PROCESSAMENTO DE CONSULTAS SOBRE BASES XML DISTRIBUÍDAS

Guilherme Coelho de Figueiredo

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENACAO DOS PROGRAMAS DE POS-GRADUACAO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENCAO DO GRAU DE MESTRE EM CIENCIAS EM ENGENHARIA DE SISTEMAS E COMPUTACAO.

Aprovada por:

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

Profª. Vanessa de Paula Braganholo, D.Sc.

Prof. Geraldo Zimbrão da Silva, D.Sc.

Profª. Julia Célia Mercedes Strauch, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2007

FIGUEIREDO, GUILHERME COELHO DE

Processamento de Consultas sobre Bases XML Distribuídas [Rio de Janeiro] 2007

XII, 99p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2007)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Banco de Dados Distribuídos
2. Processamento de Consultas
3. Banco de Dados XML

I. COPPE/UFRJ II. Titulo (série)

A Daniela Saback de Figueiredo.

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

PROCESSAMENTO DE CONSULTAS SOBRE BASES XML DISTRIBUÍDAS

Guilherme Coelho de Figueiredo

Março/2007

Orientadores: Marta Lima de Queirós Mattoso
Vanessa de Paula Braganholo

Programa: Engenharia de Sistemas e Computação

O constante aumento do volume de dados armazenados na forma de documentos XML nativos faz com que a aplicação de técnicas de fragmentação de bases XML se torne uma alternativa importante para o problema de desempenho no processamento de consultas sobre estas bases de dados. Para que uma base XML possa ser fragmentada, é fundamental que exista uma forma de se consultar esta base distribuída de maneira transparente ao usuário. Esta dissertação apresenta a nossa metodologia para o processamento de consultas Xquery sobre bases de dados XML distribuídas e fragmentadas, que consiste nas etapas de decomposição da consulta, incluindo a representação da consulta em sua forma algébrica TLC; localização dos dados; otimização global; execução e consolidação dos resultados. Essa metodologia pode ser aplicada tanto em um banco de dados que permita a fragmentação de bases XML, quanto em um sistema que proporcione uma visão integrada de bancos de dados XML semi-autônomos homogêneos. Propomos o uso de uma arquitetura baseada em um Mediador com Adaptadores acoplados aos bancos de dados remotos. O Mediador fornece uma visão XML global dos dados distribuídos, que pode ser consultada pelos usuários de forma transparente. Um protótipo do Mediador e de Adaptadores para dois processadores de consulta XQuery diferentes foram implementados, e possibilitaram a execução de experimentos que mostraram os ganhos de desempenho e os impactos de diferentes consultas executadas sobre bases XML distribuídas.

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

QUERY PROCESSING OVER DISTRIBUTED XML DATABASES

Guilherme Coelho de Figueiredo

March/2007

Advisors: Marta Lima de Queirós Mattoso
Vanessa de Paula Braganholo

Department: Computer and Systems Engineering

The constant increase of the volume of data stored as native XML documents makes fragmentation techniques an important alternative to the performance issues in query processing over these data. Fragmented databases are feasible only if there is a transparent way to query the distributed database, without the need of knowing the fragmentation details and where each fragment is located. This document presents our methodology for XQuery query processing over distributed XML databases, which consists on the steps of query decomposition, including the query's TLC algebra representation; data localization; global optimization; global query execution and final result assembly. This methodology can be used in an XML database that allows fragmentation and also in a system that publishes an integrated view of semi-autonomous and homogeneous XML databases. We propose an architecture based on a Mediator with Adaptors (wrappers) attached to remote databases. The Mediator publishes a global XML view of the distributed data, which can be queried by users in a transparent way. A Mediator and two Adapters prototypes have been implemented and experiments were executed, where we could analyze the performance improvements and impacts of different queries over distributed XML databases.

Índice do Texto

Índice de Figuras

Índice de Tabelas

Nomenclatura

APT - *Annotated Pattern Tree*, ou padrão de árvore anotado: Consiste em uma forma de representação de uma consulta sobre uma base XML utilizada na álgebra TLC.

TLC - *Tree Logical Classes*: Álgebra para consultas sobre bases XML baseada em padrões de árvores anotados (APTs).

W3C - *World Wide Web Consortium*: Consórcio internacional de organizações com o objetivo de desenvolver padrões para tecnologias na *Web*.

XML - *eXtensible Markup Language*: Meta-linguagem definida pela W3C que permite representação de dados em formato customizado e semi-estruturado, muito utilizado na *Web* para troca de informações e mais recentemente como uma forma de armazenamento de dados.

Capítulo 1 - Introdução

1.1 Motivação

A distribuição em bancos de dados é um assunto bastante estudado na literatura desde a década de 1970, de onde surgiram os primeiros protótipos como o R* (WILLIAMS et al., 1986), SDD-1 (BERNSTEIN et al., 1981), e o INGRES (STONEBRAKER, 1986). Segundo Kossman (2000), em alguns aspectos, estes bancos de dados distribuídos estavam à frente do seu tempo. A tecnologia de comunicação, por exemplo, ainda não era estável o suficiente para transportar o volume de dados requerido por estes bancos de dados distribuídos, o que prejudicou o sucesso destes trabalhos naquela época.

Hoje, o processamento distribuído dos dados é viável e cada vez mais necessário. Muitas empresas estão substituindo suas plataformas centralizadas por sistemas distribuídos por razões como o alto custo dos sistemas centralizados e necessidades de escalabilidade e alta disponibilidade, mais viáveis econômica e tecnicamente em um ambiente distribuído; a necessidade de integração de diferentes módulos de softwares; a necessidade de integração com sistemas legados; novas aplicações sendo desenvolvidas somente para ambientes distribuídos; e, com isso, o direcionamento de todo o mercado para ambientes distribuídos (KOSSMAN, 2000). Além destas razões, quando estamos lidando com bancos de dados distribuídos, podemos utilizar técnicas de fragmentação de bases de dados, o que poderá trazer ainda ganhos de desempenho no processamento de consultas, principalmente sobre grandes volumes de dados (ÖZSU et al., 1999).

Com o advento da Internet e o uso de documentos XML (W3C, 2006) para a troca de mensagens entre aplicações remotas e inclusive como forma de armazenamento de dados, as linguagens de consulta sobre documentos e coleções XML se tornam cada vez mais maduras, como é o caso da XQuery que se tornou uma recomendação da W3C em 23 de janeiro de 2007 (BOAG et al., 2007). Em paralelo, surgem bancos de dados XML nativos para gerenciar grandes coleções de dados XML armazenados em sua forma nativa, como o TIMBER (JAGADISH et al., 2002), o TAMINO (SCHONING, 2001), o eXist (EXIST DEVTEAM, 2006), dentre outros.

O processamento distribuído de consultas XML ganhou importância com a popularidade do uso de documentos XML pela *Web* (SMILJANI´ et al., 2003). O comportamento instável e imprevisível de um sistema distribuído pela *Web* apresenta alguns problemas ao processamento distribuído de consultas, como a grande autonomia das bases de dados remotas; a necessidade de processar um *streaming* de dados XML, para melhorar o desempenho das consultas; e a instabilidade dos tempos de resposta devido às características da Internet. Neste contexto, surgiram uma série de sistemas para processamento de consultas sobre bases XML distribuídas na *Web*, como (GUPTA et al., 2000; SUCIU, 2002; AGUILERA et al., 2002; IVES et al., 2002; RE et al., 2004; SILVEIRA et al., 2005). Outros trabalhos enfocam o processamento de consultas XML sobre bases heterogêneas distribuídas, como (BARU et al., 1999; LEE et al., 2002; GARDARIN et al., 2002).

Com o aumento do volume de dados XML armazenado, cresce o interesse em se utilizar as técnicas de banco de dados distribuídos em bases XML nativas. Técnicas de fragmentação de documentos e bases XML já foram elaboradas (MA et al., 2003; BREMER et al., 2003; ANDRADE et al., 2006; ANDRADE, 2006). Em especial, as funções de fragmentação propostas por Andrade et al. fazem uso de uma álgebra XML e são também definidas regras formais de reconstrução de documentos XML a partir de seus fragmentos. Experimentos que mostram o potencial de ganho em desempenho para consultas realizadas sobre bases XML fragmentadas também foram explorados (ANDRADE et al., 2006). Porém, ainda não encontramos na literatura e nem no mercado, até o momento da conclusão desta dissertação, uma metodologia para processamento de consultas XQuery sobre bases XML distribuídas e fragmentadas.

Para que esse processamento seja automático e genérico, se faz necessário o uso de um formalismo XML. Através de uma álgebra XML, uma consulta XQuery pode ser reescrita através de expressões algébricas. Ao usar fragmentos definidos com operações dessa mesma álgebra, regras formais de equivalência entre um documento XML e seus fragmentos podem ser usadas. Assim, torna-se possível substituir um documento XML, referenciado numa expressão algébrica de uma consulta XQuery, por fragmentos que compõem esse documento. Essa substituição pode ser realizada de forma correta e automática, quando os fragmentos são definidos por operações da mesma álgebra usada na reescrita da consulta XQuery.

Entretanto, esse formalismo não é suficiente. Um processamento automático distribuído necessita de uma metodologia que defina etapas que possibilitem o processamento de consultas XQuery distribuídas.

Além do ponto da fragmentação de uma base XML nativa por questões de desempenho, fatores externos podem fazer com que um conjunto de bases de dados semi-autônomas possa ser considerado como uma base de dados XML distribuída e fragmentada. O dinamismo do mundo atual faz com que muitas empresas cresçam mais rápido do que a sua própria infra-estrutura tecnológica, fazendo com que surjam bases de dados descentralizadas armazenando informações globais da empresa. Filiais de pequenas e médias empresas, como uma rede de livrarias, uma franquia de cafeterias, etc. possuem bases de dados em cada filial que podem ser consideradas fragmentos do que seria uma representação global dos dados corporativos. Cada filial da rede de livrarias, por exemplo, teria uma base de dados autônoma apenas com os registros de estoque e de vendas daquela filial, o que representa de fato uma fragmentação horizontal de uma visão global dos registros das vendas de toda a rede de livrarias. No entanto, quando se deseja consultar os dados globais da organização, como o volume de vendas no mês em todas as filiais, os usuários têm que submeter consultas a cada base, e processar os resultados posteriormente. A outra alternativa é ter uma base centralizada com réplicas todas as bases das filiais. Supondo que é crítico para a empresa obter os resultados de uma consulta global em tempo real, o que inviabiliza de certa forma a solução através da replicação dos dados das filiais para uma base corporativa centralizada, podemos aplicar as técnicas de bases de dados distribuídas para criar uma representação global dos dados distribuídos da organização, facilitando assim a consulta sobre os dados corporativos. Esta representação global virtual das bases de dados, chamada a partir de agora de *visão global*, passaria a ser consultada diretamente pelos usuários, tornando transparente a consulta sobre as bases individuais. As consultas submetidas sobre uma visão global seriam então decompostas em sub-consultas e executadas sobre as *visões locais* das bases autônomas remotas. A visão global e as visões locais podem ser consideradas visões XML (ABITEBOUL, 1999) e a linguagem de consulta poderia ser a XQuery, já que esta é a linguagem padrão para consultas a dados XML.

A representação global dos dados de uma organização com bases distribuídas e semi-autônomas, como o caso do exemplo da rede de livrarias, feita através de visões XML, exige que as visões XML disponibilizadas pelas bases de

dados das lojas sejam totalmente homogêneas, já que não é o nosso objetivo tratar de integração de bases heterogêneas nesta dissertação. Entretanto, os dados que compõem esta visão XML local de uma loja podem estar armazenados em diferentes formas e modelos de representação, de forma totalmente transparente ao sistema que irá processar a consulta distribuída. Extrapolando o exemplo da rede de livrarias, podemos sugerir a utilização da nossa proposta de utilização de uma visão XML global de dados distribuídos também para casos de integração de bases de dados heterogêneas, desde que as visões XML locais de cada base sejam homogêneas. Desta forma, cada base terá que se publicar na forma de uma visão XML local, o que é viável para todos os modelos de dados, tanto relacional (SHANMUGASUNDARAM et al., 2000; FERNÁNDEZ et al., 2002; BRAGANHOLO et al., 2004), orientado a objetos (através de propostas para bancos de dados objeto-relacionais) (CAREY et al., 2000) e o próprio semi-estruturado.

1.2 Objetivos

O principal objetivo desta dissertação é contribuir com uma solução para o processamento de consultas XQuery sobre bases XML distribuídas e fragmentadas. Para isso, propomos uma metodologia de processamento de consultas XQuery distribuídas, inspirada na metodologia para SQL no modelo relacional (ÖZSU et al., 1999), que contempla as etapas de decomposição da consulta, que inclui a representação algébrica da consulta em TLC (PAPARIZOS et al., 2004); localização dos dados; otimização global; execução e consolidação dos resultados. Para implementar a metodologia propomos uma arquitetura baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992). A função do Mediador é prover um ponto único de acesso para os dados das fontes distribuídas, realizando todas as transformações necessárias na consulta original de forma a tornar a distribuição da base transparente ao usuário. O Mediador irá conter informações sobre a distribuição da base e seus fragmentos, armazenadas em um Catálogo, e realizará o processamento de uma consulta XQuery sobre a visão XML global dos dados para gerar sub-consultas destinadas aos Adaptadores. Cada Adaptador irá gerenciar a execução da sua sub-consulta sobre a base de dados do SGBD a ele acoplada, retornando o resultado ao Mediador. O Mediador fará a consolidação de todos os resultados dos Adaptadores, e enviará o resultado final ao usuário.

Para avaliar experimentalmente a nossa metodologia e arquitetura, desenvolvemos um protótipo da solução, incluindo o Mediador e dois tipos

diferentes de Adaptadores: um para o banco de dados XML nativo eXist (EXIST DEVTEAM, 2006) e outro para o Saxon (SAXONICA LIMITED, 2006), uma biblioteca para consulta XQuery de arquivos XML. O protótipo foi desenvolvido totalmente de acordo com as definições apresentadas nesta dissertação, permitindo a execução de experimentos para análise de desempenho de consultas em ambientes com e sem fragmentação de documentos XML.

Foram realizados diversos testes e experimentos em laboratório, com até três bases distribuídas. O objetivo dos experimentos visa analisar o desempenho da metodologia proposta e o conseqüente ganho em desempenho no processamento das consultas XQuery. Nesses experimentos, para cada consulta executada em ambiente distribuído, foram comparados os resultados da mesma consulta em ambiente centralizado; e monitorados os tempos de processamento e execução das consultas, comparando-os aos tempos em ambiente centralizado. Foram monitorados os tempos de processamento da consulta no Mediador, nos Adaptadores remotos, os tempos de comunicação e os tempos de execução da consulta pelos bancos de dados XML nativos. Utilizamos consultas com e sem benefício da fragmentação das bases, de forma a poder observar não só os casos em que a fragmentação melhora o desempenho da consulta, mas também os casos onde a fragmentação pode ser prejudicial ao desempenho.

1.3 Organização dos Capítulos

Os estudos e análises realizados no desenvolvimento desta Dissertação foram organizados em Capítulos, como segue.

O Capítulo 2 apresenta uma revisão bibliográfica sobre as características do processamento de consultas em ambientes distribuídos, tanto nos modelos relacional, orientado a objetos e no modelo semi-estruturado (XML). São apresentadas as principais álgebras para processamento de consultas em XML, algumas relacionadas à linguagem XQuery. Por fim, apresentamos algumas técnicas de fragmentação de documentos ou coleções XML.

No Capítulo 3 é feita uma revisão teórica sobre a técnica de fragmentação e sobre a álgebra para consultas em bases XML escolhidas para serem utilizadas na nossa proposta. O objetivo do capítulo é fornecer ao leitor fundamentos teóricos antes da apresentação da proposta de decomposição de consultas XQuery distribuídas.

O Capítulo 4 apresenta a proposta deste trabalho: uma metodologia para processamento de consultas XQuery sobre bases XML distribuídas e fragmentadas. São apresentadas as etapas, da metodologia, necessárias para o processamento da consulta distribuída, compreendendo as etapas de decomposição da consulta; localização das visões globais dos fragmentos distribuídos; otimização global da consulta; e composição do resultado final.

A implementação da metodologia em uma arquitetura baseada em um Mediador com Adaptadores, para validar experimentalmente a proposta, é apresentada no Capítulo 5. Nele, são detalhadas questões de implementação do Mediador e dos Adaptadores, bem como a interface de comunicação entre os componentes da arquitetura e o Catálogo, que armazena as definições das visões globais e de seus fragmentos.

No Capítulo 6 apresentamos os experimentos realizados em laboratório sobre o protótipo implementado. Descrevemos a metodologia de testes utilizada, as bases, fragmentos e consultas utilizadas nos experimentos. Os resultados são discutidos e apresentados graficamente.

Por fim, as conclusões deste trabalho são apresentados no Capítulo 7, onde são reportadas algumas considerações finais e as contribuições desta dissertação, além de algumas indicações para trabalhos futuros.

Capítulo 2 - Consultas sobre Bases de Dados XML Distribuídas

2.1 Introdução

Ambientes distribuídos já se tornaram uma realidade. Para se manterem competitivas, empresas estão precisando modificar sua infra-estrutura de TI, aderindo à plataforma distribuída, integrando sistemas legados, comprando sistemas mais modernos e se integrando eletronicamente com empresas parceiras através de trocas de mensagens pela Internet. O volume de dados armazenados e trafegados pela rede cresce exponencialmente e o seu armazenamento físico se torna crítico para poder se extrair algum tipo de informação ou conhecimento a partir deles.

Com este aumento do volume de dados armazenados nas organizações, surge a necessidade de se aplicar técnicas que permitam a realização de consultas sobre estas bases de dados de forma mais eficiente. Uma técnica possível é a fragmentação da base de dados, que visa dividir uma base de dados em fragmentos e alocá-los em diversos nodos de um ambiente distribuído. A fragmentação da base de dados procura obter ganhos de desempenho através do paralelismo da execução da consulta, já que a consulta será distribuída e enviada aos diferentes nodos que a executarão em paralelo, sobre um volume menor de dados em cada nodo. A fragmentação da base de dados também pode reduzir a quantidade de dados acessados pela consulta, se esta utilizar em seu critério de seleção o mesmo atributo utilizado na fragmentação. Se uma base está fragmentada pelo atributo *ano*, por exemplo, e a consulta submetida busca informações sobre o ano atual, apenas o fragmento com dados do ano atual precisará ser acessado, evitando o acesso aos fragmentos com dados de anos anteriores.

Por outro lado, a fragmentação de uma base de dados também pode degradar ainda mais o desempenho de uma consulta. Existem situações onde o trabalho para se agrupar os resultados de cada nodo do ambiente distribuído faz com que o desempenho da consulta distribuída seja inferior ao desempenho em um ambiente centralizado. Por isso, o projeto da fragmentação da base de dados analisa os padrões de consultas mais freqüentes para que a fragmentação

apresente ganho de desempenho para a maioria das consultas realizadas sobre a base de dados. Além disso, com a distribuição da base de dados também devemos levar em conta o tempo gasto com a comunicação entre os nodos e as etapas do processamento da consulta distribuída que não podem ser paralelizadas, como funções de agregação de resultados.

Por esses motivos, a fragmentação da base de dados deve ser projetada de acordo com técnicas ou metodologias, para que a fragmentação seja eficiente no contexto em questão.

Técnicas de projeto de fragmentação de bases de dados são bem conhecidas tanto para modelos relacionais (ÖZSU et al., 1999) quanto para modelos orientados a objeto (BAIÃO et al., 2004). Para o modelo semi-estruturado, algumas técnicas de projeto de fragmentação estão sendo propostas (BREMER et al., 2003).

Outro problema decorrente da fragmentação de uma base de dados é o controle semântico dos dados, principalmente o controle de integridade. Tarefas de verificação de integridade irão envolver a consulta a informações em nodos remotos, o que aumentará a complexidade das operações, acarretando possivelmente em perda de desempenho.

Esta dissertação apresenta uma metodologia para o processamento de consultas sobre bases de dados XML fragmentadas e distribuídas. Antes disso, na seção 2.2 iremos apresentar as técnicas de processamento de consultas distribuídas. Técnicas para fragmentação de bases XML, fundamentais para o nosso trabalho, serão apresentadas na seção 2.3. Por tratarmos de consultas sobre bases XML, também precisaremos analisar as álgebras para consultas em XML, apresentadas na seção 2.4. Após esta visão geral sobre as técnicas existentes para o processamento de consultas distribuídas, a fragmentação de documentos XML e álgebras para consultas XML, fazemos uma apresentação mais detalhada, no Capítulo 3, sobre a técnica de fragmentação e a álgebra para XML escolhidas para utilização na dissertação.

2.2 Consultas Distribuídas

O processamento de consultas sobre bases fragmentadas e distribuídas apresenta algumas dificuldades adicionais em relação ao processamento de consultas sobre um ambiente centralizado. Em um ambiente distribuído, o SGBD precisa conhecer a localização dos nodos do ambiente e incluir parâmetros de

custo de comunicação e carga de processamento em cada nodo nas funções de custo do otimizador de consultas. A fragmentação da base adiciona ainda complexidades relacionadas à redução da consulta distribuída de forma a ser executada apenas nos nodos contendo dados relevantes ao resultado. Em (ÖZSU et al., 1999), encontramos uma excelente referência sobre bancos de dados distribuídos e sobre uma metodologia de processamento de consultas distribuídas em bases de dados relacionais, cujas idéias gerais podem ser adaptadas para outros modelos de dados. Tópicos mais avançados sobre o processamento de consultas distribuídas, como técnicas de otimização e de execução, replicação dinâmica de dados no ambiente distribuído, *caching*, arquiteturas, etc. são encontrados em (KOSSMAN, 2000).

De forma geral, o processamento de consultas distribuídas consiste na transformação de uma consulta de alto nível, sobre uma visão global (centralizada) dos dados distribuídos, em uma ou mais sub-consultas de mais baixo nível, direcionadas aos nodos do ambiente distribuído. Apresentaremos a seguir um resumo do que se encontra na literatura sobre as técnicas de fragmentação e de processamento de consultas distribuídas no modelo relacional, orientado a objetos e no modelo semi-estruturado.

2.2.1 Modelo Relacional

O modelo relacional, baseado em atributos, tuplas e relações, é o modelo de dados mais maduro e difundido no mercado. Atributos representam um valor atômico, com um tipo pré-definido; uma tupla representa um conjunto de atributos; e, por sua vez, uma relação representa um conjunto de tuplas. Uma relação pode ser representada por uma tabela, com várias colunas e linhas. As linhas correspondem às tuplas e as colunas aos atributos (ELMASRI et al., 2001).

Para a fragmentação de uma relação, pode-se adotar três estratégias: a fragmentação horizontal, vertical e/ou híbrida. A fragmentação horizontal divide uma relação em sub-conjuntos de suas tuplas, gerando um conjunto de novas relações com o mesmo esquema da relação original. A fragmentação horizontal é definida através da operação de seleção da álgebra relacional. Existe também um tipo especial de fragmentação horizontal: a fragmentação horizontal derivada, onde uma relação dependente (que possui chave estrangeira) de uma relação já fragmentada horizontalmente também é fragmentada com base nesta dependência. O objetivo da fragmentação horizontal derivada é otimizar a execução de operações de junção entre as relações dependentes. A fragmentação

vertical separa uma relação em um conjunto de relações menores, cada uma com um sub-conjunto de atributos da relação original, mantendo sempre os atributos da chave primária para que a relação original possa ser reconstruída. A fragmentação vertical é definida através da operação de projeção da álgebra relacional. Já a fragmentação híbrida é obtida aplicando uma fragmentação horizontal seguida de uma vertical ou vice e versa. A grande vantagem da definição de fragmentos através de operações da álgebra relacional é permitir a composição dessas operações de fragmentação nas consultas algébricas derivadas das consultas SQL sobre as relações globais. Maiores detalhes sobre técnicas de fragmentação no modelo relacional podem ser obtidos em (ÖZSU et al., 1999).

Para que o processamento de consultas sobre uma base distribuída seja possível, é necessário que a técnica de fragmentação da base forneça as regras de reconstrução dos fragmentos, de forma a permitir que a relação global original possa ser reconstruída automaticamente a partir de operadores da álgebra.

Ozsü e Valduriez (1999) apresentam uma metodologia para o processamento de consultas distribuídas sobre o modelo relacional que consiste de etapas básicas de processamento distribuído, a saber, a etapa de decomposição da consulta; a etapa de localização dos dados; a etapa de otimização global da consulta; e a etapa de otimização local, conforme mostrado na Figura 1.

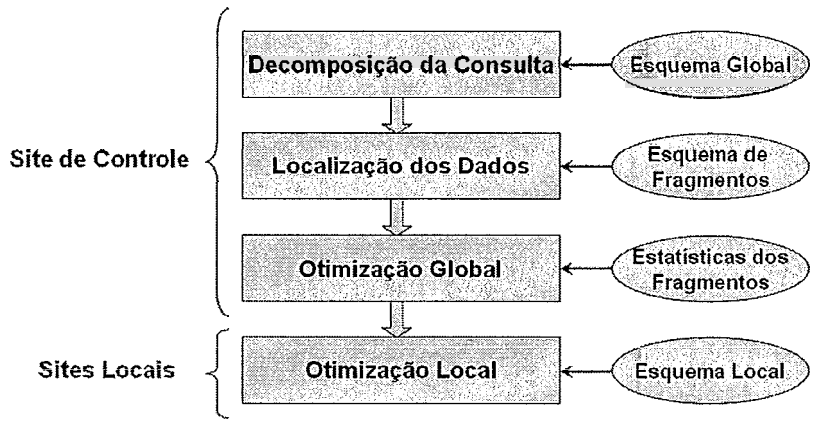


Figura 1: Metodologia de processamento de consultas distribuídas (ÖZSU et al., 1999)

A primeira etapa decompõe a consulta em uma consulta algébrica sobre relações globais. Neste processo são feitas análises sintáticas e semânticas sobre a consulta submetida; a consulta é simplificada e reescrita na forma algébrica. Informações sobre a distribuição dos dados ainda não são utilizadas nesta etapa.

A etapa de localização dos dados tem como objetivo a localização dos dados da consulta com o uso de informações sobre a distribuição (fragmentação e alocação) dos dados. São determinados os fragmentos que estão envolvidos na consulta, e é feita uma substituição na consulta algébrica para inclusão dos fragmentos no lugar das relações globais. Essa substituição pode ser feita de modo automático quando o projeto de fragmentação obedece a regras de correção (ÖZSU et al., 1999) como a reconstrução da relação a partir de seus fragmentos.

A etapa de otimização global da consulta procura encontrar uma estratégia (ou plano) de execução da consulta global que seja próxima da estratégia ótima. A otimização é feita, em geral, a partir da minimização de uma função de custo, que é utilizada para cada estratégia de execução. A função de custo é uma combinação ponderada de custos de CPU, de comunicação e de Entrada e Saída. Para bancos de dados distribuídos sobre redes mais lentas, o custo de comunicação tende a ser o fator mais significativo da função de custo.

Após a otimização global da consulta, são geradas sub-consultas destinadas aos *sites* remotos do ambiente distribuído. Cada sub-consulta executada em um *site* é otimizada com o uso do esquema local do *site*, na etapa de otimização local. Os resultados das sub-consultas remotas são processados pelo SGBD de acordo com a estratégia de execução para a composição do resultado final da consulta.

Maiores informações sobre as etapas da metodologia de processamento de uma consulta distribuída podem ser encontradas em (ÖZSU et al., 1999).

2.2.2 Modelo Orientado a Objetos

Em fevereiro de 2006, o *Object Management Group* (OMG) anunciou que irá desenvolver o padrão para a quarta geração de bancos de dados orientados a objeto, objetivando facilitar a adoção da tecnologia internacionalmente. Para isso, foi criado o *Object Database Technology Working Group* (ODBT WG) e adquiridos os direitos para o desenvolvimento de uma nova especificação da OMG baseada nos trabalhos conduzidos pelo extinto *Object Data Management Group* (ODMG), que lançou o último padrão ODMG 3.0 em 2001 (ODBMS.ORG, 2006).

Embora ainda não exista um consenso sobre o modelo de dados formal para bancos de dados orientados a objetos (ÖZSU et al., 1999; ELMASRI et al., 2001; BAIÃO et al., 2004), existem trabalhos na literatura relacionados à fragmentação de bases orientadas a objeto (BAIÃO et al., 2000), a decomposição

de consultas sobre bases OO (JOSIFOVSKI et al., 2002), e uma metodologia para a fragmentação de bases OO (BAIÃO et al., 2004).

Segundo o modelo utilizado em (BAIÃO et al., 2004), um tipo representa aspectos estruturais e comportamentais de um objeto. Um tipo é identificado pelas suas propriedades (atributos e relacionamentos) que refletem o estado do objeto, e as operações que podem ser realizadas sobre o objeto. Uma classe é um agrupamento de todas as instâncias de objetos de um dado tipo. Uma classe é formada por um identificador, um conjunto de atributos, um conjunto de métodos e um conjunto de objetos (instâncias). Um objeto é a primitiva fundamental de modelagem e acesso em um SGBDOO. Ele é um encapsulamento de um identificador de objeto (OID) representando sua identidade única no sistema e um conjunto de variáveis de instância (atributos) definindo seu estado. O conjunto de variáveis de instância do objeto determina o valor atual dos atributos definidos em sua classe. Uma coleção é um agrupamento de objetos definido pelo usuário. Este pode ser um conjunto, um *bag* (um conjunto onde repetições são permitidas), uma lista (ordenada) ou um vetor (acesso direto aos elementos através de sua posição) de objetos.

A fragmentação de uma base orientada a objetos também pode ser feita seguindo as estratégias de fragmentação horizontal, vertical e híbrida. Na fragmentação horizontal, são distribuídas instâncias de uma classe (objetos) pelos fragmentos através de critérios de seleção aplicados sobre os atributos da classe. A fragmentação horizontal derivada pode ocorrer de várias maneiras: derivada da fragmentação de uma subclasse; derivada da fragmentação de um atributo complexo; ou derivada da fragmentação de métodos complexos. A fragmentação vertical procura distribuir atributos e métodos da classe entre os fragmentos. A fragmentação híbrida é obtida aplicando as duas estratégias ao mesmo tempo. Maiores informações sobre estratégias de fragmentação de bases orientadas a objeto podem ser encontradas em (ÖZSU et al., 1999; ELMASRI et al., 2001; BAIÃO et al., 2004).

2.2.3 Modelo Semi-estruturado (XML)

O modelo de dados semi-estruturado foi definido para dados que não se adequam a um esquema rígido (BUNEMAN, 1997; ABITEBOUL et al., 1999). Este modelo permite que o dado seja parcialmente estruturado, de forma que componentes do dado possam estar faltando em alguns itens, possam ter tipos diferentes em itens diferentes, e coleções de itens possam ser heterogêneas. Em

um modelo de dados semi-estruturado, uma base de dados é modelada como um grafo rotulado e com uma raiz. Os nós representam objetos e possuem um identificador associado (*oid*). O XML (W3C, 2006), formato padrão para a *Web*, é essencialmente uma sintaxe para dados semi-estruturados.

Um repositório de dados XML pode ser de dois tipos: Único Documento (*Single Document*, SD) e Múltiplos Documentos (*Multiple Document*, MD) (YAO et al., 2002). Em repositórios do tipo SD, todas as informações estão armazenadas em um único documento XML (com um esquema definido em XML *Schema* ou DTD). Já em repositórios MD, também existe a definição de um esquema, porém, o repositório é formado por múltiplas instâncias (documentos XML) desse esquema, formando uma coleção de documentos.

A natureza hierárquica e semi-estruturada do XML representa uma dificuldade a mais no tratamento de consultas distribuídas. Alguns trabalhos na literatura (SUCIU, 2002; RE et al., 2004; SILVEIRA et al., 2005) tratam do processamento de consultas sob modelos XML distribuídos.

Um dos primeiros trabalhos sobre processamento de consultas distribuídas sobre o modelo semi-estruturado é apresentado em (SUCIU, 2002), onde o autor analisa o problema para dois tipos de consultas sobre bases XML: expressões de caminho e consultas do tipo *select-where*. O autor analisa a avaliação eficiente de consultas distribuídas, a partir da minimização dos tempos gastos com comunicação e da minimização do volume de dados trafegados entre os nodos remotos. A proposta não contempla operações de junção entre bases remotas e também não trata da fragmentação das bases.

Uma extensão da XQuery é proposta em (RE et al., 2004), que permite a execução de sub-consultas nas bases remotas declaradas diretamente dentro da sintaxe estendida da XQuery. Esta proposta tem como objetivo possibilitar uma pré-seleção remota sobre o documento que será utilizado na consulta, evitando que o documento inteiro tenha que ser transportado para o servidor que executa a consulta. Uma desvantagem desta abordagem é que o conhecimento da estrutura das bases remotas fica sob responsabilidade do usuário, o que limita a sua utilização.

Em (SILVEIRA et al., 2005), os autores tratam o problema da integração de diferentes fontes de dados XML, possivelmente heterogêneas (diferentes *schemas* XML em um mesmo domínio de aplicação), através de um mediador onde são definidos o modelo conceitual global, e definições dos modelos

conceituais locais e seus mapeamentos para os *schemas* XML das fontes de dados. O problema da decomposição de consultas CXPath (*Conceptual XPath* – uma linguagem de consulta definida sobre o modelo conceitual Global proposta em (CAMILLO et al., 2003)) em consultas XQuery sobre as diferentes fontes de dados XML também é abordado. Esta tradução considera os predicados de fragmentação dos modelos conceituais locais. Os autores definem operações para fragmentação dos modelos conceituais (a partir de um modelo conceitual global), e também um algoritmo para decomposição de consultas globais, que utiliza informações de fragmentação do modelo conceitual global em modelos locais e de mapeamentos dos modelos conceituais locais nos *schemas* XML das fontes de dados.

Alguns trabalhos tratam da integração de bases heterogêneas distribuídas utilizando o XML como o padrão comum entre as bases heterogêneas e um mediador (BARU et al., 1999; LEE et al., 2002; GARDARIN et al., 2002). Estes trabalhos não lidam com fragmentação das bases, possuindo maior foco no tratamento dos esquemas heterogêneos das bases acopladas ao Mediador. Estas bases são acessadas por adaptadores que fornecem uma visão XML dos dados e suportam a execução de consultas sobre esta visão.

2.3 Técnicas de Fragmentação de Bases XML

Para trabalharmos com consultas distribuídas em XML, precisamos de uma definição formal para a fragmentação das bases de dados XML. Esta definição deve nos permitir aplicar regras de reconstrução da coleção global a partir dos fragmentos, necessárias para a decomposição da consulta distribuída. Diversas técnicas para fragmentação de repositórios XML já foram propostas na literatura (MA et al., 2003; BREMER et al., 2003; ANDRADE et al., 2006).

Em (MA et al., 2003) são propostos três tipos de fragmentação: horizontal, que agrupa elementos de um documento XML de acordo com um predicado de seleção; vertical, que reestrutura um documento XML; e "*split*", que quebra um documento XML em um conjunto de novos documentos. Estas definições de fragmentação não são acompanhadas de uma álgebra XML que proporcione a definição algébrica de reconstrução correta do documento original, o que impossibilita a sua utilização para a decomposição automática de consultas algébricas sobre os documentos XML distribuídos e conseqüente composição do resultado.

No trabalho de (BREMER et al., 2003) é proposta uma abordagem para o projeto de distribuição de uma base XML, que contempla tanto a fragmentação da base quanto a alocação dos fragmentos. Entretanto, a abordagem considera apenas repositórios de um único documento (*single document* - SD), e não há uma distinção formal entre fragmentação horizontal e vertical, que são combinados em um tipo híbrido. Os fragmentos são definidos em uma linguagem derivada da XPath, e as suas definições são armazenadas em um repositório com os metadados da base distribuída. Este repositório de metadados será utilizado no processamento das consultas distribuídas, e deverá ser replicado em todos os nodos do ambiente distribuído.

Em (ANDRADE et al., 2005; ANDRADE et al., 2006), os autores definem formalmente as fragmentações horizontal, vertical e híbrida de repositórios XML. Os fragmentos são definidos através de operações da álgebra TLC (PAPARIZOS et al., 2004), o que permite a decomposição de consultas sobre os fragmentos, ao utilizar a álgebra TLC para consultas XQuery. Um fragmento é considerado horizontal quando é criado a partir de uma operação de seleção sobre o documento original; vertical quando criado a partir de uma operação de projeção sobre o documento original; e híbrido quando são utilizadas operações de seleção e projeção sobre o documento original para criação do fragmento. Além do aspecto formal integrado a uma álgebra de consultas XQuery, as definições de Andrade e outros, pode ser aplicada tanto a repositórios de um único documento (*single document* - SD) como a coleções de documentos (*multiple document* - MD). Um outro aspecto interessante deste trabalho é que as definições de fragmentação se assemelham muito com as definições propostas para o modelo relacional (ÖZSU et al., 1999), assim, a adaptação de técnicas bem sucedidas no modelo relacional para o modelo XML se torna uma opção atraente e promissora. Além disso, regras de correção são apresentadas pelos autores para cada tipo de fragmentação, o que é essencial para a decomposição, automática e correta, da consulta sobre o repositório global em fragmentos correspondentes.

A partir da análise das técnicas de fragmentação de bases XML disponíveis na literatura, optamos pela técnica definida em (ANDRADE et al., 2006), pois contempla tanto bases formadas por um único documento quanto coleções de documentos e possui as definições formais das regras de correção e reconstrução da base original sobre a álgebra TLC, o que é fundamental para o processamento de consultas sobre bases XML distribuídas. Maiores detalhes

sobre a técnica de fragmentação definida por Andrade e outros são apresentadas no Capítulo 3.

2.4 Álgebras para Consultas em XML

Os trabalhos já realizados sobre os modelos relacional e orientado a objetos nos mostraram que a utilização de uma álgebra formal é fundamental para o processamento e a otimização de consultas. Documentos XML possuem uma estrutura muito flexível que, embora permita a criação de documentos semi-estruturados, torna mais complexa a criação de uma álgebra para consultas em XML. Embora ainda não exista uma álgebra de consenso para o processamento de consultas sobre bases XML, existem diversos trabalhos na literatura sobre o assunto (FERNÁNDEZ et al., 2000; JAGADISH et al., 2001; ZHANG et al., 2002; FRASINCAR et al., 2002; CHEN et al., 2003; PAPARIZOS et al., 2004).

O processador de XQuery Galax (THE GALAX TEAM, 2006) utiliza uma álgebra interna (FERNÁNDEZ et al., 2000) para o processamento das consultas. Esta álgebra define operações de projeção, iteração, seleção, quantificação, junção, ordenação, agrupamento e funções de agregação (*avg, count, max, min e sum*), além de operações para reestruturação da consulta. As operações são ortogonais, fortemente tipadas, e obedecem a leis de equivalência e de otimização, utilizadas pelo otimizador do Galax para simplificação e otimização da consulta.

A álgebra XAL - *XML ALgebra* (FRASINCAR et al., 2002), segundo seus autores, se destaca das demais devido à possibilidade de se utilizar heurísticas de otimização semelhantes às heurísticas de otimização em álgebra relacional. A álgebra define três grupos de operações: operações de extração que recuperam a informação necessária através dos documentos XML; meta- operações, que controlam a avaliação das expressões; e operações de construção que constroem novos documentos XML com base nos dados extraídos. Entre as operações disponíveis, temos seleção, projeção, ordenação, junção, produto cartesiano, união, diferença, interseção, distinção (eliminação de duplicatas) e uma operação para retirada da ordem de uma coleção.

A XAT – *XML Algebra Tree* foi utilizada em um projeto de um processador de consultas XQuery sobre visões XML de bases relacionais (ZHANG et al., 2002). A álgebra é utilizada para otimizar uma consulta XQuery antes de transformá-la em uma consulta SQL para ser executada sobre a base relacional. As operações da álgebra são classificados em três categorias: operações XML, utilizadas para

representar operações sobre o documento XML, como operações de navegação sobre coleções; operações SQL, que correspondem às operações similares aos da álgebra relacional, como as operações de seleção, projeção, junção, união, agrupamento, ordenação, etc.; e operações especiais, utilizadas temporariamente nas etapas de otimização da consulta.

2.4.1 Tree Algebra for XML - TAX

TAX (JAGADISH et al., 2001) é uma álgebra para manipulação de dados em XML modelados como florestas de árvores rotuladas e ordenadas. A TAX foi criada a partir da álgebra relacional, procurando, para isso, um equivalente no XML para uma coleção de tuplas do modelo relacional, de forma a poder utilizar operações equivalentes às operações da álgebra relacional em uma álgebra para XML.

A TAX apresenta o conceito de padrão de árvore (*pattern tree*), que serve para identificar um subconjunto de nodos de interesse em uma árvore, possibilitando a definição de predicados em qualquer um dos nodos. Todas as operações da álgebra manipulam nodos e atributos identificados por meio de um padrão de árvore, o que possibilita a sua aplicação sobre coleções de árvore heterogêneas. Desta forma, a TAX implementa grande parte das operações da álgebra relacional, e algumas operações adicionais necessárias para a manipulação de estruturas em árvores. A aplicação de um padrão de árvore sobre uma árvore resulta em uma ou mais árvores testemunha (*witness trees*), que contém os nodos do padrão de árvore que satisfazem os seus predicados, mantendo a estrutura da árvore original.

Todas as operações da TAX recebem um conjunto de árvores como entrada e produzem um conjunto de árvores como saída. A TAX implementa as operações da álgebra relacional, como seleção, projeção, produto (junção), operações de agrupamento, ordenação e agregação, além de operações para inserção, atualização e exclusão de nodos de uma árvore. Em (JAGADISH et al., 2001), é apresentado um algoritmo para a transformação de uma consulta XQuery em uma expressão algébrica em TAX.

A TAX se popularizou e foi implementada no banco de dados XML nativo TIMBER (JAGADISH et al., 2002). A partir dela, surgiram derivações como a *Generalized Tree Pattern* - GTP (CHEN et al., 2003), que é uma extensão dos padrões de árvores de forma a representar totalmente uma XQuery em um único

padrão de árvore; e a álgebra *Tree Logical Class* - TLC (PAPARIZOS et al., 2004), que será analisada em mais detalhes a seguir.

2.4.2 Tree Logical Class - TLC

A álgebra *Tree Logical Class* - TLC - (PAPARIZOS et al., 2004), também implementada no banco de dados nativo XML TIMBER (JAGADISH et al., 2002), é uma evolução da TAX, pois permite o acesso a conjuntos heterogêneos de árvores através dos "padrões de árvore anotados" (*annotated pattern trees*, ou APTs), que estendem o conceito de padrão de árvore permitindo a sua utilização com conjuntos heterogêneos. Para facilitar a identificação dos nodos da árvore de saída em relação ao padrão de árvore para futuras operações, foi criado o conceito de Classes Lógicas (*logical classes*) que rotulam os nodos da árvore de saída de acordo com o padrão de árvore.

Além dos conceitos de padrão de árvore anotados e classes lógicas mencionados anteriormente, a TLC define ainda as seguintes operações algébricas: filtro, junção, seleção, projeção, eliminação de duplicatas, funções de agregação e construção, os quais serão explicados com mais detalhes na seção 3.2 do Capítulo 3.

Em (PAPARIZOS et al., 2004) encontramos um esboço de algoritmo para conversão de uma consulta XQuery em álgebra TLC. O autor também destaca a superioridade da TLC em relação a outras abordagens como a TAX (JAGADISH et al., 2001) e a GTP (CHEN et al., 2003) após uma série de testes experimentais.

2.5 Considerações quanto à execução de consultas em bases XML fragmentadas e distribuídas

Uma parte fundamental desta dissertação foi a análise de técnicas de fragmentação de bases XML e de álgebras para processamento de consultas sobre bases XML. Como vimos nas seções 2.3 e 2.4, existem diversas técnicas de fragmentação de bases XML e álgebras para processamento de consultas sobre estas bases. Em nosso trabalho, precisaremos utilizar uma técnica de fragmentação de bases XML e uma álgebra para o processamento das consultas XML distribuídas. É fundamental que a técnica de fragmentação seja compatível com a álgebra adotada, já que precisaremos construir as regras de reconstrução das bases a partir das definições dos fragmentos e, para isso, deverão ser utilizadas as mesmas operações da álgebra utilizada no processamento da consulta.

Diante das técnicas de fragmentação de bases XML expostas na seção 2.3, a única que apresenta formalmente as definições para fragmentação horizontal, vertical e híbrida válidas para bases de um único documento (SD) e coleções de documentos (MD), e que ainda utiliza uma álgebra para consultas sobre bases XML para a definição dos fragmentos e para a composição das regras de reconstrução das bases a partir dos fragmentos foi a técnica de (ANDRADE et al., 2006). Utilizando esta técnica em conjunto com a álgebra TLC (PAPARIZOS et al., 2004) para o processamento das consultas XQuery, é possível obter uma solução completa tanto para o processamento e otimização da consulta quanto para a etapa de localização dos dados, onde podemos utilizar as definições dos fragmentos e as expressões algébricas de reconstrução das visões XML. Desta forma, podemos compreender todas as etapas do processamento de uma consulta distribuída (Figura 1) com a mesma notação algébrica. No Capítulo 3 aprofundamos os aspectos teóricos destas técnicas, para dar ao leitor maior embasamento antes de passarmos para os detalhes do processamento das consultas distribuídas no Capítulo 4.

Capítulo 3 - Fundamentos Teóricos

3.1 Introdução

No Capítulo 2 foram apresentadas algumas técnicas de fragmentação e álgebras para consultas sobre bases XML. Verificamos que, para a elaboração da nossa proposta, que se preocupa com automação e correção, precisamos de uma definição de fragmentação em sintonia com a álgebra utilizada no processamento da consulta., Desta forma, as expressões algébricas que definem a reconstrução das visões globais podem ser inseridas dentro do plano algébrico de execução da consulta, através de regras de equivalência dessa álgebra. A técnica de fragmentação de bases XML apresentada em (ANDRADE et al., 2006) define formalmente as regras de reconstrução dos fragmentos a partir de operações da álgebra TLC (PAPARIZOS et al., 2004). A álgebra para consultas em XML e a definição adotada para fragmentação de bases XML são apresentadas em mais detalhes neste capítulo com o objetivo de fornecer ao leitor maior embasamento teórico antes de entrarmos nos detalhes da nossa metodologia de processamento de consultas distribuídas no Capítulo 4.

3.2 A Álgebra TLC

A álgebra TLC (PAPARIZOS et al., 2004) é a álgebra utilizada para a decomposição das consultas XQuery pelo Mediador. Para isso, precisamos compreender os elementos e operações dessa álgebra, a técnica de transformação de uma consulta XQuery para a sua representação algébrica TLC, e definir a gramática da XQuery que será utilizada pela nossa proposta.

Como apresentado na seção 2.4.2, a álgebra TLC utiliza "padrões de árvore anotados" (*annotated pattern trees*, ou APTs) para o acesso a conjuntos heterogêneos de árvores. Através das anotações de cardinalidade apresentadas na Tabela 1, cada ramo do padrão de árvore possui uma especificação de quantos filhos ele poderá ter na árvore de saída.

Tabela 1: Anotações de cardinalidade em uma APT

-	Apenas um elemento para o casamento da árvore de entrada
?	Zero ou apenas um elemento para o casamento da árvore de entrada
+	Um ou mais elementos para o casamento da árvore de entrada
*	Zero ou mais elementos para o casamento da árvore de entrada

A introdução das anotações de cardinalidade no padrão de árvore permite que sejam geradas árvores heterogêneas na saída. A Figura 2 exemplifica a aplicação de um APT em três árvores de entrada (rotuladas por B1, B2 e B3), gerando três árvores de saída. É importante observar que a árvore B2 não produziu árvores de saída, já que ela não possui nenhum elemento C filho de B, como especificado pelo APT. Já a árvore B3 gerou dois elementos de saída, pois possuía dois elementos D filhos de B. Como o APT especifica que na saída devem existir zero ou um elemento D filho de B (anotação "?"), foi possível criar duas árvores de saída para a mesma árvore de entrada.

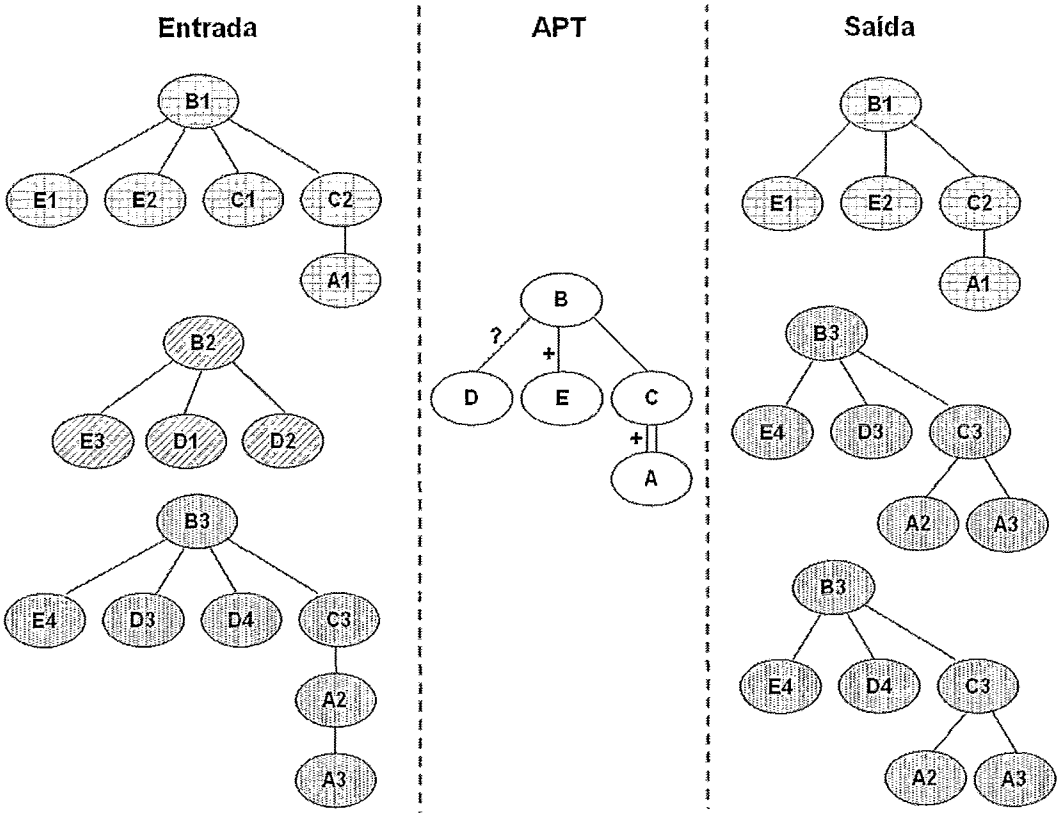


Figura 2: Exemplo da aplicação de um padrão de árvore anotado (APT) sobre um conjunto de árvores de entrada.

Classes Lógicas (*logical classes*) são utilizadas para rotular os nodos da árvore de saída de acordo com o padrão de árvore. Cada nodo do padrão de árvore será uma classe lógica, e cada nodo da árvore de saída será rotulado com a classe lógica do padrão de árvore. Mesmo podendo ter árvores de saída heterogêneas, as suas reduções em classes lógicas serão sempre homogêneas. Na Figura 2, os nodos "E1", "E2" e "E4" das árvores de saída são nodos da classe lógica "E" do APT. As classes lógicas servem para podermos mapear o nodo da árvore de saída com o nodo do APT. Para facilitar o acesso às classes lógicas, são associados rótulos para cada classe lógica do APT,, que são chamados *logical class labels* – LCLs. Os LCLs são utilizados no processamento da consulta como uma referência para os nodos dos APTs dentro das operações algébricas .

3.2.1 Operações Algébricas

Operações algébricas da TLC recebem como operando um ou mais conjuntos de árvores, produzindo como resultado um outro conjunto de árvores. Trata-se de modelo operacional fechado, assim como no relacional, onde um resultado pode ser usado como operando em outra operação e assim expressões podem ser compostas. No caso do modelo relacional, operandos são relações e operações são : seleção, projeção, junção, etc. Na TLC são definidas as seguintes operações algébricas: filtro, junção, seleção, projeção, funções de agregação e construção, as quais são explicados a seguir. Uma lista mais completa de operações da TLC é apresentada em (PAPARIZOS et al., 2004).

- Filtro $F[LCL_f, p, m](S)$: Aplicação de um predicado seletivo p sobre os nodos rotulados por LCL_f presentes no conjunto de árvores operando S, gerando como resultado o sub-conjunto de árvores que satisfazem o predicado. A operação m é utilizado para definir o modo de iteração sobre o conjunto de árvores operando S para a produção do resultado: (i) todas as árvores que satisfizerem p; (ii) existe pelo menos uma árvore de S que satisfaz p; ou (iii) apenas uma árvore de S satisfaz p.

- Junção $J[apt, p](S_l, S_r)$: Junção de dois conjuntos de árvores operando S_l e S_r de acordo com um predicado p que define o critério de junção das árvores operando. A estrutura da árvore de resultado é definida por um padrão de árvore anotado *apt.* .

- União $U[apt](S_l, S_r)$: União de dois conjuntos de árvores operando S_l e S_r cuja estrutura da árvore resultante é definida por um padrão de árvore anotado *apt.*

- Seleção *S[apt](S)*: Aplicação de um predicado, na forma de um padrão de árvore anotado *apt* sobre um conjunto de árvores operando *S*, tendo como resultado um conjunto de árvores que satisfizeram o *apt* para todas as árvores operando.

- Projeção *P[nl](S)*: Mantém no resultado apenas as classes lógicas *nl* definidas na entrada para um conjunto de árvores operando de *S*.

- Ordenação *O[ol](S)*: ordena um conjunto de árvores operando *S* de acordo com uma lista de nodos *ol* que contém a especificação da ordenação: crescente ou decrescente.

- Função de agregação *AF[nomeF, LCLa, novaLCL](S)*: Execução de uma função de agregação *nomeF* (*count, max, min, sum, avg*) em um determinado nodo rotulado por *LCLa* de um conjunto de árvores operando *S*, gerando uma nova classe lógica *novaLCL* onde o resultado da função será armazenado no resultado.

- Construção *C[c](S)*: Aplicação de uma árvore padrão de construção *c* sobre um conjunto de árvores operando *S*, gerando uma árvore que representará o resultado final da consulta.

As operações da TLC são utilizadas na representação algébrica de consultas XQuery sobre bases XML. Em nossa proposta, a TLC é utilizada para essa representação e subseqüente decomposição da consulta algébrica, expressa sobre a coleção global, em sub-consultas algébricas, expressas sobre os fragmentos desta coleção.

3.2.2 Re-escrita de XQuery em TLC

Em (PAPARIZOS et al., 2004) encontramos um esboço de algoritmo para a re-escrita de uma consulta XQuery em uma expressão da álgebra TLC. Este algoritmo foi utilizado como referência na dissertação, inclusive na implementação do protótipo do Mediador, como visto no Capítulo 5. A re-escrita de uma consulta XQuery em sua representação algébrica em TLC compreende a transformação de funções da XQuery em operações da TLC, como será exemplificado a seguir.

A consulta apresentada na Figura 3 pode ser dividida em três partes: o *for*, que será responsável pela iteração sobre as árvores da coleção; o *where*, que restringirá o resultado da consulta através de um predicado de seleção; e o *return*, para composição do resultado final. A primeira parte da consulta, o *for*, produz uma operação algébrica de seleção, com um APT inicialmente formado pela expressão

de caminho sobre a coleção consultada. A segunda parte, o *where*, adiciona um nó com o predicado restritivo ao APT da operação de seleção. Por fim, o *return* introduz dois novos operações à representação algébrica: uma nova operação de seleção, que será responsável apenas pela definição dos LCLs que serão apresentados no resultado final; e uma operação de construção, que utilizará estes LCLs para a montagem da árvore de saída da consulta.

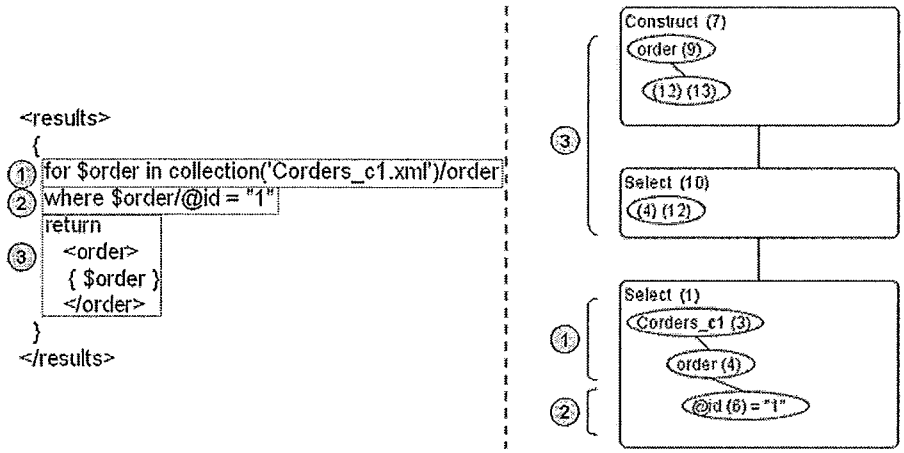


Figura 3: Exemplo 1 de re-escrita de uma consulta em sua representação algébrica.

A representação gráfica da estrutura algébrica da TLC, produzida pelo protótipo implementado para este trabalho, consiste em uma árvore de operações cuja ordem de processamento ocorre das folhas para a raiz (de baixo para cima). Para a consulta apresentada na Figura 3, temos três operações em seqüência: duas seleções e uma construção. Operações de seleção e de construção possuem APTs como parâmetros, conforme apresentado na seção 3.2.1. APTs são representados graficamente como árvores das classes lógicas, cada uma com o seu LCL associado (número entre parênteses dentro da elipse que representa a classe lógica). No caso da primeira operação de seleção, a classe lógica *order* possui LCL 4. Este LCL, por sua vez, é uma classe lógica no APT da segunda operação de seleção, que recebe um novo LCL (12) e que será utilizado no APT da operação de construção. Esta transformação do LCL 4 para o 12 na segunda operação de seleção ocorre devido ao algoritmo de transformação da XQuery para a TLC, mas é equivalente a um ponteiro para um ponteiro. O LCL 4 (ou o 12) é a representação da variável *$order* da consulta XQuery.

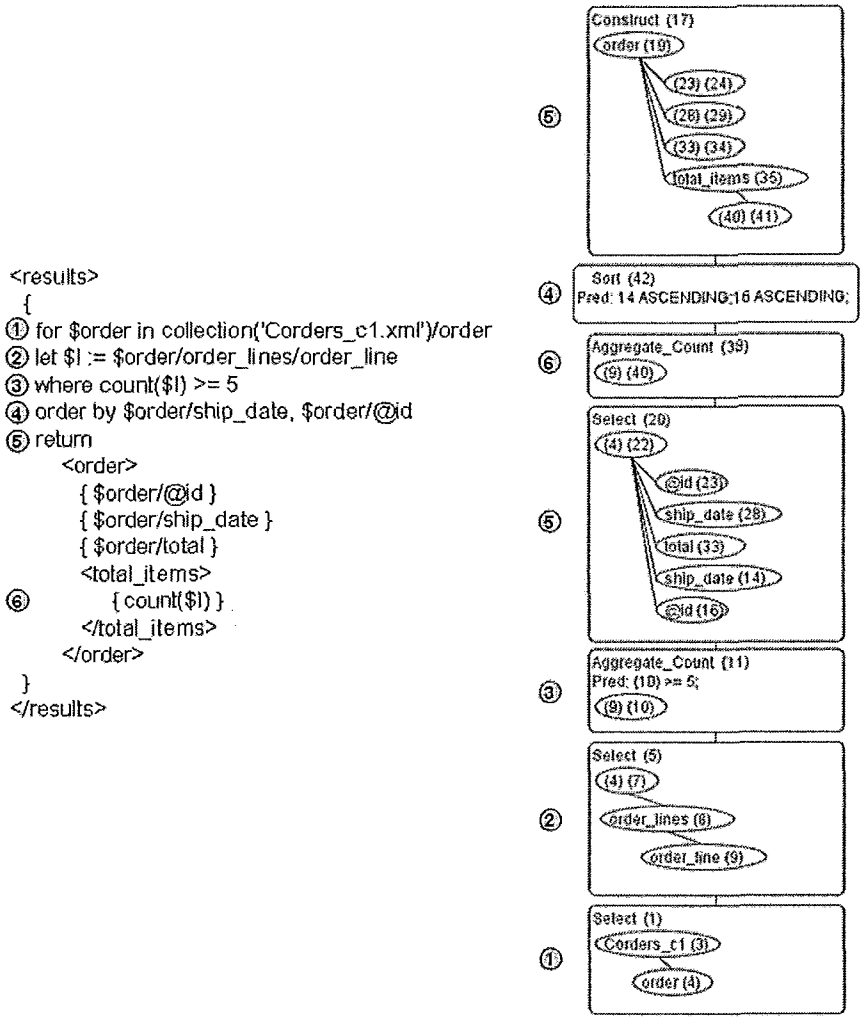


Figura 4: Exemplo 2 de re-escrita de uma consulta em sua representação algébrica.

Um exemplo com uma consulta mais complexa é apresentado na Figura 4. Nesta consulta, que utiliza todos os elementos do *FLWOR* da XQuery, a montagem da sua representação algébrica segue os mesmos passos do exemplo anterior, com exceção da função de agregação *count* da cláusula *where*, que produz uma nova operação de função de agregação no plano algébrico. Outra nova operação inserida neste exemplo é a operação de ordenação, produzida pela cláusula *order by* da consulta.

3.2.3 Gramática da XQuery Considerada

Devido à grande quantidade de recursos e funções da XQuery e para que possamos nos concentrar em aspectos específicos da metodologia de processamento de consultas distribuídas em bases XML, iremos considerar um sub-conjunto da gramática da XQuery original (BOAG et al., 2007) que contém as

principais operações desta linguagem. Este sub-conjunto, apresentado na Figura 5, é rico o suficiente para explorar a metodologia.

DirElemConstructor	::=	"<" QName DirAttributeList ("/>" | (">" DirElemContent* "</" QName S? ">"))
DirAttributeList	::=	(S (QName S? "=" S? DirAttributeValue)?)*
DirAttributeValue	::=	("" (EscapeQuot | QuotAttrValueContent)* "")
QuotAttrValueContent	::=	QuotAttrContentChar
DirElemContent	::=	DirElemConstructor | EnclosedExpr | ElementContentChar
EnclosedExpr	::=	"{" Expr "}"
Expr	::=	ExprSingle ("," ExprSingle)*
ExprSingle	::=	FLWORExpr | AndExpr
FLWORExpr	::=	(ForClause | LetClause)+ WhereClause? OrderByClause? "return" ExprSingle
ForClause	::=	"for" "$" VarName "in" ExprSingle
LetClause	::=	"let" "$" VarName ":=" ExprSingle
WhereClause	::=	"where" ExprSingle
OrderByClause	::=	"order" "by" OrderSpecList
OrderSpecList	::=	OrderSpec ("," OrderSpec)*
OrderSpec	::=	ExprSingle OrderModifier
OrderModifier	::=	("ascending" | "descending")?
AndExpr	::=	ComparisonExpr ("and" ComparisonExpr)*
ComparisonExpr	::=	ValueExpr (GeneralComp ValueExpr)?
ValueExpr	::=	PathExpr | ExtensionExpr
GeneralComp	::=	"=" | "!=" | "<" | "<=" | ">" | ">="
ExtensionExpr	::=	"{" Expr? "}"
PathExpr	::=	("/" RelativePathExpr?) | ("//" RelativePathExpr) | RelativePathExpr
RelativePathExpr	::=	PrimaryExpr (("/" | "//") PrimaryExpr)*
PrimaryExpr	::=	Literal | VarRef | FunctionCall | DirElemConstructor
Literal	::=	NumericLiteral | StringLiteral
NumericLiteral	::=	IntegerLiteral | DecimalLiteral | DoubleLiteral
VarRef	::=	"$" QName
FunctionCall	::=	QName "(" (ExprSingle ("," ExprSingle)*)? ")"

Figura 5: Gramática da XQuery considerada

3.3 Fragmentação de bases XML

A definição de fragmentos sobre bases XML proposta em (ANDRADE et al., 2005; ANDRADE et al., 2006) contempla três tipos de fragmentos: horizontal, vertical e híbrido. Estes fragmentos são definidos a partir de operações da álgebra TLC sobre coleções de árvores XML, que também serão utilizadas nas expressões de reconstrução das coleções globais. A definição desses três tipos de fragmentação se assemelha aos tipos definidos para o modelo relacional em (ÖZSU et al., 1999), pelo fato de serem baseadas operações algébricas sobre operandos que correspondem a conjuntos, i.e., no relacional, tuplas e no XML árvores.

Fragmentos podem ser definidos sobre uma coleção homogênea de documentos (MD) ou sobre uma coleção de um único documento (SD). Um fragmento *F* é definido em (ANDRADE et al., 2006) como: $F := \langle C, \gamma \rangle$, onde *C* representa uma coleção MD ou SD e γ uma operação definida sobre *C*. Instâncias de um fragmento *F* são criadas aplicando-se a operação γ sobre a coleção *C*. Para simplificar a notação das definições dos fragmentos, Andrade definiu operações de seleção e projeção que utilizam uma lista de predicados no lugar dos APTs das operações da TLC. Estas operações podem ser convertidas em operações de seleção e projeção da TLC através da transformação desta lista de predicados em um APT, o que pode ser feito por inferência direta. Maiores detalhes sobre a transformação das definições das fragmentações em operações da TLC podem ser obtidas em (ANDRADE et al., 2005; ANDRADE, 2006). Apresentamos a seguir as definições para cada um dos tipos de fragmentação.

3.3.1 Fragmentação Horizontal

A fragmentação horizontal é utilizada para agrupar dados que são freqüentemente acessados por consultas com um determinado predicado de seleção. Um fragmento horizontal *F* de uma coleção *C* é definido por uma operação de seleção (σ) aplicada sobre a coleção operando *C*, como exemplificado a seguir.

$$COrders\_c2\_fh1 := \langle C_{orders}, \sigma_{/order/total \leq 4000} \rangle$$

Neste exemplo, é definido um fragmento *COrders_c2_fh1* sobre uma coleção *Corders* que conterá apenas os documentos de *Corders* que satisfazem a restrição de */order/total* $\leq$ *4000.*

Bases SD não podem ser fragmentadas horizontalmente, já que a fragmentação horizontal é definida sobre uma coleção de árvores (documentos), e uma base SD possui apenas uma árvore. A fragmentação horizontal de uma base SD pode ser obtida através de uma fragmentação híbrida, como será visto mais adiante.

3.3.2 Fragmentação Vertical

A fragmentação vertical tem como objetivo dividir a estrutura de dados da árvore de forma a melhorar o desempenho de consultas que acessem somente um sub-conjunto dos elementos desta coleção. Um fragmento vertical *F* de uma coleção *C* é definido por uma operação de projeção ($\Pi_{P\Gamma}$) aplicado sobre *C*, onde *P* é a expressão de caminho do elemento que será projetado e Γ uma lista de expressões de caminho que serão podadas de *P*, como mostrado no exemplo:

$$CLoja\_c2\_fv1 := \langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$$

O exemplo acima define um fragmento vertical *CLoja_c2_fv1* sobre uma coleção *Cloja* projetando o atributo */Loja* sem a subárvore */Loja/Itens*.

É importante ressaltar que a expressão de caminho que define o nodo projetado não pode apontar para um elemento com cardinalidade maior do que um. Esta restrição faz com que os resultados da fragmentação sejam documentos bem formados, sem a necessidade de se criar elementos artificiais para servir de raiz para os fragmentos.

3.3.3 Fragmentação Híbrida

A fragmentação híbrida é obtida a partir da definição da fragmentação horizontal seguida da fragmentação vertical, ou vice-versa, como exemplificado abaixo.

$$CLoja\_c3\_fy6 := \langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="CD"} \rangle$$

Neste exemplo, é definido um fragmento híbrido *CLoja_c3_fy6* sobre a base *Cloja*, onde é feita a projeção do elemento */Loja/Itens* sem poda de elementos filhos, e, posteriormente, a seleção dos elementos que satisfaçam ao predicado */Item/Seção="CD"*.

Um uso interessante da fragmentação híbrida é para produzir a fragmentação horizontal de bases SD, através de uma fragmentação vertical

seguida de uma fragmentação horizontal sobre a coleção gerada pelo fragmento vertical.

3.3.4 Regras de Correção

Para garantirmos que a fragmentação de uma base está correta, devemos verificar se os fragmentos atendem às seguintes regras de correção (ANDRADE et al., 2005):

- **Completude**: cada item da coleção *C* deve aparecer em pelo menos um fragmento. Na fragmentação horizontal, este item corresponde a uma árvore, enquanto que na fragmentação vertical corresponde a um nó.

- **Disjunção**: se um item da coleção *C* pertence ao fragmento *Fi*, ele não poderá pertencer a mais nenhum outro fragmento da mesma coleção.

- **Reconstrução**; deve ser possível definir uma operação capaz de reconstruir a coleção original *C* a partir de seus fragmentos *Fi*. Para fragmentos horizontais, esta operação será a operação de união, enquanto que para fragmentos verticais será a operação de junção.

As regras de correção permitem que uma consulta sobre a base global seja reescrita automaticamente sobre a base fragmentada, garantindo que a consulta "fragmentada" equivale à consulta "global". Dentre as regras de correção, a mais importante para o nosso trabalho é a regra de reconstrução, pois ela define a expressão algébrica de reconstrução da coleção global. Essa expressão é usada para substituir a coleção global, que aparece na consulta algébrica XQuery, pelos seus fragmentos correspondentes. Dessa forma, na etapa de localização no processamento da consulta global, fragmentos desnecessários podem ser eliminados do processamento, como visto no Capítulo 4.

Capítulo 4 - Metodologia de Processamento Distribuído de Consultas XQuery

4.1 Introdução

Este capítulo apresenta a nossa metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas, que envolve a decomposição de uma consulta principal em sub-consultas que serão executadas em *sites* remotos contendo os fragmentos de uma coleção global. Sobre a consulta principal, são executadas as etapas de decomposição, localização, otimização global, a criação as sub-consultas e a sua execução nas bases XML remotas.

Essa metodologia pode ser aplicada tanto em um banco de dados que permita a fragmentação de bases XML, quanto em um sistema que proporcione uma visão integrada de bancos de dados XML semi-autônomos homogêneos (pois a metodologia não contempla esquemas heterogêneos). Nesses dois casos, um Catálogo irá armazenar as informações sobre a base distribuída, contendo os esquemas das coleções globais e de seus fragmentos, as definições dos fragmentos, informações sobre a alocação dos fragmentos e estatísticas dos fragmentos e nodos remotos.

As etapas do processamento de uma consulta XQuery distribuída são apresentadas na seção 4.2. A etapa de localização da consulta original, que envolve também a eliminação dos fragmentos inúteis para o resultado da consulta é apresentada em mais detalhes na seção 4.2.2, enquanto que a seção 4.3 apresenta em mais detalhes o processo de criação e execução das sub-consultas e a composição do resultado final.

4.1.1 Exemplos Utilizados neste Capítulo

Neste capítulo, para exemplificar o processamento das consultas distribuídas, iremos utilizar duas bases de dados: uma coleção de documentos de pedidos de compra *COrders*, do *benchmark* XBENCH (YAO et al., 2002); e um documento com dados de lojas *CLoja* (ANDRADE et al., 2006). As estruturas das bases são apresentadas na Figura 6 e na Figura 7.

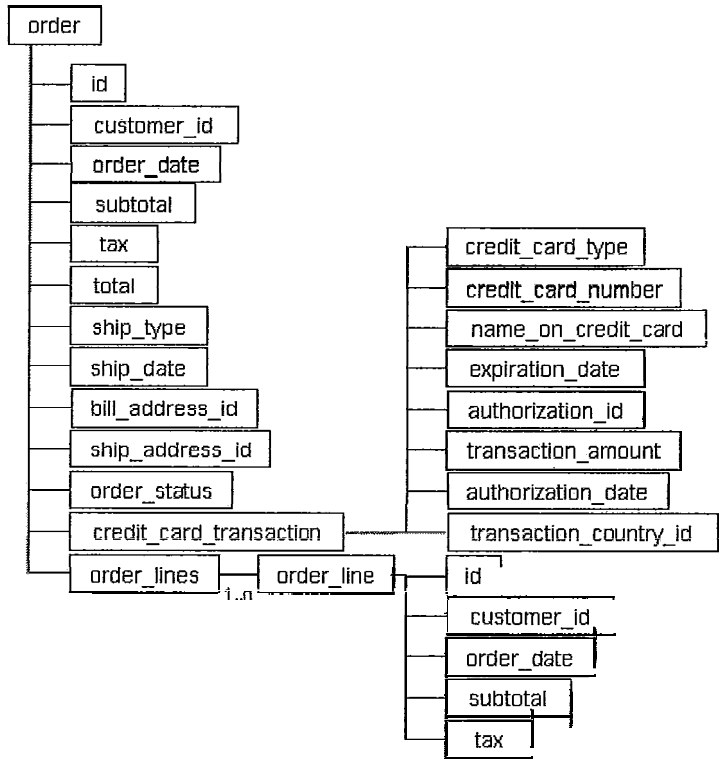


Figura 6: Estrutura da coleção de documentos de pedidos de compra *COrders*.

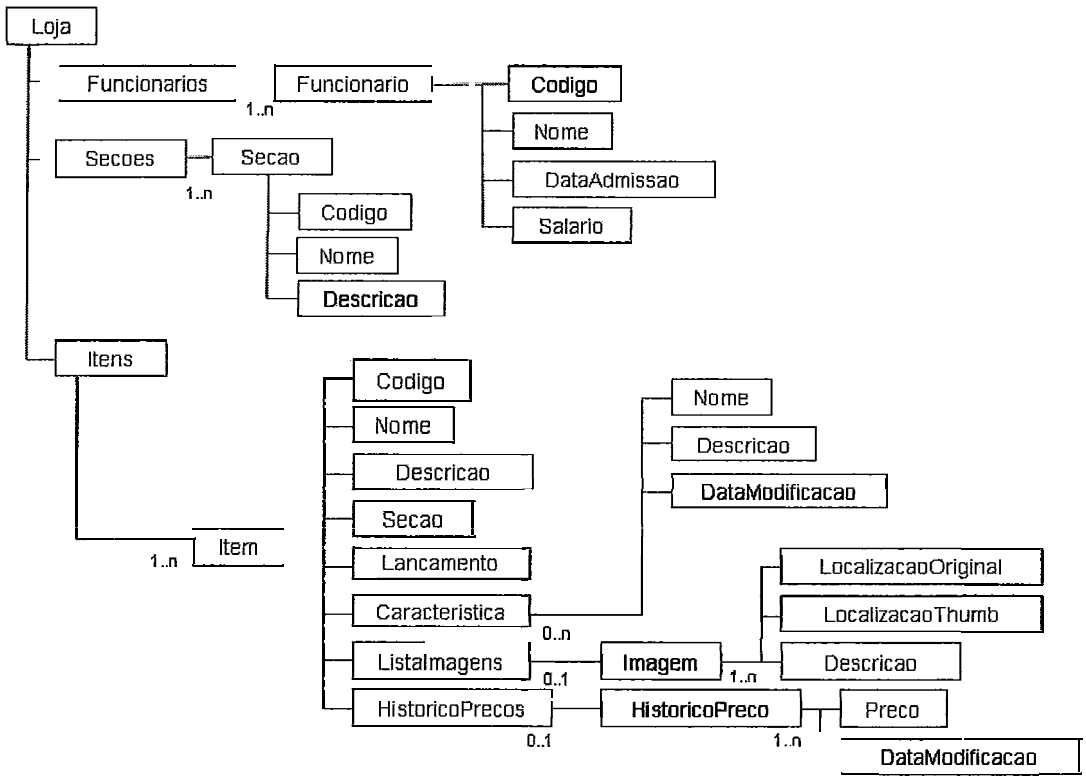


Figura 7: Estrutura do documento de dados de lojas *CLoja*.

A base *Corders*, de múltiplos documentos (MD), será fragmentada horizontalmente em três fragmentos definidos sobre faixas de valores do total do

pedido de compra (atributo *total* do documento). A base *CLoja*, de um único documento (SD), será fragmentada verticalmente para separar o elemento Itens dos demais elementos filhos de Loja (Figura 8). Posteriormente, aplicaremos uma fragmentação horizontal sobre este fragmento de Itens para separá-los por seção, de forma a representar uma fragmentação híbrida (horizontal sobre vertical). Estes cenários são apresentados na Tabela 2.

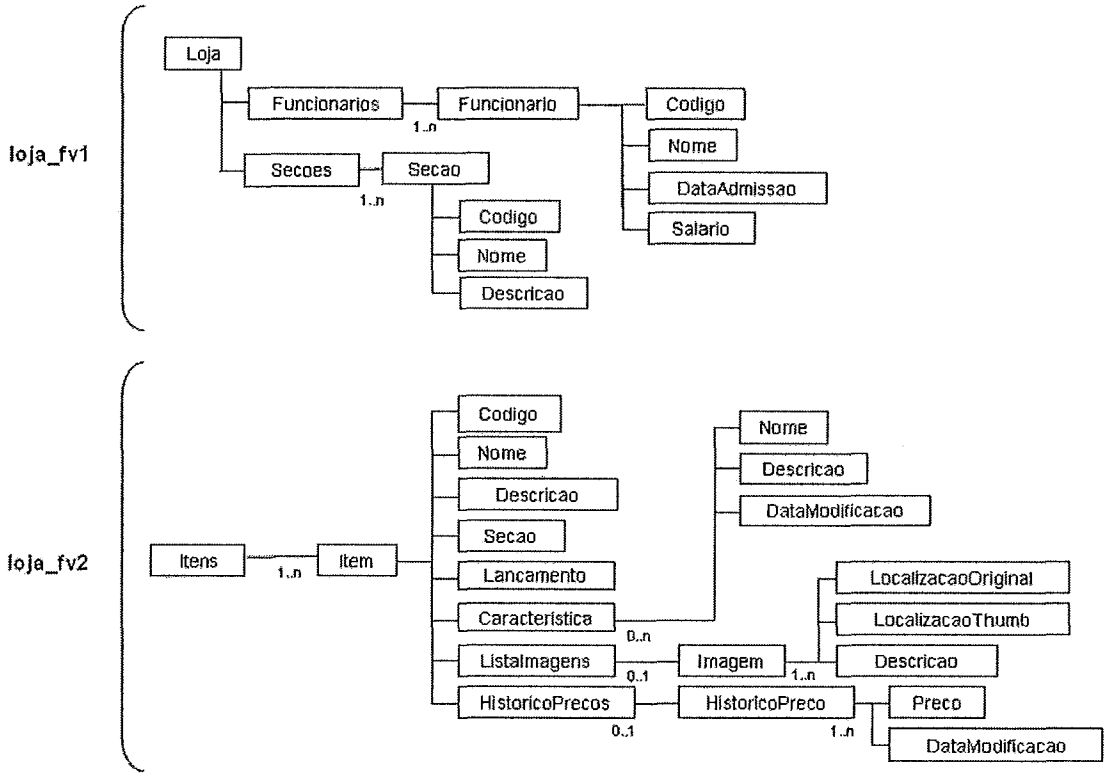


Figura 8: Fragmentos verticais de CLoja

Sobre as coleções *COrders* e *CLoja*, fragmentadas como descrito na Tabela 2, serão aplicadas as consultas XQuery apresentadas no Anexo I.

Tabela 2: Bases e fragmentos considerados nos exemplos deste capítulo

Bases	Fragmentos	Tipo
COrders	COrders_c2_fh1 := $\langle C_{orders}, \sigma_{/order/total \leq 4000} \rangle$	MD
	COrders_c2_fh2 := $\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total < 8000} \rangle$	MD
	COrders_c2_fh3 := $\langle C_{orders}, \sigma_{/order/total \geq 8000} \rangle$	MD
CLoja	CLoja_c2_fv1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$	SD
	CLoja_c2_fv2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \rangle$	SD
CLoja	CLoja_c3_fy1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$	SD
	CLoja_c3_fy2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Brinquedos"} \rangle$	SD
	CLoja_c3_fy3 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Games"} \rangle$	SD
	CLoja_c3_fy4 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Perfumaria"} \rangle$	SD
	CLoja_c3_fy5 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Eletronicos"} \rangle$	SD
	CLoja_c3_fy6 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="CD"} \rangle$	SD
	CLoja_c3_fy7 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="DVD"} \rangle$	SD
	CLoja_c3_fy8 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Livraria"} \rangle$	SD
	CLoja_c3_fy9 := $\left\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{\substack{/Item/Secao \neq "Brinquedos" \\ \wedge /Item/Secao \neq "Games" \\ \wedge /Item/Secao \neq "Perfumaria" \\ \wedge /Item/Secao \neq "Eletronicos" \\ \wedge /Item/Secao \neq "CD" \\ \wedge /Item/Secao \neq "DVD" \\ \wedge /Item/Secao \neq "Livraria"}} \right\rangle$	SD

4.2 Metodologia de Processamento de Consultas XQuery Distribuídas

A metodologia para o processamento de consultas XQuery distribuídas foi definida através de uma adaptação das quatro etapas básicas do modelo

relacional, apresentadas na seção 2.2: decomposição da consulta, localização dos dados, otimização global e otimização local, conforme ilustrado na Figura 1. Cada uma destas etapas é descrita a seguir considerando o contexto de consultas XQuery distribuídas.

4.2.1 Decomposição da Consulta

A primeira etapa no processamento distribuído de uma consulta XQuery é a sua decomposição em uma expressão algébrica sobre coleções globais. Nesta etapa utilizamos o catálogo de coleções globais para validar as coleções expressas na consulta. No entanto, as informações sobre a distribuição dos dados ainda não são utilizadas. Esta etapa é muito semelhante ao processamento de consultas XQuery em ambientes centralizados e possui as fases de validação sintática e semântica da consulta; simplificação da consulta; e representação da consulta em uma forma algébrica, como será visto a seguir.

4.2.1.1 Validação Sintática e Semântica

A validação sintática consiste na verificação de que a consulta original, expressa em XQuery, está de acordo com a gramática suportada (ver Seção 3.2.3). Palavras-chave da linguagem escritas ou posicionadas de forma errada na consulta são exemplos de erros de sintaxe.

Já a validação semântica consiste na verificação de que as visões globais utilizadas pela consulta estão de acordo com as especificações presentes no catálogo do mediador, que armazena não só as definições dos fragmentos como também os esquemas XML das visões globais. Nomes de visões globais errados ou inexistentes no catálogo, ou atributos e elementos inexistentes no esquema de uma visão global são exemplos de erros semânticos.

Consultas invalidadas sintática ou semanticamente são descartadas pelo mediador, já que o seu processamento é impossível. Neste caso, o usuário receberá uma mensagem de erro informando o problema detectado, para que ele possa corrigir a consulta submetida.

4.2.1.2 Simplificação da Consulta

Estando a consulta validada sintática e semanticamente, podemos passar para a atividade de simplificação da consulta, que consiste na verificação dos predicados de seleção e na eliminação de predicados redundantes. Para isso, é utilizado um conjunto de regras de idempotência, de forma semelhante ao que é

realizado em consultas sobre o modelo relacional (ÖZSU et al., 1999). Estas regras de idempotência têm como objetivo a simplificação da lógica booleana dos predicados de seleção da consulta (cláusula *where* da XQuery), como nos exemplos mostrados na Tabela 3.

Tabela 3: Regras de idempotência para simplificação de predicados de seleção.

Regras de Idempotência
$p \wedge p \Leftrightarrow p$
$p \vee p \Leftrightarrow p$
$p \wedge true \Leftrightarrow p$
$p \vee false \Leftrightarrow p$
$p \wedge false \Leftrightarrow false$
$p \vee true \Leftrightarrow true$
$p \wedge \neg p \Leftrightarrow false$
$p \vee \neg p \Leftrightarrow true$
$p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
$p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

4.2.1.3 Representação em Forma Algébrica

O produto final da etapa de decomposição da consulta é a sua representação algébrica sobre as coleções globais. A representação de uma consulta em sua forma algébrica consiste na execução de um algoritmo, discutido na seção 3.2.2, que analisa sintaticamente a consulta e monta as operações algébricas, os padrões de árvores e as classes lógicas da TLC. A Figura 9 apresenta a representação algébrica da consulta *COrders_bts_c10.xq* (vide Anexo I) sobre a coleção COrders.

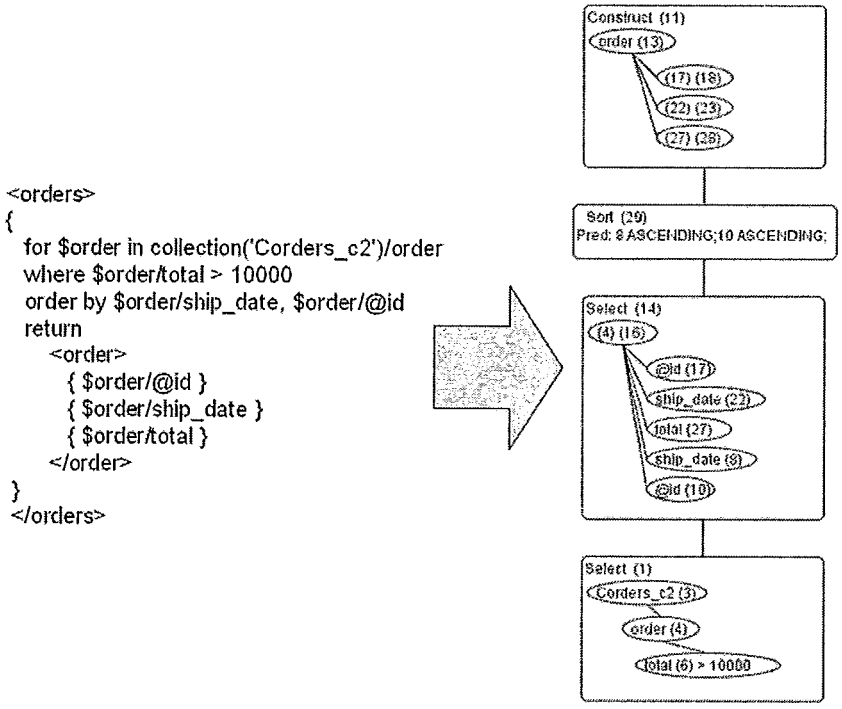


Figura 9: Representação da consulta *COrders_bts_c10.xq* em sua forma algébrica TLC

Note que esta consulta é realizada sobre as coleções globais. Ela não possui nenhuma referência a fragmentos, pois tais detalhes devem permanecer transparentes para o usuário. Esta representação algébrica inicial sobre as coleções globais é pré-requisito para a próxima etapa do processamento da consulta distribuída: a localização dos dados.

4.2.2 Localização dos Dados

A etapa de localização dos dados tem duas sub-etapas principais: substituir as referências a coleções globais do plano algébrico por referências a fragmentos locais, e aplicar a redução destes fragmentos de acordo com os predicados de seleção e projeção da consulta original, de maneira semelhante ao proposto em (ÖZSU et al., 1999). Esta etapa é responsável pelo maior benefício de uma fragmentação de uma base de dados, já que é através da eliminação dos fragmentos irrelevantes que uma consulta poderá ter o seu desempenho melhorado devido à diminuição do volume de dados consultados.

Para realizar a substituição de uma coleção global na representação algébrica da consulta, é necessário conhecer os fragmentos e seus predicados de formação: seleção, no caso de fragmento horizontal; projeção, no caso de fragmento vertical; ou ambos, no caso de fragmento híbrido. Estas informações estão armazenadas no catálogo do banco de dados distribuído.

A operação de localização de uma consulta sobre uma coleção global corresponde à substituição de todas as referências à coleção global pelos seus fragmentos. Se os fragmentos obedecerem às regras de correção de fragmentos XML (ANDRADE et al., 2006), poderemos utilizar a propriedade de reconstrução para garantir que existirá uma operação da própria TLC capaz de reconstruir a coleção global a partir de seus fragmentos. Esta operação será definida de acordo com o tipo de fragmentação, sendo união para fragmentos horizontais e junção para fragmentos verticais. A seguir apresentamos as regras para localização dos dados para os diferentes tipos de fragmentação.

4.2.2.1 Fragmentação Horizontal

Fragmentação "puramente" horizontal implica que todos os fragmentos que compõem a coleção global possuem apenas predicados de seleção. Desta forma, a reconstrução poderá ser feita a partir da operação união da TLC destes fragmentos (ANDRADE et al., 2006).

Definição 1: Seja *CG* uma coleção global, e FH_1, FH_2, ..., FH_n seus fragmentos horizontais. Segundo (ANDRADE et al., 2006), $CG = FH_1 \cup FH_2 \cup ... \cup FH_n$.

A Figura 10 apresenta o resultado da localização da operação de seleção da consulta *COrders_bts_c10.xq* sobre a coleção global *COrders* exibida na Figura 9. Esta coleção *COrders* é formada por três fragmentos horizontais, conforme a Tabela 2 (e reapresentados a seguir por questões de legibilidade), e pode ser reconstruída através da operação de união destes três fragmentos. Como a união é uma operação binária da TLC, são necessárias duas operações de união neste exemplo, para que os três fragmentos sejam unidos. A ordem da aplicação das uniões não influi no resultado final.

COrders_c2_fh1 := $\langle C_{orders}, \sigma_{/order/total \leq 4000} \rangle$
COrders_c2_fh2 := $\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total < 8000} \rangle$
COrders_c2_fh3 := $\langle C_{orders}, \sigma_{/order/total \geq 8000} \rangle$

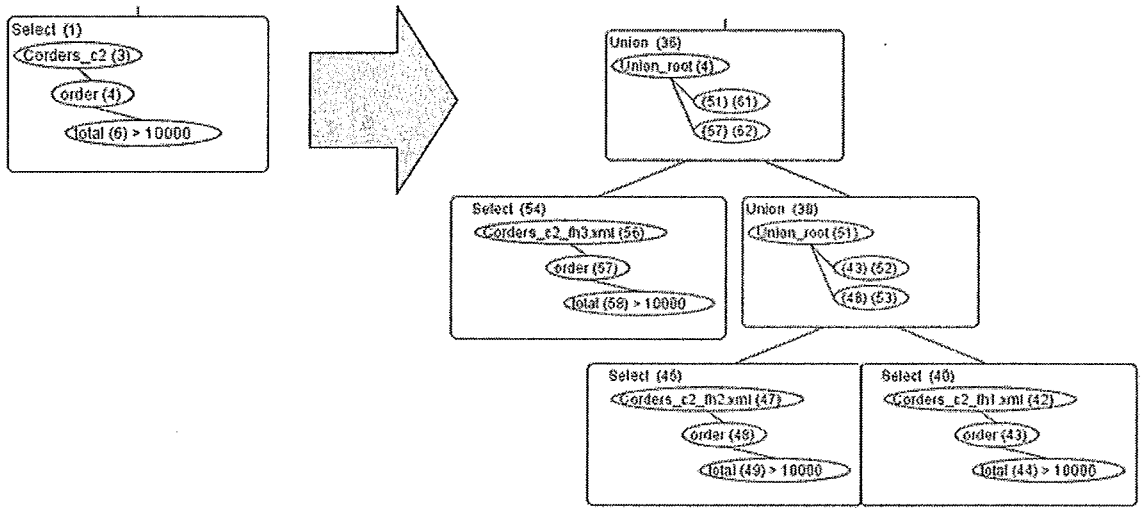


Figura 10: Localização de uma coleção global formada por fragmentos horizontais

Observe na Figura 10 que a operação de seleção sobre a coleção global possui em seu APT a classe lógica *order* com LCL igual a 4. Esta classe lógica representa a variável *$order* da consulta XQuery original (Figura 9), e será utilizada por outras operações algébricas da consulta. Quando substituímos esta operação de seleção sobre a coleção global pela operação de união dos seus fragmentos, teremos duas alternativas para não perder a referência à variável *$order* na consulta algébrica: ou atualizamos as referências ao LCL 4 da variável *$order* para o novo LCL do APT da operação de união, ou substituímos o valor do LCL do APT da operação de união pelo 4. Optamos pela segunda alternativa por ser mais simples, necessitando de apenas uma atualização de LCL.

Após a substituição da coleção global pela sua expressão de reconstrução através de seus fragmentos, passamos para o processo de redução dos fragmentos irrelevantes para o resultado final da consulta. No caso de fragmentos horizontais, que possuem apenas predicados de seleção, a redução consiste na análise da operação de seleção aplicada sobre o fragmento para identificar se os seus predicados não contradizem os predicados de definição do próprio fragmento. Predicados incompatíveis geram árvores vazias, e, portanto, podem ser eliminadas da consulta. Formalmente, a eliminação de fragmentos pode ser definida pela seguinte regra:

Definição 2: Seja *CG* uma coleção global, e FH_1, FH_2, ..., FH_n seus fragmentos horizontais. Seja p_i o predicado de seleção que define o fragmento FH_i. Seja *pq* o predicado de seleção utilizado numa consulta *q* sobre a coleção global *CG*. Seja *A* a árvore algébrica que representa a consulta *q* sobre *CG*. Um fragmento FH_i pode ser eliminado de *A* se $\sigma_{pq}(FH_i) = \varnothing$. Para que esta seleção seja vazia, tem-se a

seguinte regra: σ_{pq} (FH_j) = ∅ se $\forall s$ em *CG*: $\neg(pq(s) \wedge (p_i(s))$, onde *p(s)* denota que a sub-árvore *s* satisfaz o predicado *p*.

A Figura 11 exemplifica a redução de dois fragmentos horizontais na consulta *COrders_bts_c10.xq* para a coleção *COrders*. Neste exemplo, como pode ser visto também na Tabela 4, os fragmentos 1 e 2 não atendem ao critério de seleção da consulta, já que a consulta original seleciona *orders* com valor total maior do que 10000 e, de acordo com a Tabela 2 que contém a definição dos predicados da coleção *COrders*, estes fragmentos possuem *orders* com valor total menor ou igual a 4000 e valor total entre 4000 e 8000. Apenas o fragmento 3 pode possuir algum resultado relevante para a consulta, pois ele contém *orders* com valor total maior ou igual a 8000, o que é compatível com o critério de seleção da consulta original.

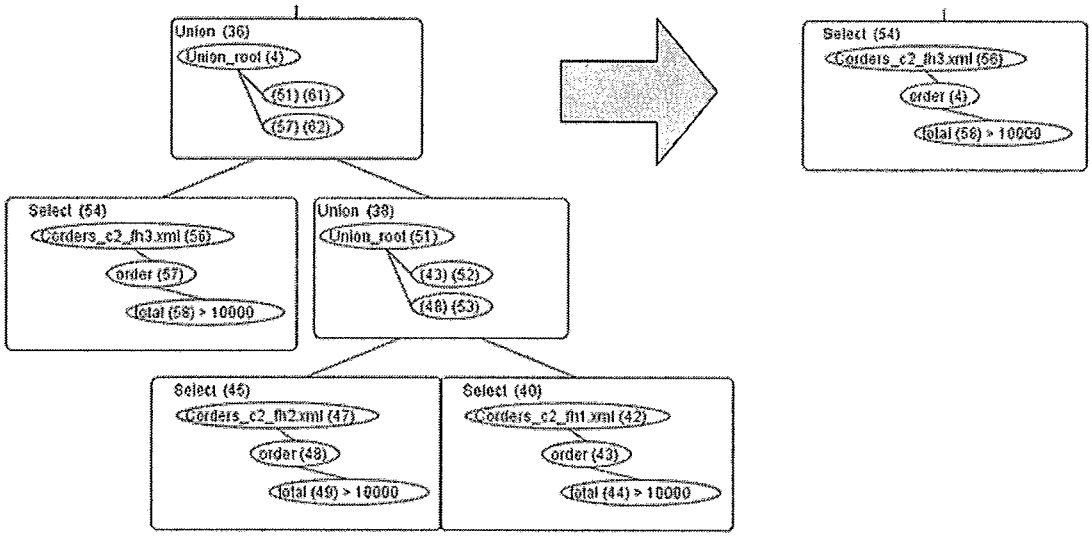
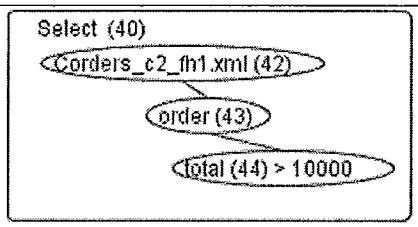
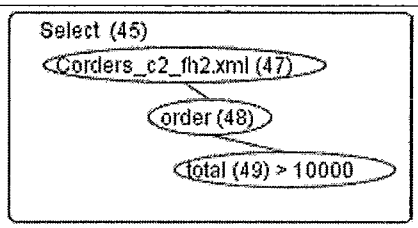
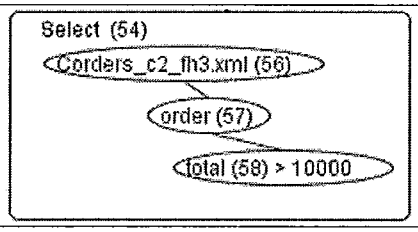


Figura 11: Redução de fragmentos horizontais

Tabela 4: Redução dos fragmentos horizontais na consulta *COrders_bts_c10.xq*

Operador de Seleção	Predicado de Seleção da Consulta	Predicado do Fragmento	Compatível?
	/order/total > 10000	/order/total <= 4000	Não
	/order/total > 10000	/order/total > 4000 ^ /order/total < 8000	Não
	/order/total > 10000	/order/total >= 8000	Sim

A remoção de uma operação de seleção sobre um fragmento do plano algébrico, que será filho de uma operação de união no caso da fragmentação puramente horizontal, implica também na remoção da operação de união pai, que será substituída pela sua operação irmã. Formalmente, se VG = A ∪ (B ∪ C), e o fragmento C puder ser eliminado, a expressão resultante é A ∪ B, como pode ser observado na Figura 12. Este procedimento é aplicado em todas as operações removidas, produzindo o plano algébrico reduzido. Da mesma forma como na localização de uma coleção global, onde a operação sobre a coleção global é substituída pela operação de união de seus fragmentos horizontais, uma especial atenção deve ser dada aos LCLs dos APTs das operações reduzidas para que não haja uma quebra de referências entre as operações algébricas.

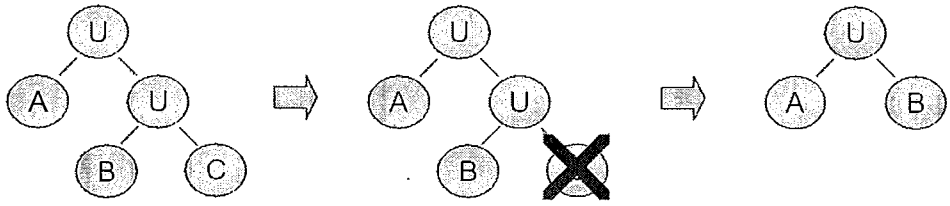


Figura 12: Redução de uma operação de uma árvore algébrica.

4.2.2.2 Fragmentação Vertical

Na fragmentação "puramente" vertical, de forma semelhante à fragmentação horizontal, os fragmentos possuirão apenas predicados de projeção. Neste caso, a reconstrução da coleção global poderá ser feita através da junção dos fragmentos (ANDRADE et al., 2006).

Definição 3: Seja *CG* uma coleção global, e FV_1, FV_2, ..., FV_n seus fragmentos verticais. Segundo (ANDRADE et al., 2006), $CG = FV_1 \bowtie FV_2 \bowtie \ldots \bowtie FV_n$.

A Figura 13 apresenta o resultado da representação algébrica e a localização da coleção global *CLoja* fragmentada verticalmente na consulta *CLoja_bps_c11.xq*. Observe neste caso que a operação utilizada para a reconstrução da coleção global pelos fragmentos foi a operação de Junção da TLC. Os fragmentos de *CLoja*, mostrados na Tabela 2, são reapresentados abaixo:

CLoja_c2_fv1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$
CLoja_c2_fv2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \rangle$

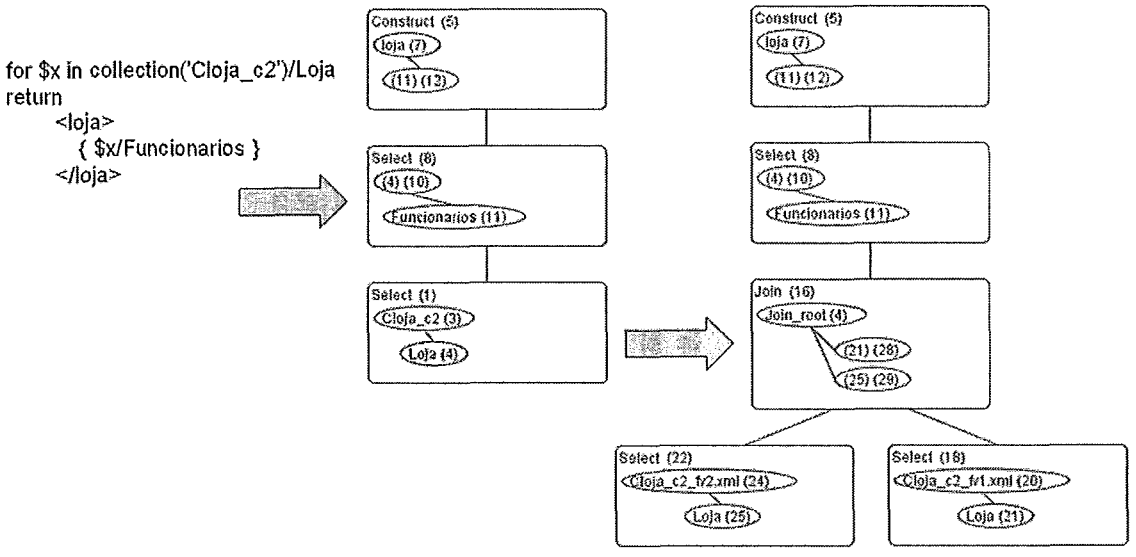


Figura 13: Localização de uma coleção global formada por fragmentos verticais

A fragmentação vertical exige um cuidado adicional nesta etapa de localização. Como os fragmentos verticais não possuem todos os elementos da coleção global, pois estes são distribuídos pelos fragmentos, será necessária a realização de uma poda nos APTs das operações de seleção aplicadas sobre os

fragmentos para eliminar os elementos que não pertencem ao fragmento vertical. Cada operação de seleção será analisada para verificação se os elementos do APT da operação fazem parte do fragmento vertical, podando os elementos do padrão de árvore quando ele não fizer parte. Este procedimento não é necessário na fragmentação horizontal devido à homogeneidade dos esquemas dos fragmentos horizontais. Esta poda é exemplificada na Figura 14, onde o predicado de seleção da consulta só é válido para um dos fragmentos verticais da base CLoja.

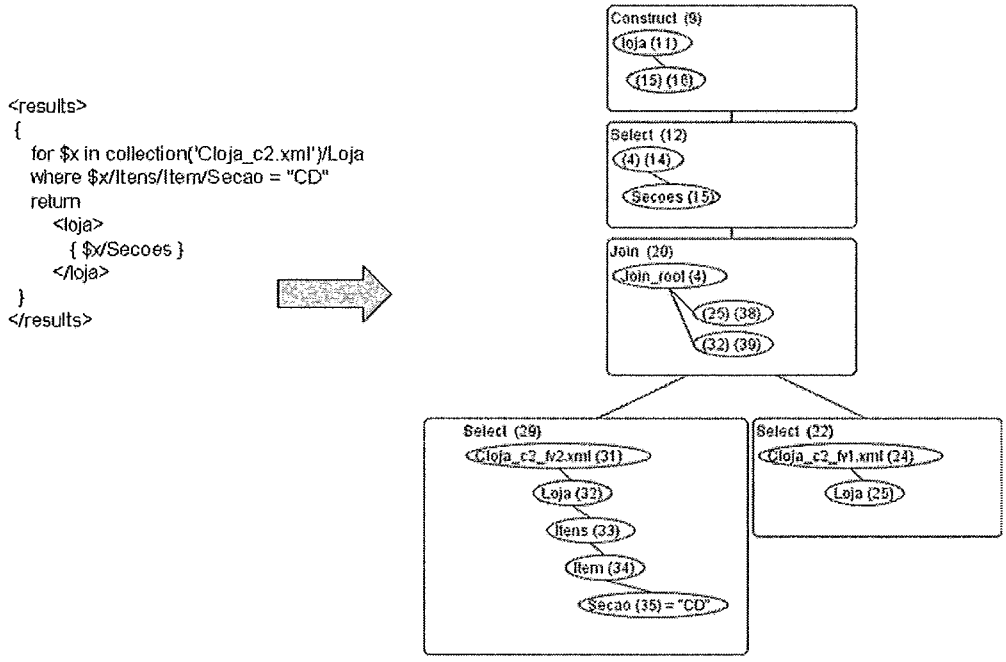


Figura 14: Poda de predicados de seleção em operações sobre fragmentos verticais.

De forma semelhante ao que ocorre na fragmentação puramente horizontal, após a reconstrução da coleção global pelas junções dos fragmentos verticais poderemos reduzir os fragmentos irrelevantes para o resultado final da consulta. Neste caso de fragmentação vertical, este processo consiste na análise das sub-árvores utilizadas pelas operações no fluxo de processamento do plano algébrico da consulta. Todo o plano algébrico deverá ser verificado nesta etapa, e não apenas a operação de seleção sobre o fragmento, como ocorre na fragmentação horizontal. Nesta análise, deverão ser verificadas quantas sub-árvores contidas no fragmento são utilizados pela consulta, descontando as sub-árvores utilizadas nas junções dos fragmentos, quando for o caso. Caso alguma sub-árvore do fragmento vertical seja necessária em alguma etapa da consulta, como na construção do resultado final, na ordenação dos resultados ou em qualquer outra operação da consulta, este fragmento será mantido no plano de

execução. Caso contrário, poderá ser excluído do plano. Este procedimento pode ser definido da seguinte forma:

Definição 4: Seja *CG* uma coleção global, e FV_1, FV_2, ..., FV_n seus fragmentos verticais. *CG* possui um conjunto de sub-árvores $A = \{A_1, A_2, ..., A_n\}$. Por definição, cada fragmento vertical FV_i contém uma projeção, tal que $FV_i = \Pi_{A'}(CG)$, onde $A' \subseteq A$. Seja *Q* a árvore algébrica que representa a consulta *q* sobre *CG*, e *Q'* o resultado da etapa de localização dos dados de *Q*. Seja *P* o conjunto de sub-árvores ($P \subseteq A$) utilizado pelas operações de *Q'*. Um fragmento FV_i pode ser eliminado de *Q'*, se *Q'* não utilizar como operando nenhuma sub-árvore contida em *A'*, de forma que $\Pi_P(FV_i) = \varnothing$. Para que esta projeção seja vazia, tem-se a seguinte regra: $\Pi_P(FV_i) = \varnothing$ se o conjunto de sub-árvores P não contiver nenhuma sub-árvore em A'.

A Figura 15 exemplifica a redução de um fragmento vertical da consulta *CLoja_bps_c11.xq* sobre a coleção de *CLoja*. Neste exemplo, o fragmento vertical 2, que não contém o elemento *Funcionários* exigido pela consulta original como apresentado na Tabela 2, pode ser desconsiderado do plano algébrico.

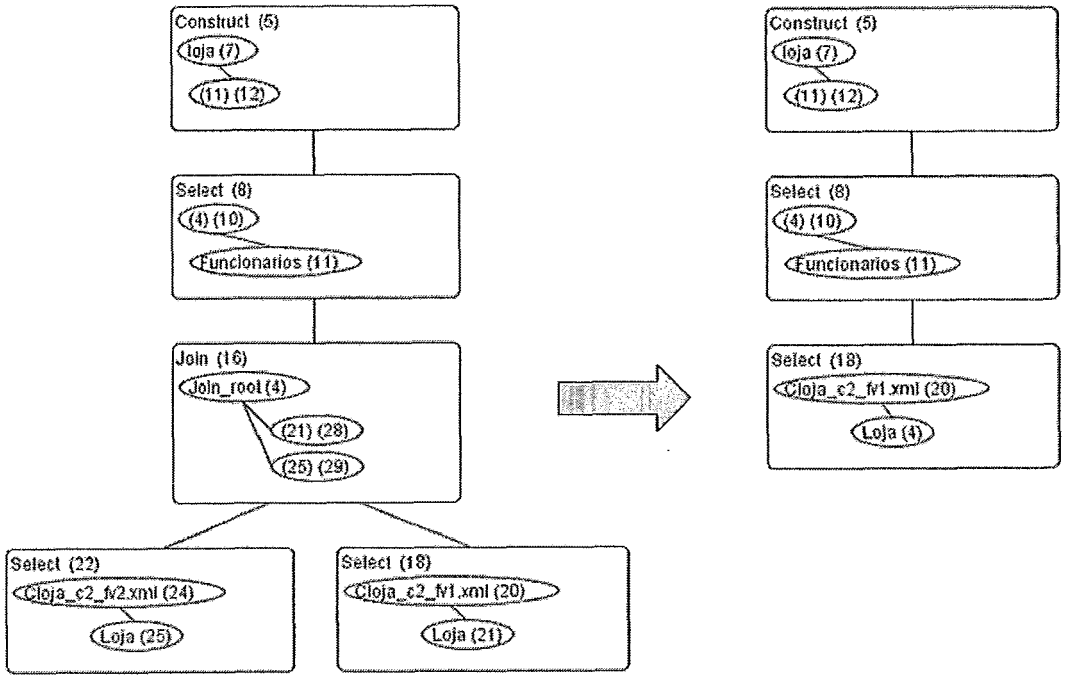


Figura 15: Redução de fragmentos verticais

4.2.2.3 Fragmentação Híbrida

A fragmentação híbrida é caracterizada por fragmentos formados por operações de seleção e/ou de projeção, contendo pelo menos um fragmento definido pelos dois tipos de operação (seleção e projeção).

A reconstrução da coleção global formada por fragmentos híbridos será formada por operações de união e de junção, aplicadas sobre os fragmentos. A regra de reconstrução da coleção global deverá ser construída a partir das definições das operações de seleção e de projeção dos fragmentos. Uma fragmentação híbrida poderá ser reconstruída a partir de uma operação de junção aplicada sobre operações de união, ou de uma operação de união aplicada sobre operações de junção, dependendo do tipo da fragmentação híbrida.

Uma fragmentação híbrida será do tipo *horizontal primária* se for criada por uma fragmentação horizontal seguida de uma fragmentação vertical destes fragmentos horizontais. E será do tipo *vertical primária* se for criada por uma fragmentação vertical seguida da fragmentação horizontal destes fragmentos verticais. O tipo da fragmentação híbrida é fundamental para criarmos a função de reconstrução da coleção global, e poderá ser identificado automaticamente da seguinte forma:

Definição 5: Seja *CG* uma coleção global, e FY_1, FY_2, ..., FY_n seus fragmentos híbridos ordenados pela seqüência em que foram criados. Se FY_1 é definido apenas por uma operação de seleção, então a fragmentação híbrida é do tipo horizontal primária. Se FY_1 é definido apenas por uma operação de projeção, então a fragmentação híbrida é do tipo vertical primária. Caso contrário, deveremos ler as definições do fragmento FY_2, irmão de FY_1, que será um fragmento com uma operação de seleção ou de projeção idêntica à mesma operação de FY_1. Se esta operação idêntica for uma operação de seleção, então a fragmentação híbrida é do tipo horizontal primária. Se não, será do tipo vertical primária.

Após a identificação do tipo da fragmentação híbrida, poderemos criar a função de reconstrução da coleção global. A regra a seguir apresenta como criar esta função de reconstrução para uma fragmentação híbrida do tipo vertical primária. A regra para fragmentação híbrida do tipo horizontal primária pode ser facilmente obtida a partir de adaptações nesta regra.

Definição 6: Seja *CG* uma coleção global, e FY_1, FY_2, ..., FY_n seus fragmentos definidos em uma fragmentação híbrida do tipo vertical primária. Este tipo de fragmentação é definido inicialmente por uma fragmentação vertical, e de

acordo com a **Definição 3**, temos que a operação raiz da função de reconstrução da coleção global será uma operação de junção. Se existirem fragmentos FY_i definidos apenas pela operação de projeção, estes fragmentos poderão ser tratados como fragmentos verticais e adicionados à função de reconstrução da coleção global utilizando as regras da **Definição 3** para reconstrução de uma coleção fragmentada verticalmente. Para cada fragmento FY_j definido por operações de seleção e projeção deveremos buscar os seus fragmentos irmãos, ou seja, fragmentos que possuírem operações de projeção idênticas à do fragmento FY_j. Estes fragmentos irmãos são o resultado de uma fragmentação horizontal de um fragmento vertical, e poderão ser unidos pela operação de união de acordo com a **Definição 1**. Desta forma, podemos criar a função de reconstrução de uma fragmentação híbrida a partir das regras de reconstrução da fragmentação horizontal e da fragmentação vertical.

Para o caso da fragmentação híbrida apresentada na Tabela 2, se aplicarmos as definições apresentadas, descobriremos que se trata de uma fragmentação híbrida do tipo vertical primária, devido ao primeiro fragmento ser formado apenas por uma operação de projeção. A árvore de operações da função de reconstrução desta fragmentação híbrida é apresentada na Figura 16.

CLoja_c3_fy1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$
CLoja_c3_fy2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Brinquedos"} \rangle$
CLoja_c3_fy3 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Games"} \rangle$
CLoja_c3_fy4 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Perfumaria"} \rangle$
CLoja_c3_fy5 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Eletronicos"} \rangle$
CLoja_c3_fy6 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="CD"} \rangle$
CLoja_c3_fy7 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="DVD"} \rangle$
CLoja_c3_fy8 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Livraria"} \rangle$
CLoja_c3_fy9 := $\left\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{\substack{/Item/Secao \neq "Brinquedos" \\ \wedge /Item/Secao \neq "Games" \\ \wedge /Item/Secao \neq "Perfumaria" \\ \wedge /Item/Secao \neq "Eletronicos" \\ \wedge /Item/Secao \neq "CD" \\ \wedge /Item/Secao \neq "DVD" \\ \wedge /Item/Secao \neq "Livraria"}} \right\rangle$

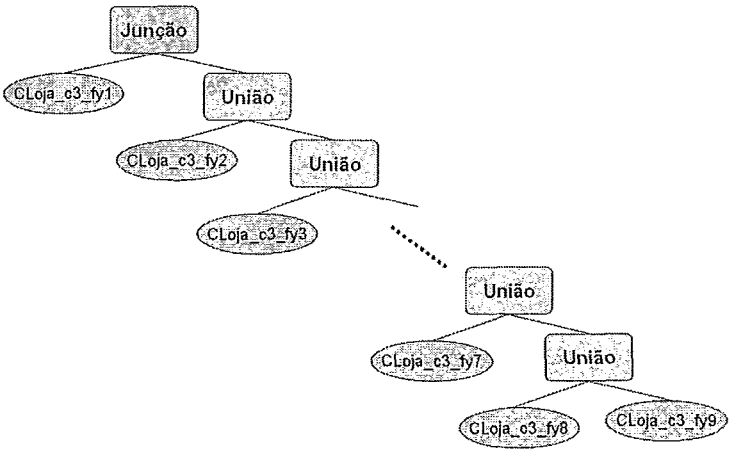


Figura 16: Operações para reconstrução da fragmentação híbrida apresentada na Tabela 2

Após a montagem das expressões para reconstrução da coleção global, podemos aplicar os mesmos recursos apresentados para fragmentação horizontal e vertical sobre cada fragmento híbrido para a redução dos fragmentos irrelevantes ao resultado da consulta. Desta forma, a redução de um fragmento híbrido poderá ocorrer devido a critérios de predicado de seleção e/ou a critérios de predicado de projeção.

4.2.3 Otimização Global

A etapa de otimização global da consulta procura obter um plano algébrico de execução de custo mínimo, criando variações equivalentes do plano algébrico obtido na etapa de Localização da consulta através de transformações algébricas e aplicando uma função de custo para determinação do melhor plano. A busca do plano ótimo pode se tornar, além de muito difícil e até mesmo impossível, muito cara para o processamento da consulta, o que acaba reduzindo os ganhos obtidos com otimização do plano algébrico. Portanto, esta etapa também deve utilizar heurísticas de otimização algébrica e técnicas de minimização da função de custo que busquem um plano próximo ao ótimo com o menor custo de processamento possível. Apesar de não ser o foco desta dissertação, nesta seção damos uma visão geral sobre a etapa de otimização global.

A otimização global de uma consulta distribuída começa com a criação de variações equivalentes do plano algébrico localizado, onde poderão ser aplicadas transformações algébricas como a troca de posição de operações dentro do plano; a substituição da localização de fragmentos, quando existirem réplicas destes fragmentos em diferentes nodos do ambiente distribuído; a inversão de operações filhas de uma operação de união ou junção; etc.

Para cada plano algébrico equivalente produzido será calculado o seu custo estimado a partir de uma função de custo. O plano com menor custo será escolhido para execução da consuta distribuída. Esta função de custo deverá utilizar e estimar parâmetros para o cálculo do custo de cada operação do plano, de forma a obter o custo total do plano. Dentre estes parâmetros, podemos destacar o volume de dados processados pela operação, o custo de acesso a disco, custo de transmissão de dados por rede, estimativas de volume de dados a partir de histogramas das bases de dados, estimativas do volume de dados retornado por uma operação, etc.

Maiores detalhes sobre estas transformações algébricas e funções de custo para otimização global de consultas estão fora do escopo deste trabalho por se tratar de um campo muito abrangente que deve ser analisado em um trabalho futuro.

Um importante algoritmo para otimização global de consultas distribuídas, o algoritmo de programação dinâmica é avaliado em (KOSSMAN et al., 2000). Este algoritmo constrói planos de execução global a partir da localização de réplicas das relações envolvidas nas consultas, descartando planos de qualidade inferior o quanto antes puder. O grande problema deste algoritmo é que ele possui complexidade exponencial, o que o torna inviável para consultas mais complexas.

Esta etapa de Otimização Global tem como produto final um plano de execução próximo do ótimo. Cada operação do plano algébrico otimizado deverá conhecer o *site* onde será executado, que pode ser um *site* remoto ou o próprio SGBD distribuído. Este plano algébrico final será utilizado para a montagem das sub-consultas em XQuery, que executarão todas as operações direcionadas a um mesmo nodo. Caso mais de um nodo remoto seja envolvido na consulta, o mediador será responsável pela composição do resultado final, a partir de uma consulta de composição, como visto em detalhes na seção 4.3.

4.2.4 Otimização Local

A otimização local é realizada pelos nodos remotos, através do próprio banco de dados que armazena o fragmento XML. A princípio, qualquer banco de dados XML capaz de receber consultas em XQuery poderá ser utilizado para este papel, pois as sub-consultas geradas serão em XQuery. Detalhes de otimização de consultas XQuery em bases XML nativas estão fora do escopo deste trabalho. Maiores informações sobre o processamento de consultas em bancos de dados

XML nativos podem ser encontradas em (SCHONING, 2001; JAGADISH et al., 2002; MEIER, 2002; FIEBIG et al., 2002).

4.3 Execução e Composição do Resultado Final

As etapas de execução das consultas remotas e a composição do resultado final não fazem parte da metodologia propriamente dita pois dependem da implementação e do contexto da aplicação. No caso de um SGBD distribuído, estas etapas de execução e composição do resultado poderão ser implementadas utilizando seus próprios recursos para execução de consultas XQuery, enquanto que em um mediador responsável pela distribuição da consulta precisaremos utilizar recursos externos para execução das operações algébricas de composição do resultado.

De modo geral, para a execução da consulta distribuída utilizamos o plano algébrico localizado, reduzido e otimizado, onde devemos identificar o local de execução de cada operação algébrica do plano, tendo como ponto de partida a localização dos fragmentos da coleção global. A partir daí, poderemos agrupar as operações algébricas, onde cada grupo irá conter operações a serem executadas em um mesmo nodo. Cada grupo de operações corresponderá a uma sub-consulta a ser executada por um nodo remoto ou pelo próprio SGBD distribuído, para a composição do resultado final. Quando existirem fragmentos localizados em nodos distintos no plano algébrico de uma consulta, obrigatoriamente existirá uma sub-consulta que será executada no SGBD distribuído para realizar a junção dos resultados dos fragmentos remotos e a composição do resultado final. A Figura 17 mostra o plano algébrico otimizado para a consulta *COrders_bps_c12.xq* decomposto em três grupos, dois para execução em nodos remotos e um para execução no próprio SGBD distribuído a partir dos resultados das execuções remotas.

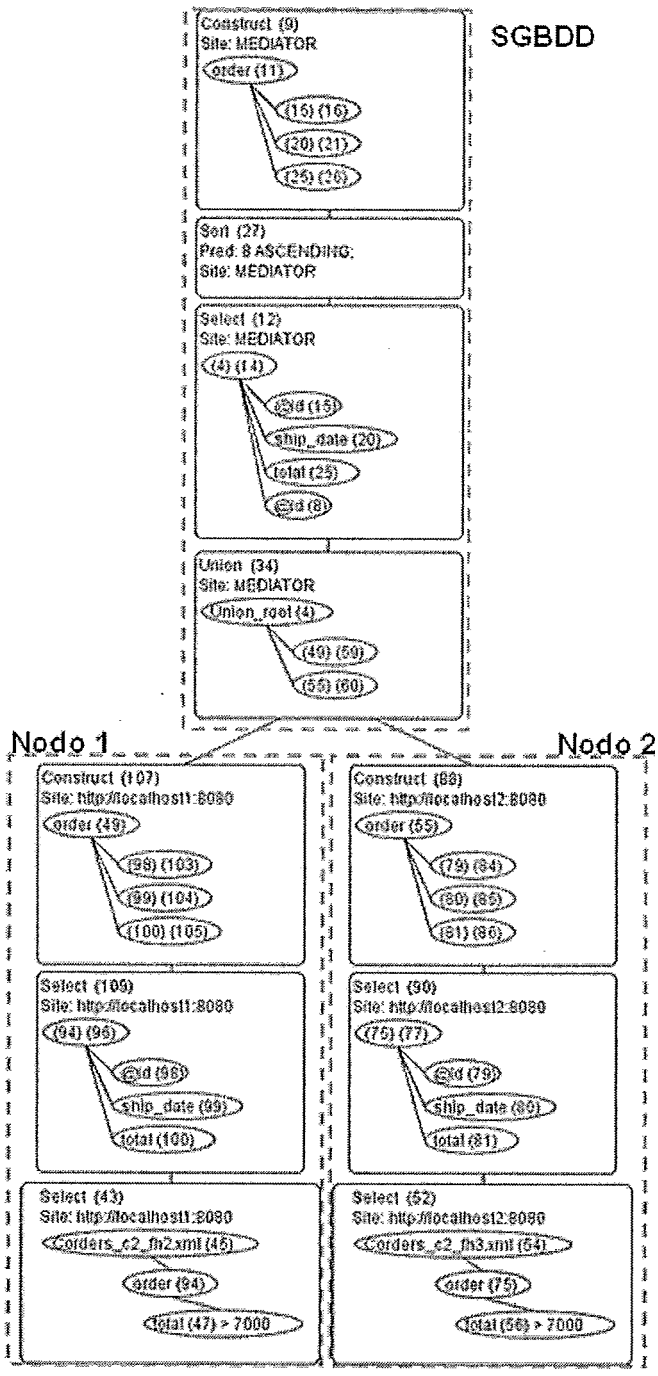


Figura 17: Decomposição do plano algébrico em sub-consultas.

Como foi dito anteriormente, a técnica para execução destas operações nos nodos remotos e a composição do resultado final dependerá da forma como a metodologia foi implementada no ambiente distribuído. Em um banco de dados XML distribuído, por exemplo, as operações de composição dos resultados remotos poderão ser executadas diretamente pelo processador de operações algébricas do próprio SGBD. Por outro lado, no caso de um mediador que não possui um processador nativo para execução das operações algébricas, poderá

ser criada uma sub-consulta em XQuery para ser executada sobre os resultados dos nodos remotos para composição do resultado final.

Após a criação das sub-consultas, é iniciada a etapa de execução da consulta global. As sub-consultas serão enviadas aos bancos de dados remotos em paralelo, para obtermos o paralelismo intra-consulta. Após a obtenção de todos os resultados remotos, o SGBD distribuído executará as operações de composição do resultado final sobre os resultados remotos.

Caso exista apenas uma consulta remota a ser executada, como em situações onde apenas um fragmento está envolvido na consulta, ou de fragmentos que estejam localizados em um mesmo nodo, não será necessária a composição do resultado final, pois o resultado da sub-consulta remota já será o resultado final da consulta global. Neste caso, apenas um grupo de operações será gerado e o seu resultado será repassado diretamente ao usuário sem necessidade de pós-processamento.

Capítulo 5 - Implementação de um Sistema para Processamento de Consultas XQuery Distribuídas

5.1 Introdução

Com o objetivo de avaliar a viabilidade técnica da nossa metodologia e analisar questões de desempenho para o processamento de consultas XQuery em ambiente distribuído, propomos uma arquitetura baseada em camadas onde foi implementada a metodologia descrita no Capítulo 4,. Os detalhes da arquitetura proposta e da implementação são apresentados neste capítulo, já os experimentos realizados e seus resultados são discutidos no Capítulo 6.

5.2 Arquitetura Proposta

A distribuição de uma base de dados introduz algumas dificuldades no processamento de consultas. Dentre estas dificuldades podemos destacar a transparência dos dados distribuídos para o usuário, que envolve a decomposição da consulta original em sub-consultas destinadas a cada base distribuída; a composição do resultado final da consulta original a partir dos resultados parciais de cada sub-consulta; o gerenciamento de um catálogo com as definições das bases distribuídas através de seus fragmentos; o gerenciamento do próprio ambiente distribuído; dentre outros. Um sistema transparente "oculta" dos usuários os detalhes de implementação da distribuição, sejam estes detalhes relacionados a questões de rede, replicação de dados ou fragmentação. Para que a distribuição de uma base de dados seja totalmente transparente aos seus usuários, se faz necessário criar um ponto único de acesso para consultas (não necessariamente centralizado) que represente uma visão global dos dados distribuídos (ÖZSU et al., 1999).

Para compor esta visão global e servir de ponto único de acesso, consideraremos o uso de um *mediador* (WIEDERHOLD, 1992) como mostrado na Figura 18, que seria responsável pelo processamento das consultas distribuídas, ocultando dos usuários os detalhes da localização e da fragmentação da base. As consultas submetidas sobre uma visão global seriam decompostas em um conjunto de sub-consultas, que seriam então executadas pelos nodos remotos sobre os fragmentos. Os resultados de cada sub-consulta retornariam ao mediador para

construção do resultado final. No contexto de bases XML distribuídas, consultas XQuery (BOAG et al., 2007) seriam submetidas sobre visões globais de fragmentos XML distribuídos.

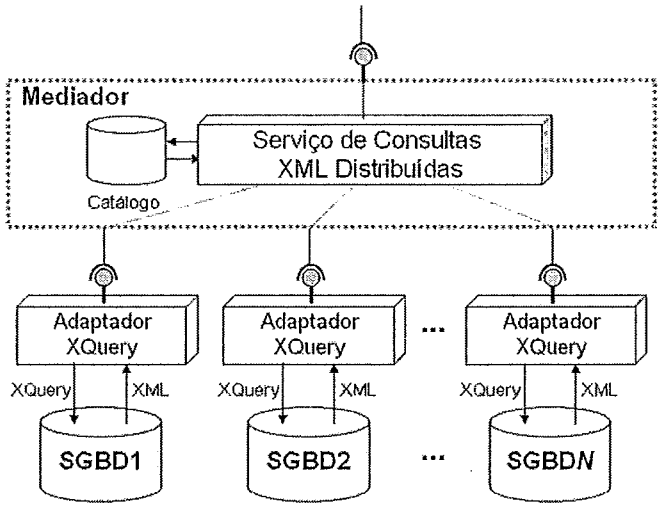


Figura 18: Arquitetura *Mediador – Adaptadores* para a execução das consultas em bases de dados XML distribuídas

A arquitetura apresentada na Figura 18, sobre a qual a implementação do protótipo foi totalmente baseada, contempla os seguintes componentes:

Um **Mediador**, responsável pelo processamento da consulta XQuery sobre as coleções globais de fragmentos distribuídos de bases XML; envio das sub-consultas geradas aos Adaptadores; e composição do resultado das sub-consultas para obtenção do resultado final.

Um **Catálogo**, que armazena as configurações do ambiente distribuído, como a referência de cada Adaptador remoto e os fragmentos por ele disponibilizados, a definição de cada fragmento (através de operações de projeção e de seleção), a coleção global a eles relacionada e os esquemas das coleções.

Um conjunto de **Adaptadores**, que executam as sub-consultas nos SGBDs XML a eles acoplados através de uma biblioteca ou protocolo específico de comunicação. O resultado da sub-consulta é retornado ao Mediador para composição do resultado final.

Uma característica importante de uma arquitetura desenvolvida para aplicações distribuídas é a interface de comunicação entre os seus componentes. Na arquitetura adotada para este protótipo, optamos por utilizar uma interface baseada em serviços *Web* (BOOTH et al., 2004) implementada pelo Mediador e pelos Adaptadores. Esta interface tem como objetivo tornar a arquitetura mais

flexível e com suporte a um ambiente mais heterogêneo, possibilitando a criação de Adaptadores em diferentes linguagens de programação, a utilização de diferentes sistemas operacionais nos nodos do ambiente distribuído, e a utilização de outros Mediadores no lugar de Adaptadores, possibilitando a criação de níveis hierárquicos de mediação.

A interface projetada possui a operação *executeXQuery* cujos parâmetros de entrada e saída são apresentados na Tabela 5.

Tabela 5: Descrição da interface implementada pelo Mediador e Adaptadores.

executeXQuery		
Entrada		Descrição
query	string	Consulta XQuery a ser executada pelo Adaptador/Mediador
Saída		Descrição
numberQueriesExecuted	int	Número de consultas executadas pelo Adaptador/Mediador
result	string	Representação em string do XML com o resultado da consulta
success	boolean	Flag indicando sucesso ou falha na execução
timeMsCommunicRemote	long	Tempo em milisegundos gasto com comunicação com nodo remoto
timeMsCompile	long	Tempo em milisegundos gasto com a compilação da consulta pelo Mediador ou pelo SGBD
timeMsLocal	long	Tempo em milisegundos gasto com a execução local da consulta pelo Mediador ou pelo SGBD
timeMsRemote	long	Tempo em milisegundos gasto com a execução remota da consulta (compilação remota + execução)
totalBytes	long	Quantidade de bytes do resultado da consulta (result)

Apesar de apresentar vantagens do ponto de vista de interoperabilidade, uma interface baseada em serviços *Web* apresenta desempenho inferior se comparada com interfaces proprietárias. Nos experimentos executados, veremos que o tempo gasto com comunicação foi significativo em relação ao tempo total das consultas, principalmente para consultas com maior volume de dados trafegados entre os Adaptadores e o Mediador e entre o Mediador e o cliente. Em interfaces através de serviços Web, além do tempo gasto com o tráfego dos dados pela rede, existe ainda o agravante de que as mensagens, transmitidas como documentos XML, precisam ser serializadas para envelopes SOAP pelo emissor e desserializadas pelo receptor. Apesar de ter sido utilizada para o protótipo, esta

interface pode ser implementada utilizando outras tecnologias, como Java RMI (SUN MICROSYSTEMS, 2006b) e CORBA (OMG, 2004), dependendo da aplicação e dos requisitos de desempenho e de interoperabilidade exigidos. Alguns trabalhos na literatura comparam os recursos e o desempenho destas tecnologias (DAVIS et al., 2002; JURIC et al., 2004; JURIC et al., 2006).

5.3 Mediador

O Mediador é o principal componente da arquitetura pois ele é responsável pelo processamento da consulta distribuída, realizando as etapas de decomposição, localização e otimização global da consulta, conforme a metodologia descrita na seção 4.2 deste documento.

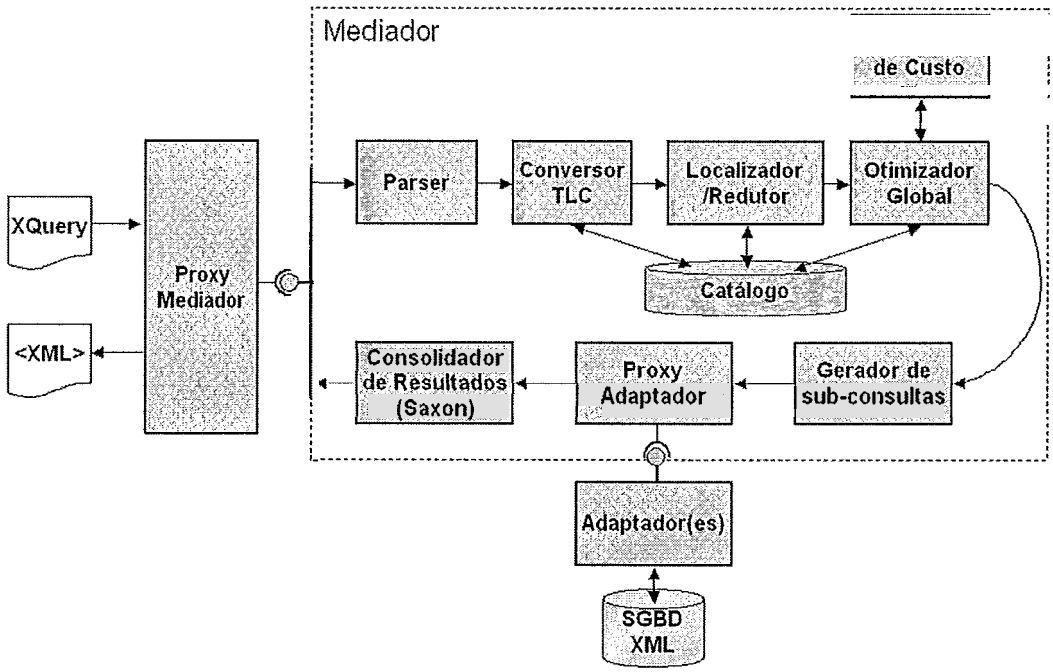


Figura 19: Diagrama de blocos dos componentes do Mediador.

A arquitetura para o processamento das consultas distribuídas pelo Mediador segue a arquitetura básica para processamento de consultas apresentada em (KOSSMAN, 2000), estendida para o Mediador como esquematizado na Figura 19. Esta arquitetura possui uma série de módulos, responsáveis por partes isoladas do processamento da consulta distribuída, como será descrito a seguir.

Parser: responsável pela validação sintática da consulta XQuery submetida pelo usuário de acordo com a gramática suportada pelo Mediador (seção 3.2.3). O *parsing* da consulta é a primeira parte da etapa de decomposição

no processamento de uma consulta XQuery distribuída, conforme foi apresentado na seção 4.2.1.1. Para a implementação do *Parser* no Mediador, utilizamos a biblioteca *JavaCC* com *JJtree* (SUN MICROSYSTEMS, 2006a), que permite a construção automática de um *parser* a partir de um arquivo com a definição da gramática. Além da validação sintática, o *Parser* transforma a consulta textual em uma representação interna que será utilizada na próxima fase do processamento.

Conversor TLC: módulo responsável pela representação algébrica TLC da consulta XQuery. Implementa o algoritmo de conversão da XQuery para a TLC como descrito na seção 3.2.2 que transforma uma consulta em sua representação algébrica, que será utilizada pelas próximas fases do processamento da consulta. O diagrama de classes simplificado da implementação das operações algébricas da TLC para o Mediador é apresentado na Figura 20.

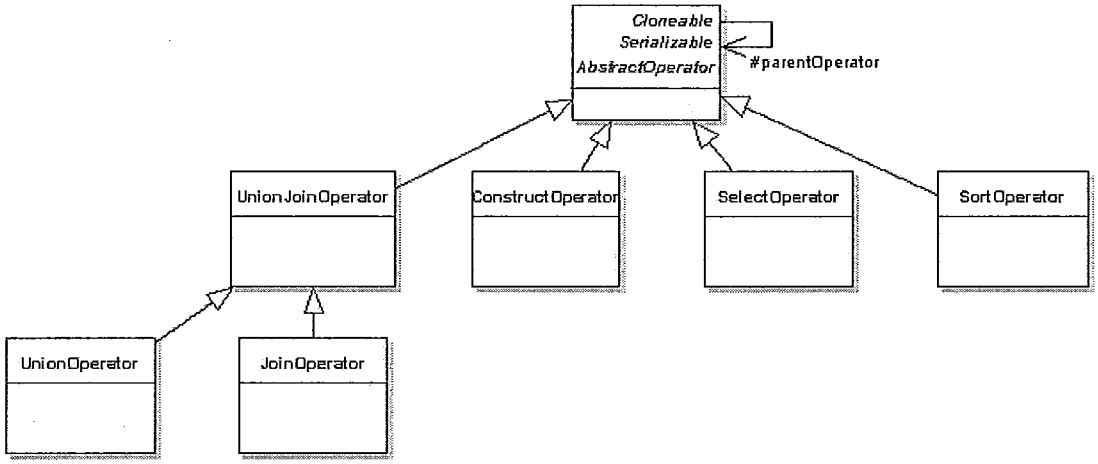


Figura 20: Diagrama de classes das operações da TLC utilizados na representação algébrica das consultas XQuery pelo Mediador.

Localizador/Redutor: responsável pela etapa de localização do plano algébrico da consulta. Essa etapa compreende duas atividades: (i) a substituição das referências a coleções globais por referências a fragmentos destas coleções, e (ii) a eliminação dos fragmentos irrelevantes ao resultado da consulta, utilizando a abordagem de substituição/redução do plano algébrico apresentada na seção 4.2.2. Utiliza os dados disponibilizados no Catálogo (descrito na seção 5.4) para localização das coleções globais.

Otimizador Global: responsável pela otimização global do plano algébrico. Na implementação do protótipo, desenvolvemos um otimizador que produz um conjunto de planos algébricos equivalentes a partir das réplicas dos fragmentos existentes no ambiente distribuído para descobrir, com o uso de uma função de custo, o plano de menor custo total dentre os planos gerados. Outras

abordagens e heurísticas poderiam ser utilizadas para otimização do plano algébrico global, como foi comentado na seção 4.2.3.

Função de Custo: módulo responsável pelo cálculo do custo de um plano algébrico a partir dos dados estatísticos de cada fragmento, como o número de nodos, tamanho médio em *bytes* de cada nodo, parâmetros de seletividade, peso de leitura em disco, peso de comunicação, etc. Na implementação do protótipo utilizamos apenas o peso da comunicação e a estimativa do número total de nodos do fragmento para o cálculo do custo do plano algébrico. O cálculo do custo é feito das folhas do plano algébrico para a raiz. É feita uma estimativa do volume de dados processados por cada operação e o volume que é trafegado entre as operações do plano, considerando o peso da comunicação entre as operações. Operações executadas em um mesmo nodo não possuem custo de comunicação, mas também não poderão ser executadas em paralelo, o que poderia compensar o custo de comunicação devido à distribuição do processamento da consulta.

Gerador de Sub-consultas: responsável pela extração e composição das sub-expressões (sub-consultas) do plano algébrico otimizado. Cada sub-consulta é transformada em uma representação em XQuery e enviada para o seu Adaptador correspondente, ou permanece no próprio Mediador para realizar a composição do resultado final. As sub-consultas são criadas a partir de operações da TLC, através de um algoritmo inverso ao algoritmo de conversão da XQuery, produzindo uma consulta textual em XQuery a partir de expressões de operações algébricas.

***Proxy* do Adaptador**: componente que permite a comunicação entre o Mediador e os Adaptadores, através da execução dos protocolos para chamada de serviços *Web*. O *proxy* permite a definição do endereço do Adaptador que será invocado, tornando todo o processo de comunicação com o serviço *Web* transparente para o resto do Mediador.

Consolidador dos Resultados: módulo responsável pela etapa de composição do resultado final. Em nossa implementação, a composição do resultado final é realizada através da execução de uma consulta XQuery local sobre os resultados das sub-consultas retornadas pelos Adaptadores, como mostrado na Figura 21. Utilizamos o processador de XQuery Saxon (SAXONICA LIMITED, 2006) para execução da consulta em memória, sem necessidade de armazenamento dos resultados dos Adaptadores em disco. Esta forma de implementação foi utilizada para facilitar o desenvolvimento de um protótipo para a nossa metodologia. Outra alternativa seria que o Mediador executasse fisicamente

as operações algébricas de composição dos resultados. Desta forma ele poderia, por exemplo, utilizar técnicas de processamento de *streaming* de dados, o que poderia melhorar o desempenho das consultas distribuídas com grandes volumes de dados. Se houver apenas uma sub-consulta, não será necessário compor resultados e o resultado final será o próprio resultado desta sub-consulta.

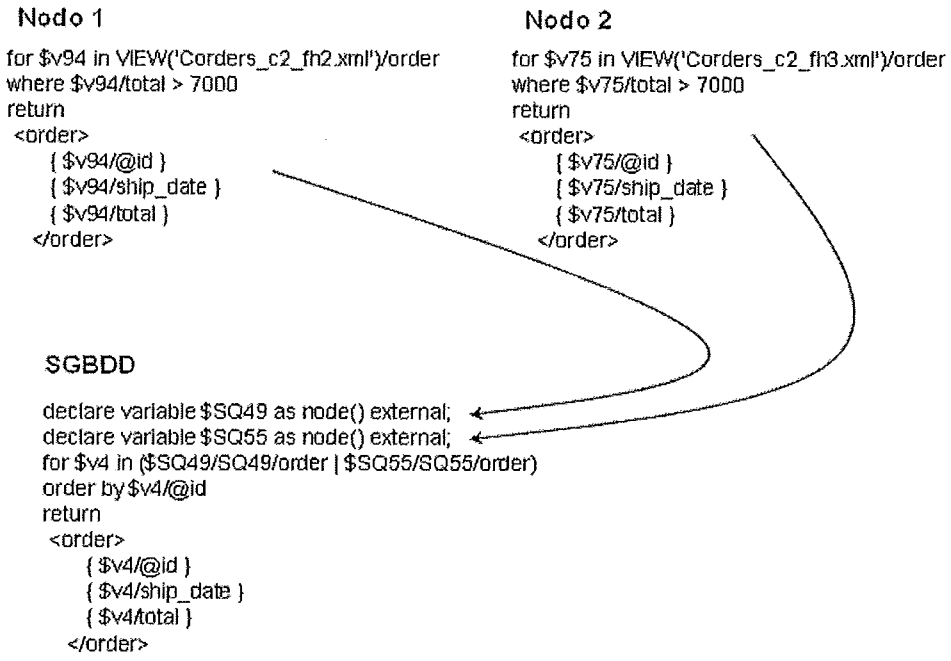


Figura 21: Sub-consultas destinadas aos nodos remotos e ao Mediador.

Observe que as sub-consultas remotas da Figura 21 utilizam a expressão "*VIEW*" para se referir a um fragmento. Esta sub-consulta será processada por um adaptador no banco de dados remoto, onde deverá ser feita a substituição da expressão "*VIEW*" pela expressão correta da sintaxe da XQuery que represente o endereço local do documento ou da coleção consultada.

Um outro detalhe deste exemplo é a execução da operação de ordenação dos resultados apenas na sub-consulta do Mediador, e não nas sub-consultas remotas. Como utilizamos a API do Saxon para execução da sub-consulta de consolidação dos resultados no Mediador, não temos controle sobre o algoritmo utilizado na operação de união dos fragmentos. Caso contrário, se estivéssemos implementando a nossa metodologia em um banco de dados XML nativo, por exemplo, poderíamos executar a operação de ordenação nas sub-consultas remotas e fazer com que o Mediador una os resultados através do algoritmo de *merge*, que mantém a ordenação dos dados.

***Proxy* do Mediador**: componente que, da mesma forma que o *proxy* do Adaptador, permite a comunicação de um cliente com o Mediador, através da

implementação dos protocolos de comunicação e da configuração de atributos específicos do Mediador. É importante observar que, apesar de serem *proxies* diferentes, a interface implementada pelo Mediador e pelos Adaptadores é exatamente a mesma (descrita na Tabela 5). O *proxy* do Mediador foi implementado apenas para facilitar o desenvolvimento de aplicações clientes.

5.4 Catálogo

O Catálogo armazena todas as informações necessárias para o processamento da consulta distribuída, em especial para a etapa de localização, como o nome e o *schema* das visões globais; os fragmentos que formam a visão global da coleção distribuída; as definições de cada fragmento; o endereço de cada Adaptador remoto que possui uma cópia do fragmento; estatísticas dos fragmentos, como número total de nodos, características de seletividade, etc. O Catálogo foi implementado como um conjunto de objetos em Java que pode ser serializado e desserializado em um documento XML para edição manual. A estrutura do documento que representa o Catálogo é apresentada na Figura 22.

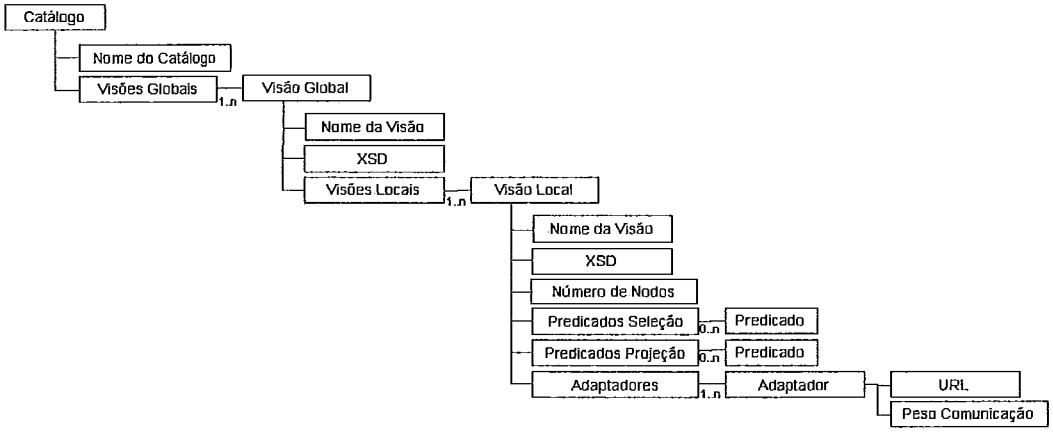


Figura 22: Estrutura do conteúdo do Catálogo do Mediador.

Uma das informações mais importantes armazenadas no Catálogo são as definições dos fragmentos das coleções globais. A partir da relação dos critérios de seleção ou de projeção que formam os fragmentos, o Mediador será capaz de montar uma operação para reconstrução da coleção global a partir dos seus fragmentos, possibilitando a etapa de localização da coleção global. A relação de predicados de formação dos fragmentos também permite ao Mediador fazer a redução do plano algébrico, removendo as operações sobre fragmentos inúteis para o resultado da consulta.

O Catálogo também permite a inclusão de campos com informações estatísticas sobre os fragmentos e sobre os Adaptadores remotos, que poderão ser utilizados na otimização global da consulta, como o número de nodos de um fragmento ou o peso de comunicação de um Adaptador. O Catálogo foi implementado de forma a permitir múltiplas localizações para um determinado fragmento, através da definição dos Adaptadores que contêm uma réplica do fragmento (cardinalidade 1..n). Desta forma, o otimizador global pode escolher dentre os Adaptadores que possuem uma réplica do fragmento, o melhor para enviar a sub-consulta sobre aquele fragmento.

Estes parâmetros de otimização, como o número total de nodos em um fragmento e o peso de comunicação de um Adaptador remoto, são apenas para exemplificar a possibilidade de se definir e incluir novos parâmetros no Catálogo para serem utilizados pelo otimizador do Mediador. Além disso, também seria interessante que estes parâmetros fossem constantemente atualizados para refletirem sempre a realidade atual da base de dados. Esta funcionalidade não foi incluída na implementação do protótipo, pois o nosso foco neste trabalho não era a otimização global da consulta. No entanto, este é um trabalho futuro interessante, dentre outros que são discutidos no Capítulo 7. Quanto mais informações estiverem disponíveis ao otimizador global, maiores serão as suas possibilidades de montar um plano de execução mais eficiente a partir de suas heurísticas e algoritmos de otimização.

O Catálogo também contém uma base de arquivos com os *schemas* das coleções XML globais e dos fragmentos, que são utilizados pelo Mediador para validações semânticas nas consultas e para as etapas de localização e redução no processamento de consultas sobre fragmentos verticais ou híbridos.

5.5 Adaptadores para execução das sub-consultas XQuery

Foram implementados dois tipos de adaptadores para execução de sub-consultas XQuery: um que utiliza o eXist (EXIST DEVTEAM, 2006) (um SGBD XML nativo) e outro que utiliza o Saxon (SAXONICA LIMITED, 2006) (uma biblioteca Java para processamento de consultas XQuery em documentos armazenados em disco ou em memória). Com a utilização de uma interface através de serviço Web publicada pelos adaptadores, a natureza da base de dados XML é transparente para o Mediador, podendo ser tanto um SGBD XML nativo, como o eXist, quanto uma API de execução de consultas XQuery sobre documentos armazenados diretamente em disco, como o Saxon. Essa

transparência para o Mediador permite que sejam implementados adaptadores para diferentes bases de dados XML, ou até mesmo para bases de dados relacionais ou heterogêneas que publiquem visões XML correspondentes aos fragmentos esperados pelo Mediador.

A implementação de um adaptador é relativamente simples, já que a consulta XQuery enviada pelo Mediador já está praticamente pronta para execução. A única responsabilidade do Adaptador, além de implementar a interface de comunicação apresentada na Tabela 5, é atualizar a localização do documento XML, ou da coleção sendo consultada, para o seu endereço na base de dados local (ou no disco local). Para isso, o Adaptador possui um arquivo de configuração que contém o mapeamento entre o nome do documento (fragmento) com o seu endereço completo no servidor local. Após a realização deste mapeamento, o Adaptador pode executar a consulta utilizando a interface ou API do banco de dados por ele acoplado ao ambiente. Este recurso permite fazer com que o Mediador não tenha que saber detalhes sobre o armazenamento físico dos dados nos adaptadores, delegando para eles esta responsabilidade. O diagrama de blocos dos componentes de um Adaptador é apresentado na Figura 23.

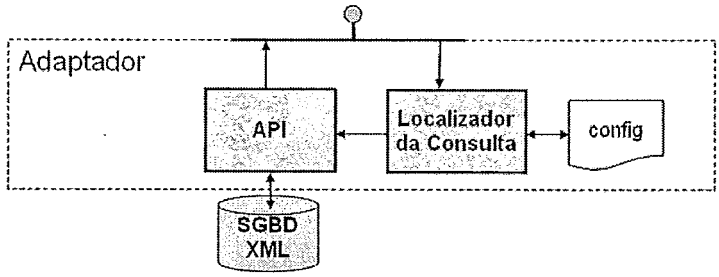


Figura 23: Diagrama de blocos dos componentes do Adaptador

A partir da implementação do Mediador e de Adaptadores, foram executados testes locais para avaliar todos os módulos da implementação. Passados estes testes, iniciamos o planejamento de uma série de experimentos a serem executados em laboratório para análise dos resultados, que serão apresentados no Capítulo 6.

Capítulo 6 - Avaliação Experimental

6.1 Introdução

Com a implementação do protótipo do Mediador e Adaptadores como descrito no Capítulo 5, foi possível realizar experimentos em laboratório descritos neste capítulo. O objetivo dos experimentos é avaliar, para um escopo limitado de consultas, porém representativas, a adequação da metodologia de processamento das consultas XQuery descrita no Capítulo 4, em especial, as etapas de decomposição de consultas e composição do resultado. Além disso, também procuramos analisar aspectos técnicos sobre a fragmentação de documentos XML em ambiente distribuído em comparação com ambientes centralizados, e características de implementação relacionados a desempenho, flexibilidade da arquitetura e viabilidade de uso.

Em (ANDRADE et al., 2006) são apresentados resultados experimentais que mostram que o uso de fragmentação em bases XML é vantajoso em termos do desempenho no processamento de consultas. Em seus experimentos, o autor obteve ganhos de até 72 vezes no desempenho de algumas consultas. É importante lembrar que os ganhos de desempenho com a fragmentação de uma base de dados são mais expressivos para consultas que se beneficiam da fragmentação adotada. Consultas que não se beneficiam da fragmentação podem ter o seu desempenho prejudicado mesmo em ambiente distribuído, devido ao custo da composição do resultado final a partir dos resultados de cada nodo remoto.

Neste capítulo, apresentamos na seção 6.2 detalhes sobre a preparação dos experimentos, como a descrição das bases de dados e dos servidores utilizados, a fragmentação das bases e a alocação dos fragmentos no ambiente distribuído e a metodologia de testes utilizada para execução e análise dos dados. Na seção 6.3, são apresentados os resultados, as análises e as conclusões obtidas a partir deles.

6.2 Preparação dos Experimentos

Para a execução de experimentos com o protótipo implementado, é necessário, previamente: definir os objetivos dos experimentos; planejar a sua

execução, como o ambiente onde serão executados, as bases de dados e fragmentos que serão utilizados, e a alocação destes fragmentos no ambiente distribuído; e definir uma metodologia de execução que será utilizada para permitir uma análise conclusiva dos resultados.

6.2.1 Objetivos

Antes de executar um experimento é preciso definir os objetivos da sua execução para garantir que os seus resultados serão relevantes para o trabalho sendo executado. A partir destes objetivos, poderá ser elaborada uma metodologia de testes para execução dos experimentos (seção 6.2.3), garantindo, desta forma, que todos serão cumpridos.

Os objetivos definidos para os experimentos com o protótipo do Mediador implementado são:

Objetivo 1: Avaliação experimental da decomposição das consultas XQuery, a partir da comparação dos resultados de consultas em ambiente distribuído com os resultados das mesmas consultas em ambiente centralizado (sem o Mediador).

Objetivo 2: Comparação do desempenho de consultas sobre um ambiente centralizado com consultas sobre o ambiente distribuído em cenários com e sem fragmentação das bases.

Objetivo 3: Avaliação de desempenho na execução de consultas que se beneficiam da fragmentação e para consultas que não se beneficiam da fragmentação no ambiente distribuído e fragmentado.

Estes objetivos norteiam todo o planejamento dos experimentos, como a preparação do ambiente, das bases e dos fragmentos, que são vistos na seção 6.2.2, e a elaboração da metodologia de execução apresentada na seção 6.2.3.

6.2.2 Ambiente, Bases e Fragmentos

O ambiente para execução dos experimentos é formado por três computadores em rede local que fazem o papel de nodos do ambiente distribuído (Nodo 0, Nodo 1 e Nodo 2). Cada computador possui processador Pentium Dual Core de 1,8 GHz com 1 GB de memória RAM e sistema operacional Windows XP.

Cada nodo do ambiente distribuído possui um servidor *Web* Tomcat 5.5 (THE APACHE SOFTWARE FOUNDATION, 2006) onde foi implantado o serviço *Web* do Adaptador desenvolvido para eXist e o próprio banco de dados XML nativo

eXist. No nodo 0, além do eXist e do Adaptador, implantamos também o serviço *Web* do Mediador. Desta forma, o nodo 0 possui papel de base de dados e de Mediador. Como o Mediador sempre envia as consultas para os Adaptadores e aguarda o seu resultado, e os experimentos não seriam concorrentes, o impacto de termos papéis de Mediador e Adaptador para um mesmo nodo não seria relevante e nos permitiria executar experimentos em um ambiente mais interessante. A Figura 24 apresenta a distribuição das aplicações nos nodos do ambiente distribuído.

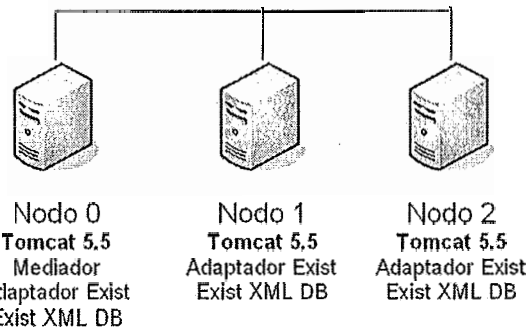


Figura 24: Ambiente para execução dos experimentos.

A partir da definição da quantidade de nodos disponíveis para montagem do ambiente de execução dos experimentos, podemos pensar na fragmentação de bases e na alocação destes fragmentos pelos nodos.

As bases e fragmentos utilizados nos experimentos são os mesmos já apresentados na Tabela 2. Ambas as bases foram geradas utilizando o gerador de bases *ToXgene* (BARBOSA et al., 2002), a partir de *templates*, e carregadas no eXist através de sua ferramenta de administração.

Analisando os objetivos definidos na seção 6.2.1, projetamos quatro cenários para execução dos experimentos:

Cenário 0: Centralizado, onde as consultas são executadas sem intermediação do Mediador, para servir de referência aos resultados dos demais cenários. A definição das bases deste cenário é apresentada na Tabela 6.

Tabela 6: Definição do Cenário 0 para execução dos experimentos.

Cenário 0				
Bases	**Fragmentos**	**Tipo**	**Tamanho**	**Localização**
CLoja	CLoja_c0	SD	4,71 MB	Nodo 0
COrders	COrders_c0	MD	10,1 MB	Nodo 0

Cenário 1: Centralizado, com intermediação do Mediador nas consultas, mas sem fragmentação das bases. Tem como objetivo a comparação com o cenário 0 para avaliação do impacto em desempenho da inclusão do Mediador no processamento das consultas. A definição das bases deste cenário é apresentada na Tabela 7.

Tabela 7: Definição do Cenário 1 para execução dos experimentos.

Cenário 1				
Bases	**Fragmentos**	**Tipo**	**Tamanho**	**Localização**
CLoja	CLoja_c1	SD	4,71 MB	Nodo 1
COrders	COrders_c1	MD	10,1 MB	Nodo 1

Cenário 2: Distribuído, com fragmentação vertical de CLoja e fragmentação horizontal de COrders (3 fragmentos). A definição das bases e fragmentos deste cenário é apresentada na Tabela 8.

Tabela 8: Definição do Cenário 2 para execução dos experimentos.

Cenário 2				
Bases	**Fragmentos**	**Tipo**	**Tamanho**	**Localização**
CLoja	CLoja_c2_fv1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$	SD	4 KB	Nodo 1
	CLoja_c2_fv2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \rangle$	SD	4,71 MB	Nodo 2
COrders	COrders_c2_fh1 := $\langle C_{orders}, \sigma_{/order/total \leq 4000} \rangle$	MD	3,23 MB	Nodo 0
	COrders_c2_fh2 := $\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total < 8000} \rangle$	MD	3,70 MB	Nodo 1
	COrders_c2_fh3 := $\langle C_{orders}, \sigma_{/order/total \geq 8000} \rangle$	MD	3,16 MB	Nodo 2

Cenário 3: Distribuído, com fragmentação híbrida de CLoja (vertical com horizontal) e fragmentação horizontal em COrders (6 fragmentos). A definição das bases e fragmentos deste cenário é apresentada na Tabela 9.

Tabela 9: Definição do Cenário 3 para execução dos experimentos.

Cenário 3				
Bases	**Fragmentos**	**Tipo**	**Tamanho**	**Local**
CLoja	CLoja_c3_fy1 := $\langle C_{loja}, \Pi_{/Loja,\{/Loja/Itens\}} \rangle$	SD	4 KB	Nodo 1
	CLoja_c3_fy2 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Brinquedos"} \rangle$	SD	1 MB	Nodo 0
	CLoja_c3_fy3 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Games"} \rangle$	SD	284 KB	Nodo 0
	CLoja_c3_fy4 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Perfumaria"} \rangle$	SD	97 KB	Nodo 1
	CLoja_c3_fy5 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Eletronicos"} \rangle$	SD	727 KB	Nodo 1
	CLoja_c3_fy6 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="CD"} \rangle$	SD	907 KB	Nodo 2
	CLoja_c3_fy7 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="DVD"} \rangle$	SD	477 KB	Nodo 2
	CLoja_c3_fy8 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao="Livraria"} \rangle$	SD	896 KB	Nodo 2
	CLoja_c3_fy9 := $\langle C_{loja}, \Pi_{/Loja/Itens,\{\}} \bullet \sigma_{/Item/Secao \neq "Brinquedos" \wedge /Item/Secao \neq "Games" \wedge /Item/Secao \neq "Perfumaria" \wedge /Item/Secao \neq "Eletronicos" \wedge /Item/Secao \neq "CD" \wedge /Item/Secao \neq "DVD" \wedge /Item/Secao \neq "Livraria"} \rangle$	SD	648 KB	Nodo 2
COrders	COrders_c3_fh1 := $\langle C_{orders}, \sigma_{/order/total \leq 2000} \rangle$	MD	1,36 MB	Nodo 0
	COrders_c3_fh2 := $\langle C_{orders}, \sigma_{/order/total > 2000 \wedge /order/total \leq 4000} \rangle$	MD	1,86 MB	Nodo 0
	COrders_c3_fh3 := $\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total \leq 6000} \rangle$	MD	1,87 MB	Nodo 1
	COrders_c3_fh4 := $\langle C_{orders}, \sigma_{/order/total > 6000 \wedge /order/total \leq 8000} \rangle$	MD	1,83 MB	Nodo 1
	COrders_c3_fh5 := $\langle C_{orders}, \sigma_{/order/total > 8000 \wedge /order/total \leq 10000} \rangle$	MD	1,87 MB	Nodo 2
	COrders_c3_fh6 := $\langle C_{orders}, \sigma_{/order/total \geq 10000} \rangle$	MD	1,29 MB	Nodo 2

A partir dos objetivos traçados e dos cenários projetados, podemos definir uma metodologia de execução dos experimentos de forma a obter resultados capazes de serem analisados de acordo com os objetivos definidos.

6.2.3 Metodologia de Execução

Para que os resultados dos experimentos sejam relevantes aos objetivos do trabalho, é necessário elaborar um planejamento das atividades para que elas contemplem todos os resultados desejados.

A metodologia de execução dos experimentos neste trabalho consiste na execução de consultas XQuery repetidamente sobre as bases de dados nos quatro cenários apresentados na seção 6.2.2. Cada cenário possui um propósito, o que permitirá a comparação dos resultados de forma a nos permitir fazer uma avaliação do desempenho das consultas em diferentes ambientes e configurações.

As consultas serão executadas automaticamente por uma aplicação cliente desenvolvida para este fim, que implementa a metodologia de execução através dos passos apresentados na Tabela 10.

Tabela 10: Procedimento para execução dos experimentos.

Passo 1: Leitura das consultas armazenadas em um diretório local.
Passo 2: Para cada rodada de execução, faça:
Passo 3: Para cada consulta, faça:
Passo 4: Executar consulta no cenário 0 e gerar "gabarito" do resultado;
Passo 5: Para cada cenário (1, 2 e 3), faça:
Passo 6: Executar a consulta através do Mediador;
Passo 7: Comparar o resultado da consulta com o "gabarito" gerado no Passo 4;
Passo 8: Adicionar tempos medidos na execução ao arquivo de saída.

A aplicação cliente gera um arquivo de saída no formato CSV (*comma-separated values*) com tempos medidos em todas as execuções de todas as consultas em todos os cenários. Este formato pode ser aberto em uma planilha eletrônica para análise dos dados e criação de gráficos dos resultados, como foi feito para este trabalho.

A partir deste arquivo de saída gerado pela aplicação, poderemos obter os tempos médios e os parâmetros definidos na Tabela 11. Além disso, o arquivo de saída também contém informações sobre a comparação dos resultados das consultas executadas através do Mediador com o "gabarito" gerado pela execução no Cenário 0.

Tabela 11: Parâmetros coletados para cada consulta executada.

Parâmetro	Descrição
Tempo total	Tempo total de execução da consulta, do ponto de vista do cliente, entre o envio da consulta e a recuperação do resultado final.
Tempo de comunicação com o Mediador	Tempo gasto com a comunicação entre o cliente e o Mediador, incluindo os tempos de envio da consulta e o recebimento do resultado final.
Tempo de compilação do Mediador	Tempo gasto na compilação da consulta pelo Mediador, que inclui as etapas de *parsing*, decomposição, localização, otimização global e criação das sub-consultas.
Tempo de execução no Mediador	Tempo gasto na execução da consulta pelo Mediador, para consultas que exigem a consolidação de resultados de mais de um nodo remoto.
Tempo máximo de comunicação com nodos remotos	Tempo máximo gasto na comunicação com um nodo remoto entre envio da sub-consulta e recebimento do resultado.
Tempo máximo de execução nos nodos remotos	Tempo máximo gasto na execução de uma sub-consulta por um nodo remoto, incluindo o tempo de execução do Adaptador e do banco de dados eXist.
Tempo total sem tempos de comunicação	Tempo total de execução da consulta descontando os tempos gasto com comunicação entre nodos e Mediador e entre cliente e Mediador.
Número de sub-consultas executadas	Número de sub-consultas produzidas para uma determinada consulta do experimento, que dependerá da fragmentação da base.

Na execução dos experimentos, trabalhamos com dez rodadas de execução para cada consulta. As rodadas com os dois melhores e os dois piores resultados são desconsiderados para o cálculo dos tempos médios. O experimento tem como pré-requisito a comparação dos resultados das consultas em todos os cenários quando comparados com o cenário 0, de acordo com o Objetivo 1. Com todos os resultados corretos, passamos para a etapa de análise dos dados colhidos.

6.2.4 Consultas Utilizadas

As consultas utilizadas nos experimentos foram criadas levando-se em consideração a fragmentação das bases de dados. Foram criadas consultas que

se beneficiavam e que não se beneficiavam da fragmentação da base, para que fosse possível avaliar o desempenho nestes dois tipos de consulta. Para facilitar a identificação do tipo de consulta e da base consultada, foi adotada a seguinte nomenclatura:

Nome da consulta = *Coleção_Benefício_cNúmeroSequencial*.xq

Onde,

Coleção é o nome da coleção consultada (CLoja ou COrders);

Benefício é um dos seguintes indicadores:

bns: Benefício nenhum da fragmentação da base.

bnc: Benefício nenhum da fragmentação da base com complexidade de agregação dos resultados.

bps: Benefício parcial da fragmentação da base.

bpc: Benefício parcial da fragmentação da base com complexidade de agregação dos resultados.

bts: Benefício total da fragmentação da base.

btc: Benefício total da fragmentação da base com complexidade de agregação dos resultados.

NúmeroSequencial é um número identificador da consulta.

A relação das consultas utilizadas nos experimentos é apresentada no Anexo I.

6.3 Análise dos Resultados

Os experimentos foram executados em laboratório, com os servidores do ambiente distribuído totalmente dedicados para os testes. Os resultados são analisados nesta seção, a partir dos dados coletados pelo programa de execução dos experimentos.

A análise dos resultados foi feita a partir da comparação dos tempos totais médios de execução das consultas entre os diferentes cenários. Posteriormente, analisamos também os tempos totais médios desconsiderando o tempo gasto com comunicação entre o Mediador e os Adaptadores. Por fim, analisamos, para cada consulta em cada cenário, o tempo médio de compilação pelo Mediador a fim de verificar individualmente o custo da decomposição da consulta distribuída.

É importante destacar que a implementação do Mediador e do Adaptador utilizada neste experimento é um protótipo, tendo como objetivo a realização de uma prova de conceito, não tendo sido desenvolvido de forma otimizada para obtenção de melhor desempenho. Desta forma, os resultados que serão apresentados podem ser melhorados em futuras versões ou em outras implementações.

6.3.1 Comparação dos Tempos de Execução

A comparação dos tempos médios de execução das consultas nos diferentes cenários foi feita através de um gráfico que apresenta os tempos de execução das consultas normalizados pelo tempo de execução da mesma consulta no cenário 0. Portanto, calculamos, para cada consulta, a relação entre o seu tempo médio de execução nos cenários 1, 2 e 3 com o tempo médio da sua execução no cenário 0, de forma a poder apresentar um gráfico normalizado com o tempo de execução no cenário 0 sempre igual a 1. A Figura 25 apresenta a comparação de desempenho na execução das consultas sobre a coleção *CLoja*, enquanto que a Figura 26 apresenta os resultados para as consultas sobre a base *COrders*.

Para as consultas realizadas sobre a base CLoja (Figura 25), não foi observado nenhum ganho de desempenho com a distribuição e a fragmentação da base. Mesmo para as consultas com benefício total da fragmentação (consultas com "*bts*" no nome, vide seção 6.2.4) que consultavam apenas um único fragmento da base, os tempos introduzidos pela arquitetura distribuída (como os tempos de comunicação entre os nodos e o tempo de compilação da consulta no Mediador) foram significativos em relação ao tempo total da consulta no ambiente centralizado (cenário 0), tornando o seu desempenho em ambiente distribuído inferior.

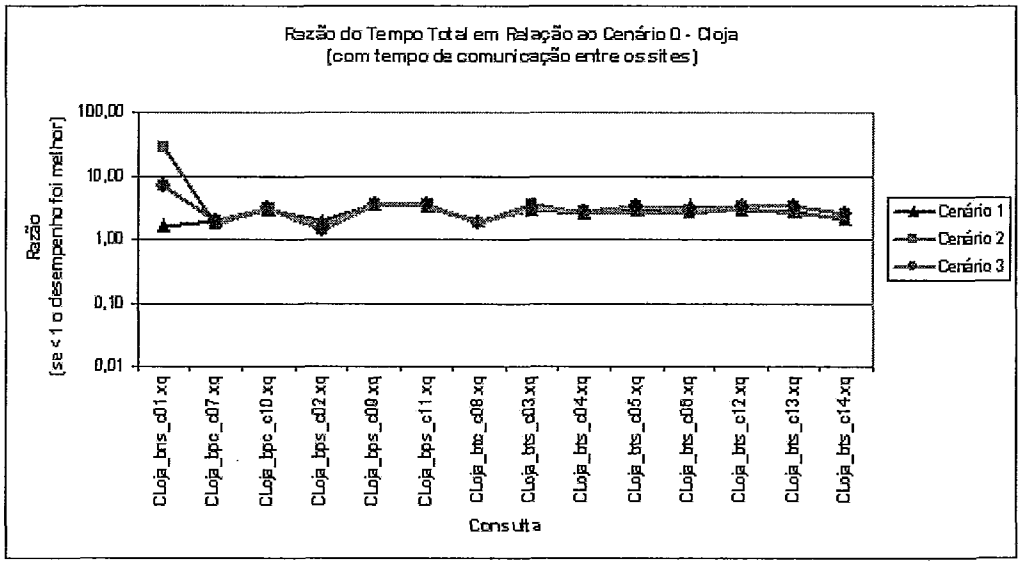


Figura 25: Comparação do Tempo Total de execução das consultas sobre a base CLoja nos cenários 0 (normalizado), 1, 2 e 3.

Algumas consultas realizadas sobre a base COrders (Figura 26) apresentaram ganhos de desempenho no ambiente distribuído e fragmentado, chegando a reduções da ordem de 95% em relação ao tempo de execução em ambiente centralizado. Os maiores ganhos em desempenho foram observados para as consultas com benefício total da fragmentação e complexidade de agregação do resultado, como o caso das consultas *Corders_btc_c14* e *Corders_btc_c16*. As outras consultas com benefício total da fragmentação, mas sem funções de agregação, obtiveram ganhos entre 10 e 40%.

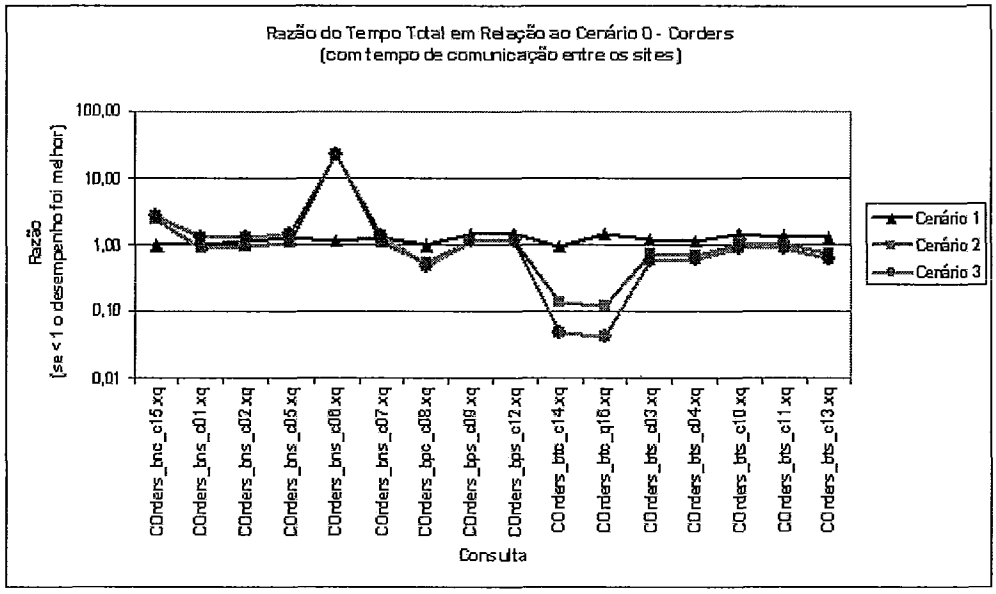


Figura 26: Comparação do Tempo Total de execução das consultas sobre a base COrders nos cenários 0 (normalizado), 1, 2 e 3.

Durante a análise dos resultados, verificamos que o tempo gasto com a comunicação dos nodos com o Mediador, e entre o Mediador e o cliente, era elevado em relação ao tempo total de execução das consultas, devido a problemas de desempenho da interface de serviços *Web* utilizadas no Mediador (JURIC et al., 2004; JURIC et al., 2006). As consultas executadas no cenário 0 diretamente sobre o banco de dados XML, sem intermediação do Mediador, apresentaram taxas de comunicação entre o cliente e o banco de dados bastante superiores às taxas de comunicação medidas nos cenários com uso do Mediador e sua interface de serviço Web. A Figura 27 apresenta a razão entre as taxas de comunicação obtidas com o uso da API do banco de dados Exist e as taxas de comunicação obtidas com o uso do serviço Web do Mediador para todas as consultas utilizadas nos experimentos. Esta diferença média de 38 vezes entre as taxas de comunicação representa o custo por se optar pelo uso de uma arquitetura mais flexível e que permita a heterogeneidade entre os bancos de dados acoplados ao Mediador.

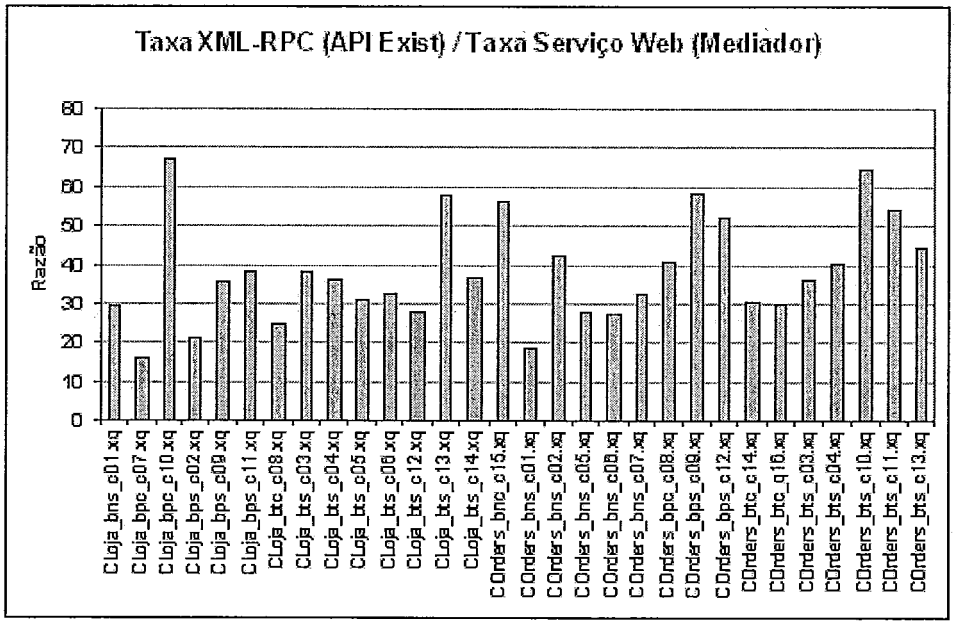


Figura 27: Razão entre as taxas de comunicação através da API do Exist e as taxas de comunicação através de serviço Web.

Para poder avaliar o desempenho do processamento das consultas distribuídas sem interferência dos tempos gastos com comunicação, refizemos os gráficos comparativos removendo os tempos de comunicação do tempo total de execução da consulta. Estes gráficos são apresentados na Figura 28 para as consultas sobre a base *CLoja* e na Figura 29 para as consultas sobre a base *COrders*.

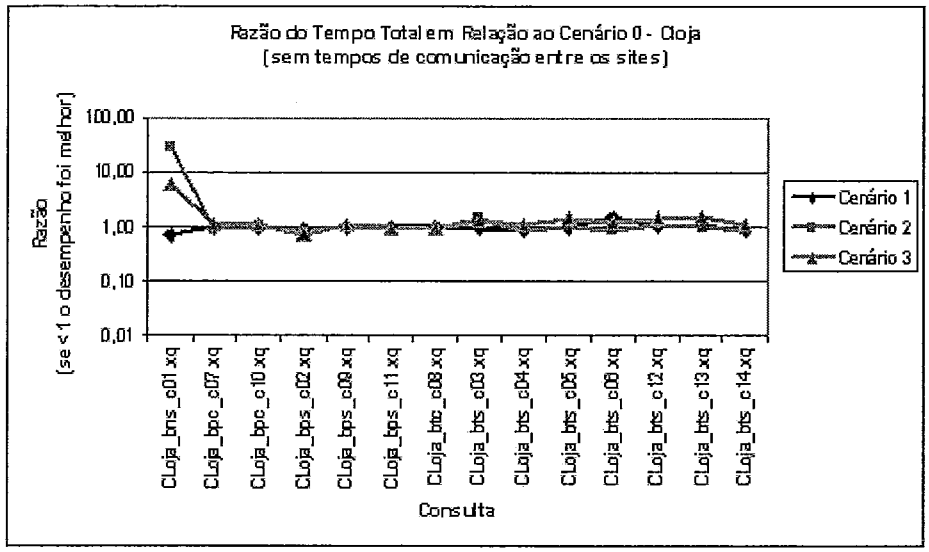


Figura 28: Comparação do Tempo Total de execução das consultas (sem os tempos com comunicação) sobre a base CLoja nos cenários 0 (normalizado), 1, 2 e 3.

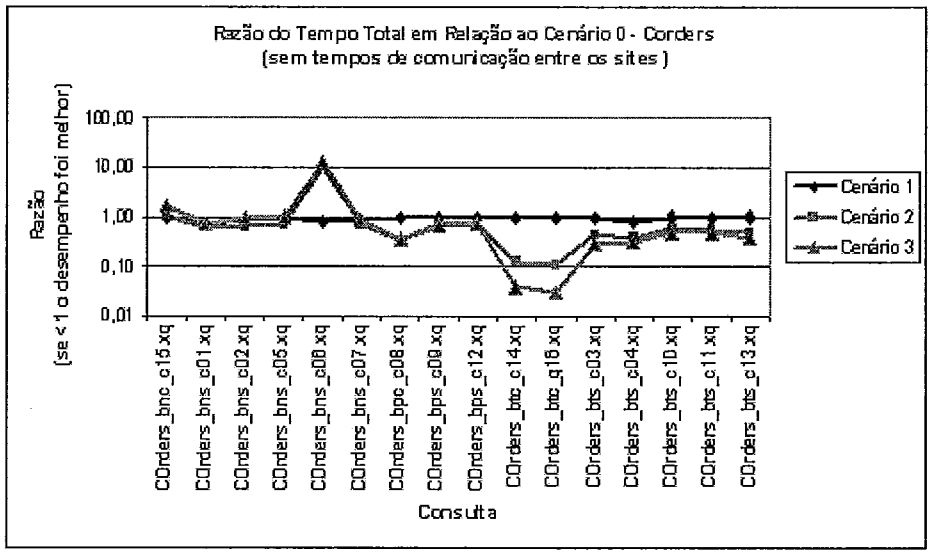


Figura 29: Comparação do Tempo Total de execução das consultas (sem os tempos com comunicação) sobre a base COrders nos cenários 0 (normalizado), 1, 2 e 3.

Com esses resultados, podemos concluir que os problemas de desempenho encontrados nas consultas sobre a base CLoja eram relacionados ao peso da comunicação, para a maioria dos casos. Porém, mesmo com a exclusão dos tempos de comunicação, essas consultas ainda não apresentaram ganhos de desempenho. Comparando os resultados obtidos para a base CLoja e a base COrders, podemos concluir facilmente que a fragmentação da base MD COrders apresentou melhores resultados de desempenho. Observamos que o processamento das consultas diretamente no eXist sobre a base SD foi mais eficiente do que sobre a base MD, o que explica o melhor resultado da

fragmentação da base MD. Em uma base MD, além da forma de armazenamento físico dos documentos, o banco de dados nativo ainda precisa fazer o *parsing* de cada documento da coleção, o que aumenta o tempo das consultas. Por este mesmo motivo, uma base SD fragmentada de forma híbrida em vários fragmentos também terá estes custos adicionais da base MD de recuperação física dos documentos e *parsing* de cada fragmento, podendo fazer com que a fragmentação não se torne tão interessante neste caso.

6.3.2 Tempo de Compilação das Consultas pelo Mediador

Outro parâmetro importante coletado pelo programa de execução dos experimentos é o tempo gasto com a compilação da consulta distribuída pelo Mediador, ou seja, o tempo que ele leva para realizar todas as etapas de processamento descritas no Capítulo 4, desde o *parsing* da consulta original até a criação das sub-consultas.

A importância do tempo de compilação se justifica por refletir o trabalho adicional introduzido pelo Mediador em cenários onde a base de dados não está fragmentada (cenário 1), ou correspondendo ao custo para se trabalhar em um ambiente com fragmentação (cenários 2 e 3). Como podemos observar na Figura 30, o tempo de compilação das consultas no protótipo do Mediador foi aumentando à medida em que foram sendo introduzidos mais fragmentos nas bases utilizadas pelas consultas. Consultas que não se beneficiam da fragmentação utilizada geram um número maior de sub-consultas, exigindo um maior processamento por parte do Mediador e, conseqüentemente, um maior tempo de compilação. Consultas que se beneficiam da fragmentação eliminam operações na etapa de localização, o que reduz a necessidade de processamento pelo Mediador. No Anexo II são apresentados outros gráficos com os tempos totais de execução das consultas e comparações entre os tempos de comunicação, compilação e execução local e remota das consultas nos quatro cenários.

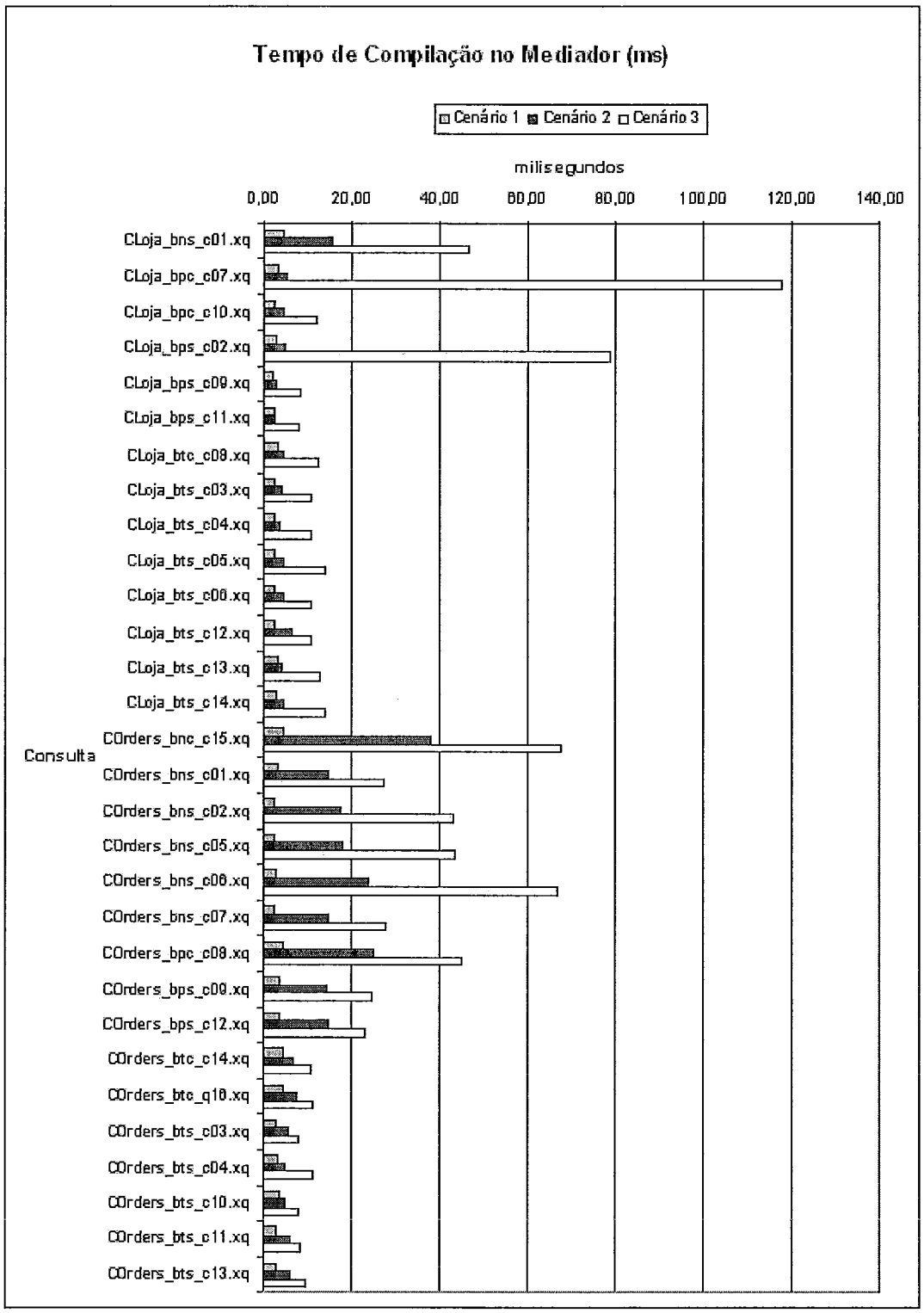


Figura 30: Tempos de compilação das consultas distribuídas pelo Mediador.

Podemos concluir com base nos resultados destes experimentos que a fragmentação de uma base XML é possível a partir de um sistema de decomposição da consulta XQuery como o proposto neste trabalho. Os resultados obtidos nos mostram que se pode reduzir o tempo de execução de consultas em

até 90%, dependendo do tipo de fragmentação, da consulta e do volume de dados consultado. Entretanto, a fragmentação de uma base XML precisa ser muito bem planejada, pois ela pode melhorar muito o desempenho de consultas que se beneficiam da fragmentação adotada, mas também pode piorar significativamente o desempenho daquelas que não se beneficiam da fragmentação. Quanto mais fragmentada a base, mais acentuados serão estes comportamentos. Por estes motivos, se torna necessária uma metodologia para projetos de fragmentação de bases XML, nos moldes do que já existe para o modelo relacional (ÖZSU et al., 1999; ELMASRI et al., 2001) e para o modelo Orientado a Objetos (BAIÃO et al., 2004), que possa auxiliar ou até mesmo automatizar, a partir do histórico de consultas sobre a base, o projeto de fragmentação de uma base XML.

Capítulo 7 - Conclusões e Trabalhos Futuros

7.1 Considerações Finais

Esta dissertação apresentou uma solução para o processamento de consultas XQuery sobre bases de dados XML distribuídas e fragmentadas. O objetivo foi atingido a partir de uma metodologia que se baseia no uso da álgebra TLC (PAPARIZOS et al., 2004) para as etapas de processamento da consulta distribuída, em conjunto com uma definição de fragmentação de documentos XML (ANDRADE et al., 2006) capaz de nos fornecer o formalismo necessário para a aplicação algébrica das regras de reconstrução da coleção original através de seus fragmentos. A partir da consulta algébrica sobre coleções fragmentadas, pudemos criar regras para a localização e para a redução da consulta, de forma que apenas as operações algébricas sobre fragmentos relevantes ao resultado final fossem executadas, o que pode melhorar de modo significativo o desempenho da consulta.

Podemos destacar as seguintes contribuições deste trabalho:

- Uma metodologia para o processamento de consultas sobre coleções distribuídas, genérica o suficiente para poder ser aplicada tanto em um SGBD XML distribuído quanto em um sistema para integração de bases semi-autônomas. A metodologia se baseia em técnicas de processamento de consultas distribuídas para bancos de dados relacionais (ÖZSU et al., 1999), a partir de correlações do modelo relacional com o modelo semi-estruturado.

- Regras para as etapas de localização dos dados e de composição do resultado final de uma consulta algébrica TLC expressa sobre uma coleção fragmentada e distribuída. Estas regras permitem a automatização destas etapas, possibilitando a sua implementação e aplicação em um sistema distribuído. Além disso, a regra para localização dos dados elimina as operações algébricas sobre fragmentos que não contribuirão ao resultado final da consulta, minimizando o acesso aos dados.

- Proposta de uma arquitetura não-intrusiva para implementação da metodologia apresentada, baseada no uso de um mediador com adaptadores acoplados aos bancos de dados remotos (WIEDERHOLD, 1992). O mediador é responsável pelo processamento da consulta distribuída a partir das etapas descritas em nossa metodologia. Os adaptadores são responsáveis pela execução

das sub-consultas enviadas pelo mediador sobre os fragmentos contidos no banco de dados XML a eles acoplados. Especificamos uma interface de comunicação entre o mediador e os adaptadores que permite a construção de adaptadores para diversos SGBDs ou processadores de consulta com suporte a linguagem XQuery.

- Implementação de um protótipo da arquitetura proposta, onde o mediador é capaz de realizar todas as etapas da metodologia de processamento de consultas distribuídas sobre coleções XML fragmentadas. Foram implementados dois adaptadores, um para o banco de dados XML nativo eXist (EXIST DEVTEAM, 2006) e outro para a biblioteca de XQuery Galax (THE GALAX TEAM, 2006). Esta implementação nos possibilitou evidenciar a viabilidade da metodologia, das regras para localização dos dados e composição do resultado final, e da arquitetura proposta. Diversos testes foram conduzidos, utilizando diferentes cenários de fragmentação e de consultas XQuery. Através do gerador Toxgene, muito utilizado em avaliações sobre XML, criamos bases para os testes utilizando os três tipos de fragmentação possíveis: horizontal, vertical e híbrida. As consultas utilizadas nos testes exploraram todas as funções suportadas pela gramática da XQuery adotada, definida na seção 3.2.3.

- Resultados experimentais de desempenho tanto em consultas que se beneficiam da fragmentação adotada quanto em consultas que não se beneficiam da fragmentação. Os experimentos realizados em laboratório, em um ambiente com um mediador e três adaptadores remotos, mostraram que a solução pode apresentar ganhos de até 90% em relação ao tempo da mesma consulta em ambiente centralizado para consultas que se beneficiam da fragmentação. Esta redução no tempo de processamento das consultas foi obtida graças à redução da consulta original feita pelo mediador, através das regras para localização dos dados, e ao paralelismo intra-consulta do ambiente distribuído. Consultas que não se beneficiam da fragmentação, apesar de estar em ambiente distribuído, apresentaram desempenho inferior ao desempenho em ambiente centralizado devido ao processamento adicional efetuado pelo mediador, principalmente para composição dos resultados. Os experimentos também mostraram que a utilização de serviços *Web* para a interface entre os componentes da arquitetura comprometeu o desempenho das consultas devido ao tempo gasto com comunicação entre os nodos, principalmente para maiores volumes de dados. Por outro lado, o uso de serviços *Web* possibilita uma maior interoperabilidade entre os componentes da arquitetura, permitindo que os nodos sejam mais heterogêneos, como para o caso da integração de bases semi-autônomas.

Durante a elaboração deste trabalho, obtivemos uma publicação de um artigo no IV Workshop de Teses e Dissertações em Banco de Dados (FIGUEIREDO et al., 2005). Atualmente, estamos trabalhando em um relatório técnico e um novo artigo apresentando todas as contribuições desta dissertação.

7.2 Trabalhos Futuros

O processamento de consultas distribuídas é um assunto bastante complexo, principalmente quando aplicado ao modelo semi-estruturado. As diferentes etapas do processamento de uma consulta em ambiente distribuído e fragmentado podem ser continuamente evoluídas. Podemos enumerar os seguintes trabalhos futuros para as etapas do processamento de consultas XQuery distribuídas descritas em nossa metodologia:

- Elaboração de uma proposta para um otimizador de consultas XQuery distribuídas, utilizando heurísticas de otimização, estatísticas dos fragmentos e uma função de custo para determinação do melhor plano algébrico (HAAS et al., 1997; ROTH et al., 1999; RUBERG et al., 2003; ZHANG et al., 2005);

- Inclusão de suporte na etapa de redução dos fragmentos ao tratamento de operações com junção de duas bases fragmentadas pelo mesmo atributo, cujos fragmentos envolvidos na junção não possuem predicados compatíveis (ÖZSU et al., 1999);

- Aumento do escopo da gramática da XQuery suportada pelo Mediador, como a inclusão de expressões de quantificação (*"some", "every" ... "satisfies"*), expressões condicionais (*"if", "then","else"*) e expressões de adição e multiplicação (*"+", "-", "*", "div", "idiv", "mod"*) (BOAG et al., 2007);

- Elaboração de uma proposta para tratamento de atualizações na base distribuída, utilizando Adaptadores que suportem a atualização das visões XML locais. Poderiam ser utilizados trabalhos existentes na literatura sobre a atualização de visões XML de bases relacionais (BRAGANHOLO et al., 2004);

Do ponto de vista da arquitetura proposta, da sua implementação e de questões técnicas levantadas durante os testes do protótipo desenvolvido, podemos destacar as seguintes áreas de trabalho:

- Avaliação de diferentes tecnologias de interfaces para o mediador e os adaptadores, como Java RMI e CORBA e a comparação do desempenho das consultas distribuídas com estas diferentes tecnologias;

- Solução para o problema de estouro de memória pelo Mediador (*Java.OutOfMemory*) na etapa de composição do resultado final, quando os resultados dos adaptadores ultrapassa o limite físico de memória disponível. A solução para este problema poderia envolver o uso de um banco de dados XML nativo para armazenar temporariamente os resultados;

- Avaliação de técnicas de processamento de consultas em *stream* de dados para ser utilizada pelo mediador no processamento das operações sobre os resultados dos adaptadores. Esta técnica poderia reduzir o consumo de memória do mediador, e aumentar o desempenho das consultas, devido ao maior paralelismo intra-consulta (KOSSMAN, 2000; JOSIFOVSKI et al., 2005).

Referências Bibliográficas

ABITEBOUL, S., 1999, "On views and XML". In: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 1-9, ACM Press, Philadelphia, Pennsylvania, United States.

ABITEBOUL, S., BUNEMAN, P., SUCIU, D., 1999, "Data on the Web: From Relations to Semistructured Data and XML", Morgan Kaufmann Publishers, San Francisco, California, USA.

AGUILERA, V., CLUET, S., MILO, T., et al., 2002, "Views in a Large Scale XML Repository", *The VLDB Journal*, v. 11, 3, pp. 238-255.

ANDRADE, A., 2006, *PARTIX: Projeto de fragmentação de dados XML*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ANDRADE, A., RUBERG, G., BAIÃO, F., et al., 2005, "PartiX: processing XQuery queries over fragmented XML repositories". In: Technical Report ES-691, COPPE/UFRJ, Rio de Janeiro, RJ.

ANDRADE, A., RUBERG, G., BAIÃO, F., et al., 2006, "Efficiently processing XML queries over fragmented repositories with PartiX". In: DATAX - EDBT Workshop Proceedings, pp. 150-163, Munich, Germany.

BAIÃO, F., MATTOSO, M., ZAVERUCHA, G., 2000, "Horizontal Fragmentation in Object DBMS: New Issues and Performance Evaluation". In: Proceedings of the 19th IEEE International Performance, Computing, and Communications Conference, pp. 108-114, IEEE CS Press, Phoenix, AZ, USA.

BAIÃO, F., MATTOSO, M., ZAVERUCHA, G., 2004, "A Distribution Design Methodology for Object DBMS". In: Distributed and Parallel Databases, v. 16, pp. 45-90, Kluwer Academic Publishers, Hingham, MA, USA.

BARBOSA, D., MENDELZON, A., KEENLEYSIDE, J., et al., 2002, "ToXgene: An Extensible Template-Based Data Generator for XML". In: Proceedings of 5th International WebDB Workshop, pp. 49-54, Wisconsin, USA.

BARU, C., GUPTA, A., LUDAESHER, B., et al., 1999, "XML-Based Information Mediation with MIX". In: Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pp. 597-599, ACM Press.

BERNSTEIN, P. A., GOODMAN, N., WONG, E., et al., 1981, "Query processing in a system for distributed databases (SDD-1)", *ACM Transactions on Database Systems (TODS)*, v. 6, 4, pp. 602-625.

BOAG, S., CHAMBERLIN, D., FERNÁNDEZ, M., et al., 2007, "XQuery 1.0: An XML Query Language - W3C Recommendation 23 January 2007". Disponível em: http://www.w3.org/TR/xquery/, acessado em 12/03/2007.

BOOTH, D., HAAS, H., MCCABE, F., et al., 2004, "Web Services Architecture". Disponível em: http://www.w3.org/TR/ws-arch, acessado em 30/01/2007.

BRAGANHOLO, V., DAVIDSON, S., HEUSER, C., 2004, "From XML View Updates to Relational View Updates: old solutions to a new problem". In: VLDB - International Conference on Very Large Data Bases, pp. 276-287, Toronto, Canada.

BREMER, J.-M., GERTZ, M., 2003, "On Distributing XML Repositories". In: International Workshop on Web and Databases, WebDB, pp. 73-78, San Diego, California.

BUNEMAN, P., 1997, "Semistructured Data". In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 117-121, ACM Press, Tucson, Arizona.

CAMILLO, S., MELLO, R., HEUSER, C., 2003, "Querying Heterogeneous XML Sources through a Conceptual Schema". In: CONCEPTUAL MODELING - ER 2003, pp. 186-199, Chicago, USA.

CAREY, M., FLORESCU, D., ZACHARY, I., et al., 2000, "XPERANTO: Publishing Object-Relational Data as XML". In: WebDB - International Workshop on the Web and Databases, pp. 105-110.

CHEN, Z., JAGADISH, H. V., LAKSHMANAN, L. V. S., et al., 2003, "From Tree Patterns to Generalized Tree Patterns: On Efficient Evaluation of XQuery". In: VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, pp. 237-248, Berlin, Germany.

DAVIS, D., PARASHAR, M. P., 2002, "Latency Performance of SOAP Implementations". In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 407-412, IEEE Computer Society.

ELMASRI, R., NAVATHE, S., 2001, "Fundamentals of Database Systems". 3 ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

EXIST DEVTEAM, 2006, "eXist: Open Source Native XML Database", v. 1.1. Disponível em http://exist.sourceforge.net/.

FERNÁNDEZ, M., KADISYSKA, Y., SUCIU, D., et al., 2002, "SilkRoute: A Framework for Publishing Relational Data in XML". In: ACM Transactions on Database Systems (TODS), v. 27, pp. 438-493.

FERNÁNDEZ, M., SIMÉON, J., WADLER, P., 2000, "An Algebra for XML Query". In: FST TCS 2000: Proceedings of the 20th Conference on Foundations of

Software Technology and Theoretical Computer Science, pp. 11-45, Springer-Verlag, London, UK.

FIEBIG, T., HELMER, S., KANNE, C., et al., 2002, "Anatomy of a native XML base management system", *The VLDB Journal*, v. 11, 4, pp. 292-314.

FIGUEIREDO, G., BRAGANHOLO, V., MATTOSO, M., 2005, "Consultas sobre visões XML globais de Bases de Dados Distribuídas". In: WTDBD - Workshop de Teses e Dissertações em Banco de Dados, v. I, Uberlândia, MG, Brazil.

FRASINCAR, F., HOUBEN, G.-J., PAU, C., 2002, "XAL: an algebra for XML query optimization". In: ADC '02: Proceedings of the 13th Australasian database conference, pp. 49-56, Australian Computer Society, Inc., Melbourne, Victoria, Australia.

GARDARIN, G., MENSCH, A., DANG-NGOC, T.-T., et al., 2002, "Integrating Heterogeneous Data Sources with XML and XQuery". In: Proceedings of the 13th International Workshop on Database and Expert Systems Applications, pp. 839-846, IEEE Computer Society.

GUPTA, N., HARITSA, J., RAMANATH, M., 2000, "Distributed Query Processing on the Web". In: Proceedings of the 16th International Conference on Data Engineering, pp. 1-20, IEEE Computer Society.

HAAS, L., KOSSMANN, D., WIMMERS, E., et al., 1997, "Optimizing Queries Across Diverse Data Sources", *Proceedings of the Twenty-third International Conference on Very Large Databases*, Athens, Greece, VLDB Endowment, Saratoga, Calif.

IVES, Z. G., HALEVY, A. Y., WELD, D. S., 2002, "An XML query engine for network-bound data", *The VLDB Journal*, v. 11, 4, pp. 380-402.

JAGADISH, H. V., AL-KHALIFA, S., CHAPMAN, A., et al., 2002, "TIMBER: A native XML database", *VLDB Journal*, v. 11, 4, pp. 274-291.

JAGADISH, H. V., LAKSHMANAN, L. V. S., SRIVASTAVA, D., et al., 2001, "TAX: A Tree Algebra for XML". In: Database Programming Languages, 8th International Workshop, DBPL, pp. 149-164.

JOSIFOVSKI, V., RISCH, T., 2002, "Query decomposition for a distributed object-oriented mediator system". In: Distributed and Parallel Databases, v. 11, pp. 307-336.

JOSIFOVSKI, V., FONTOURA, M., BARTA, A., 2005, "Querying XML streams", *The VLDB Journal*, v. 14, 2, pp. 197-210.

JURIC, M. B., KEZMAH, B., HERICKO, M., et al., 2004, "Java RMI, RMI tunneling and Web services comparison and performance analysis". In: SIGPLAN, v. 39, pp. 58-65, ACM Press.

JURIC, M. B., ROZMAN, I., BRUMEN, B., et al., 2006, "Comparison of performance of web services, WS-security, RMI, and RMI-SSL", *Journal of Systems and Software*, v. 79, 5, pp. 689-700.

KOSSMAN, D., 2000, "The State of the Art in Distributed Query Processing". In: ACM Computing Surveys, v. 32, pp. 422-469.

KOSSMAN, D., STOCKER, K., 2000, "Iterative dynamic programming: a new class of query optimization algorithms". In: ACM Transactions on Database Systems (TODS), v. 25, pp. 43-82.

LEE, K., MIN, J., PARK, K., et al., 2002, "A Design and Implementation of XML-Based Mediation Framework (XMF) for Integration of Internet Information Resources". In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, v. 7, pp. 202-211, IEEE Computer Society.

MA, H., SCHEWE, K.-D., 2003, "Fragmentation of XML documents". In: XVIII Simpósio Brasileiro de Banco de Dados, pp. 200-214, Manaus, AM, Brasil.

MEIER, W., 2002, "eXist: An Open Source Native XML Database". In: Web, Web-Services, and Database Systems, v. 2593, pp. 169-183, Springer, Erfurt, Germany.

ODBMS.ORG, 2006, "4th Generation Standard for Object Databases on its Way". Disponível em: http://www.odbms.org/about_news_20060218.html, acessado em 21/1/2007.

OMG, 2004, "CORBA: Common Object Request Broker Architecture". Disponível em: http://www.omg.org/technology/documents/formal/corba_2.htm, acessado em 30/01/2007.

ÖZSU, M. T., VALDURIEZ, P., 1999, "Principles of Distributed Database Systems". 2 ed., Prentice Hall

PAPARIZOS, S., WU, Y., LAKSHMANAN, L. V. S., et al., 2004, "Tree Logical Classes for Efficient Evaluation of XQuery". In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, pp. 71-82, ACM.

RE, C., BRINKLEY, J., HINSHAW, K. P., et al., 2004, "Distributed XQuery". In: Proceedings of VLDB Workshop on Information Integration on the Web (IIWeb), pp. 116-121, Toronto, Canada.

ROTH, M. T., OZCAN, F., HAAS, L., 1999, "Cost Models DO Matter: Providing Cost Information for Diverse Data Sources in a Federated System". In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 599-610, Morgan Kaufmann Publishers Inc.

RUBERG, N., RUBERG, G., MATTOSO, M., 2003, "Digging Database Statistics and Costs Parameters for Distributed Query Processing". In: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, pp. 301-318, Springer.

SAXONICA LIMITED, 2006, "Open Source SAXON XSLT Processor, v.8.8". Disponível em http://saxon.sourceforge.net/.

SCHONING, H., 2001, "Tamino - A DBMS designed for XML". In: Proceedings of the 17th International Conference on Data Engineering, pp. 149-154, IEEE Computer Society, Washington, DC, USA.

SHANMUGASUNDARAM, J., SHEKITA, E., BARR, R., et al., 2000, "Efficiently Publishing Relational Data as XML Documents". In: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 65-76, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

SILVEIRA, F. V., HEUSER, C., 2005, "Decomposição de Consultas sobre Múltiplas Fontes XML". In: I Escola Regional de Banco de Dados, Porto Alegre, RS, Brasil.

SMILJANI´, M., FENG, L., JONKER, W., 2003, "Web-Based Distributed XML Query Processing", *Intelligent Search on XML Data*, chapter 14, Springer.

STONEBRAKER, M., 1986, "The design and implementation of distributed INGRES", *The INGRES papers: anatomy of a relational database system*, Boston, MA, USA, Addison-Wesley Longman Publishing Co.,Inc.

SUCIU, D., 2002, "Distributed Query Evaluation on Semistructured Data", *ACM Transactions on Database Systems*, v. 27, 1, pp. 1-62.

SUN MICROSYSTEMS, 2006a, "Java Compiler Compiler 4.0 (JavaCC)". Disponível em https://javacc.dev.java.net/.

SUN MICROSYSTEMS, 2006b, "Java Remote Method Invocation - Distributed Computing for Java". Disponível em: http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper/index.jsp, acessado em 30/01/2007b.

THE APACHE SOFTWARE FOUNDATION, 2006, "Apache Tomcat 5.5". Disponível em http://tomcat.apache.org/.

THE GALAX TEAM, 2006, "Galax: An implementation of XQuery", v. 0.6.5. Disponível em http://www.galaxquery.org/.

W3C, W. W. W. C., 2006, "Extensible Markup Language (XML) 1.0". Disponível em: http://www.w3.org/TR/REC-xml/, acessado em 20/01/2007.

WIEDERHOLD, G., 1992, "Mediators in the Architecture of Future Information Systems". In Michael N.Huhns and Munindar P.Singh, *Readings in Agents*, San Francisco, CA, USA, Morgan Kaufmann.

WILLIAMS, R., DANIELS, D., HAAS, L., et al., 1986, "R*: an overview of the architecture", *Distributed systems, Vol. II: distributed data base systems*, Artech House, Inc.

YAO, B., ÖZSU, M. T., KEENLEYSIDE, J., 2002, "XBench: A Family of Benchmarks for XML DBMSs". In: In Proceedings of EEXTT 2002.

ZHANG, N., HAAS, P. J., JOSIFOVSKI, V., et al., 2005, "Statistical Learning Techniques for Costing XML Queries". In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 289-300, ACM, Trondheim, Norway.

ZHANG, X., PIELECH, B., RUNDESNTEINER, E., 2002, "Honey, I shrunk the XQuery!: an XML algebra optimization approach". In: WIDM '02: Proceedings of the 4th international workshop on Web information and data management, pp. 15-22, ACM Press, New York, NY, USA.

Anexo I Consultas XQuery utilizadas nos experimentos

CLoja_bns_c01.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja
	for $a in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $a/Secao = "CD"
	return
	<loja>
	{ $a/Nome }
	{ $x/Secoes }
	</loja>
	}
	</results>

CLoja_bps_c02.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Lancamento = "T"
	order by $x/Codigo
	return
	<lancamento_t>
	{ $x }
	</lancamento_t>
	}
	</results>

CLoja_bts_c03.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "CD"
	return
	<output>
	{ $x/Nome }
	</output>
	}
	</results>

CLoja_bts_c04.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "Livraria"
	return
	<output>
	{ $x/Nome }
	</output>
	}
	</results>

CLoja_bts_c05.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "CD"
	and $x/Lancamento = "T"
	return
	<output>
	{ $x/Nome }
	</output>
	}
	</results>

CLoja_bts_c06.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "Perfumaria"
	return
	<output>
	{ $x/Nome }
	{ $x/Preco }
	</output>
	}
	</results>

CLoja_bpc_c07.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where count($x/Caracteristica) >= 4
	order by $x/Codigo
	return
	<output>
	{ $x }
	</output>
	}
	</results>

CLoja_btc_c08.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "CD"
	and count($x/Caracteristica) >= 4
	return
	<output>
	{ $x }
	</output>
	}
	</results>

CLoja_bps_c09.xq	<results>
	{
	for $x in
	collection('Cloja_c?.xml')/Loja/Funcionarios/Funcionario
	return
	<output>
	{ $x }
	</output>
	}
	</results>

CLoja_bpc_c10.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Funcionarios
	return
	<TotalPagamento>
	{ sum($x/Funcionario/Salario) }
	</TotalPagamento>
	}
	</results>

CLoja_bps_c11.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja
	return
	<loja>
	{ $x/Funcionarios }
	</loja>
	}
	</results>

CLoja_bts_c12.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "Brinquedos"
	return
	<output>
	{ $x/Nome }
	{ $x/Preco }
	</output>
	}
	</results>

CLoja_bts_c13.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "Brinquedos"
	and $x/Preco > 50
	return
	<output>
	{ $x/Nome }
	{ $x/Preco }
	</output>
	}
	</results>

CLoja_bts_c14.xq	<results>
	{
	for $x in collection('Cloja_c?.xml')/Loja/Itens/Item
	where $x/Secao = "Perfumaria"
	and $x/Preco > 40
	return
	<output>
	{ $x/Nome }
	{ $x/Preco }
	</output>
	}
	</results>

COrders_bns_c01.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/@id = "1"
	return
	<order>
	{ $order }
	</order>
	}
	</results>

COrders_bns_c02.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where $a/@id = "3"
	return
	<items>
	{ $a//order_line/item_id }
	</items>
	}
	</results>

COrders_bts_c03.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where $a/total > 11000
	order by $a/ship_type, $a/@id
	return
	<Output>
	{$a/@id}
	{$a/order_date}
	{$a/ship_type}
	</Output>
	}
	</results>

COrders_bts_c04.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where $a/total > 11000.0
	order by $a/total descending, $a/@id
	return
	<Output>
	{$a/@id}
	{$a/order_date}
	{$a/total}
	</Output>
	}
	</results>

COrders_bns_c05.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where $a/@id = "5"
	return
	<Output>
	{$a/order_lines}
	</Output>
	}
	</results>

COrders_bns_c06.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where count($a/order_lines/order_line) = 1
	order by $a/@id
	return
	<Output>
	{$a/@id}
	</Output>
	}
	</results>

COrders_bns_c07.xq	<results>
	{
	
	for $a in collection('Corders_c?.xml')/order
	where $a/@id = "6"
	return
	<Output>
	{$a}
	</Output>
	}
	</results>

COrders_bpc_c08.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	let $l := $order/order_lines/order_line
	where $order/total > 7000
	and count($l) >= 5
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	<total_items>
	{ count($l) }
	</total_items>
	</order>
	}
	</results>

COrders_bps_c09.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/total > 7000
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>

COrders_bts_c10.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/total > 10000
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>

COrders_bts_c11.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/total > 10000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>

COrders_bps_c12.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/total > 7000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>

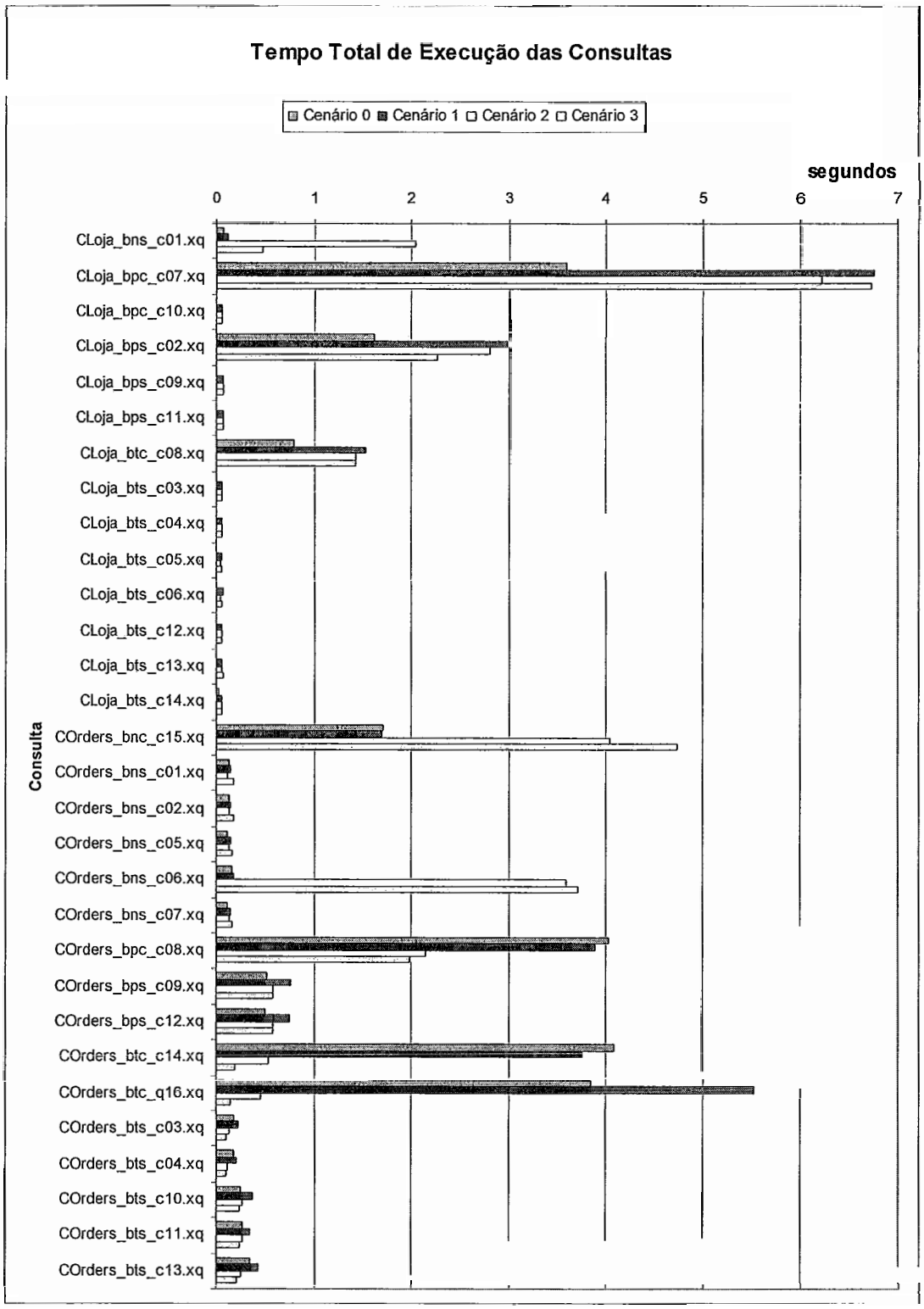
COrders_bts_c13.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	where $order/total > 7000
	and $order/total < 8000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>

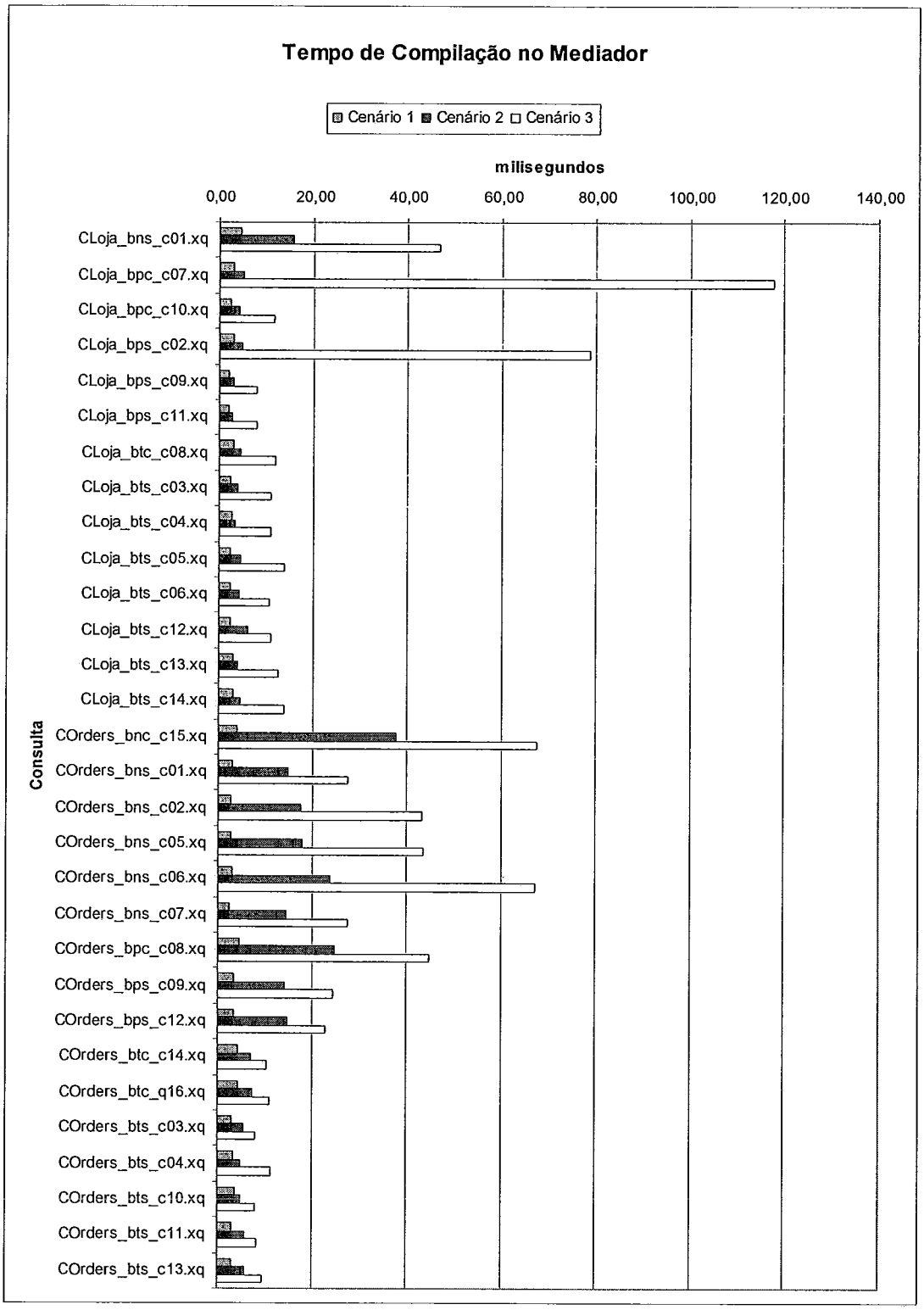
COrders_btc_c14.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	let $l := $order/order_lines/order_line
	where $order/total < 2000
	and count($l) >= 5
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	<total_items>
	{ count($l) }
	</total_items>
	</order>
	}
	</results>

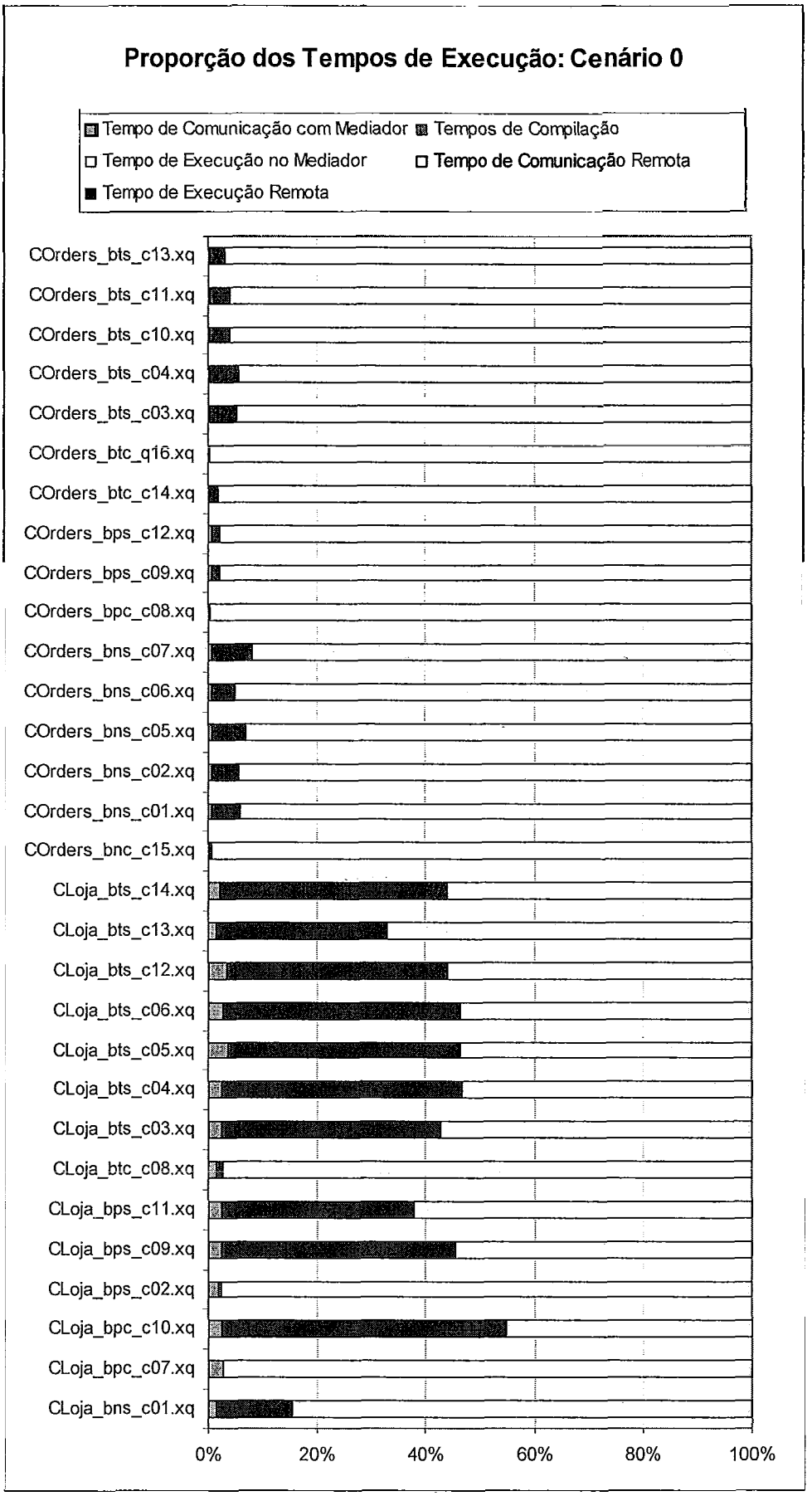
COrders_bnc_c15.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	let $l := $order/order_lines/order_line
	where count($l) >= 5
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	<total_items>
	{ count($l) }
	</total_items>
	</order>
	}
	</results>

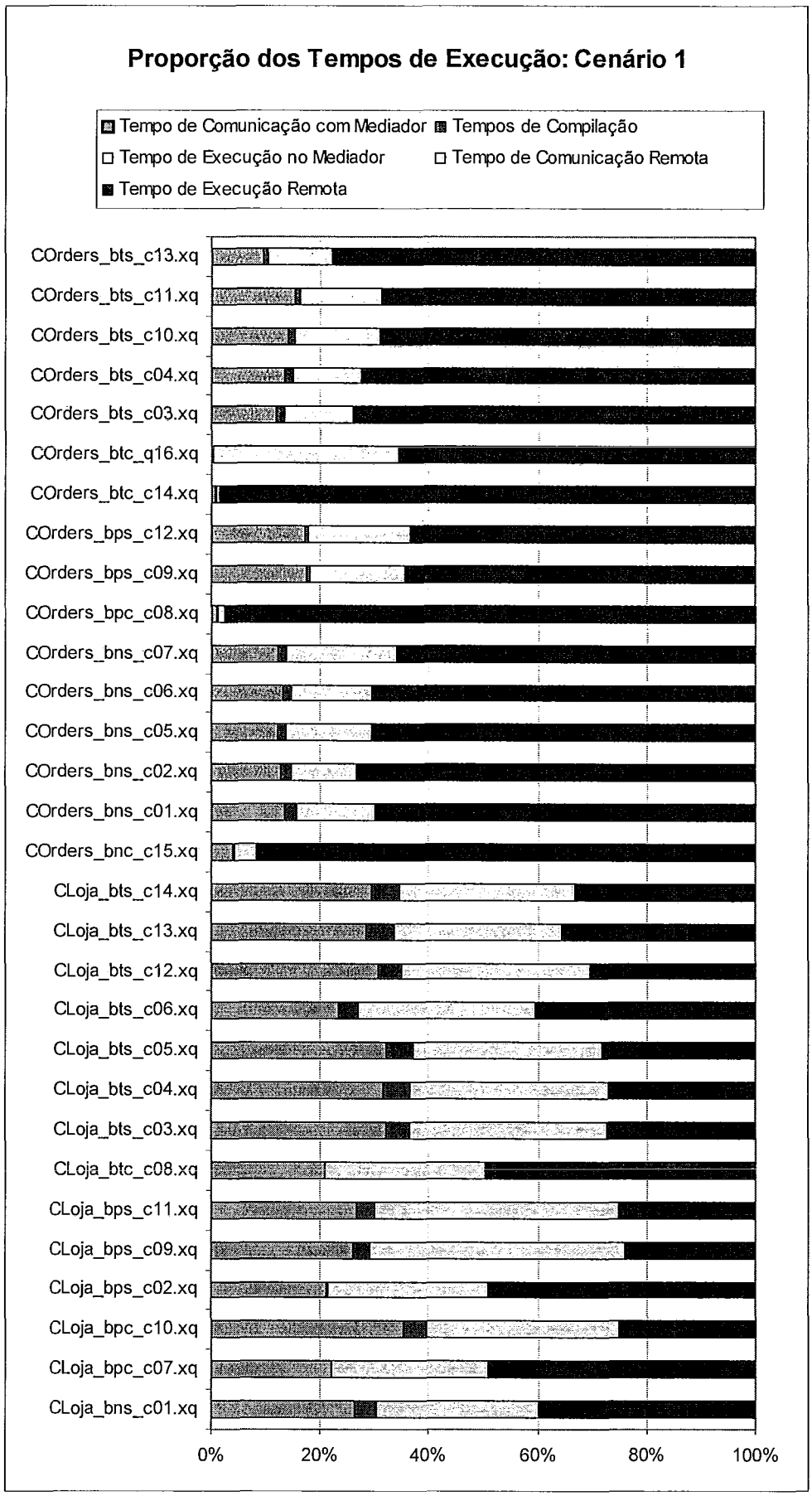
COrders_btc_q16.xq	<results>
	{
	for $order in collection('Corders_c?.xml')/order
	let $l := $order/order_lines/order_line
	where $order/total > 11000
	and count($l) >= 5
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	<total_items>
	{ count($l) }
	</total_items>
	</order>
	}
	</results>

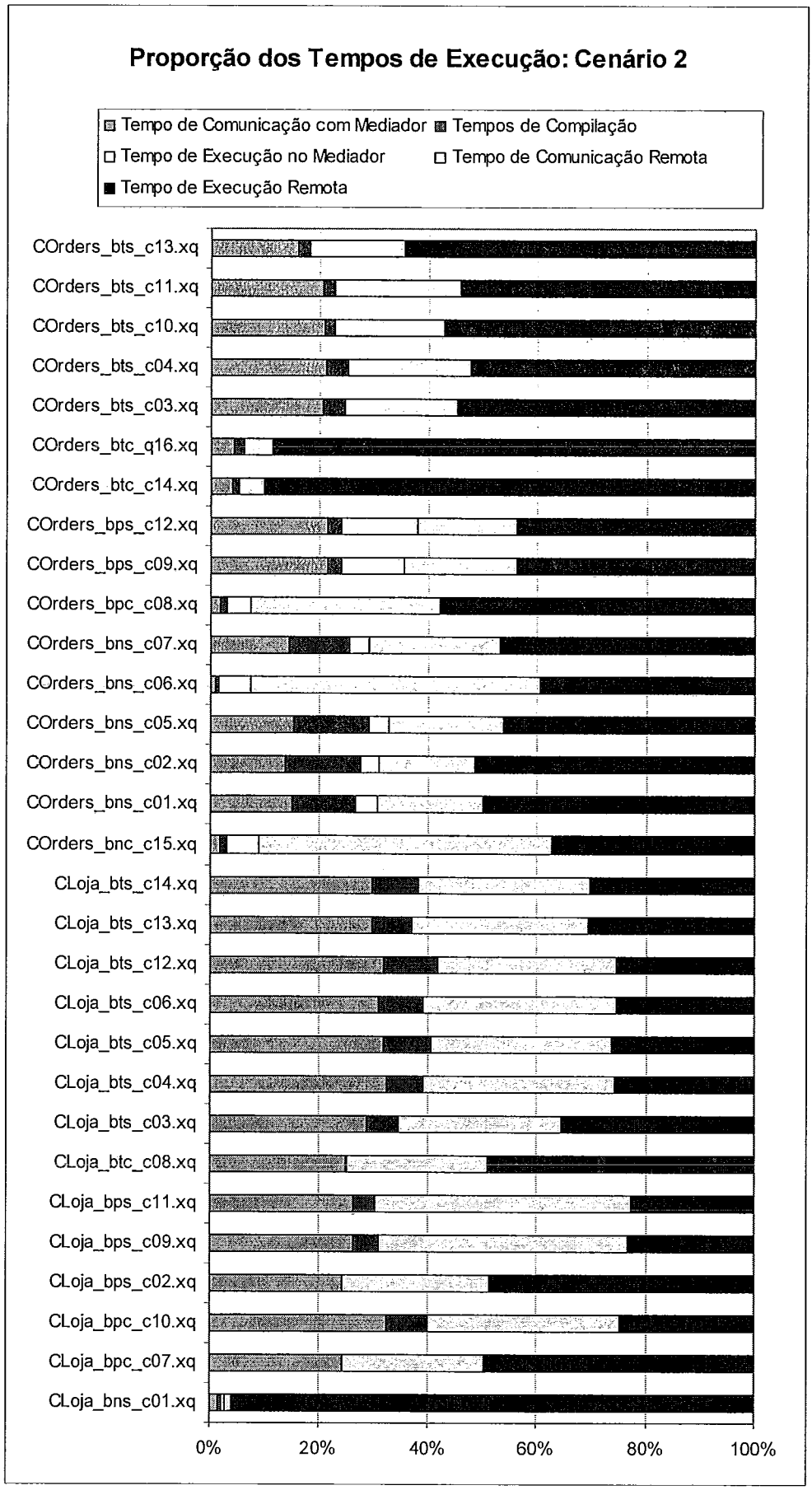
Anexo II Tempos de Execução das Consultas

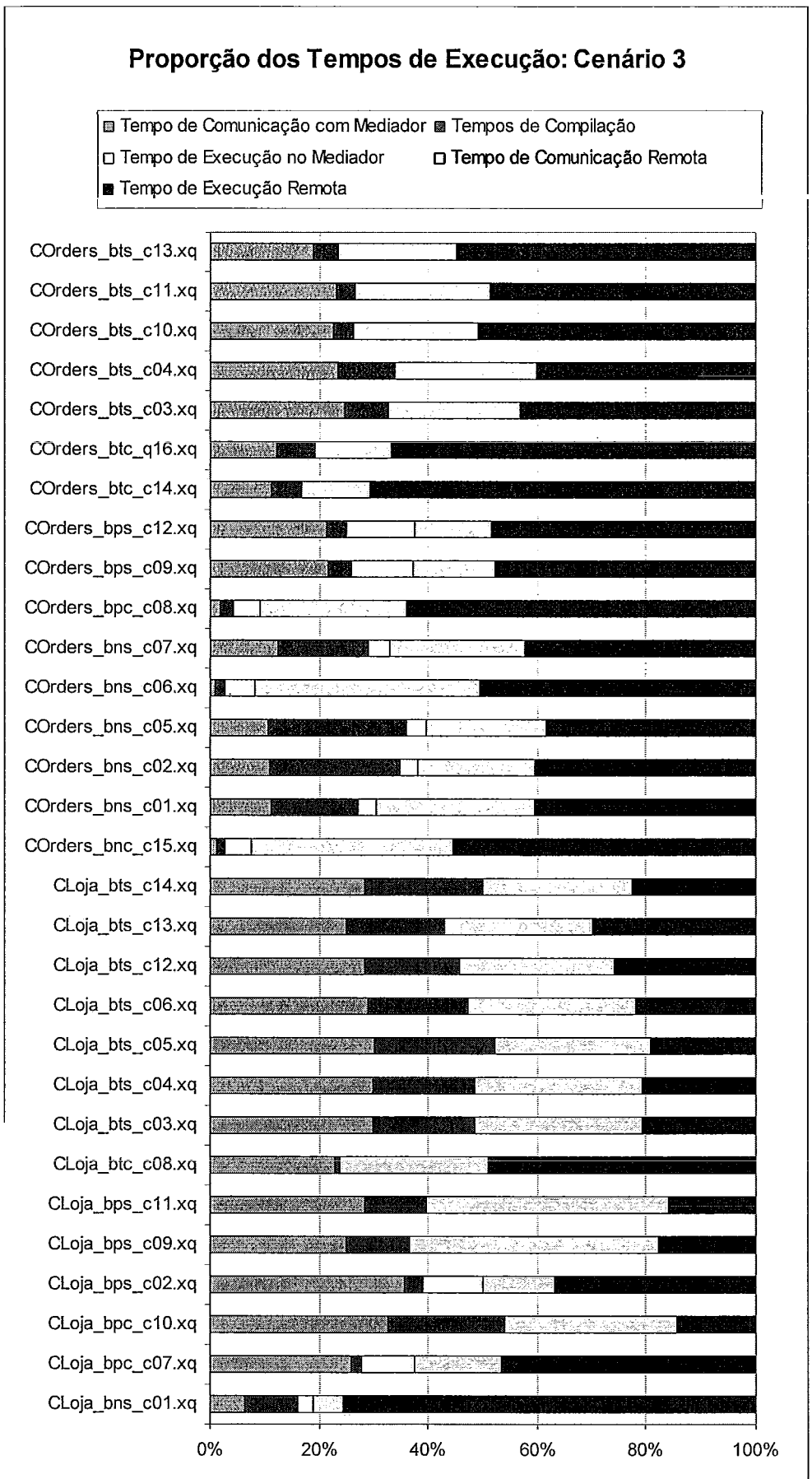



Tempo de Compilação no Mediador
Cenário 1
Cenário 2
Cenário 3
milisegundos
0,00
20,00
40,00
60,00
80,00
100,00
120,00
140,00
Consulta
CLoja_bns_c01.xq
CLoja_bpc_c07.xq
CLoja_bpc_c10.xq
CLoja_bps_c02.xq
CLoja_bps_c09.xq
CLoja_bps_c11.xq
CLoja_btc_c08.xq
CLoja_bts_c03.xq
CLoja_bts_c04.xq
CLoja_bts_c05.xq
CLoja_bts_c06.xq
CLoja_bts_c12.xq
CLoja_bts_c13.xq
CLoja_bts_c14.xq
COrders_bnc_c15.xq
COrders_bns_c01.xq
COrders_bns_c02.xq
COrders_bns_c05.xq
COrders_bns_c06.xq
COrders_bns_c07.xq
COrders_bpc_c08.xq
COrders_bps_c09.xq
COrders_bps_c12.xq
COrders_btc_c14.xq
COrders_btc_q16.xq
COrders_bts_c03.xq
COrders_bts_c04.xq
COrders_bts_c10.xq
COrders_bts_c11.xq
COrders_bts_c13.xq


Proporção dos Tempos de Execução: Cenário 0
Tempo de Comunicação com Mediador
Tempos de Compilação
Tempo de Execução no Mediador
Tempo de Comunicação Remota
Tempo de Execução Remota
COrders_bts_c13.xq
COrders_bts_c11.xq
COrders_bts_c10.xq
COrders_bts_c04.xq
COrders_bts_c03.xq
COrders_btc_q16.xq
COrders_btc_c14.xq
COrders_bps_c12.xq
COrders_bps_c09.xq
COrders_bpc_c08.xq
COrders_bns_c07.xq
COrders_bns_c06.xq
COrders_bns_c05.xq
COrders_bns_c02.xq
COrders_bns_c01.xq
COrders_bnc_c15.xq
CLoja_bts_c14.xq
CLoja_bts_c13.xq
CLoja_bts_c12.xq
CLoja_bts_c06.xq
CLoja_bts_c05.xq
CLoja_bts_c04.xq
CLoja_bts_c03.xq
CLoja_btc_c08.xq
CLoja_bps_c11.xq
CLoja_bps_c09.xq
CLoja_bps_c02.xq
CLoja_bpc_c10.xq
CLoja_bpc_c07.xq
CLoja_bns_c01.xq
0%
20%
40%
60%
80%
100%


Proporção dos Tempos de Execução: Cenário 1
Tempo de Comunicação com Mediador
Tempos de Compilação
Tempo de Execução no Mediador
Tempo de Comunicação Remota
Tempo de Execução Remota
COrders_bts_c13.xq
COrders_bts_c11.xq
COrders_bts_c10.xq
COrders_bts_c04.xq
COrders_bts_c03.xq
COrders_btc_q16.xq
COrders_btc_c14.xq
COrders_bps_c12.xq
COrders_bps_c09.xq
COrders_bpc_c08.xq
COrders_bns_c07.xq
COrders_bns_c06.xq
COrders_bns_c05.xq
COrders_bns_c02.xq
COrders_bns_c01.xq
COrders_bnc_c15.xq
CLoja_bts_c14.xq
CLoja_bts_c13.xq
CLoja_bts_c12.xq
CLoja_bts_c06.xq
CLoja_bts_c05.xq
CLoja_bts_c04.xq
CLoja_bts_c03.xq
CLoja_btc_c08.xq
CLoja_bps_c11.xq
CLoja_bps_c09.xq
CLoja_bps_c02.xq
CLoja_bpc_c10.xq
CLoja_bpc_c07.xq
CLoja_bns_c01.xq
0%
20%
40%
60%
80%
100%


Proporção dos Tempos de Execução: Cenário 2
Tempo de Comunicação com Mediador
Tempos de Compilação
Tempo de Execução no Mediador
Tempo de Comunicação Remota
Tempo de Execução Remota
COrders_bts_c13.xq
COrders_bts_c11.xq
COrders_bts_c10.xq
COrders_bts_c04.xq
COrders_bts_c03.xq
COrders_btc_q16.xq
COrders_btc_c14.xq
COrders_bps_c12.xq
COrders_bps_c09.xq
COrders_bpc_c08.xq
COrders_bns_c07.xq
COrders_bns_c06.xq
COrders_bns_c05.xq
COrders_bns_c02.xq
COrders_bns_c01.xq
COrders_bnc_c15.xq
CLoja_bts_c14.xq
CLoja_bts_c13.xq
CLoja_bts_c12.xq
CLoja_bts_c06.xq
CLoja_bts_c05.xq
CLoja_bts_c04.xq
CLoja_bts_c03.xq
CLoja_btc_c08.xq
CLoja_bps_c11.xq
CLoja_bps_c09.xq
CLoja_bps_c02.xq
CLoja_bpc_c10.xq
CLoja_bpc_c07.xq
CLoja_bns_c01.xq
0%
20%
40%
60%
80%
100%


Proporção dos Tempos de Execução: Cenário 3
Tempo de Comunicação com Mediador
Tempos de Compilação
Tempo de Execução no Mediador
Tempo de Comunicação Remota
Tempo de Execução Remota
COrders_bts_c13.xq
COrders_bts_c11.xq
COrders_bts_c10.xq
COrders_bts_c04.xq
COrders_bts_c03.xq
COrders_btc_q16.xq
COrders_btc_c14.xq
COrders_bps_c12.xq
COrders_bps_c09.xq
COrders_bpc_c08.xq
COrders_bns_c07.xq
COrders_bns_c06.xq
COrders_bns_c05.xq
COrders_bns_c02.xq
COrders_bns_c01.xq
COrders_bnc_c15.xq
CLoja_bts_c14.xq
CLoja_bts_c13.xq
CLoja_bts_c12.xq
CLoja_bts_c06.xq
CLoja_bts_c05.xq
CLoja_bts_c04.xq
CLoja_bts_c03.xq
CLoja_btc_c08.xq
CLoja_bps_c11.xq
CLoja_bps_c09.xq
CLoja_bps_c02.xq
CLoja_bpc_c10.xq
CLoja_bpc_c07.xq
CLoja_bns_c01.xq
0%
20%
40%
60%
80%
100%