analytical Hierarchy Process – AHP*) [Saaty99] é composto de quatro passos: (i) o desenvolvimento de uma hierarquia de objetivos; (ii) comparação entre pares de objetivos; (iii) verificação da consistência das comparações; (iv) agregação das comparações obtidas.

A Figura 4.3 ilustra a hierarquia de objetivos do exemplo que está sendo analisado.

O objetivo principal da resolução do problema é selecionar e classificar as fontes de dados. Este é subdividido nas metas de custo e qualidade, as quais têm subordinados os critérios requeridos para a seleção. O nível mais baixo da hierarquia apresenta as fontes de informação [Zhu02].

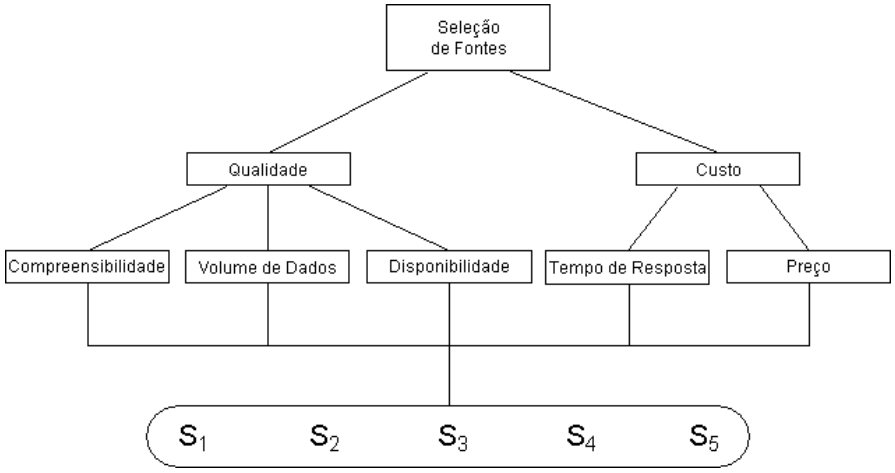


Figura 4.3 – Hierarquia de objetivos no método AHP [Naumann98b]

Após a decomposição do problema em uma hierarquia, as alternativas de um mesmo nível são comparadas em pares para avaliar a importância ou preferência relativa entre objetivos. Valores de importância são fornecidos pelo usuário. Os resultados das comparações são representados em uma matriz, onde a comparação entre uma alternativa *x* com a alternativa *y* deve resultar em um valor *x* escalonado entre 1 (mesma importância) e 9 (extremamente mais importante). Em contrapartida, a comparação de *y* com *x* resultará no valor recíproco *1/x*. A comparação entre *x* e *x* retorna o valor 1.

Seguindo essas regras, uma matriz de comparação de critérios é construída de acordo com a Tabela 4.3:

Tabela 4.3 – Matriz de comparações entre pares de critérios

Critério	Compreensibilidade	Volume	Disponibilidade	Tempo	Preço
Compreensibilidade	1	3	1/5	1/3	1/2
Volume	1/3	1	1/6	1/5	4
Disponibilidade	5	6	1	2	3
Tempo	3	5	1/2	1	2
Preço	2	1/4	1/3	1/2	1

Em seguida, serão executados os seguintes passos:

1. Normalizar cada coluna dividindo cada célula pelo somatório total da coluna;
2. Somar cada uma das linhas em uma nova coluna;
3. Normalizar a nova coluna dividindo cada valor pela soma total da coluna;
4. A nova coluna normalizada representa um vetor de pesos de cada atributo.

O resultado para a execução desses passos está na Tabela 4.4:

Tabela 4.4 – Matriz de comparações normalizada

Critério	Compreensibilidade	Volume	Disponibilidade	Tempo	Preço	∑	Peso
Compreensibilidade	0,09	0,20	0,09	0,08	0,05	0,51	0,10
Volume	0,03	0,07	0,08	0,05	0,38	0,60	0,12
Disponibilidade	0,44	0,39	0,45	0,50	0,29	2,07	0,41
Tempo	0,26	0,33	0,23	0,25	0,19	1,26	0,25
Preço	0,18	0,02	0,15	0,12	0,10	0,56	0,11
Total	1,00	1,00	1,00	1,00	1,00	5,00	1,00

O próximo passo consiste em criar uma matriz com preferências de valores para cada um dos critérios em questão. No exemplo representado pelas matrizes da Figura 4.2, essa matriz foi criada para *tempo de resposta*, como mostra a Tabela 4.5:

Tabela 4.5 – Matriz de tempo de resposta das fontes

Fonte de Dados	S_1	S_2	S_3	S_4	S_5
S_1	1	9	2	1	2
S_2	1/9	1	1/8	1/9	1/8
S_3	1/2	8	1	1/3	1
S_4	1	9	3	1	3
S_5	1/2	8	1	1/3	1

No exemplo, pode-se ver que a fonte de dados S_1 tem um tempo de resposta 9 vezes maior que S_2. Em contrapartida, a entrada da linha S_2 com a coluna S_1 deverá receber o valor oposto 1/9. A Tabela 4.6 mostra os valores das comparações de relevância entre as fontes de dados normalizados:

Tabela 4.6 – Matriz de tempo de resposta das fontes normalizada

Fonte de Dados	S_1	S_2	S_3	S_4	S_5	∑	Peso
S_1	0,32	0,26	0,28	0,63	0,28	1,77	0,35
S_2	0,04	0,03	0,02	0,02	0,02	0,12	0,02
S_3	0,16	0,23	0,14	0,07	0,14	0,74	0,15
S_4	0,32	0,26	0,42	0,21	0,42	1,63	0,33
S_5	0,16	0,23	0,14	0,07	0,14	0,74	0,15
Total	1,00	1,00	1,00	1,00	1,00	5,00	1,00

As tabelas de normalização devem ser calculadas para todos os critérios. O escore de preferência final de cada fonte é calculado pela soma ponderada dos valores ao longo do caminho na árvore que parte da fonte de dados até o objetivo principal. O valor computado pelo método AHP é determinado pela equação 4.8:

$$AHP_i = \sum_{j=1}^{N} a_{ij} w_j, \quad \text{para i = 1,2,3,...,M} \tag{4.8}$$

onde M é o número de fontes de dados; N é o número de critérios;
a_{ij} denota o escore do j-ésimo critério para a i-ésima fonte e w_j é o peso associado ao j-ésimo critério

O ranking produzido pelo método AHP foi: 1. S_3 = 0,2476; 2. S_5= 0,2215 3. S_1 = 0,2114; 4. S_4 = 0,2011; 5. S_2 = 0,1185.

4.3.1.4 Análise por Envoltória de Dados

O método Análise por Envoltória de Dados (*Data Envelopment Analysis – DEA*) [Zhu02, Naumann98a] é um método genérico utilizado para classificar uma série de observações e foi inicialmente desenvolvido para comparar a eficiência de unidades de tomada de decisão (DMU) e posteriormente projetado como uma ferramenta de suporte à decisão. No escopo de nosso exemplo, as DMU são as fontes de dados.

Diferentemente dos métodos descritos anteriormente, o método DEA não fornece um *ranking* total de classificação das fontes. Em nosso problema, a abordagem DEA pode ser empregada para encontrar uma ou mais fontes de dados com o maior escore dentro do conjunto de critérios de qualidade. No modelo DEA, o grau de eficiência de cada fonte S_{jo} é determinado pela fórmula da equação 4.9:

$$\begin{aligned} &\text{maximizar } E_{S_i} = w1 \cdot c1_{i0} + w2 \cdot c2_{i0} + w3 \cdot c3_{i0} + w4 \cdot c4_{i0} + w5 \cdot c5_{i0} \qquad (4.9)\\ &\text{satisfazendo } w1 \cdot c1_i + w2 \cdot c2_i + w3 \cdot c3_i - w4 \cdot c4_i - w5 \cdot c5_i \le 1 \ \forall \ S_i \\ &\qquad\qquad w4 \cdot c4_{i0} + w5 \cdot c5_{i0} = 1 \\ &\qquad\qquad w1, w2, w3, w4, w5 \ge \partial > 0 \end{aligned}$$

onde: E_{S_i} é o escore de eficiência da fonte de informação S_i;

$c j_i$ representa o valor do j-ésimo critério c para a fonte S_i;

w_i é o peso atribuído à fonte de dados S_i.

No modelo DEA os pesos não são especificados pelos usuários, são variáveis determinadas na execução do método. O valor ótimo para uma alternativa é 1, ou seja, a fonte de dados é eficiente, e valores menores do que 1 determinam o grau de ineficiência. Para nosso exemplo o DEA retornou os valores: 1. S_1 = 1; 2. S_3 = 1; 3. S_4 = 1; 4. S_5 = 0,9849; 5. S_2 = 0,947. Essa classificação indica que as fontes de dados S_1, S_3 e S_4 são eficientes, e as fontes de dados S_5 e S_2 são ineficientes em ordem decrescente.

4.3.1.5 Comparação entre os Métodos

Os quatro métodos aqui analisados para o exemplo adotado produziram *rankings* de classificação resumidos na Tabela 4.7.

Os métodos SAW, TOPSIS e AHP produziram a mesma classificação das fontes de dados. O método DEA produziu uma classificação diferente devido à falta de uso de pesos para os critérios e devido à sua natureza não classificatória.

Tabela 4.7 – *Rankings* das Fontes de dados

Classificação	SAW	TOPSIS	AHP	DEA
1	S_3	S_3	S_3	S_1,S_3, S_4
2	S_5	S_5	S_5	
3	S_1	S_1	S_1	
4	S_4	S_4	S_4	
5	S_2	S_2	S_2	

O método SAW é um método bastante popular e apresenta duas vantagens que é simplicidade e robustez. Seus resultados normalmente se aproximam muito de resultados produzidos por métodos mais sofisticados [Zhu02, Naumann99]. O estudo mostrado em [Azar00], que é um trabalho de comparação entre os métodos, elaborado para o domínio da Bioengenharia, também atesta essas vantagens e indica que o SAW é ainda mais robusto que o TOPSIS.

A atribuição de pesos feita pelo usuário requerida pelos métodos SAW, TOPSIS e AHP diminui a objetividade do processo de tomada de decisão. O método AHP ainda perde um pouco mais devido à matriz de preferências que é construída subjetivamente pelo usuário, o que também pode trazer inconsistências na análise dos critérios [Zhu02]. O método AHP apesar de também ser um método robusto, requer muitas comparações de pares das matrizes de decisão o que pode acarretar um processamento maior e conseqüentemente, perda de desempenho.

Naumann em [Naumann98b] sugere que os métodos SAW e TOPSIS sejam escolhidos quando a quantidade de critérios e o número de fontes a ser analisado sejam pequenos. Em caso contrário é sugerido o método DEA que efetua a seleção automaticamente sem intervenção do usuário, e assim, é considerado o mais objetivo dos quatro métodos. A grande desvantagem do método DEA é a falta de classificação das fontes quanto à sua eficiência.

4.4 Seleção de Planos de Consulta

O trabalho descrito em [Naumann99] propõe um *framework* para integração de dados voltado para a qualidade da informação. O tratamento da qualidade ocorre para cada consulta de usuário e é feito em níveis diferentes: primeiro é processada uma seleção das fontes de dados para eliminar fontes que apresentam baixa qualidade para responder à consulta. Em seguida, para as definições de visões que descrevem o conteúdo das fontes, são computados escores de qualidade. Por último, a qualidade das alternativas de planos de consultas é determinada através da agregação dos escores e é feita uma classificação desses planos a fim de encontrar um conjunto de planos com a melhor qualidade. Para demonstrar a abordagem, será utilizado um

exemplo de sistema de integração com arquitetura de mediadores, com dados a respeito de genes [Naumann99]. O esquema global é um esquema relacional mostrado na Tabela 4.8:

Tabela 4.8 – Esquema global do mediador

Relação	Descrição
gene(Gn,Do)	Nomes de genes e doenças relacionadas
sequencia(Gn,Se,Or,Cm)	Seqüências genéticas, origem e comentários
cromo(En,Cr,Po,P1,P2)	Armazena o cromossomo, posição do gene no cromossomo e as duas primeiras seqüências
gc(Gn,En)	Relaciona genes com cromossomos

O mediador considera cinco fontes de dados distintas e hipotéticas, brevemente descritas na Tabela 4.9.

Tabela 4.9 – Descrições das fontes do mediador

S_1	Armazena seqüências que são copiadas de outro sites com pouca freqüência, algumas vezes apresentando erros de *parsing*
S_2	Copia seqüências de outros sites, atualizadas com pouca freqüência, mas, utiliza uma quantidade de sites maior que S_1, portanto, sendo mais completa.
S_3	Servidor Web de uma instituição de pesquisa que produz suas próprias seqüências. Os dados são altamente atualizados, mas com poucos comentários. O servidor está freqüentemente indisponível.
S_4	Provedor comercial de dados de cromossomos. Fornece duas interfaces: *www* que é um servidor Web, *free*, com conexão lenta e somente disponibiliza a localização do cromossomo. *direct* é uma conexão SQL rápida através da qual clientes podem recuperar todos os atributos, mas para isso têm de pagar uma taxa por consulta. Seqüências iniciais são disponibilizadas para 95% dos cromossomos e posições no cromossomo para apenas 60%.
S_5	Banco de dados relacional diretamente acessível que armazena dados de mapeamentos e seqüências de genes. O esquema é diferente do esquema global do mediador, o qual usa duas consultas: a primeira relaciona genes e seqüências e a segunda relaciona os genes aos cromossomos.

Para responder às consultas dos usuários e selecionar fontes de dados, o mediador deve ter conhecimento do conteúdo de cada fonte com respeito ao esquema global. Esse conhecimento é determinado pelas consultas de mediação aqui denominadas *QCA (Query Correspondence Assertions)*, que são um conjunto de mapeamentos entre relações nas fontes de dados e relações no esquema global do mediador. Uma QCA tem a seguinte forma geral:

$$QCA = MQ \leftarrow S_i.v_j \leftarrow WQ$$

MQ é a consulta do mediador que representa um conjunto de relações que formam uma consulta que especifica os dados do esquema global que são fornecidos pela fonte. $S_i.v_j$ é uma visão qualquer da fonte S_i. *WQ* é uma consulta gerada pelo programa *wrapper* sobre o esquema exportado da fonte. As variáveis da visão devem aparecer tanto na consulta *MQ* com na consulta *WQ*.

A Tabela 4.10 mostra as QCA hipotéticas que definem correspondências entre os esquemas das fontes e o esquema global.

Tabela 4.10 – Conjunto de QCA do mediador

S_1	QCA_1	sequencia(Gn,Se,Or,Cm) ← $S_1.v_1$(Gn,Se,Or,Cm) ← seq(Gn,Se,Or,Cm)
S_2	QCA_2	sequencia(Gn,Se,Or,Cm) ← $S_2.v_1$(Gn,Se,Or,Cm) ← seq(Gn,Se,Or,Cm)
S_3	QCA_3	sequencia(Gn,Se,Or,Cm) ← $S_3.v_1$(Gn,Se,Or,Cm) ← seq(Gn,Se,Or,Cm)
S_4	QCA_4	cromo(En,Cr,-,-,-) ← $S_4.v_1$(En,Cr) ← www(En,Cr)
	QCA_5	cromo(En,Cr,Po,P1,P2) ← $S_4.v_2$(En,Cr,Po,P1,P2) ← direct(En,Cr,Po,P1,P2)
S_5	QCA_6	gene(Gn,Do), sequencia(Gn,Se,-,Cm) ← $S_5.v_1$(Gn,Do,Se,Cm) ← genes(GID,Gn,Do), partegene(GID,PID,-),parte(PID,Se,Cm)
	QCA_7	gene(Gn,-), gc(Gn,En), cromo(En,Cr,-,-,-) ← $S_5.v_2$(Gn,En,Cr,P1,P2) ← c(Gn,En,Cl), c1(Cl,Cr), primeiros(En,P1,P2)

Para responder a uma consulta de usuário, o mediador tenta encontrar combinações de QCA (planos) que estejam semanticamente contidas na consulta. São chamados de planos *corretos*. Os resultados obtidos da execução de planos diferentes podem ser mais de um e serão diferentes se acessam fontes distintas. A resposta completa para uma consulta de usuário com respeito as QCA será a união de todas as respostas obtidas para todos os planos corretos. Entretanto, pode existir um número muito grande de planos corretos. O objetivo desse trabalho é incorporar critérios de qualidade para selecionar os melhores planos.

4.4.1 Classificação dos Critérios de Qualidade

Para efetuar a seleção de fontes e planos de consulta, Naumann [Naumann99] agrupou os critérios de qualidade discutidos na seção 4.2 em três classes de níveis diferentes:

- *Critérios específicos das fontes de dados* – Determinam a qualidade de uma fonte de informação. Os escores de critérios dessa natureza permanecem imutáveis ao longo do tempo em que a fonte não sofre grandes modificações;
- *Critérios específicos de QCA* – Determinam aspectos de qualidade das consultas de mediação que são computadas a partir das fontes. Com essa granularidade, é possível, por exemplo, impor escores diferentes para consultas. No nosso exemplo, a fonte S_4 cobra uma taxa para consultas definidas pela QCA_5, e não cobra pela QCA_4;
- *Critérios específicos de consultas de usuários* – Atestam a qualidade de uma fonte de informação em termos das respostas que ela fornece às consultas de usuário. Dessa forma escores desse tipo só podem ser determinados em tempo de processamento de consulta.

A Tabela 4.11 resume e classifica os critérios de QI utilizados nesse trabalho:

Tabela 4.11 – Critérios de QI para integração de dados

Dependência	Critério	Descrição
Fonte de Dados	Facilidade de entendimento Reputação Confiabilidade Atualidade	Ranking do usuário Ranking do usuário Ranking de métodos experimentais Idade média dos dados
QCA	Disponibilidade Preço Representação Consistente Tempo de Resposta Precisão Relevância	Tempo (%) em que a fonte está disponível Valor em moeda cobrado pela consulta Carga de trabalho do wrapper Média do tempo de espera pelo resultado Percentual de objetos do mundo real sem erros Percentual de objetos do mundo real sem redundâncias
Consulta de Usuário	Completude Volume	Volume de dados completos da relação Número de atributos não solicitados

Na Tabela 4.11, a coluna *Descrição* mostra como cada critério é fornecido ao sistema de integração.

Esse trabalho propõe uma abordagem de integração de informações dirigida à qualidade e executada em três fases que são processadas sempre que uma consulta de usuário é submetida ao sistema de integração:

- Na primeira fase, é feita uma redução do custo computacional para a segunda fase através da filtragem e eliminação das fontes de dados de baixa qualidade baseada nos escores dos critérios específicos das fontes. A segunda fase será iniciada apenas com as melhores fontes de dados;
- A segunda fase usa as QCA para gerar todos os planos corretos, estabelecendo assim, o espaço de busca para a próxima fase;
- Na terceira fase ocorre a exploração do espaço de busca utilizando os critérios remanescentes e selecionando os melhores planos de execução.

Os problemas de criação de um *ranking* das fontes de dados na fase 1 e dos planos de consultas na fase 3 com base em critérios múltiplos devem ser solucionados através de métodos de tomada de decisão (seção 4.3.1). Para encontrar as melhores fontes, foi utilizado o método DEA (*Data Envelopment Analysis*) e para classificar os planos de consulta foi utilizado o método SAW (*Simple Additive Method*).

Nas próximas seções os três passos do *framework* serão detalhados através de um exemplo. O esquema global a ser utilizado é o esquema descrito na Tabela 4.8. O conteúdo de cada fonte de dados com respeito ao esquema global é descrito pelas sete QCA da Tabela 4.10. Os valores hipotéticos de escores s_{ij}, (escore *i* da fonte *j*) para critérios das fontes de dados e QCA que

serão utilizados para exemplificação ao longo das próximas seções foram obtidos de [Naumann99] e são mostrados nas Tabelas 4.12 e 4.13.

Tabela 4.12 – Escores das fontes de dados

Critério	S_1	S_2	S_3	S_4	S_5
FC (grau)	5	7	7	8	6
Rp (grau)	5	5	7	8	7
Cf (grau)	2	6	4	6	6
At (dias)	30	30	2	1	7

Tabela 4.13 – Escores das QCA

Critério	QCA_1	QCA_2	QCA_3	QCA_4	QCA_5	QCA_6	QCA_7
Di (%)	99	99	60	80	99	95	95
Pr (R$)	0	0	0	0	1	0	0
RC (Seg)	1	1	0,5	0,7	0,2	0,7	0,7
TR (Seg)	0,2	0,2	0,2	3	0,1	1	1
Pe (%)	99,90	99,90	99,80	99,95	99,95	99,95	99,95
Re (%)	60	80	90	80	80	60	60

4.4.2 Fase 1 – Seleção das Fontes de Dados

Para selecionar as melhores fontes de dados e, conseqüentemente, diminuir o custo da próxima fase serão utilizados os critérios de qualidade específicos das fontes. Fontes de dados que não são qualitativamente boas serão eliminadas. O objetivo dessa fase é encontrar um certo número ou percentual de melhores fontes independentemente de nenhuma intervenção do usuário. Para isso foi escolhido o método de tomada de decisão DEA (ver seção 4.3.1.4) o qual retornará quais fontes são "adequadas" (escore = 1) e quais são "inadequadas" (escore < 1). A equação 4.10 vai determinar o escore global de cada fonte S_{j0} com escores de critérios s_{ij}:

$$\begin{aligned} &\text{maximizar} && IQ(S_{j0}) := \textstyle\sum_i w_i \cdot s_{ij0} \\ &\text{satisfazendo a} && IQ(S_j) := \textstyle\sum_i w_i \cdot s_{ij} \le 1 \quad \text{para todas as fontes } j = 1,2,...,n \\ & && w_i \ge \partial > 0 \text{ para } i = 1,2,3,4 \end{aligned} \tag{4.10}$$

Pode-se variar o número de fontes "boas" através do ajuste do parâmetro ∂. Existe um risco de eliminação de uma fonte que tem QI baixo, mas é a única a fornecer um determinado atributo do esquema global. O algoritmo do DEA é processado repetidamente para evitar esse tipo de problema: para cada atributo do esquema global são determinadas as fontes que o fornecem e daí é extraído um conjunto de fontes adequadas. Nas fases a seguir, apenas serão consideradas as fontes obtidas a partir da união desses conjuntos.

Após o processamento do DEA, só foi excluída a fonte S_1, que é dominada por S_2, ou seja, S_2 é igual, ou melhor, do que S_1. A característica de que S_2 é mais *completa* que S_1 interferiu diretamente nesse resultado.

4.4.3 Fase 2 – Criação dos Planos de Consulta

O objetivo dessa segunda fase é encontrar todas as combinações de QCA que juntas podem retornar respostas semanticamente corretas para uma consulta de usuário. Cada QCA define uma visão no esquema do mediador.

Para exemplificar o processo de criação dos planos corretos suponhamos uma consulta de usuário solicitando *todos os genes do cromossomo X juntamente com doenças relacionadas, origem da seqüência e comentários*. O resultado para essa consulta é obtido através de um *join* da relação *gene* com a relação *sequencia* (para os atributos *origem* e *comentários*). Também deverá haver um *join* com a relação *cromo* para determinar que sejam recuperadas tuplas do cromossomo *X*:

$$UQ(Gn,Do,Se,Or,Cm) \leftarrow gene(Gn,Do) \bowtie sequencia(Gn,Se,Or,Cm) \bowtie gc(Gn,En) \bowtie cromo(En,Cr,-,-,-) \wedge Cr = \text{“X”}$$

Inicialmente, para cada relação presente na consulta de usuário (UQ) são determinadas as QCA que englobam e exportam os atributos necessários da relação de acordo com a Tabela 4.10:

$C(gene) = \{QCA_6\}$ $\quad C(sequencia) = \{QCA_2, QCA_3\}$
$C(gc) = \{QCA_7\}$ $\quad C(cromo) = \{QCA_4, QCA_5, QCA_7\}$

A QCA_7 não ocorre no conjunto da relação *gene* porque não exporta o atributo *Do*, o mesmo ocorre para a QCA_6 em *sequencia* com o atributo *Or*. A QCA_1 não aparece no conjunto da relação *sequencia* porque a fonte S_1 foi eliminada na fase 1.

Em um segundo passo, são enumerados os produtos cartesianos de todos os conjuntos e para cada combinação ocorrem as seguintes verificações: (i) Se não existem contradições, por exemplo, contradições de condições; (ii) se a combinação está semanticamente contida na *UQ*; (iii) se existem QCA redundantes. No exemplo, todas as combinações estão contidas na *UQ* e não existem contradições. Entretanto, todos os conjuntos que possuem uma mesma QCA geram planos com redundâncias. Ao final restaram os planos a seguir, cada um deles produzindo um conjunto diferente de tuplas corretas para a *UQ*:

$P_1 = QCA_6 \bowtie QCA_2 \bowtie QCA_7 \bowtie QCA_4$ $\quad P_4 = QCA_6 \bowtie QCA_3 \bowtie QCA_7 \bowtie QCA_4$
$P_2 = QCA_6 \bowtie QCA_2 \bowtie QCA_7 \bowtie QCA_5$ $\quad P_5 = QCA_6 \bowtie QCA_3 \bowtie QCA_7 \bowtie QCA_5$
$P_3 = QCA_6 \bowtie QCA_2 \bowtie QCA_7$ $\quad P_6 = QCA_6 \bowtie QCA_3 \bowtie QCA_7$

A próxima fase irá selecionar os melhores planos dentre os acima, de acordo com os critérios de qualidade.

4.4.4 Fase 3 – Seleção dos Planos de Consulta

O objetivo dessa fase é criar um *ranking* qualitativo dos planos obtidos na fase anterior e garantir que a consulta seja executada somente com os melhores planos.

Os planos de consulta podem ser representados como uma estrutura em árvore onde as folhas correspondem às QCA e os nós são as operações entre elas.

A seleção dos planos se dá em três passos: (i) os escores de QI das QCA são calculados; (ii) o modelo de qualidade agrega os escores ao longo dos caminhos da árvore para obter um escore geral da raiz da árvore; (iii) os escores das raízes são utilizados para criar o *ranking* dos planos. Nas subseções a seguir, serão detalhadas as três etapas da seleção.

4.4.4.1 Qualidade das QCA

Um vetor de tamanho 8 será associado a cada QCA, com uma ocorrência para cada critério de QCA e de consulta de usuário. Os critérios estão dispostos nessas duas categorias da seguinte forma:

- *Específicos de QCA* – são critérios que têm valor fixo para cada QCA (ver Tabela 4.13). São determinados apenas uma vez e alterados quando as fontes correspondentes sofrem modificações drásticas;
- *Específicos de consultas de usuário* – os escores desses critérios são calculados a cada nova consulta de usuário. Nessa categoria estão inseridos os critérios *completude (Com)* e *volume de dados (Vol)*. O volume de dados é o número de atributos retornados pela QCA que não são requisitados na consulta de usuário e a completude da QCA é obtida com relação aos atributos, isto é, o escore de completude de cada atributo na QCA é ponderado pela sua importância especificada pelo usuário (1-100). Os escores resultantes são somados e divididos pela soma total dos pesos. O resultado é uma média ponderada que é o escore para o critério *completude* da QCA.

O vetor de escores associado as QCA pode ser representado da seguinte forma:

$$QI(QCA_i) := (s_{i5}, s_{i6}, ..., s_{i12}) = (Di, Pr, RC, TR, Pe, Re, Com(UQ), Vol(UQ))$$

Os valores dos seis primeiros escores são extraídos da Tabela 4.13 e os escores para completude e volume de dados são calculados utilizando as consultas de usuário. Esses valores estão resumidos na Tabela 4.14:

Tabela 4.14 – Vetores de escores das QCA

	Di	Pr	RC	TR	Pe	Re	Com	V
$QI(QCA_2)$	99	0	1	0,2	99,90	80	52,80	0
$QI(QCA_3)$	60	0	0,5	0,2	99,80	90	49	0
$QI(QCA_4)$	80	0	0,7	3	99,95	80	20	1
$QI(QCA_5)$	99	1	0,2	0,1	99,95	80	20	4
$QI(QCA_6)$	95	0	0,7	1	99,95	60	48,20	0
$QI(QCA_7)$	95	0	0,7	1	99,95	60	38	3

Nesse ponto, para cada nó da árvore do plano é atribuído um vetor de QI. O próximo passo irá calcular o vetor total do plano de consulta.

4.4.4.2 Qualidade do Plano

Esse passo consiste em obter o escore total de QI de cada plano (P_i) considerado. A árvore do plano tem como folhas as QCA e os nós são as operações realizadas. Assim, o vetor para um nó da árvore é calculado de acordo com a equação a seguir:

$$QI(e \bowtie d) = QI(e) \bowtie QI(d) \quad := (s_{e5} \bowtie s_{d5}, s_{e6} \bowtie s_{d6}, \ldots, s_{e12} \bowtie s_{d12})$$

O operador $\bowtie$ deverá ser resolvido de acordo com a Tabela 4.15.

Tabela 4.15 – Funções de cálculo dos critérios de qualidade

Critério	**Função de Agregação**	**Descrição**
Di	$S_{e5} \bullet S_{d5}$	Probabilidade de ambas as fontes estarem acessíveis
Pr	$S_{e6} + S_{d6}$	Valor a se pago pelas duas consultas
RC	$\max(S_{e7}, S_{d7})$	Carga de trabalho dos *Wrappers* que operam nas fontes paralelamente
TR	$\max(S_{e8}, S_{d8})$	Ambos os nós são processados em paralelo
PE	$S_{e9} \bullet S_{d9}$	Probabilidade dos dados de ambos os lados não conter erro
Re	$S_{e10} \bullet S_{d10}$	Probabilidade de resolução do join
Com	$S_{e11} + S_{d11} - S_{e11} \bullet S_{d11}$	Probabilidade do lado direito ou esquerdo não ter valores nulos
Vol	$S_{e12} + S_{d12}$	Todos os atributos desnecessários

A árvore do plano de consulta $P_3 = QCA_6 \bowtie QCA_2 \bowtie QCA_7$ com seus respectivos vetores de escores associados a cada folha e nó da mesma está na Figura 4.4:

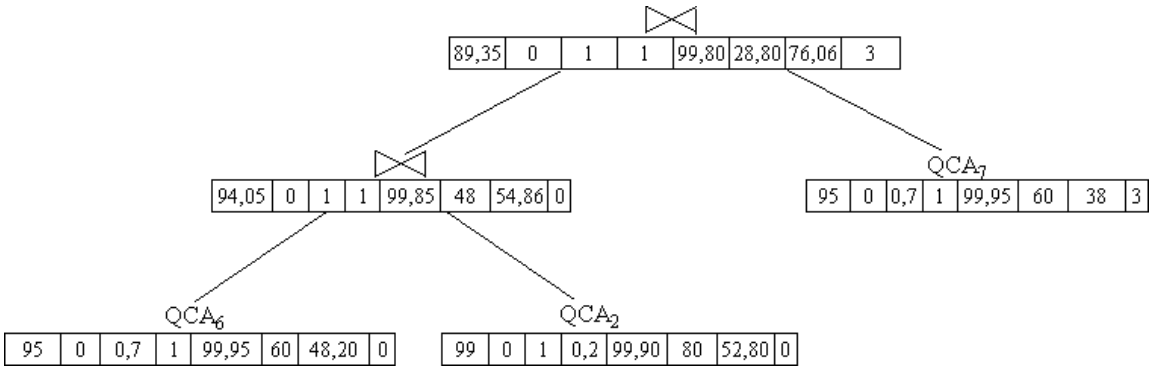


Figura 4.4 – Agregação de vetores de QI [Naumann99]

Após o processamento das funções de agregação, os seis planos de consulta do exemplo ficarão com os vetores mostrados na Tabela 4.16:

Tabela 4.16 – Vetores agregados

$QI(P_1)$	71,00	0	1	3	99,75	23,04	78,06	4
$QI(P_2)$	88,45	1	1	1	99,75	23,04	78,06	7
$QI(P_3)$	89,35	0	1	1	99,80	28,80	78,06	3
$QI(P_4)$	43,32	0	0,7	3	99,65	25,92	75,94	4
$QI(P_5)$	53,61	1	0,7	1	99,65	25,92	75,94	7
$QI(P_6)$	54,50	0	0,7	1	99,70	32,40	73,94	3

Até esse ponto, os escores são medidos com unidades distintas. O próximo passo vai tratar o escalonamento desses valores para que possam ser somados e comparados.

4.4.4.3 Classificação dos Planos

A execução do passo anterior gera como saída um conjunto de vetores de QI, um para cada plano de consulta, cujos valores são medidos em unidades diferentes. O propósito do terceiro e último passo é: escalonar, ponderar, somar e comparar os escores para encontrar um *ranking* dos planos de consultas. Para tal, será utilizado o método SAW (ver seção 4.3.1.1).

Os escores dos critérios *disponibilidade*, *precisão*, *relevância* e *completude* são denominados *positivos*, uma vez que, quanto maior seus valores, melhor é a avaliação do plano de consulta. Por outro lado, os critérios *preço*, *representação consistente*, *tempo de resposta* e *volume*, são critérios *negativos*, pois, quanto maior seus valores, mais baixa é a qualidade. Os critérios negativos são escalonados de acordo com a equação 4.11 e critérios negativos através da equação 4.12:

$$v_{ij} = \frac{s_{ij} - s_j^{min}}{s_j^{max} - s_j^{min}} \quad (4.11) \qquad v_{ij} = \frac{s_j^{max} - s_{ij}}{s_j^{max} - s_j^{min}} \quad (4.12)$$

onde s_j^{max} e s_j^{min} são os valores máximo e mínimo do critério j,
s_{ij} é o valor do critério j para o plano i e v_{ij} é o valor do critério j para o plano i escalonado.

Com o uso dessas funções de escalonamento, os valores dos escores (v_{ij}) estarão todos no intervalo [0,1], onde os melhores estão próximos de 1 e os de baixa qualidade estão próximos de 0. Essa propriedade assegura a possibilidade de comparação de *escores* de diferentes critérios.

Para o passo de ponderação, o método SAW requer um vetor de valores de pesos, $W = (w_1, w_2, ...w_m)$, um para cada critério analisado, e satisfazendo à condição $\sum_{j=1}^{m} w_j = 1$. O vetor de pesos indica a importância de cada critério individualmente e pode ser definido pelo usuário ou pela aplicação. Para um plano de consulta P_i, o seu escore global de qualidade é calculado pela soma ponderada da equação 4.13:

$$QI(P_i) = \sum_{j=1}^{m} w_j \cdot v_{ij} \quad (4.13)$$

O escore global também está no intervalo [0,1] e após todos os valores de todos os planos terem sido calculados, pode-se fazer a classificação entre eles. Para o exemplo, utilizou-se o vetor de pesos $W = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$, onde todos os critérios são considerados de igual importância. Os resultados de escores globais e o *ranking* dos planos de consultas são mostrados a seguir:

1. $QI(P_3) = 0,7663$ 2. $QI(P_6) = 0,6970$ 3. $QI(P_1) = 0,5023$

4. $QI(P_2) = 0,4559$ 5. $QI(P_4) = 0,4429$ 6. $QI(P_5) = 0,3771$

Foi concluído em [Naumann99] que, com exceção do critério *completude*, todas as funções de agregação diminuem os escores globais de QI a cada QCA adicional. Isso mostra uma tendência natural de favorecimento dos planos menores, ou seja, de planos que envolvem um número menor de QCA.

4.5 Considerações Finais

Nesse capítulo, foram detalhados alguns estudos e trabalhos de pesquisa voltados para a integração de dados, levando em consideração a qualidade da informação (QI) das fontes de dados, consultas de mediação e das consultas de usuário.

O trabalho descrito em [Wang98] define categorias de critérios que devem ser analisados para determinar a qualidade da informação proveniente de fontes de dados individuais. Naumann [Naumann00] refinou, agrupou e reclassificou esses critérios para torná-los mais adequados a aplicações de integração de dados.

Zhu [Zhu02] utiliza métodos de tomada de decisão com múltiplos atributos para selecionar as melhores fontes a serem armazenadas em um *data warehouse*. No final é feita uma comparação entre os métodos utilizados.

Em [Naumann99], critérios de qualidade são usados para selecionar as fontes de dados de um sistema de mediação mais adequadas a responder as consultas de usuário. Em seguida, é feita uma seleção dos melhores planos de consulta de mediação para restringir o espaço de busca otimizando ainda mais o processamento da consulta submetida. As seleções de fontes de dados e planos de consultas também são executadas com o uso de métodos de tomada de decisão com múltiplos atributos.

Como dito anteriormente, para otimizar o trabalho do mediador, o ambiente de integração de dados proposto deverá contar com uma porção de dados que será materializada em um *data warehouse*. Esses dados deverão ser selecionados através do uso de alguns critérios de qualidade e métodos de tomada de decisão com múltiplos atributos que foram descritos aqui, a fim de que seja garantida a materialização de dados menos atualizados e menos disponíveis. Para o ambiente de integração, dos critérios descritos em [Naumann99], foram extraídos três para efetuar a seleção dos dados a serem materializados no *data warehouse*, são eles: *atualidade*, *disponibilidade* e *tempo de resposta*. Portanto, os trabalhos aqui relacionados, mais estreitamente o de [Naumann99], têm ligação com a proposta de ambiente de integração.

Capítulo 5

Consultas com Uso de Cache e Materialização de Dados

5.1 Introdução

Cada uma das arquiteturas de sistemas de integração de dados possui vantagens e desvantagens. A arquitetura de *data warehouse* é uma boa opção para acesso rápido às informações integradas, mas não é aconselhável em aplicações onde as fontes de dados sofrem alterações com muita freqüência e assim a manutenção da consistência do *data warehouse* com os dados das fontes requer um esforço especial. A arquitetura de Mediadores fornece as informações integradas sempre atualizadas, porém pode-se ter problemas de desempenho nesse caso, devido às características de distribuição das fontes na Web.

A nossa abordagem propõe um ambiente de integração de dados com uma arquitetura baseada em mediadores com recursos adicionais de materialização de uma parte dos dados originários das fontes Web em um *data warehouse*, caracterizando uma arquitetura híbrida baseada em uma seleção criteriosa desses dados. A proposta inclui também recursos para armazenamento dos resultados das consultas de usuário mais freqüentemente processadas pelo sistema de integração em uma estrutura de *cache*.

Como vimos no capítulo 3, inserir mecanismos de materialização de dados e *caching* em um sistema de integração de dados baseado em mediadores, vem a ser um fator importante para trazer ganhos significativos de desempenho no processamento das consultas de usuários. Principalmente se for feita uma seleção criteriosa dos dados que serão materializados ou recuperados sob demanda. A nossa proposta visa trazer contribuições nesse sentido de otimização de processamento para o sistema de integração de dados. Acreditamos que o objetivo principal será atingido com a inserção dos recursos de *data warehouse* e *caching* juntos,

a fim de maximizar os ganhos de desempenho da arquitetura híbrida tradicional que normalmente utiliza uma opção ou a outra em separado.

Nesse capítulo será detalhada a arquitetura geral do sistema de integração proposto: todos os módulos, seus funcionamentos juntamente com as interações entre eles e os recursos criados para otimização de desempenho.

5.2 Visão Geral da Arquitetura do Híbrida

A proposta é a criação de um sistema de integração de dados no ambiente Web e sobre fontes de dados distribuídas, autônomas e heterogêneas. A arquitetura do sistema é baseada em uma abordagem híbrida de mediadores em conjunto com a materialização de dados em um *data warehouse*. Existem alguns diferenciais a mais na nossa proposta que são o uso de uma *cache* para armazenar resultados das consultas mais freqüentemente submetidas ao sistema e soluções para problemas relacionados com a geração e a manutenção do esquema do mediador em ambientes dinâmicos. A arquitetura geral do sistema é mostrada na Figura 5.1.

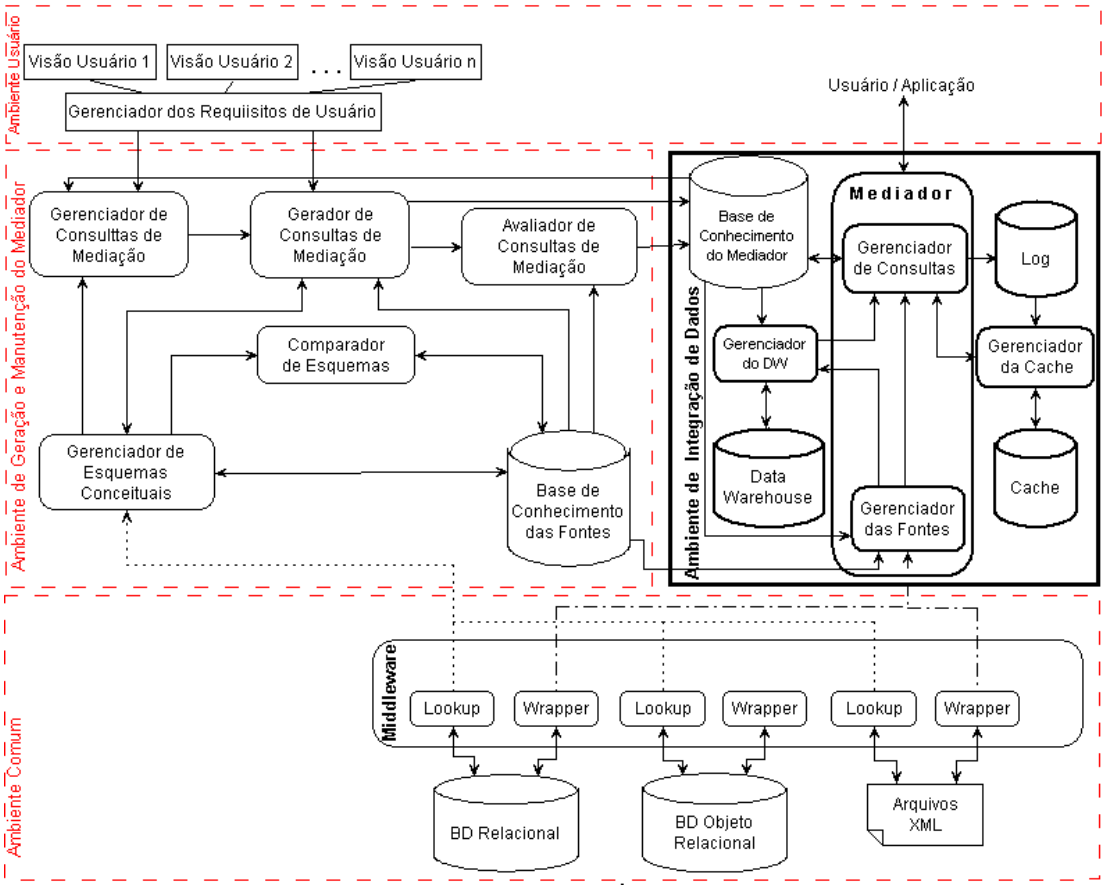


Figura 5.1 – Arquitetura do sistema

A arquitetura se divide em quatro ambientes:

- Ambiente comum: é aonde encontramos as fontes de dados que participam do sistema de integração. Esse ambiente fornece ao sistema de integração informações sobre os esquemas das fontes, recebe as subconsultas direcionadas para as fontes de dados e retorna as respostas das mesmas;
- Ambiente de integração de dados: recebe as consultas de usuário, verifica se os resultados estão na *cache*, processa a decomposição da consulta original e submete as subconsultas para as fontes de dados e/ou para o *data warehouse*. Alguns componentes desse espaço são responsáveis pela otimização do desempenho no processamento das consultas de usuário. Vale ressaltar que o escopo de nosso trabalho restringe-se a esse ambiente de integração de dados e o mesmo está destacado na Figura 5.1;
- Ambiente de geração e manutenção do mediador: é o espaço responsável pelos processos semi-automáticos de manutenção e geração das consultas de mediação. As consultas de mediação são mantidas através de um processo global de evolução que recebe eventos disparados quando ocorrem modificações nos esquemas das fontes de dados ou inserção de uma nova fonte e propaga esses eventos para o mediador;
- Ambiente de usuário: nesse ambiente são especificados e mantidos os requisitos do usuário para o esquema de mediação. A partir daqui, modificações nos requisitos são propagadas para o esquema do mediador e para os módulos responsáveis pela manutenção das consultas de mediação que compõem o esquema.

O escopo desta dissertação está relacionado com os módulos do ambiente de integração de dados. Entretanto, a seguir faremos uma breve descrição de todos os módulos dos ambientes da arquitetura proposta:

- ***Fontes de Dados***: são bases de onde o sistema de integração recebe esquemas e dados. As fontes de dados em ambientes dinâmicos normalmente suportam aplicações locais que manipulam suas informações sem haver cooperação com o nosso ambiente de integração de dados. A participação da fonte de dados no sistema de integração de dados é mantida pelo módulo *Lookup* que mantém o esquema da fonte atualizado e pelo módulo Wrapper que é responsável pelo recebimento e processamento das consultas enviadas à fonte de dados.
- ***Middleware***: o middleware é uma camada intermediária entre as fontes de dados e o sistema de integração com tarefas de tradução de dados e extração e envio de esquemas feitas pelos seguintes submódulos:

- ***Wrappers***: como citado anteriormente, um programa *Wrapper* é um componente de software que converte dados e consultas de um modelo para outro, a fim de encapsular as heterogeneidades que ocorrem no acesso a fontes de dados diversas [Liu00]. No nosso sistema, os *Wrappers* trabalham como tradutores dos dados das fontes para o padrão XML [Bray00] e transformam consultas da linguagem usada no sistema de integração para o formato nativo da fonte de dados. Nós utilizamos XML como modelo comum para troca de dados;
- ***Lookup***: o *Lookup* tem como principal tarefa analisar os esquemas exportados das fontes de dados e verificar periodicamente modificações que possam ocorrer nesses esquemas. Os esquemas no sistema de integração são representados na linguagem XML Schema [Biron01, Thompson01]. Esse módulo tem também a tarefa de traduzir o esquema definido no modelo nativo da fonte de dados para a linguagem XML Schema.

- ***Mediador***: software que fornece uma visão integrada das fontes de dados subjacentes. Um esquema para essa visão integrada é disponibilizado pelo *Mediador* e as consultas de usuários são submetidas e processadas sobre esse esquema. Para executar as consultas de usuários o Mediador utiliza dois submódulos:
  - ***Gerenciador de Consultas***: é responsável por identificar aonde os dados relevantes para responder a uma dada consulta de usuário estão armazenados, decompor a consulta original, identificar as subconsultas a serem executadas e recuperar e integrar esses dados;
  - ***Gerenciador das Fontes***: retorna respostas a consultas virtuais através do acesso aos programas *wrappers* associados às fontes de dados. Cabe também ao Gerenciador de Fontes selecionar os dados mais adequados para serem materializados no *data warehouse*.
- ***Gerenciador da Cache***: a tarefa principal desse módulo é retornar resultados de consultas armazenados na *cache*, efetuar a manutenção da *cache* com relação ao espaço ocupado pelos resultados armazenados e também com relação a operações de *refresh* de conteúdo;
- ***Gerenciador do Data Warehouse***: é responsável pela manutenção do *data warehouse* também com relação ao seu conteúdo.

- ***Gerenciador de Esquemas Conceituais***: a fim de fornecer um alto nível de abstração para as informações descritas pelo XML Schema, Bernadette Lóscio em [Lóscio03] propôs um modelo conceitual denominado *X-Entity*. Trata-se de uma extensão do ER [Chen76] para dados XML. O módulo Gerenciador de Esquemas Conceituais converte o esquema exportado pelo *Lookup* e dá como saída um esquema no formato *X-Entity*. Esse módulo também trata modificações nos esquemas das fontes.
- ***Comparador de Esquemas***: esse módulo é responsável por efetuar comparações entre esquemas *X-Entity* e identificar correspondências entre eles.
- ***Gerador de Consultas de Mediação***: a geração das consultas de mediação define o conjunto de consultas que processa uma dada entidade para o esquema do mediador. As consultas de mediação são baseadas em XML e são geradas em três etapas: (i) seleção das fontes relevantes para a entidade de mediação; (ii) identificação de possíveis operadores para aplicar entre as fontes e (iii) geração de todas as consultas de mediação possíveis com o conjunto de fontes e o conjunto de operadores. Todas essas etapas são executadas pelo módulo Gerador de Consultas de Mediação.
- ***Gerenciador das Consultas de Mediação***: é o módulo responsável pela propagação das modificações nos esquemas das fontes e evolução dos requisitos de usuário.
- ***Avaliador da Qualidade das Consultas de Mediação***: mudanças ocorridas tanto nos requisitos do usuário como nos esquemas das fontes de dados podem acarretar impactos na qualidade do esquema de mediação. Esse módulo é responsável por avaliar o impacto da propagação dessas mudanças para o sistema de integração através do uso de critérios de qualidade associados às consultas de mediação como descrito em [Naumann99].
- ***Gerenciador de Requisitos de Usuário***: esse módulo trabalha através de uma interface onde o usuário é capaz de ver as informações que as fontes subjacentes podem oferecer e definir um esquema de mediação de acordo com seus requisitos.

O sistema possui duas bases de metadados principais: a base de conhecimento das fontes de dados e a base de conhecimento do mediador. Na primeira estão armazenadas descrições das fontes de dados, esquemas exportados e correspondências entre esses esquemas. A base de conhecimento do mediador armazena descrições importantes para o mediador, o esquema de mediação em XML Schema e as consultas de mediação das entidades.

Os outros componentes do ambiente de integração de dados são: o *data warehouse*, a *cache* e o Log de Consultas submetidas ao sistema. O Log de Consultas guarda informações históricas sobre as consultas de usuário.

O ambiente proposto possui dois repositórios de dados: o *data warehouse* que armazena os dados mais indisponíveis e que sofrem menos alterações; a *cache* que armazena resultados prontos para as consultas mais freqüentemente submetidas ao sistema de integração. É importante pontuar que a construção do *data warehouse* e a construção da *cache* são operações que serão efetuadas em duas fases distintas:

(i) o *data warehouse* é criado durante a construção inicial da visão integrada e a partir daí será mantido pelo módulo *Gerenciador do Data Warehouse*. Os dados a serem materializados são selecionados a partir das fontes de dados de acordo com a avaliação de certos critérios pré-definidos;

(ii) a *cache* será populada em tempo de execução com resultados de consultas.

Em resumo, o nosso trabalho consiste no desenvolvimento de um ambiente de integração de dados na Web que tem como principais objetivos fornecer três formas de execução de consultas:

(a) Consultas a dados oriundos de fontes remotas que são processadas sob demanda;

(b) Consultas a dados materializados que serão obtidos a partir de um repositório, o *data warehouse*;

(c) Resultados das consultas de usuário mais freqüentemente submetidas ao sistema de integração armazenados em uma *cache* local.

Vale ressaltar que para uma mesma consulta de usuário os dados podem ser recuperados tanto das fontes de dados como do *data warehouse*. A próxima seção irá descrever detalhadamente o ambiente de integração de dados.

5.3 Ambiente de Integração de Dados

Como visto na seção anterior o nosso trabalho consiste em especificar o ambiente de integração de dados da arquitetura do sistema. Esse ambiente é responsável basicamente por responder às consultas de usuários submetidas a um sistema de integração de dados. Alguns recursos adicionais são inseridos em uma arquitetura baseada em mediadores [Lóscio01] para obter ganhos maiores de desempenho no processamento das consultas.

As próximas seções mostram em separado os componentes do ambiente de integração de dados: as funcionalidades de cada módulo do ambiente de integração de dados, as comunicações entre eles e recursos inseridos com fins de obter melhorias no desempenho das consultas.

5.3.1 Mediador

O Mediador é o componente de software do sistema que processa todas as consultas de usuário submetidas ao sistema de integração. Esse módulo recebe consultas XQuery [Boag02] como entrada e devolve como saída o resultado da consulta em XML. Para tal é necessário que o mediador apresente uma visão das fontes de dados e um esquema para essa visão denominado *Esquema do Mediador*. Todas as consultas de usuário são processadas sobre esse esquema.

Na arquitetura proposta para o nosso ambiente de integração de dados existem dois submódulos que compõem o Mediador: O Gerenciador de Consultas e o Gerenciador das Fontes de Dados.

Nas próximas seções serão discutidos o Esquema do Mediador, os módulos Gerenciador de Consultas e Gerenciador das Fontes de Dados.

5.3.1.1 O Esquema do Mediador

Alguns trabalhos de integração de dados podem ser classificados de acordo com a abordagem utilizada para definir os mapeamentos de mediação entre as fontes de dados e o esquema global. Uma das abordagens, denominada *GAV* ou *global as a view*, requer que cada objeto do esquema do mediador, ou *esquema global*, seja expresso como uma visão (ex.: uma consulta) sobre as fontes de dados. Na outra abordagem, denominada *LAV* ou *local as a view*, os mapeamentos são definidos de maneira oposta, onde cada objeto em uma fonte de dados é criado como uma visão no esquema global [Halevy00, Levy00, Ullman97].

No trabalho descrito em [Lóscio03], que trata da manutenção dos esquemas das fontes e do mediador, é proposto um sistema de integração baseado em mediadores que adota a abordagem GAV. Um recurso novo deste trabalho é que além de integrar os dados, ele também trata dos problemas relacionados com a geração e manutenção de consultas de mediação. Para isso considera-se um contexto específico de integração de dados, onde um esquema de mediação representa uma conciliação entre requisitos de usuários e capacidades das fontes de acordo com a Figura 5.2.

O esquema de mediação descreve uma visão integrada sobre múltiplas fontes de dados. Assim, usuários submetem consultas sobre esse esquema sem haver a necessidade de tomar

conhecimento dos esquemas das fontes de dados. Cada conceito no esquema de mediação é um conceito virtual, ou seja, os dados que compõem o esquema não estão armazenados em lugar nenhum e o esquema é computado a partir da integração dos dados distribuídos nas fontes.

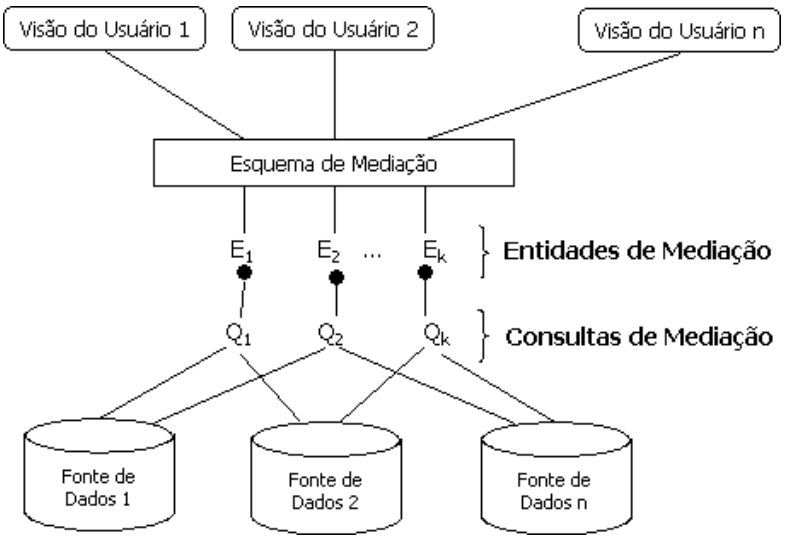


Figura 5.2 – Definição de um esquema de mediação [Lóscio03]

No nosso trabalho, utilizaremos o modelo conceitual *X-Entity* proposto em [Lóscio03] para definir o esquema de mediação. Esse modelo possui um conceito principal que é o de *entidade*. Cada entidade representa um conceito do domínio do sistema de integração (Ex.: Professor, Aluno, Disciplina). Assim, podemos dizer que o esquema de mediação deverá conter todas as entidades necessárias para responder às consultas de usuários e que podem ser computadas a partir das fontes de dados. Cada entidade E_i no esquema de mediação está associada com uma consulta de mediação Q_i que computa a entidade E_i sobre o conjunto de fontes de dados. Em abordagens GAV, uma consulta de mediação descreve como computar um elemento do esquema de mediação sobre as fontes de dados. Desta forma, as consultas de mediação expressam como obter os dados para cada entidade do esquema de mediação.

Baseado nas consultas de mediação criadas off-line, o sistema computa em tempo de execução a re-escrita mais apropriada para responder a uma consulta de usuário simplesmente através da execução das consultas de mediação necessárias e combinação de seus resultados. Nessa abordagem, as fontes relevantes para computar uma dada entidade do esquema de mediação são determinadas previamente.

Inicialmente o esquema do mediador seria criado com uso de XML Schema [Biron01, Thompson01]. A fim de fornecer uma abstração de alto nível para informações descritas em XML Schema foi proposto um esquema conceitual denominado Modelo *X-Entity*. O propósito principal desse modelo é fazer a descrição dos esquemas exportados das fontes de uma maneira

mais simples que com XML Schema. Na concepção do modelo conceitual *X-Entity* foram utilizados alguns recursos básicos do modelo E-R [Chen76] juntamente com a criação de algumas extensões para melhor representar estruturas de XML Schema. O esquema do mediador é representado por um esquema *X-Entity*, o qual é composto basicamente por entidades e relacionamentos entre elas.

Mais formalmente, pode-se dizer que a definição de uma consulta de mediação para uma entidade de mediação E_m consiste em decompor E_m em n outras entidades $T_1,...,T_n$ de forma que $E_m = T_1\ \theta_1 T_2\ \theta_2 \ldots\ \theta_{n-1} T_n$, onde θ_i é um operador binário e cada entidade T_i é derivada através do uso de uma expressão $E(S_i)$ sobre uma única fonte de dados. As entidades T_i são denominadas *entidades relevantes* para computar E_m. O processo de encontrar as consultas de mediação que definem uma entidade do mediador pode ser resumido em três passos principais:

- Selecionar as entidades relevantes que potencialmente possibilitam a construção de uma entidade de mediação;
- Identificar os operadores que podem ser aplicados entre as entidades contidas nas fontes de dados;
- Para o conjunto de entidades e operadores selecionados nos dois passos anteriores proceder com a geração de todas as consultas possíveis sobre esses conjuntos.

O processo de geração das consultas de mediação é largamente baseado na existência de metadados descrevendo tanto as fontes individuais, quanto o esquema do mediador e em assertivas semânticas descrevendo similaridades entre conceitos pertencentes a fontes distintas.

A abordagem descrita em [Lóscio03] para a geração das consultas de mediação fornece um formalismo eficiente para a representação de tais consultas. Esse formalismo é baseado no conceito de grafos de operações, os quais descrevem as fontes relevantes e as formas possíveis de combiná-las. Um grafo de operações é uma estrutura associada a uma entidade de mediação composta de entidades de mapeamento e operadores aplicados sobre essas entidades. Um exemplo de um grafo de operações para uma entidade de mediação `filme`$_m$ pode ser visto na Figura 5.3. O esquema do mediador é composto de um grafo de operações para cada consulta de mediação que está associada a uma entidade de mediação.

Entidade de Mediação

$filme_m(\{titulo_m, duracao_m, genero_m, diretor_m\}, \{ator_m\})$

Entidades de Mapeamento

$T_{filme1}(\{titulo_1, duracao_1\}, \{ator_1\})$

$T_{filme2}(\{titulo_2, genero_2\}, \{\})$

$T_{filme3}(\{titulo_3, diretor.nome_3\}, \{\})$

Assertivas Inter-esquema

$filme_1 \cap filme_2 \neq \emptyset$

$filme_1 \cap filme_3 \neq \emptyset$

$filme_2 \cap filme_3 \neq \emptyset$

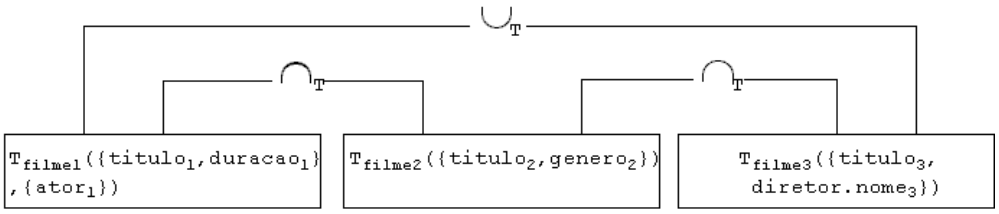
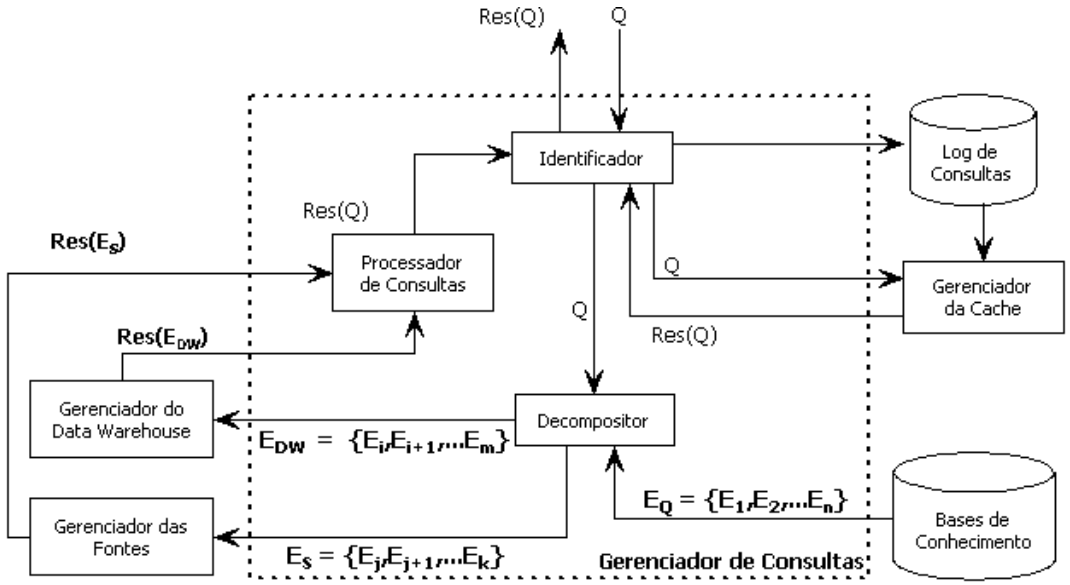


Figura 5.3 – Exemplo de um grafo de operações de entidade de mediação

5.3.2 Gerenciador de Consultas

O *Gerenciador de Consultas (GC)* é responsável por receber a consulta do usuário, processá-la e compor os resultados da mesma. Para tal, esse módulo tem acesso a resultados armazenados na *cache*, a entidades materializadas no *data warehouse* e a fontes de dados. O funcionamento do GC pode ser visto na Figura 5.4.



Notações:
E_Q = Conjunto das entidades de mediação $E_1, E_2, \ldots E_n$ solicitadas na consulta de usuário Q;
E_{DW} = Subconjunto de E_Q contendo as entidades presentes no DW;
E_S = Subconjunto de E_Q contendo as entidades presentes nas fontes de dados;
Res(Q) = Resultado em XML para a consulta Q;
$Res(E_s)$ = Resultado em XML para as consultas de entidades recuperadas das fontes;
$Res(E_{dw})$ = Resultado em XML para as consultas de entidades recuperadas do DW.

Figura 5.4 – Módulo Gerenciador de Consultas

Foi feita uma divisão do GC em três submódulos: *Identificador*, *Decompositor* e *Processador de Consultas* onde cada um executa tarefas específicas. Quando o sistema recebe a consulta de usuário, o Identificador inicialmente irá verificar se o resultado da mesma está armazenado na *cache* através de comunicação com o módulo Gerenciador da Cache. Se o resultado da consulta estiver na *cache*, será prontamente retornado ao usuário. Em caso contrário, a consulta será entregue ao módulo Decompositor.

Na Figura 5.4, o componente denominado Bases de Conhecimento é uma generalização para representar as duas bases de conhecimento presentes na arquitetura original: a base de conhecimento do mediador e a base de conhecimento das fontes (ver Figura 5.1). O Decompositor acessa as Bases de Conhecimento do Mediador para identificar as entidades de mediação que estão envolvidas na consulta e o local onde seus dados estão armazenados. Cabe a esse módulo determinar as entidades que serão recuperadas do *data warehouse* e as entidades que serão recuperadas a partir das fontes de dados. Assim, o Decompositor finaliza o seu processamento enviando duas listas de entidades, uma para o Gerenciador do Data Warehouse e a outra para o Gerenciador das Fontes. Estes dois últimos procederão com a recuperação dos dados das entidades armazenadas no *data warehouse* e nas fontes de dados respectivamente, enviando o resultado em XML para o módulo Processador de Consultas.

O Processador de Consultas combina as instâncias das entidades de mediação recuperadas do *data warehouse* e das fontes fornecendo ao módulo Identificador o resultado final da consulta de usuário.

O passo final executado pelo módulo Identificador é a atualização da entrada no *Log* para a consulta requisitada informando alguns itens tais como: o *script* da consulta, tempo de resposta, freqüência de submissão, tamanho do resultado obtido, entre outros. A seção a seguir detalhará as informações que estão contidas no *Log* de consultas.

Uma observação importante deve ser feita quanto ao processamento de consultas de usuário: quando ocorrer de uma consulta apresentar condições sobre os atributos de entidades, essas condições deverão ser propagadas para as fontes de dados ou data warehouse para evitar que o resultado traga um grande volume de dados e para evitar que o Gerenciador de Consultas tenha de processar as condições diminuindo o desempenho do processamento das consultas.

5.3.2.1 O Log de Consultas

O *Log* de consultas é uma base de dados históricos de todas as consultas submetidas ao sistema de integração. Através dele o Gerenciador de Consultas pode identificar consultas, controlar espaço dos resultados armazenados na cache, contabilizar a quantidade de vezes em

que uma mesma consulta é requisitada ao sistema de integração, verificar o tempo de resposta da consulta e manter algumas informações adicionais. A estrutura de um registro de resumo de consulta no *Log* de Consultas está na Figura 5.5:

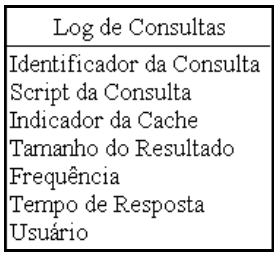


Figura 5.5 – Log de Consultas

Para o Gerenciador de Consultas, uma consulta de usuário é representada pelo seu script, isto é, pelo string da consulta XQuery.

O atributo *Identificador de Consulta* é obtido da seguinte forma: cada vez que o Gerenciador de Consultas consulta o *Log* e é verificado que a consulta ainda não foi submetida ao sistema de integração, é criado automaticamente pelo sistema um novo identificador e uma nova entrada no *Log* para essa consulta. Nesse caso é retornada para o Gerenciador de Consultas a informação de que a consulta não estava na cache.

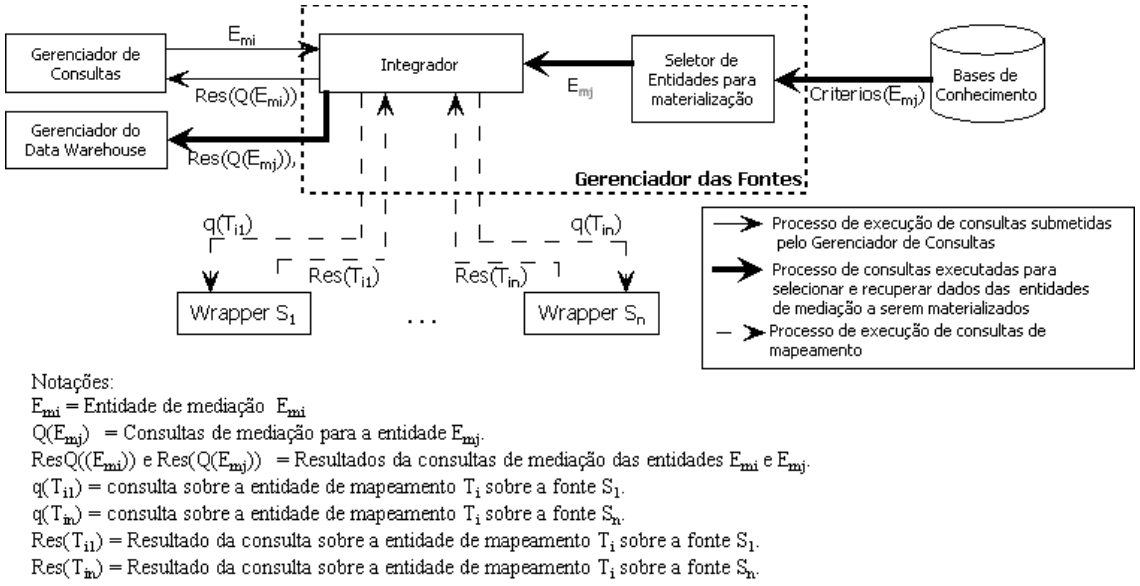
O atributo *Script* da Consulta contém o string que constitui a consulta em si. O Indicador da Cache informa se o resultado da consulta está ou não armazenado na *cache*.

O *Log* também armazena informações como: o *Tamanho do Resultado* da consulta em *Kbytes*; o número de vezes em que ela foi submetida ao sistema de integração (atributo *Freqüência*); o *Tempo de Resposta* que o sistema levou processando desde a chegada da consulta de usuário até o retorno do resultado integrado; o *login* do usuário que solicitou a consulta pela última vez (atributo.*Usuário*). Todos esses atributos, à exceção de *Freqüência* que é cumulativo, são referentes à última execução de uma consulta. Portanto, o *Log* de consultas é fonte de informações de controle das consultas do sistema.

5.3.3 Gerenciador das Fontes de Dados

Durante o processamento de uma consulta de usuário, o módulo Gerenciador das Fontes de Dados (GF), recebe como entrada uma lista de entidades que devem ser recuperadas a partir das fontes de dados, processa as consultas de mediação correspondentes e retorna os resultados obtidos ao Gerenciador de Consultas. O GF é o único módulo do sistema que tem acesso aos dados das fontes de dados

Uma outra tarefa do GF é avaliar periodicamente os critérios de materialização relacionados com as entidades e fontes de dados e avaliar quais entidades do mediador devem ser armazenadas no *data warehouse* (ver seção 5.3.3.1). Os submódulos do Gerenciador das Fontes de Dados podem ser visualizados na Figura 5.6:



Notações:
E_{mi} = Entidade de mediação E_{mi}
$Q(E_{mj})$ = Consultas de mediação para a entidade E_{mj}.
$ResQ((E_{mi}))$ e $Res(Q(E_{mj}))$ = Resultados da consultas de mediação das entidades E_{mi} e E_{mj}.
$q(T_{i1})$ = consulta sobre a entidade de mapeamento T_i sobre a fonte S_1.
$q(T_{in})$ = consulta sobre a entidade de mapeamento T_i sobre a fonte S_n.
$Res(T_{i1})$ = Resultado da consulta sobre a entidade de mapeamento T_i sobre a fonte S_1.
$Res(T_{in})$ = Resultado da consulta sobre a entidade de mapeamento T_i sobre a fonte S_n.

Figura 5.6 – Módulos do Gerenciador das Fontes de Dados

Na Figura 5.6, o componente denominado Bases de Conhecimento também é uma generalização para representar as duas bases de conhecimento presentes na arquitetura original: a base de conhecimento do mediador e a base de conhecimento das fontes (ver Figura 5.1).

Quando o Gerenciador de Consultas solicita os dados de uma entidade de mediação E_{mi}, o Integrador irá decompor a consulta de mediação $Q(E_{mi})$ da entidade em subconsultas que endereçam as entidades de mapeamento $(T_{i1},...T_{in})$ relevantes para computar E_{mi}. Para cada entidade de mapeamento obtida, o Integrador deverá executar a consulta de mapeamento $q(T_{in})$ sobre a fonte de dados S_n. O processo de execução de consultas de mapeamento e de integração de dados será descrito na seção 5.3.3.2.

O submódulo Seletor de Entidades para Materialização, deverá periodicamente executar o processo de materialização das entidades de mediação armazenadas no *data warehouse*. Inicialmente, o Seletor recupera os critérios de materialização das bases de conhecimento do mediador e das fontes de dados para poder calcular *rankings* dos valores de escores para cada uma das entidades de mediação que compõem o esquema de mediação e proceder com a seleção das que serão materializadas baseada na classificação obtida. Em seguida, o Integrador executa as consultas de mediação das entidades selecionadas e retorna seus resultados para o

Gerenciador do Data Warehouse que os armazenará no *data warehouse*. A seção 5.3.3.1 irá detalhar os processos de seleção e classificação de dados para materialização.

5.3.3.1 Seleção e Classificação de Dados para Materialização

Um problema chave no projeto de um *data warehouse* é a seleção de um conjunto de visões obtidas a partir das fontes de dados a fim de atender a requisitos de desempenho, que são minimizar o custo de processamento das consultas de usuário e minimizar também o custo de manutenção dos dados materializados. As abordagens existentes para solucionar problemas desse tipo, ou propõem algoritmos que exploram exaustivamente o espaço de busca[2] para enumerar e avaliar a possível materialização de todas as soluções encontradas, ou introduzem funções heurísticas que restringem o espaço de busca para obter reduções nos custos de consultas e manutenção [Bouzeghoub00]. A segunda solução apresenta algoritmos mais baratos, mas não dá garantias de que a melhor solução será utilizada, uma vez que, o espaço de busca é reduzido arbitrariamente pelas funções heurísticas. Ambas as abordagens executam o algoritmo sobre um espaço de busca construído de acordo com as consultas de usuários previamente submetidas.

Em ambas as soluções, é uma tarefa complexa inserir requisitos adicionais nos algoritmos de seleção de visões (ex.: *disponibilidade da fonte de dados*), que também possam influenciar os custos.

Como visto no capítulo anterior, é possível criar algoritmos de seleção de dados com base em múltiplos critérios de qualidade. Portanto nós optamos em criar uma abordagem, na qual as entidades do mediador serão classificadas e selecionadas para materialização no *data warehouse*. A cada entidade será atribuído um valor de escore de critério e esses valores classificam as entidades para o processo de materialização. A seleção de entidades para materialização será efetuada com base em quatro critérios de qualidade e custo:

- Freqüência de Atualizações ou *atualidade* – entidades que são obtidas a partir de fontes que comumente não têm seus dados alterados com muita freqüência são boas candidatas à materialização;
- Tempo de Resposta – em casos onde o tempo de resposta na execução da consulta de mediação de uma entidade for muito alto, essa entidade deverá ser materializada;

[2] Um espaço de busca é o conjunto de visões que combinadas podem atender a uma determinada consulta. Geralmente é representado como um grafo que é a união de todos os planos possíveis de consultas que representam as visões.

- Disponibilidade das Fontes de Dados – pode ocorrer de determinadas entidades estarem vinculadas a consultas de mediação que são executadas sobre fontes de dados que freqüentemente estão inacessíveis ao sistema de integração. Em casos como esse, também se deve optar pela materialização dos dados da entidade;
- Freqüência de Acessos à Entidade – entidades que são mais freqüentemente requisitadas em consultas de usuários também são boas candidatas à materialização.

Uma outra característica de nossa abordagem é a não-dependência da existência de um espaço de busca de consultas de usuário. O algoritmo de seleção de dados para materialização pode ser utilizado sem acessar as estatísticas do Gerenciador de Consultas: os critérios serão mantidos na base de conhecimento do mediador e na base de conhecimento das fontes a partir do momento da inserção da fonte de dados no nosso sistema de integração.

A escolha do uso de uma *entidade do mediador* como unidade para seleção e materialização de dados, se deu devido à característica do esquema do mediador estar representado em um modelo conceitual baseado no conceito de entidade (ver seção 5.3.1.1) e porque o módulo Gerenciador de Fontes tem acesso às subconsultas da consulta original do usuário endereçadas às fontes de dados e decompostas para cada entidade, ou seja, o GF tem acesso aos dados e metadados de todas as entidades do esquema do mediador.

Fizemos uma adaptação do *framework* descrito em [Naumann99] para o nosso contexto de materialização de dados e assim obter valores reais de *escores* para os critérios que interessam ao sistema de materialização. Naumann [Naumann00] e Wang [Wang93] propõem uma relação de critérios de qualidade que comumente devem ser avaliados. O critério *freqüência de acessos à entidade* não consta da classificação proposta nesses dois trabalhos, tendo sido inserido em nossa abordagem com o propósito de aumentar ainda mais os ganhos de desempenho do sistema. Os demais critérios definidos nesses estudos de qualidade não foram utilizados por questões de simplificação, entretanto, nosso *framework* é extensível de forma a futuramente poder incorporar a avaliação de critérios de qualidade adicionais.

A Tabela 5.1 indica o que significa cada critério e quais os métodos de avaliação para a obtenção de cada um dos *escores*.

Um outro aspecto que deve ser destacado é que a nossa proposta considera os critérios de *atualidade* e *disponibilidade* como sendo próprios da fonte de dados e os critérios *tempo de acesso* e *freqüência de acessos*, como critérios específicos das consultas de mediação que geram a entidade.

Tabela 5.1 – Critérios para materialização de entidades

Critério	Descrição	Método de Avaliação
Atualidade (At)	Período de tempo em dias desde que a entidade foi atualizada pela última vez.	Extraído da fonte de dados ou informado pela mesma ao GF.
Tempo de Acesso à Entidade (TA)	Tempo em segundos que uma consulta de mediação leva para ser processada.	Calculado com base no histórico das consultas submetidas ao sistema. O GF controla essa operação.
Disponibilidade (Di)	Valor percentual em que a(s) fonte(s) de dados que contêm dados da entidade estão accessíveis pelo sistema de integração.	Calculado com base no histórico das consultas submetidas ao sistema. O GF controla essa operação.
Freqüência de Acessos (FA)	Número total de vezes em que a entidade foi solicitada em consultas submetidas ao mediador.	Obtido a partir da base de metadados do mediador. O GF controla essa operação.

Periodicamente, o módulo Gerenciador das Fontes de Dados irá processar o algoritmo de seleção dos dados para materialização. O algoritmo de materialização deverá selecionar e classificar entidades para o *data warehouse* que atendam aos requisitos estabelecidos para os critérios conforme a descrição a seguir:

- O critério *atualidade* deverá ter um valor baixo, significando que os dados da fonte não são atualizados com muita freqüência. Dados altamente dinâmicos não devem ser replicados no *data warehouse*, pois isso implica em um alto custo de manutenção da consistência;
- *Tempo de acesso* deve ter um valor alto, o que indica que a execução de algumas consultas requer um custo de processamento muito alto, degradando assim o desempenho do sistema;
- A *disponibilidade* da fonte de dados ser baixa, pois a proposta é armazenar no *data warehouse* dados de fontes que passam muito tempo inacessíveis pelo sistema de integração;
- O critério *freqüência de acessos* das entidades selecionadas deverá ter o valor alto, pois esse fator indica que ela detém dados muito solicitados pelos usuários do sistema de integração.

Os critérios *atualidade* e *disponibilidade* são critérios de custo ou *negativos*, pois, quanto maior os valores de seus *escores*, menos indicados para materialização [Naumann98b]. Em contrapartida, *tempo de acesso* e *freqüência de acessos* são critérios de qualidade, ou *positivos* e quanto maior os valores de escores para esses critérios, mais indicada à materialização é a entidade em questão. Em resumo, critérios positivos favorecem à materialização, enquanto que, os critérios negativos são desfavoráveis.

Será considerado que cada entidade do mediador E_{Mk} seja obtida através de uma consulta de mediação MQ_k, a qual é composta de uma ou mais subconsultas Q_i cada uma das quais dirigidas a uma fonte de dados que contém dados relevantes para E_{Mk}.

Para cada entidade de mapeamento, será criado um vetor de quatro ocorrências e com a consulta da entidade Q_i. Cada uma das dimensões do vetor corresponde a um *escore* de critério avaliado. O vetor de critérios de uma consulta é representado como a seguir:

$$VCr(Q_i) = (c_{i1},c_{i2},.c_{i3},c_{i4}) = (At,TA,Di,FA)$$

A consulta de mediação possui operadores θ que podem representar qualquer operação: união, interseção ou diferença de mapeamento definidas em [Lóscio03]. Assim, como em [Naumann99] (ver seção 4.4.4) teremos uma árvore binária onde as folhas são as entidades de mapeamento e os nós mais acima representam as operações. Cada elemento da árvore será associado a um vetor de escores de critérios como mostra a Figura 5.7:

$$E_{m1} = MQ_1 = (Q_1(S_1)\ \theta_1\ Q_2(S_2))\ \theta_2\ Q_3(S_3)$$

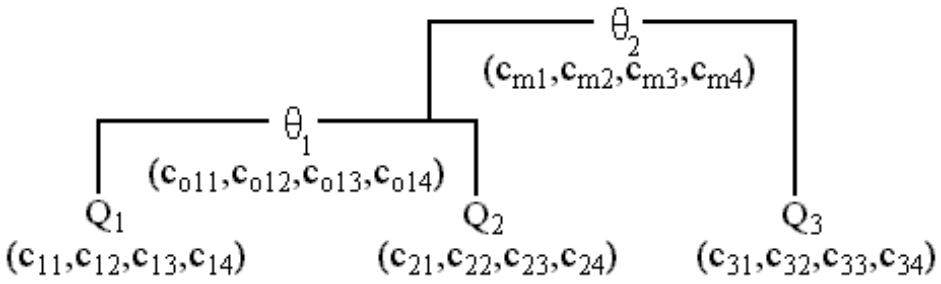


Figura 5.7 – Vetores de critérios da árvore de consulta

O vetor de critérios de um nó da árvore é recursivamente calculado de acordo com funções de agregação (*f*) que são aplicadas aos nós imediatamente inferiores. Por exemplo, para calcular o vetor de critérios associado à consulta MQ_1 da Figura 5.7, teremos a expressão:

$$VCr(\theta_1) = VCr(Q_1\ f\ Q_2) = VCr(Q_1)\ f\ VCr(Q_2) = (c_{11}\ f\ c_{21},\ c_{12}\ f\ c_{22},\ c_{13}\ f\ c_{23},\ c_{41}\ f\ c_{24})$$

$$VCr(MQ_1) = VCr(\theta_2) = VCr(\theta_1\ f\ Q_3) = VCr(\theta_1)\ f\ VCr(Q_3) =$$

$$= (c_{o11}\ f\ c_{31},\ c_{o12}\ f\ c_{32},\ c_{o13}\ f\ c_{33},\ c_{o14}\ f\ c_{34}) = (c_{m1},c_{m2},c_{m3},c_{m4})$$

As funções de agregação *f* serão aplicadas sobre cada nó da árvore, desde as folhas que representam as entidades de mapeamento até a raiz da árvore que é a entidade de mediação E_{m1}. Para os quatro critérios de materialização que o sistema considera, os escores dos elementos do vetor são calculados pelas funções de agregação *f* de acordo com a Tabela 5.2.

Tabela 5.2 – Funções de agregação para cálculo de escores

Critério	Função Aplicada	Descrição
Disponibilidade	c_{e1} x c_{d1}	Probabilidade de que ambas as fontes estejam acessíveis.
Tempo de Acesso	max[c_{e2},c_{d2}]	Ambos os nós filhos são processados em paralelo.
Atualidade	média[c_{e3},c_{d3}]	Considera-se o tempo médio de atualização.
Freqüência de Acessos	média[c_{e4},c_{d4}]	Média aritmética entre as duas freqüências

onde: c_{ej} representa o escore do critério j do nó inferior esquerdo ao nó atual;
c_{dj} representa o escore do critério j do nó inferior direito ao nó atual.

Os *escores* obtidos até esse ponto para cada entidade de mediação estão com medidas e unidades distintas. É necessário escaloná-los para poder somá-los entre si, obtendo assim um escore global da entidade e, em seguida, criar uma classificação decrescente das entidades candidatas à materialização. Para isso é utilizado um método denominado *SAW* ou *Simple Additive Weighting* (ver seção 4.3.1.1). Os escores individualmente serão uniformizados através das equações 5.1 e 5.2, onde a primeira será utilizada para os critérios positivos (*tempo de acesso* e *freqüência de acessos*) e a segunda para critérios negativos (*atualidade* e *disponibilidade*).

$$vij = \frac{c_{ij} - c_j^{min}}{c_j^{max} - c_j^{min}} \quad (5.1) \qquad vij = \frac{c_j^{max} - c_{ij}}{c_j^{max} - c_j^{min}} \quad (5.2)$$

onde: v_{ij} é o escore escalonado do critério j para a entidade i;
c_{ij} é o escore original do critério j para a entidade i;
c_j^{max} é o valor de escore máximo para o critério j;
c_j^{min} é o valor de escore mínimo para o critério j;

Uma vez aplicadas as funções de uniformização, cada escore v_{ij} terá um valor no intervalo $0 \leq v_{ij} \leq 1$. Essa propriedade assegura a possibilidade de comparação entre critérios distintos em diferentes dimensões. Assim, poderão ser somados todos os escores de critérios de uma mesma entidade.

A cada *escore* unificado, será atribuído um valor de peso que reflete a importância de um critério em relação aos outros. O vetor de pesos $W = (w_1, w_2, w_3, w_4)$ obedece a propriedade $\sum_{j=1}^{4} w_j = 1$.

Para uma determinada consulta de mediação MQ_i, o seu *escore* de materialização global é calculado pela soma ponderada definida na equação 5.3:

$$V(MQ_i) = \sum_{j=1}^{4} w_j \times v_{ij} \qquad (5.3)$$

O valor final do *escore* $V(MQ_i)$ está em [0,1] e fornece a posição de ranking para materialização da entidade de mediação E_{mi} associada à consulta de mediação MQ_i. A lista de entidades para serem materializadas é elaborada em ordem decrescente de *escores*, ou seja, quanto maior o valor obtido, mais apta à materialização é a entidade associada. No ambiente de integração de dados, ficou estabelecido que, entidades com escore no intervalo [0.5,1] são indicadas à materialização. Para materializar uma entidade de mediação o GF submete a consulta de mediação correspondente, retornando os dados obtidos para GDW que ao receber os dados selecionados, procede com as tarefas de materialização das entidades escolhidas.

5.3.3.2 O Processo de Integração de Instâncias de Dados

Para recuperar dados de uma entidade de mediação devemos computá-la através da execução da consulta de mediação correspondente e, por conseguinte, através das execuções das consultas de mapeamento que compõem a consulta de mediação. Na execução de uma consulta de mediação, os dados da entidade são recuperados das fontes e integrados.

Para descrever o processo de integração dos dados propriamente dito utilizaremos um exemplo de uma entidade $\texttt{funcionario}_m$ que é computada pela consulta de mediação representada pelo grafo de operação a seguir:

$\texttt{funcionario}_m(\{\texttt{nome}_m, \texttt{funcao}_m, \texttt{departamento}_m\}, \{\texttt{endereco}_m\})$
$Q(\texttt{funcionario}_m) = T_{\texttt{funcionario1}} \cup_T T_{\texttt{funcionario2}}$

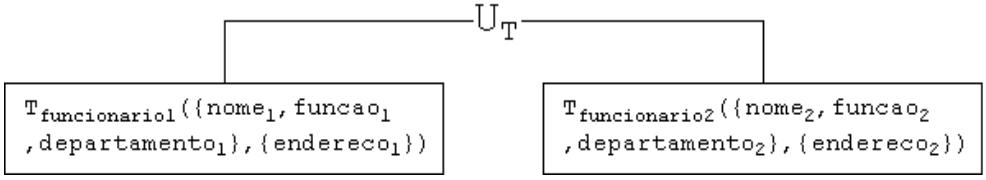


Figura 5.8 – Grafo de operação de consulta de mediação da entidade $\texttt{funcionario}_m$

O grafo da consulta $Q(\texttt{funcionario}_m)$ indica que as instâncias de $\texttt{funcionario}_m$ são obtidas das fontes S_1 e S_2 através da união das instâncias da entidade $\texttt{funcionario}_1$ com as instâncias de $\texttt{funcionario}_2$.

As instâncias de $\texttt{funcionario}_m$ são obtidas através dos seguintes passos iniciais:

1. Processamento da consulta $q(T_{funcionario1})$ sobre a fonte de dados S_1 para computar a entidade de mapeamento $T_{funcionario1}$ e obter as instâncias da mesma;
2. Processamento da consulta $q(T_{funcionario2})$ sobre a fonte de dados S_2 para computar a entidade de mapeamento $T_{funcionario2}$ e obter as instâncias da mesma.

As instâncias XML obtidas são coleções de elementos obtidos das fontes de dados. As Figuras 5.9 e 5.10 ilustram os resultados obtidos com a execução dos passos 1 e 2 sobre um exemplo de dados XML.

O passo seguinte para obter dados de $funcionario_m$ consiste em executar a consulta de mediação $Q(funcionario_m)$ sobre as duas coleções de dados $C(funcionario_1)$ e $C(funcionario_2)$ obtidas das fontes nos passos anteriores. Trata-se de processar a união entre as coleções para obter a coleção ilustrada na Figura 5.11.

C(funcionario$_1$):

```
<funcionario1>
  <nome1>Jose da Silva</nome1>
  <funcao1>Programador</funcao1>
  <departamento1>Informatica</departamento1>
  <endereco1>
     <tipo1>postal</tipo1>
     <conteudo1>Av. Caxanga, 5281 - Varzea</conteudo1>
  </endereco1>
  <endereco1>
     <tipo1>internet</tipo1>
     <conteudo1>jsilva@hotlink.com</conteudo1>
  </endereco1>
</funcionario1>
```

Figura 5.9 – Coleção de elementos de $funcionario_1$

C(funcionario$_2$):

```
<funcionario2>
  <nome2>Jose da Silva</nome2>
  <funcao2>Programador</funcao2>
  <departamento2>Informatica</departamento2>
  <endereco2>
     <tipo2>contato</tipo2>
     <conteudo2>Rua do Futuro,1385/703</conteudo2>
   </endereco2>
   <endereco2>
     <tipo2>internet</tipo2>
     <conteudo2>jsilva@hotlink.com</conteudo2>
   </endereco2>
</funcionario2>
```

Figura 5.10 – Coleção de elementos de $funcionario_2$

C(funcionario$_m$):

```
<funcionario_m>
  <nome_m>Jose da Silva</nome_m>
  <funcao_m>Programador</funcao_m>
  <departamento_m>Informatica</departamento_m>
  <endereco_1>
     <tipo_1>postal</tipo_1>
     <conteudo_1>Av. Caxanga, 5281 - Varzea</conteudo_1>
  </endereco_1>
  <endereco_1>
     <tipo_1>internet</tipo_1>
     <conteudo_1>jsilva@hotlink.com</conteudo_1>
  </endereco_1>          Elementos da Fonte de Dados S1
  <endereco_2>
     <tipo_2>contato</tipo_2>
     <conteudo_2>Rua do Futuro,1385/703</conteudo_2>
     <cep_2>52050-010</cep_2>
  </endereco_2>
  <endereco_2>
     <tipo_2>internet</tipo_2>
     <conteudo_2>jsilva@hotlink.com</conteudo_2>
  </endereco_2>          Elementos da Fonte de Dados S2
</funcionario_m>
```

Figura 5.11 – Coleção de elementos de $funcionario_m$

O elemento $funcionario_m$ possui uma entidade filha $endereco_m$ As coleções em destaque (com linha cheia) na Figura 5.11 representam instâncias de elementos da entidade $endereco_m$ obtidas da fonte S_1 e as coleções destacadas com linha pontilhada são as instâncias obtidas da fonte S_2. O próximo passo a ser executado é a integração dos dados de $endereco_m$ através da execução da consulta de mediação da entidade apresentada na Figura 5.12:

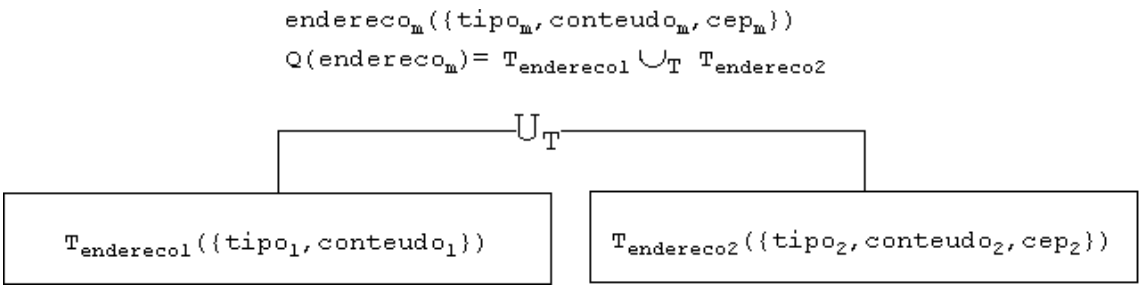


Figura 5.12 – Grafo de operação de consulta de mediação da entidade $endereco_m$

Da mesma maneira que ocorreu com a entidade $funcionario_m$, os passos de integração serão aplicados para as instâncias de $endereco_m$ contida em cada instância de $funcionario_m$. Serão executadas as consultas de mapeamento para computar as entidades $T_{endereco1}$ e $T_{endereco2}$ a partir das fontes de dados, obtendo as coleções $C(endereco_1)$ e $C(endereco_2)$. É importante notar que para cada entidade integrada esse processo deverá ser repetido sobre as

entidades filhas. No passo final da integração, as redundâncias deverão ser eliminadas. A Figura 5.13 mostra os elementos de $\texttt{funcionario}_m$ após a integração de $\texttt{endereco}_m$:

```
<funcionario_m>
  <nome_m>Jose da Silva</nome_m>
  <funcao_m>Programador</funcao_m>
  <departamento_m>Informatica</departamento_m>
  <endereco_m>
     <tipo_m>postal</tipo_m>
     <conteudo_m>Av. Caxanga, 5281 - Varzea</conteudo_m>
           </endereco_m>
  <endereco_m>
     <tipo_m>contato</tipo_m>
     <conteudo_m>Rua do Futuro,1385/703</conteudo_m>
     <cep_m>52050-010</cep_m>
  </endereco_m>
<endereco_m>
     <tipo_m>internet</tipo_m>
     <conteudo_m>jsilva@hotlink.com</conteudo_m>
  </endereco_m>
</funcionario_m>
```

Figura 5.13 – Elemento $\texttt{funcionario}_m$ integrado

Podemos observar que a ocorrência do elemento $\texttt{<endereco}_m\texttt{>}$ contendo um endereço de e-mail que estava repetido no elemento $\texttt{<funcionario}_m\texttt{>}$ foi eliminada.

Esses são os passos que o GF segue na execução de consultas de mediação para computar entidades.

5.3.4 Gerenciador da Cache

Sistemas de *Caching* são largamente utilizados para atingir melhorias de desempenho em muitos contextos [Mohan01]. Particularmente em integração de dados, alguns sistemas existentes na Web têm proposto o uso de *caching* para armazenar resultados de consultas mais freqüentes [Hammer99, Draper01b] submetidas ao sistema de integração.

No ambiente o módulo *Gerenciador da Cache (GCh)* é responsável pela manutenção de uma *cache* armazenada localmente. O processo de manutenção da *cache* diz respeito a aspectos de disponibilidade de espaço para armazenamento dos dados, políticas de substituição e *refresh* periódico do conteúdo para garantir que os resultados armazenados não fiquem longe da realidade dos dados das fontes. O *refresh* consiste em re-executar periodicamente as consultas cujos resultados estão armazenados na *cache*. Adicionalmente, durante esse processo, o sistema verifica se existem consultas que ainda não estão com os resultados na *cache*, mas que sua freqüência de submissão é mais alta do que outras que estão armazenadas.

No momento de efetuar o *refresh* do conteúdo da *cache*, o GCh acessa o *Log* de consultas para verificar as freqüências de submissão de todas as consultas e assim identificar se existem

novas consultas cujos resultados devam ser armazenados na *cache*, em detrimento de outras com menor freqüência de submissão que devam ser eliminadas. Para concluir o processo de *refresh*, o Gerenciador de Consultas computa as consultas selecionadas para a *cache* e envia seus resultados ao GCh. O *refresh* deverá ser efetuado periodicamente dentro de um espaço de tempo determinado automaticamente pelo sistema de forma que resultados armazenados na *cache* não sofram de inconsistências com relação aos dados das fontes de dados.

O GCh também verifica a alocação do espaço reservado para a cache e processa as operações de substituição necessárias. Inicialmente, a política de substituição adotada é baseada nas técnicas adotadas pela política LFU (*Least Frequently Used*) [Dar96], a qual sugere que quando o espaço da *cache* estiver se tornando insuficiente para armazenar novos resultados de consultas, o GCh deverá remover os resultados das consultas menos freqüentemente requisitadas.

5.3.5 Gerenciador do Data Warehouse

As tarefas principais do *Gerenciador do Data Warehouse (GDW)* são: proceder com a manutenção do repositório do sistema ou *data warehouse* no que se refere à materialização dos dados das entidades enviadas pelo Gerenciador das Fontes e efetuar *refresh* periódico dos dados materializados.

Quando o GDW recebe os escores de critérios e as entidades materializadas, ambos enviados pelo GF, deverá proceder com a escolha das entidades que serão materializadas e em seguida as entidades escolhidas são armazenadas no repositório. O Gerenciador das Fontes deverá suprir o GDW com todos os dados e metadados que são necessários para executar as operações de materialização. O uso de materialização de dados em sistemas de integração de dados já foi investigado em trabalhos prévios tais como [Widom95, Gupta95, Florescu98] e também atestado como sendo uma maneira eficaz de otimização de desempenho nas consultas do sistema de integração de dados.

Para o nosso ambiente ficou definido que apenas entidades filhas do elemento raiz do esquema do mediador serão materializadas por completo juntamente com todas as suas entidades filhas. Isso é feito porque se permitíssemos a materialização de entidades filhas sozinhas, perderíamos a ligação com a entidade mãe no momento da integração de dados, uma vez que o Integrador não teria como determinar essa informação. Por exemplo, suponhamos que no caso de uma entidade $\texttt{endereco}_{\texttt{m}}$ filha de $\texttt{funcionario}_{\texttt{m}}$, o sistema determinasse que as instâncias de $\texttt{funcionario}_{\texttt{m}}$ seriam recuperadas a partir das fontes de dados e apenas as instâncias de $\texttt{endereco}_{\texttt{m}}$ ficariam no data warehouse. No momento de integrar os dados de

funcionario$_m$, o módulo Integrador não teria condições de determinar que instâncias de endereco$_m$ são filhas de que instâncias de funcionario$_m$, ocorrendo uma perda de ligação entre elas.

Políticas de manutenção de *data warehouse* também já foram investigadas previamente em trabalhos como os de Rundensteiner em [Rundensteiner00] e Widom em [Widom95]. Tais políticas tratam do problema de manter o conteúdo do *data warehouse* consistente com relação aos dados das fontes. Isso pode ser feito de duas formas: descartar totalmente o conteúdo atual do *data warehouse* e novamente processar todas as consultas de mediação que recuperam os dados das entidades selecionadas ou propagar as alterações ocorridas nos dados das fontes de forma gradual e incremental. A primeira opção é menos eficiente e mais simples do que a segunda. Na segunda opção, um detalhe importante relacionado com a manutenção é que a propagação das alterações das fontes de dados para o *data warehouse* deve ser efetuada de maneira correta e imediata e esse processo é denominado *manutenção incremental*.

Será adotada a política de *refresh* completo para a manutenção do repositório. Isso se deve ao fato de que, como sempre é materializado um conjunto de entidades, não faria sentido propagar atualizações de uma parte dos dados de uma entidade. A operação correta seria proceder com a materialização completa de todas as entidades de mediação.

Periodicamente, o módulo GDW define o melhor momento para proceder com o *refresh* e em seguida, envia solicitações de manutenção ao Gerenciador das Fontes. Este último processará as consultas de mediação para re-computar as entidades do *data warehouse*.

Um outro momento em que deve-se proceder com o *refresh* do *data warehouse* é quando ocorrem modificações no esquema do mediador, pois as entidades materializadas podem ficar inconsistentes com o novo esquema.

5.4 Considerações Finais

Neste capítulo foi discutido o ambiente de integração de dados que é baseado em uma arquitetura híbrida de integração de dados.

Todos os módulos da arquitetura foram descritos. O componente Mediador fornece uma visão integrada das fontes de dados, bem como um esquema para essa visão e é sobre o esquema do mediador que todas as consultas de usuário são processadas.

O Mediador é composto pelos módulos Gerenciador de Consultas e Gerenciador das Fontes. O Gerenciador de Consultas basicamente processa a consulta de usuário e o Gerenciador das Fontes, o único módulo que acessa os dados enviados diretamente das fontes de dados, trabalha

recebendo a entidade a ser recuperada e executa a consulta de mediação desta entidade. Uma outra função do Gerenciador das Fontes é periodicamente efetuar o processo de seleção de entidades para materialização no *data warehouse* através de cálculos e análise de valores de escores associados a critérios de qualidade.

Os outros módulos do sistema são o Gerenciador da Cache e o Gerenciador do Data Warehouse. A execução de consultas em sistemas de integração baseados em *data warehouse* e recuperação de consultas em *cache* normalmente são mais rápidas de que em sistemas de mediadores. Entretanto, a maioria dos sistemas de arquitetura híbrida que utilizam esses recursos, o fazem em separado, ou utilizam técnicas de materialização em *data warehouse* ou utilizam *caching*. O nosso sistema se propõe a colocar esses recursos juntos em uma arquitetura de mediação. A principal contribuição que esperamos obter com essa junção de recursos, é a otimização do processamento das consultas de usuário, através da redução de problemas de desempenho que comumente ocorrem em sistemas de mediadores sem deixar de aproveitar de vantagens que são oferecidas, como dados atualizados e menor necessidade de manutenção dos mesmos.

Capítulo 6

Protótipo do Ambiente de Integração de Dados

6.1 Introdução

Para validar a nossa proposta de ambiente de integração de dados, construímos uma versão inicial de protótipo para validação dos módulos para a arquitetura híbrida de integração detalhada no capítulo anterior.

Nesse capítulo discutiremos aspectos da implementação do protótipo: os módulos e funcionalidades que utilizamos, linguagens de programação que empregamos para escrever os módulos, estruturas de dados e alguns algoritmos criados para o sistema de integração. Detalhes técnicos mais profundos do protótipo são discutidos em [Amaral02] que abrange toda a parte técnica da implementação.

Também falaremos a respeito das dificuldades que foram encontradas no processo de implementação do sistema e justificaremos o porque de alguns processos que não foram implementados.

O restante do capítulo está organizado da seguinte forma: a seção 6.2 explica as tecnologias que adotamos para implementação; a seção 6.3 detalha as funcionalidades do ambiente de integração de dados que compõem o protótipo implementado e; a seção 6.4 traz nossas conclusões e pendências com relação ao resultado da implementação.

6.2 Tecnologias Adotadas

A linguagem de programação que utilizamos no desenvolvimento dos módulos do sistema foi a linguagem Java[3], pela sua portabilidade, facilidade de reuso e grande número de APIs para interagir com o padrão XML e com XML Schema [Vioellau01].

A seguir listamos as APIs e os *parsers* utilizados na implementação com suas funções desempenhadas para o sistema:

- DBAccessor [Laux02] – API desenvolvida por Mathias Laux (Sun Microsystems), para encapsular algumas funcionalidades do JDBC[4] e tornar mais fácil a migração de dados de bancos de dados relacionais para dados XML. Essa API também foi utilizada na conversão de esquemas de BD relacionais para um formato específico dela própria, representado em XML. A única dificuldade encontrada para utilizar essa API foi que, originalmente, ela não estava preparada para utilizar o MS SQL Server 7[5]. Nós entramos em contato com o autor, e este gentilmente nos cedeu o código para alterarmos a API e fazer a adaptação para o SQL Server;
- JDOM [Hunter00] – JDOM é uma API *open source*, criada para facilitar a manipulação de XML como objetos Java. É utilizada pela API DBAccessor para a criação e o manuseio de estruturas XML;
- Jtds[6] – *driver open source* para a API JDBC da Sun escrito também em Java, para acessar bancos de dados relacionais, mais especificamente, o jtds fornece acesso ao banco de dados MS SQL Server. Dentre as alterações que fizemos na DBAccessor, uma delas foi utilizar esse driver no acesso ao SQL Server;
- *Xerces*[7] – *parser* para XML escrito em Java e também utilizado pela API DBAccessor. O Xerces está totalmente em conformação com XML e XML Schema;
- JAXP [Armstrong03]– conhecida como *Java API for XML Processing*, JAXP nivela os padrões SAX (Simple API for XML Parsing) [Armstrong03] e DOM (Document Object Model) [Wood00] de maneira que é possível escolher entre fazer um *parsing* em XML como uma seqüência de eventos ou construindo uma estrutura baseada em árvore.

---

[3] http://java.sun.com

[4] http://java.sun.com/products/jdbc/

[5] http://www.microsoft.com/sql/default.asp

[6] http://jtds.sourceforge.net/

[7] http://xml.apache.org/xerces2-j/index.html

- Para estruturas de armazenamento utilizamos o banco de dados MS SQL Server e arquivos XML para os dados. Os esquemas do sistema foram armazenados em XML Schema.

6.3 Funcionalidades

Algumas funcionalidades de nossa proposta foram modeladas para implementação através do uso da linguagem UML [Booch03] em diagramas de seqüência, onde podemos visualizar as classes criadas para os módulos da arquitetura, seus métodos e trocas de mensagens entre objetos das classes. As classes Java que foram criadas na implementação estão listadas a seguir:

- `Interface`: classe de controle que faz interface do sistema com eventos externos;
- `Controlador`: classe de controle responsável por tarefas periódicas;
- `GerenciadorConsultas`: classe que implementa o módulo Gerenciador de Consultas;
- `ResumoLog`: classe que implementa a entrada de resumo de uma consulta no Log de Consultas;
- `GerenciadorCache`: classe que implementa o módulo Gerenciador da Cache;
- `BCMediador`: classe que implementa a Base de Conhecimento do Mediador;
- `GerenciadorFontes`: classe que implementa o módulo Gerenciador das Fontes;
- `GerenciadorDW`: classe que implementa o módulo Gerenciador do Data Warehouse;

O Diagrama com as classes do ambiente de integração está ilustrado na Figura 6.1.

Dentro da proposta de arquitetura, os processos previstos para o nosso ambiente de integração de dados seriam:

- Execução de consulta de usuário com resultado recuperado da *cache*;
- Execução de consulta de usuário a partir das fontes de dados e *data warehouse*;
- Armazenamento de resultados na *cache*;
- Seleção de entidades para materialização;
- *Refresh* da *cache*;
- *Refresh* do *data warehouse*.

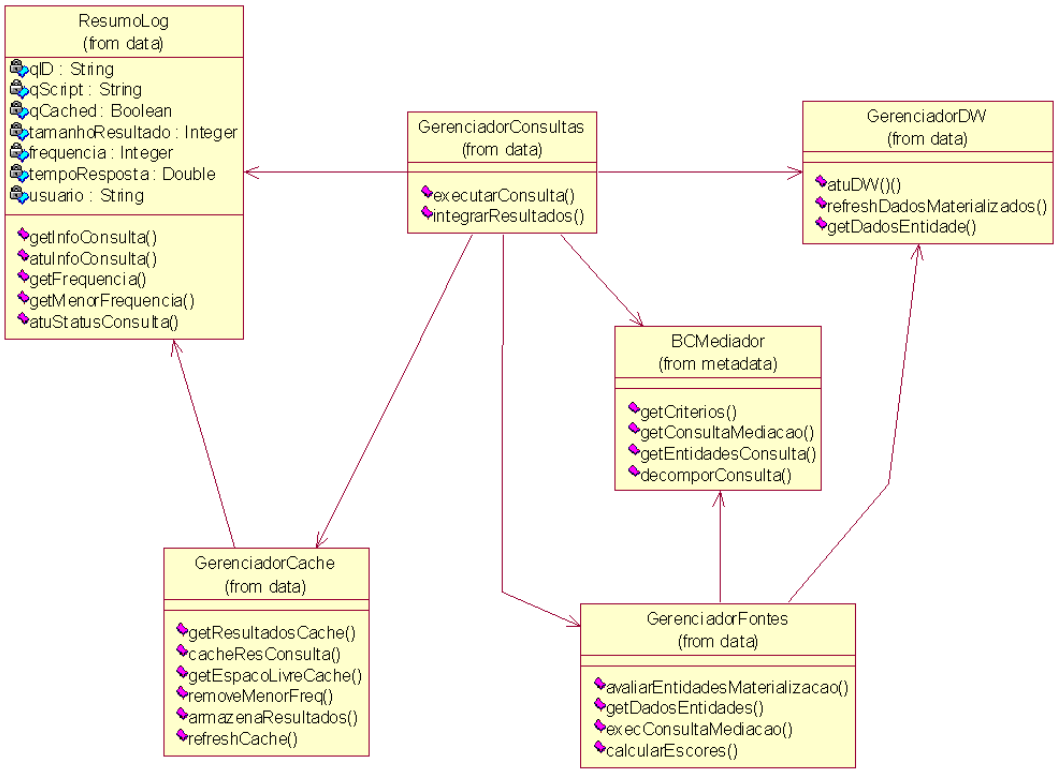


Figura 6.1 – Diagrama de classes do ambiente de integração

Os dois últimos processos de *refresh* da *cache* e do *data warehouse* não foram implementados nessa primeira versão do protótipo, já que não têm tanta relevância para a validação da arquitetura proposta para o nosso ambiente. Dos outros quatro processos descritos acima, apenas algumas funções/métodos não foram implementadas, ficando previstas suas implementações para uma próxima versão do protótipo. As próximas seções descrevem os processos implementados.

6.3.1 O Processamento de Consultas de Usuário

Como vimos na especificação da arquitetura, a execução de uma consulta de usuário é processada pelo Gerenciador de Consultas e envolve três etapas: (i) verificação da *cache* para recuperar os resultados imediatamente; (ii) decomposição da consulta de usuário em entidades de mediação; (iii) recuperação dos dados das entidades materializadas a partir do *data warehouse* enviados pelo Gerenciador do Data Warehouse e/ou recuperação dos dados das entidades a partir das fontes enviados pelo Gerenciador de Fontes; (iv) combinação dos resultados para compor a resposta da consulta de usuário.

Muito embora no sistema ocorram em conjunto, na modelagem, essas funcionalidades foram separadas em seqüências diferentes, com a finalidade de dar ao sistema uma maior visibilidade e modularidade dos processos envolvidos. São elas:

- O processamento de uma consulta com resultado na *cache*;
- O processamento de uma consulta a partir do *data warehouse* e fontes de dados;

Quando qualquer usuário externo interage com o ambiente, imediatamente é acionada uma classe de controle denominada `Interface` que é responsável por fazer a interface entre o sistema e solicitações externas. Essa classe é acionada sempre que um usuário submete consultas XQuery ao ambiente de integração de dados.

O Log de Consultas é atualizado sempre que uma consulta de usuário é submetida ao sistema com informações diversas e mantém várias entradas para uma mesma consulta, isto é, uma entrada para cada submissão. Uma classe que criamos no sistema chama-se `ResumoLog` que é um resumo do Log de Consultas contendo informações para uma consulta específica definidas na seção 5.3.2.1 que são relevantes para o controle da cache.

Nas seções a seguir discutiremos os detalhes de implementação do processamento de consultas de usuários.

6.3.1.1 Execução de Consulta com Resultado na Cache

O diagrama com a seqüência de passos ou métodos que implementam a execução de uma consulta a partir da *cache* pode ser visto na Figura 6.2:

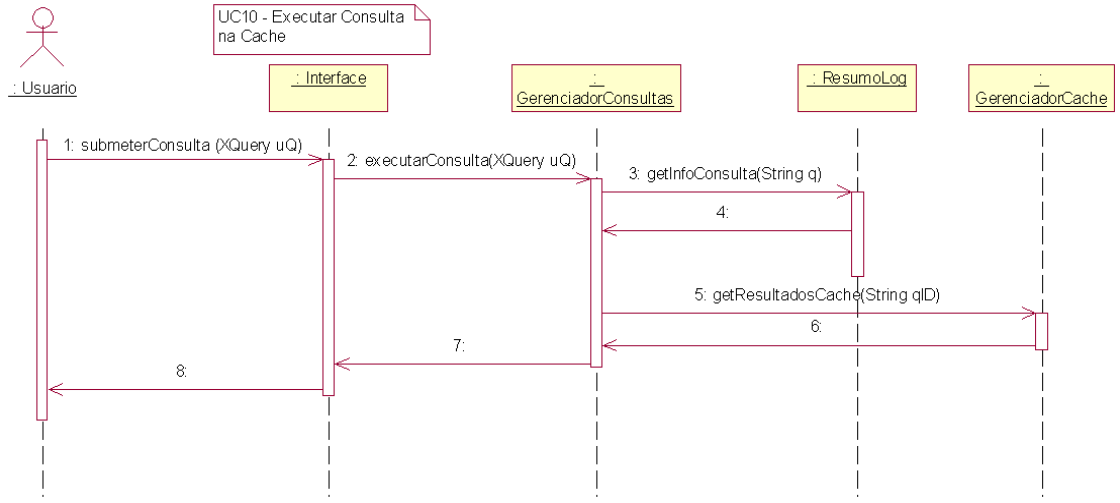


Figura 6.2 – Seqüência de processamento de uma consulta com resultados na *cache*

Quando uma consulta de usuário é recebida pelo sistema de integração, imediatamente a classe `Interface` é acionada através de seu método `submeterConsulta()` e esse método aciona a classe `GerenciadorConsultas` informando como parâmetro a consulta XQuery. A classe `GerenciadorConsultas` instancia o método `getInfoConsulta()` da classe `ResumoLog` para recuperar o identificador da consulta e a informação se o resultado está

armazenado na *cache*. Se o resultado está na cache é imediatamente recuperado pelo `GerenciadorConsultas` através da chamada do método `getResultadosCache()` presente na classe `GerenciadorCache`. Esse método devolve o resultado da consulta do usuário. Caso o resultado não esteja armazenado na *cache*, teremos então o processamento da consulta a partir das fontes de dados e do *data warehouse* que será discutido na próxima seção.

A Tabela 6.1 mostra os passos, métodos, parâmetros, argumentos de retorno e classes da execução da consulta com resultado armazenado na *cache*:

Tabela 6.1 – Componentes do processo de execução de consulta com resultados na *cache*

	Descrição	**Classe**	**Método(Parâmetros)**	**Retorno**
1	Usuário submete a consulta ao ambiente de integração de dados.	`Interface`	`submeterConsulta (XQuery uQ)`	Documento XML
2	A `Interface` solicita a execução à classe `GerenciadorConsultas`	`GerenciadorConsultas`	`executarConsulta (XQuery uQ)`	Documento XML
3	A classe `GerenciadorConsultas` recupera o status do resultado (se está ou não está na cache)	`ResumoLog`	`getInfoConsulta (String q)`	ID da consulta e o status indicando se o resultado está na cache.
4	A classe `GerenciadorConsultas` solicita os resultados da cache.	`GerenciadorCache`	`getResultadosCache (String qID)`	Documento XML

Sempre que um método de submissão ou execução de consulta é instanciado o objeto que este retorna é um documento XML contendo a resposta para a consulta. Todos os métodos e funções do processo de execução de consulta com resultado na *cache* foram implementados.

6.3.1.2 Execução de Consulta a partir das Fontes e Data Warehouse

Uma vez que o resultado de uma consulta de usuário não se encontra armazenado na *cache*, o sistema então irá decompor a consulta original e recuperar os dados a partir do *data warehouse* e das fontes de dados. É um processo um pouco mais complicado do que o descrito na seção anterior.

Após a recuperação dos dados a partir das fontes e *data warehouse* existe um passo adicional onde o sistema recupera a freqüência de submissão da consulta atual para analisar através do Gerenciador da Cache, se o resultado deve passar a ser armazenado na *cache*. O diagrama de seqüência de execução de consultas a partir das fontes e do *data warehouse* pode ser visto na Figura 6.3.

Como no caso anterior, imediatamente a classe `Interface` é acionada pela consulta de usuário através da chamada do método `submeterConsulta()` e esse método aciona a classe `GerenciadorConsultas` informando como parâmetro a consulta XQuery. A classe `GerenciadorConsultas` instancia o método `getInfoConsulta()` da classe `ResumoLog` para recuperar o identificador da consulta e a informação se o resultado está armazenado na *cache*. Nesse caso o resultado não está na cache, então o `GerenciadorConsultas` instancia o método `decomporConsulta()` da classe `BCMediador` para recuperar todas as entidades que estão inseridas na consulta de usuário e o local onde cada uma está armazenada (fonte de dados ou *data warehouse*).

De posse dessas informações, o `GerenciadorConsultas` instancia dois métodos: o método `getDadosEntidade()` da classe `GerenciadorDW` que retorna as entidades materializadas no *data warehouse* e `getDadosEntidade()` da classe `GerenciadorFontes`. Este último, para cada entidade, chama o método `getConsultaMediacao()` da classe `BCMediador` para receber a consulta de mediação e em seguida executa o método interno `execConsultaMediacao()` para recuperar os dados da entidade e retorná-los ao `GerenciadorConsultas`.

O próximo passo é a execução do método `integraResultados()` da classe `GerenciadorConsultas` que combina os resultados recebidos do *data warehouse* e das fontes de dados. Em seguida o `GerenciadorConsultas` irá interagir com a classe `ResumoLog` para atualizar as informações da consulta de usuário através da chamada do método `atuInfoConsulta()` e irá recuperar o valor da freqüência de submissão dessa consulta já atualizado (método `getFrequencia()`).

A última etapa do processo consiste em instanciar o método `cacheResConsulta()` da classe `GerenciadorCache` onde esta última procederá com os passos necessários para avaliar se o resultado obtido para a consulta de usuário atual deverá ser armazenado na *cache*. Em paralelo a classe `GerenciadorConsultas` retorna o resultado final da consulta de usuário originalmente submetida.

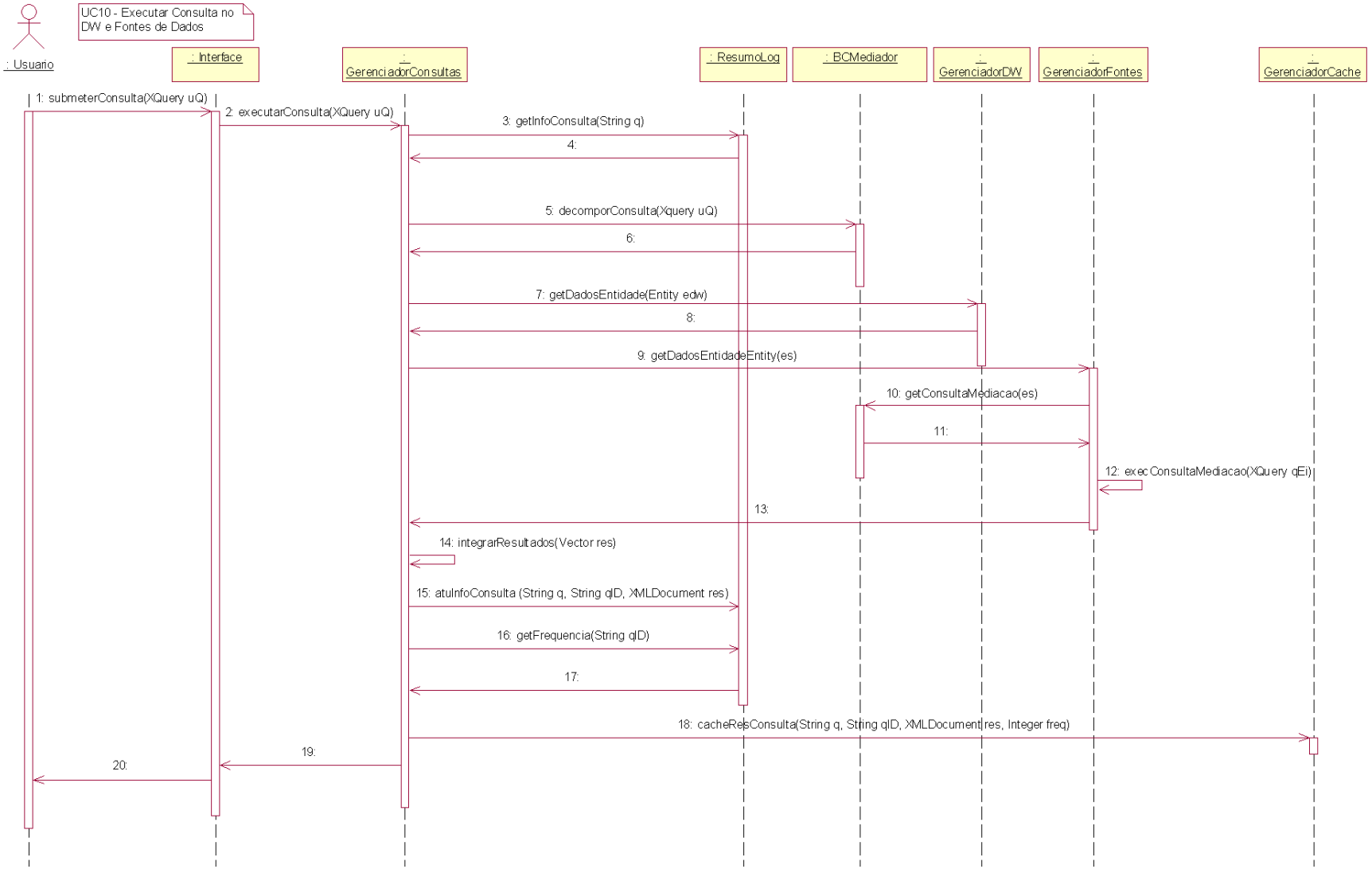


Figura 6.3 – Seqüência de processamento de uma consulta a partir das fontes e DW

A Tabela 6.2 mostra os passos, métodos, parâmetros, argumentos de retorno e classes da execução da consulta a partir das fontes de dados e do *data warehouse*.

Tabela 6.2 – Componentes do processo de execução de consulta a partir das fontes e DW

	Descrição	**Classe**	**Método(Parâmetros)**	**Retorno**
1	Usuário submete a consulta ao ambiente de integração de dados.	`Interface`	`submeterConsulta (XQuery uQ)`	Documento XML
2	A `Interface` solicita a execução à classe `GerenciadorConsultas`	`GerenciadorConsultas`	`executarConsulta (XQuery uQ)`	Documento XML
3	A classe `GerenciadorConsultas` recupera o status do resultado (se está ou não está na cache)	`ResumoLog`	`getInfoConsulta (String q)`	ID da consulta e o status indicando se o resultado está na cache.
4	`GerenciadorConsultas` solicita as entidades da consulta de usuário	`BCMediador`	`decomporConsulta (XQuery uQ)`	Entidades de mediação que fazem parte da consulta
5	`GerenciadorConsultas` solicita dados das entidades do *data warehouse*	`GerenciadorDW`	`getDadosEntidade (edw)`	Documento XML
6	`GerenciadorConsultas` solicita dados das entidades das fontes	`GerenciadorFontes`	`getDadosEntidade (es)`	Documento XML
7	`GerenciadorFontes` solicita consultas de mediação das entidades	`BCMediador`	`getConsultaMediacao (es)`	Consultas de mediação e mapeamento da entidade (XQuery)
8	`GerenciadorFontes` executa a consulta de mediação da entidade	`GerenciadorFontes`	`execConsultaMediacao (XQuery qEi)`	Documento XML
9	`GerenciadorConsultas` integra resultados	`GerenciadorConsultas`	`integrarResultados (Vector res)`	Documento XML
10	`GerenciadorConsultas` atualiza dados da consulta	`ResumoLog`	`atuInfoConsulta (String q, String qID, XMLDocument res)`	
11	`GerenciadorConsultas` recupera freqüência da consulta	`ResumoLog`	`getFrequencia (String qID)`	Valor da freqüência de submissão da consulta (inteiro)
12	`GerenciadorConsulta` solicita avaliação para armazenar resultado na *cache*	`GerenciadorCache`	`cacheResConsulta (String q, String qID, XMLDocument res, Integer freq)`	

Para o processo de execução de consultas a partir das fontes e do *data warehouse*, devido a limitações de tempo, os métodos `decomporConsulta()` da classe `BCMediador`, `execConsultaMediacao()` de `GerenciadorFontes` e `integrarResultados()` da classe `GerenciadorConsultas` não foram implementados. Assim, para poder concluir o processo sem essas implementações, tivemos de recorrer à simulação de entrada e saída dos métodos.

6.3.2 Armazenamento de Resultados na Cache

Como visto na seção anterior, sempre que uma consulta de usuário a partir das fontes e do *data warehouse* é processada, a classe `GerenciadorConsultas` instancia o método `cacheResConsulta()` pertencente à classe `GerenciadorCache` para que seja verificado na cache se o novo resultado deve ser armazenado ou não. Basicamente, através desse processo que a cache é populada e mantida pelo ambiente de integração. O processamento desse método produz uma seqüência de passos que são mostrados no diagrama da Figura 6.4.

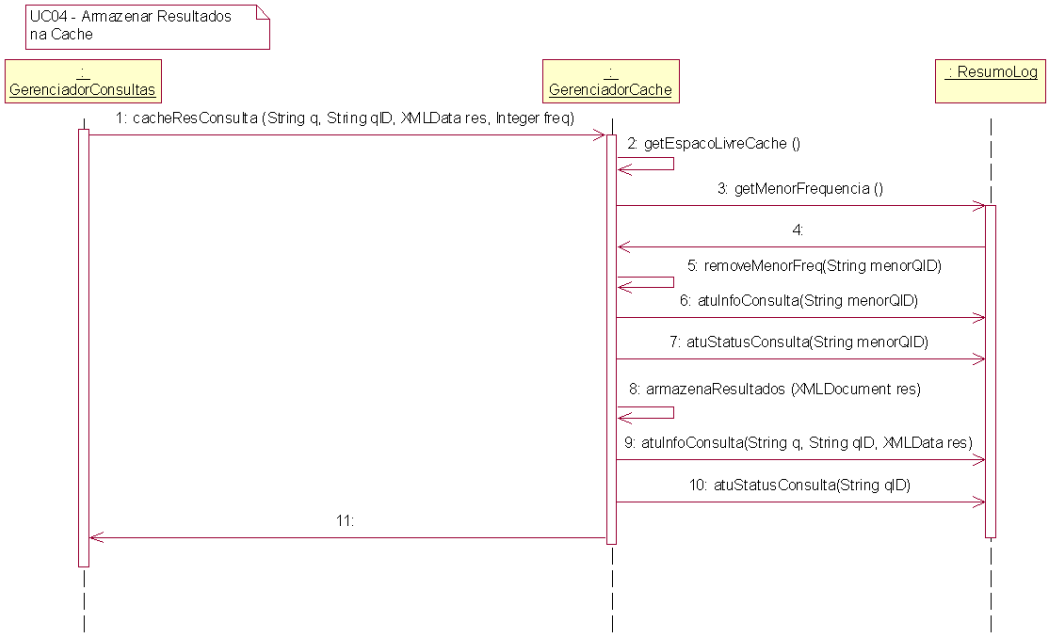


Figura 6.4 – Seqüência para armazenar resultado de consulta na *cache*

A classe `GerenciadorConsultas` instancia o método `cacheResConsulta()` da classe `GerenciadorCache`, e este por sua vez iniciará o processo de avaliação e possível substituição e armazenamento do resultado.

A classe `GerenciadorCache` calculará o espaço disponível da *cache* e fará a análise se o resultado novo cabe nesse espaço através do método próprio `getEspacoLivreCache()`. Também será instanciado o método `getMenorFrequencia()` da classe `ResumoLog` que

retorna uma identificação de consulta armazenada na *cache* juntamente com o espaço que seu o resultado ocupa, e se essa consulta tem um valor de freqüência de submissão menor do que a freqüência da consulta que está sendo avaliada para armazenamento do resultado.

De posse dessas informações o `GerenciadorConsultas` decidirá se o novo resultado deve ou não ser armazenado na *cache*. Em caso negativo, que ocorre quando os fatores espaço disponível e freqüência de submissão indicam que não vale a pena modificar o conteúdo da cache, o processo é suspenso.

Para casos em que o novo resultado pertence a uma consulta mais freqüentemente submetida e existe espaço disponível, a classe `GerenciadorConsultas` chamará o método `removerMenorFreq()` para o resultado de consulta que será retirado da *cache*. Em seguida invoca os métodos `atuInfoConsulta()` e `atuStatusConsulta()`para atualizar as informações em `ResumoLog` com relação à remoção de resultado da consulta na cache. O próximo passo é o armazenamento do novo resultado que ocorre através do método `armazenaResultados()`.

Novamente, os dois métodos da classe `ResumoLog`, `atuInfoConsulta()` e `atuStatusConsulta()`, serão instanciados desta vez para atualizar essa classe com as informações de inserção de um novo resultado de consulta que foi armazenado na *cache*. A Tabela 6.3 mostra os passos, métodos, parâmetros, argumentos de retorno e classes da execução da consulta com resultado armazenado na *cache*:

Tabela 6.3 – Componentes do processo de armazenamento de resultado na *cache*

	Descrição	**Classe**	**Método(Parâmetros)**	**Retorno**
1	`GerenciadorConsulta` solicita avaliação para armazenar resultado na *cache*	`GerenciadorCache`	`cacheResConsulta (String q, String qID, XMLDocument res, Integer freq)`	
2	`GerenciadorCache` verifica espaço disponível	`GerenciadorCache`	`getEspacoLivreCache()`	Espaço livre em KBytes (Inteiro)
3	`GerenciadorCache` solicita consulta com menor freqüência	`ResumoLog`	`getMenorFrequencia ()`	ID consulta e espaço em Kbytes que o resultado ocupa
4	`GerenciadorCache` remove consulta de menor freqüência	`GerenciadorCache`	`removeMenorFrequencia (String menorQID)`	
5	`GerenciadorCache` atualiza informações do resultado removido no Log	`ResumoLog`	`atuInfoConsulta (String menorQID)` `atuStatusConsulta (String menorQID)`	
6	`GerenciadorCache` armazena novo resultado	`GerenciadorCache`	`armazenaResultados (XMLDocument res)`	
7	`GerenciadorCache` atualiza informações do novo resultado no Log	`ResumoLog`	`atuInfoConsulta (String q, String qID, XMLDocument res)` `atuStatusConsulta (String menorQID)`	

Todos os métodos e funções do processo de armazenamento de resultado na *cache* foram implementados.

6.3.3 Seleção de Entidades para Materialização

Uma outra funcionalidade que foi implementada para o ambiente de integração de dados foi a seleção de entidades de mediação para materialização a partir da análise e classificação dos escores de critérios de materialização estabelecidos para as entidades e fontes de dados (seção 5.3.3.1).

A seleção de dados de entidades é efetuada periodicamente dentro de um período de tempo mantido pelo próprio sistema. Esse processo é iniciado por uma classe especial, a `Controlador` que é responsável por gerenciar todas as tarefas periódicas do sistema.

O diagrama da seqüência de execução da seleção está ilustrado na Figura 6.5:

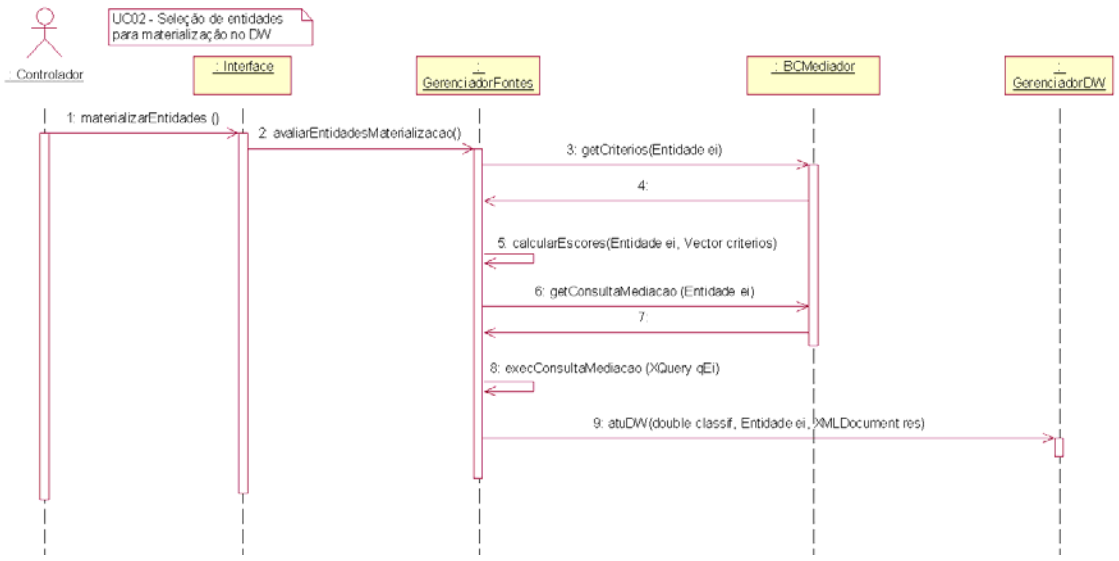


Figura 6.5 – Seqüência para o processo de seleção de entidades para materialização

A classe `Controlador` inicia o processo instanciando o método `materializarEntidades()` da classe `Interface`, que por sua vez, chama o método `avaliarEntidadesMaterializacao()` pertencente à classe `GerenciadorFontes()`.

Para cada uma das entidades de mediação o processo de seleção será repetido com os passos que serão descritos a seguir.

Inicialmente, a classe `GerenciadorFontes` acessa da classe `BCMediador` os valores dos critérios que serão avaliados através do método `getCriterios()`. De posse dos critérios e pesos, invoca o método `calcularEscores()` que fará o cálculo do escore global da entidade com base nos valores de escores obtidos do método anterior. Quando tiver calculado e

classificado os escores de todas as entidades do mediador, o `GerenciadorFontes` irá recuperar as consultas de mediação de cada uma através do método `getConsultaMediacao()` da classe `BCMediador`. Em seguida, executa cada consulta de mediação através do método `execConsultaMediacao()`.

Por último, a classe `GerenciadorFontes` entrega ao `GerenciadorDW` a classificação global dos critérios de materialização e os dados de todas as entidades a serem materializadas, para que essa classe decida quais as entidades que irão popular o *data warehouse* (método `atuDW()`).

A Tabela 6.4 mostra os passos, métodos, parâmetros, argumentos de retorno e classes da execução do processo de seleção de entidades para materialização.

Tabela 6.4 – Componentes do processo seleção de entidades para materialização

	Descrição	**Classe**	**Método(Parâmetros)**	**Retorno**
1	`Controlador` inicia o processo de seleção	`Interface`	`materializarEntidades()`	
2	`Interface` solicita avaliação de entidades para materialização	`GerenciadorFontes`	`avaliarEntidadesMaterializacao()`	
3	`GerenciadorFontes` recupera escores de critérios da entidade/fonte de dados	`BCMediador`	`getCriterios()`	Array de 4 valores, sendo um para cada critério
4	`GerenciadorFontes` calcula e uniformiza escores em um escore global da entidade	`GerenciadorFontes`	`calcularEscores (Entidade ei, Vector criterios)`	Valor do escore global da entidade
5	`GerenciadorFontes` solicita consultas de mediação das entidades	`BCMediador`	`getConsultaMediacao (ei)`	Consultas de mediação e mapeamento da entidade (XQuery)
6	`GerenciadorFontes` executa a consulta de mediação da entidade	`GerenciadorFontes`	`execConsultaMediacao (XQuery qEi)`	Documento XML
7	`GerenciadorFontes` entrega classificação e dados da entidade para o `GerenciadorDW`	`GerenciadorDW`	`atuDW (double classif, Entidade ei, XMLDocument res)`	

À exceção do método `execConsultaMediacao()`, todos os métodos especificados para o processo de seleção de entidades para materialização foram implementados.

6.4 Testes Efetuados

Uma vez que algumas operações e métodos não foram implementadas, não foi possível fazer um teste completo do sistema. Para concluir os processos que estavam incompletos tivemos de simular manualmente o processamento dos métodos que não estavam implementados. Assim, os testes que executamos foram suficientes apenas para validar a proposta do ambiente. Não foi possível testar o sistema funcionando no ambiente Web e com dados reais e conseqüentemente nós não pudemos extrair índices de otimização do processamento das consultas de usuário. A implementação dos métodos que faltaram, juntamente com os testes de carga na Web ficam para uma versão posterior de nosso protótipo.

Todos os módulos implementados no ambiente foram implementados e testados como classes Java *estáticas*. Isso significa que quando uma classe necessita interagir com uma outra, não existe necessidade de criar um objeto desta última classe para poder instanciar seus métodos. Isso implica em uma certa melhoria no desempenho do sistema como um todo já que, para todas as classes do sistema, não existe a necessidade do passo de criação de objetos de classe antes de poder instanciar os seus métodos.

6.5 Considerações Finais

Nesse capítulo foram discutidos os detalhes dos processos implementados no ambiente proposto para integração de dados.

Inicialmente foram mostradas as tecnologias adotadas: linguagem de programação, banco de dados, padrões de armazenamento de dados e esquemas e APIs de manipulação de dados e esquemas. Também foram mostradas as classes que compõem o sistema.

Dentre os processos propostos para serem executados pelos módulos da arquitetura do ambiente de integração, apenas os menos relevantes para a validação de nossa proposta não foram implementados. Para essa versão inicial do protótipo, os seguintes processos foram implementados:

- Execução de consulta de usuário com resultado recuperado da *cache*;
- Armazenamento de resultados na *cache*;
- Execução de consulta de usuário a partir das fontes de dados e *data warehouse*;
- Seleção de entidades para materialização.

Os dois primeiros processos foram totalmente implementados e validados. Nos dois últimos, não foram implementadas algumas operações, as quais tiveram de ser simuladas.

Mesmo tendo de efetuar testes pequenos e não ter sido possível verificar o comportamento do ambiente de integração de dados no ambiente Web e com dados reais, o protótipo atualmente implementado nos permitiu experimentar e explorar os recursos da arquitetura proposta.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Introdução

Sistemas de integração de dados comumente são baseados nas duas abordagens clássicas [Abiteboul99]: abordagem virtual que pode ser implementada pela arquitetura de mediadores e a abordagem materializada, implementada pela arquitetura de *data warehouse*. Sistemas de mediadores fornecem os dados integrados sempre atualizados, entretanto apresentam problemas do tipo: baixo desempenho e baixa qualidade nas respostas a consultas devido a indisponibilidade das fontes de dados. Sistemas de *data warehouse* oferecem um acesso rápido aos dados integrados, mas não garantem a consistência dos mesmos com relação ao conteúdo das fontes de dados, ou, pelo menos, apresentam custo muito alto com operações de manutenção do *data warehouse*.

A proposta de trabalho aqui discutida, é a construção de um ambiente de integração de dados na Web de arquitetura híbrida. Na arquitetura de um sistema de integração baseado em mediadores [Lóscio01], foram inseridos recursos para otimização no tempo de resposta das consultas de usuário: um repositório local ou *data warehouse* para materialização de dados e uma *cache* também local para armazenamento de resultados das consultas de usuário mais freqüentes.

Foi utilizado o padrão XML [Bray00] para representação dos dados e XML Schema [Biron01, Thompson01] para representação dos esquemas do ambiente. As consultas de usuário são expressas na linguagem XQuery [Boag02]. O mediador disponibiliza um esquema de mediação no qual as consultas de usuário são processadas. Neste sistema, o esquema do mediador foi definido no modelo *X-Entity* [Lóscio03] que é uma extensão do modelo E-R [Chen76] para XML Schema. O esquema do mediador é definido em termos de entidades e relacionamentos entre elas representando conceitos do mundo real. Cada entidade está

associada a uma consulta de mediação que computa os dados da mesma a partir das fontes de dados.

Quando falamos em otimização de desempenho, significa dizer que, queremos atingir boas taxas de processamento de consultas, mas sem incorrer na materialização de grandes volumes de dados acarretando um alto custo de manutenção de consistência. Ou seja, o ambiente foi projetado para obter ganhos significativos de desempenho dentro de parâmetros de custo de manutenção do conteúdo do *data warehouse* e da *cache* também aceitáveis. Para isso tivemos o cuidado de projetar o gerenciamento desses dois componentes com o máximo de automação de manutenção de conteúdo.

Foram abordados alguns sistemas de integração de dados construídos de acordo com uma arquitetura híbrida, um dos quais, descrito em [Ashish00], fornece um *framework* de materialização de porções de dados em uma arquitetura de mediadores através da análise das consultas de usuários previamente submetidas ao mediador. Desse estudo, nós aproveitamos para o nosso ambiente a proposta de efetuar uma materialização seletiva dos dados que serão armazenados no *data warehouse*, entretanto, a nossa proposta se mostra mais eficiente por dois motivos: (i) utiliza a entidade como unidade para materialização e; (ii) não efetua análises no espaço de busca. A nossa proposta de materialização seletiva é baseada na análise de critérios de qualidade previamente associados às entidades do mediador.

Outro recurso de nosso ambiente é a presença da *cache* para armazenar resultados das consultas que são submetidas com mais freqüência ao sistema de integração. O tempo de processamento e retorno da resposta de uma consulta cujo resultado está na *cache* é mínimo. No sistema NIMBLE [Draper01b], existe uma *cache* com finalidades semelhantes à nossa, entretanto os mecanismos de seleção de resultados para a *cache*, bem como as políticas de manutenção de conteúdo são efetuadas manualmente. No nosso ambiente, existem controles automatizados para verificação e substituição do conteúdo da *cache* e para efetuar *refresh* periódico desse conteúdo a fim de garantir que os resultados armazenados não fiquem inconsistentes por muito tempo.

Com relação a sistemas de integração de dados com arquitetura híbrida utilizando recursos de *caching* e *data warehouse*, nós não encontramos nenhum na literatura até a presente data.

Foi desenvolvida uma versão inicial de um protótipo para validação da proposta. Nele, os processos de análise de critérios de entidades para materialização, armazenamento de resultados e processamento de consultas na *cache* foram completamente implementadas. Devido às nossas limitações de tempo para conclusão deste trabalho, o processo de execução de consulta a partir das fontes de dados e do *data warehouse* e o processo de materialização de dados propriamente

dita foram parcialmente implementados. Os processos de *refresh* da *cache* e *refresh* do *data warehouse* foram os únicos que ficaram totalmente pendentes. Entretanto, são processos simples e de menor importância para a validação global do sistema.

7.2 Contribuições

Consideramos como principais contribuições do nosso trabalho, as relacionadas a seguir:

- Concepção de um ambiente de integração de dados seguindo uma arquitetura híbrida. O ambiente é baseado em mediadores com recursos de *cache* e *data warehouse* juntos com fins de maximizar ganhos de desempenho no processamento de consultas de usuários;
- Especificação de processos automatizados de controle e manutenção do conteúdo da *cache* para armazenar os resultados das consultas que são mais requisitadas ao sistema de integração;
- Especificação de processos seletivos de materialização de dados no *data warehouse* baseados na análise de critérios de qualidade e com finalidade de minimizar o custo de manutenção do *data warehouse*;
- Desenvolvimento de uma versão inicial de um protótipo para validação de nossa proposta.

O ambiente possui uma boa coleção de recursos e processos de controle para minimizar custos de processamento de consultas e custos de manutenção de dados materializados.

7.3 Trabalhos Futuros

Alguns tópicos que aqui ficaram pendentes e que acreditamos podem engrandecer nossa proposta se considerados como trabalhos futuros, são os seguintes:

- Estudar e definir uma política de substituição da cache mais eficiente [Dar96] para incorporar ao Gerenciador da Cache funções de otimização de gerenciamento e alocação de espaço para os dados armazenados na cache;
- Implementar as funções de decomposição e execução de consultas e de integração de dados que ficaram pendentes nessa versão do protótipo;
- Executar o sistema completo no ambiente Web processando consultas sobre dados reais para podermos avaliar e validar a proposta; elaborar estatísticas a respeito de melhorias no desempenho e também para analisar o processamento do ambiente como um todo;

- Incorporar novos critérios de QI (Qualidade da Informação) ao processo de seleção de dados para materialização a fim de torná-lo mais refinado, eficiente e completo;
- Incorporar técnicas mais sofisticadas para a identificação da consulta. Esse tópico está relacionado com o problema de *string matching* [Akutsu95] e pode ser solucionado com o a inserção no módulo Gerenciador de Consultas de algoritmos eficientes já existentes para a solução de problemas desse tipo [Akutsu95, Navarro99, Cho02] ou com o uso de mecanismos de assinatura para otimizar a avaliação de scripts de consultas [Koçberber96, Faloutsos97];
- Estudar formas de flexibilização da instalação da arquitetura proposta para que esta possa comportar os módulos independentes entre si de modo que seja possível adaptá-la às necessidades específicas de aplicações. Por exemplo, em certos casos uma aplicação pode requerer apenas a presença do Gerenciador de Consultas e do Gerenciador da Cache, sem necessitar materializar dados no *data warehouse*. O sistema deve então efetuar a instalação apenas dos módulos e funções requeridos.

Referências Bibliográficas

[Abiteboul97] Abiteboul, S. Querying Semi-structured Data. In Proceedings of the 6th International Conference on Database Theory, pp. 1-18, 1997.

[Abiteboul99] Abiteboul, S., Buneman, P. and Suciu, D. Gerenciando Dados na Web. 1a. Edição. Editora Campus, 272 p., 1999.

[Agrawal97] Agrawal, D., Al Abbadi, A., Singh, A. and Yurek, T. Efficient View Maintenance at Data Warehouses. In Proceedings of SIGMOD, pp. 417--427. 1997.

[Akutsu95] Akutsu, T. Approximate String Matching with Variable Length Don't Care Characters. Research Report, AL95-43-5, pp.33--38, Information Processing Society of Japan, 1995.

[Amaral02] Amaral, H. Uma Ferramenta para Gerenciamento de Consultas em um Ambiente de Integração de Dados na Web. Trabalho de Graduação do Curso de Ciência da Computação, Centro de Informática, UFPE, Recife, 2º Semestre de 2002.

[Ambite98] Ambite, J., Ashish, N., Barish, G., Knoblock, C., Minton, S. Modi, P., Muslea, I., Philpot, A. and Tejada, S. Ariadne - A System for Constructing Mediators for Internet Sources. In Proceedings of ACM SIGMOD Conf. on Management of Data, Seattle, WA, 1998.

[Armstrong03] Armstrong, E. The Java API for XML Processing Tutorial. . http://java.sun.com/webservices/docs/1.1/tutorial/doc/index.html. Acessado em Março/2003.

[Ashish00] Ashish, N. Optimizing Information Mediators By Selectively Materializing Data. PHd Thesis, USC Information Sciences Institute, 2000.

[Ashish98] Ashish, N., Knoblock, C. and Shahabi, C. Intelligent Caching for Information Mediators: A KR Based Approach. In Knowledge Representation meets Databases (KRDB), Seattle, WA, 1998.

[Ashish99a] Ashish, N., Knoblock, C. and Shahabi, C. Selectively Materializing Data in Mediators by Analyzing User Queries. In Proceedings of the 4th IFCIS International Conference on Cooperative Information Systems (CoopIS) pp. 256—266. Edinburgh, Scotland. 1999.

[Ashish99b] Ashish, N., Knoblock, C. and Shahabi, C. Selectively Materializing Data in Mediators by Analyzing Source Structure, Query Distribution and Maintenance Cost. In ACM CIKM 2nd Workshop on Web Information and Data Management (WIDM) Kansas City, Missouri, 1999.

[Azar00] Azar, F. Multiattribute Decision-Making: Use of Three Scoring Methods to Compare the Performance of Imaging Techniques for Breast Cancer Detection. Technical Report. Department of BioEngineering, University of Pennsylvania, Philadelphia, 2000.

[Baru99] Baru, C., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P. and Chu, V. XML-Based Information Mediation with MIX. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, Philadelphia, Pensylvania, USA, pp. 597-599, 1999.

[Biron01] Biron, P. and Malhotra, A. XML Schema Part 2: Datatypes – W3C Recommendation. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/. Acessado em Setembro/2002.

[Boag02] Boag, S., Chamberlin, D., Fernandez, M., Florescu, D., Robie, J., Siméon, J. and Stefanescu, M. XQuery 1.0: An XML Query Language – W3C Working Draft. http://www.w3.org/TR/2002/WD-xquery-20020430/. Acessado em Dezembro/2002.

[Booch03] Booch, G., Jacobson, I. and Rumbaugh, J. The OMG Unified Modeling Language Specification – Version 1.5. Rational Software Corporation. http://www.omg.org/technology/documents/formal/uml.htm. Acessado em Março/2003.

[Bouzeghoub00] Bouzeghoub, M. and Kedad, Z. A Quality-Based framework for Physical Data Warehouse Design. In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000), Stockhol, Sweden, June 5-6, 2000.

[Bray00] Bray, T., Paoli, J., Sperberg-McQueen, C. M. and Maler, E. Extensible Markup Language (XML) 1.0 – WC Recommendation. http://www.w3.org/TR/2000/REC-xml-20001006. Acessado em Agosto/2002.

[Brickley02] Brickley, D. and Guha, R. V. RDF Vocabulary Description Language 1.0: RDF Schema – W3C Working Draft. http://www.w3.org/TR/2002/WD-rdf-schema-20020430/. Acessado em Agosto/2001.

[Buneman97] Buneman, P. Semi-structured Data. In Proceedings of ACM Symp. on Principles of Database Systems, 1997.

[Chawathe94] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. and Widom, J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In Proc. of 10th Meeting of the Information Processing Society of Japan (IPSJ), 1994.

[Chen76] Chen, P.P. The Entity-Relationship Model: Toward a unified view of data, ACM Transactions on Database Systems, v.1, n.1, pp.1-36, 1976.

[Cho02] Cho, J. and Rajagopalan, S. A Fast Regular Expression Indexing Engine. In Proc. of 18th International Conference on Data Engineering (ICDE), San Jose, California, 2002.

[Clark99] J. Clark. XSL Transformations (XSLT) Version 1.0 – W3C Recommendation. http://www.w3.org/TR/xslt. acessado em Janeiro/2002.

[Cowan01] Cowan, J. and Tobin, R. XML Information Set – W3C Recommendation. http://www.w3.org/TR/2001/REC-xml-infoset-20011024. Acessado em Agosto/2001.

[Dar96] Dar, S. Franklin, M.J., Jónsson, B.T., Srivastava, D. and Tan, M., Semantic Data Caching and Replacement. In Proceedings of the 22nd VLDB Conference, Mumbai, India. 1996.

[Draper01a] Draper, D., Halevy, A. Y. and Weld D. S. The Nimble Integration Engine. In Proceedings ACM SIGMOD, Santa Barbara, California, USA, pp. 21-24, 2001.

[Draper01b] Draper, D., Halevy, A. Y. and Weld D. S. The Nimble XML Data Integration System[tm]. In Proceedings of Int. Conf. on Data Engineering (ICDE), 2001.

[Deutsch99] Deutsch, A., Fernandez, M. F., Florescu, D., Levy, A. and Suciu, D. A Query Language for XML. In Proceedings of the International World Wide Web Conference, Toronto, CA, 1999.

[Erdmann00] Erdmann, M. and Studer, R. How to Structure and Access XML Documents With Ontologies. Data and Knowledge Engineering (Special Issue on Intelligent Information Integration.), 2000.

[Faloutsos97] Faloutsos, C., Jagadish, H. V., Mendelzon, A. O. and Milo, T. A Signature Technique for Similarity-based Queries. In Proc. Compression and Complexity of Sequences (SEQUENCES '97), Positano, 1997.

[Fernandez02] Fernandez, M., Marsh, J. and Nagy, M. XQuery 1.0 and XPath 2.0 Data Model – W3C Working Draft. http://www.w3.org/TR/2002/WD-query-datamodel-20020430/. Acessado em Janeiro/2002.

[Florescu98] Florescu, D,. Levy, A. and Mendelzon, A. Database Techniques for the World-Wide-Web: A Survey. ACM SIGMOD Record, vol. 27, No. 3, pp. 59-74, 1998.

[Galhardas00] Galhardas, H., Florescu, D., Shasha, D., and Simon, E. An Extensible Framework for Data Cleaning. In Proceedings of the International Conference on Data Engineering (ICDE), San Diego, CA, 2000.

[Gupta95] Gupta, S. and Mumick, I. S. Maintenance of Materialized Views: Problems, Techniques and Applications. IEEE Data Engineering Bulletin. 18(2): pp. 3-18, June, 1995.

[Halevy00] Halevy, Y., Theory of Answering Queries Using Views, SIGMOD Record, v. 29, n.4, pp.40-47, 2000.

[Hammer95] Hammer, J., Garcia-Molina, H., Widom, J., Labio, W. and Zhuge, Y. The Stanford Data Warehousing Project. In Data Engineering Bulletin, Special Issue on Materialized Views and Data Warehousing, 18(2):pp. 41-48. June, 1995.

[Hammer99] Hammer, J. The Information Integration Wizard (IWiz) Project. Report on Work in Progress. Dept. of Computer & Information Science & Engineering, University of Florida, 1999.

[Hasselbring00] Hasselbring, W. Information System Integration. Communications of the ACM, vol. 43, no. 6, June, 2000.

[Hunter00] Hunter, J. and McLaughlin, B. Easy Java/XML integration with JDOM, Part 1. http://www.javaworld.com/javaworld/jw-05-2000/jw-0518-jdom.html.2000.

[Koçberber96] Koçberber, S. Partial Query Evaluation for Verticallly Partitioned Signature Files in Very Large Unformatted Databases. PHd Thesis, Computer Engineering and Information Science and Institute of Engineering and Science of Bilkent University, 1996.

[Krishnamurthy91] Krishnamurthy, R., Litwin, W. and Kent, W.. Language Features for Interoperability of Data Base with Schematic Discrepancies. ACM, 1991.

[Labio97] Labio, W. J., Zhuge, Y., Wiener, J. L., Gupta, H., Garcia-Molina, H. and Widom, J. The WHIPS Prototype for Data Warehouse Creation and Maintenance. In Proceedings of SIGMOD, pp. 557-559. 1997.

[Lassila01] Lassila, O. and Swick, R. Resource Description Framework (RDF) Model and Syntax Specification – W3C Recommendation. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. Acessado em Agosto/2001.

[Laux02] Laux, M. DBAccessor: A JDBC Wrapper Package http://developer.java.sun.com/developer/technicalArticles/Database/dbaccessor/. Acessado em Dezembro/2002.

[Lee02] Lee, Y., Strong, D., Kahn, B., and Wang, R. AIMQ: A Methodology for Information Quality Assessment. Information & Management, Volume 40, Issue 2, pp. 133-146. 2002.

[Levy00] Levy A. Y., Logic-Based Techniques in Data Integration, In J. Minker, editor Logic based Artificial Intelligence. Kluwer Publishers, 2000.

[Levy99] Levy, A. Y. Combining Artificial Intelligence and Databases for Data Integration. Artificial Intelligence Today, pp. 249-268, 1999.

[Liu00] Liu, L., Pu, C. and Han, W. XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources. In Proc. of the International Conference on Data Engineering (ICDE), pp. 611-621. 2000.

[Lóscio01] Lóscio, B. F., Salgado, A. C. and Vidal, V. M. P. Using Agents for Generation and Maintenance of Mediators in a Data Integration System on The Web. In Proc. of XVI Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, Brasil, 2001.

[Lóscio03] Lóscio, B. F., Managing the Evolution of XML-Based Mediation Queries. Tese de Doutourado em andamento. Curso de Ciência da Computação. Centro de Informática, UFPE, Recife, 2003.

[Ludascher99] Ludäscher, B., Papakonstantinou, Y., and Velikhov, P. A Brief Introduction to XMAS. 1999.

[MIX02] Página do Sistema MIX: http://www.npaci.edu/DICE/MIX. 2002.

[Mohan01] Mohan, C. Caching Technologies for Web Applications, Tutorial in 27th International Conference on Very Large Databases (VLDB2001), Rome, Italy. 2001.

[Munroe00] Munroe, K. D. and Papakonstantinou, Y. BBQ: A Visual Interface for Integrated Browsing and Querying of XML. 5th IFIP 2.6 Working Conference on Visual Database Systems. Fukuoka, Japan. 2000.

[Naumann98a] Naumann, F. and Freytag, J. C. Quality-driven Source Selection using Data Envelopment Analysis. In Proceedings of the 3rd Conference on Information Quality (IQ' 98). Cambridge, MA, 1998.

[Naumann98b] Naumann, F. Data Fusion and Data Quality. In Proceedings of the New Techniques and Technologies for Statistics Seminar (NTTS'98). Sorrent, Italy, 1998.

[Naumann99] Naumann, F. and Leser, U. Quality-driven Integration of Heterogeneous Information Systems. In Proceedings of the 25th International Conference on Very Large Databases (VLDB' 99) Edinburgh, UK. pp. 447-458, 1999.

[Naumann00] Naumann, F. and Rolker, C. Assessment Methods for Information Quality Criteria. In Proceedings of the Conference on International Quality (IQ00) Boston, 2000.

[Navarro99] Navarro, G. A Guided Tour to Approximate String Matching, ACM Computing Surveys, 33(1), 2001, 31--88.

[Nimble02] Nimble Technology, Inc. The Nimble Integration Suite White Paper. http://www.nimble.com/solutions/literature/. Acessado em Julho/2002.

[Pequeno00] Pequeno, V.M. Auto-manutenção de Classes de Fusão em Visões de Integração de Dados. Dissertação de Mestrado, Universidade Federal do Ceará. Fortaleza, Ceará, Abril, 2000.

[Rundensteiner00] Rundensteiner, E., Koeller, A. and Zhang, X. Maintaining Data Warehouses over Changing Information Sources. Communications of the ACM, Special Section on System Integration, 2000.

[Saaty99] Saaty, T. L. The Seven Pillars of The Analytic Hierarchy Process. In Proceedings of the 5th International Simposium on the Analytic Hierarchy Process (ISAHP'99), Kobe, Japan, 1999.

[Salgado01] Salgado, A.C. and Lóscio, B. F. Integração de Dados na Web. Jornada de Atualização em Informática, Congresso da SBC, Fortaleza, Brasil, 2001.

[Thompson01] Thompson, H. S., Beech, D., Maloney, M. and Mendelsohn, N. XML Schema Part 1: Structures – W3C Recommendation. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/. Acessado em Outubro/2002.

[Ullman97] Ullman, J. D., Information Integration Using Logical Views, in Proc. of ICDT'97, vol.1186 of LNCS, pp.19-40, Springer-Verlag, 1997.

[Vidal01] Vidal, V. M. P., Lóscio B. F. and Salgado, A. C. Using Correspondence Assertions for Specifying the Semantics of XML-Based Mediators. In Proc. of WIII'2001 International Workshop on Information Integration on the Web. Rio de Janeiro, 2001.

[Vioellau01] Vioellau, T. Java Technology and XML Part 1 - An Introduction to APIs for XML Processing. http://developer.java.sun.com/developer/technicalArticles/xml/JavaTechandXML/. Acessado em Março/2003.

[Wache01] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S. Ontology-based Integration of Information - a Survey of Existing Approaches. In Stuckenschmidt, H., ed., IJCAI-01 Workshop: Ontologies and Information Sharing, pp. 108--117. 2001.

[Wang93] Wang, R. Y., Kon, H. and Madnick, S. Data Quality Requirements Analysis and Modeling. In Proceedings of the 9th International Conference of Data Engineering, pp. 670—677, 1993.

[Wang96] Wang, R. Y. and Strong, D. Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management of Information Systems, 12, 4: pp.5-34, 1996.

[Widom95] Widom, J. Research Problems in Data Warehousing. In Proceedings of 4th Int´l Conference on Information and Knowledge Management (CIKM), 1995.

[Wiederhold92] Wiederhold, G. Mediators in the Architecture of Future Information Systems. IEEE Computer. 25(3): pp. 38-49. 1992.

[Wiener96] Wiener, J. L., Gupta, H., Labio, W. J., Zhuge, Y., Garcia-Molina, H. and Widom, J. A System Prototype for Warehouse View Maintenance. In Proc. of the ACM Workshop on Materialized Views: Techniques and Applications. Montreal, Canada. pp. 26 -33. 1996.

[Wood00] Wood, L., Le Hors, A., Le Hegaret, P., Nicol, G., Robie, J., Champion, M. and Byrne S. Document Object Model (DOM) Level 2 Core Specification – Version 1.0 – W3C Recommendation. http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/. Acessado em Novembro/2001.

[Zhu02] Zhu, Y. and Buchmann, A. Evaluating and Selecting Web Sources as External Information Resources of a Data Warehouse. In Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE' 2002), Singapore, 2002.

[Zhuge95] Zhuge, Y., Garcia-Molina, H., Hammer, J. and Widom, J. View Maintenance in a Warehousing Environment. In Proceedings of SIGMOD. pp. 316-327. 1995.