COPPE/UFRJ

PROCESSAMENTO DE CONSULTAS SOBRE BASES XML DISTRIBUÍDAS EM UM AMBIENTE PEER-TO-PEER

Clarissa Netto Vilela

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador(es): Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

Rio de Janeiro
Julho de 2010

PROCESSAMENTO DE CONSULTAS SOBRE BASES XML DISTRIBUÍDAS EM UM AMBIENTE PEER-TO-PEER

Clarissa Netto Vilela

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

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

_____
Profa. Vanessa Braganholo Murta, D.Sc.

_____
Profa. Ana Carolina Brandao Salgado, Ph.D.

_____
Prof. Alexandre de Assis Bento Lima, D.Sc

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 2010

Vilela, Clarissa Netto

Processamento de Consultas sobre Bases XML Distribuídas em um Ambiente Peer-to-Peer/ Clarissa Netto Vilela. – Rio de Janeiro: UFRJ/COPPE, 2010.

XII, 111 p.: il.; 29,7 cm.

Orientadores: Marta Mattoso de Queirós Lima

Vanessa Braganholo Murta

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2010.

Referencias Bibliográficas: p. 103-107.

1. Banco de dados distribuídos. 2. Processamento de consultas. 3. Banco de dados XML. 4. Ambiente peer-to-peer. I. Mattoso, Marta Lima de Queirós *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

À minha família!

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 EM UM AMBIENTE PEER-TO-PEER

Clarissa Netto Vilela

Julho/2010

Orientadores: Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

Programa: Engenharia de Sistemas e Computação

O gerenciamento de dados em ambientes *Peer-to-Peer* (P2P) representa uma tarefa complexa e desafiante devido à natureza dinâmica da abordagem P2P e à presença de dados complexos, principalmente descritos no padrão XML. Por questões de escalabilidade e com o objetivo de preservar a autonomia de um ambiente P2P, um dos maiores objetivos de um sistema P2P é permitir que usuários realizem consultas de forma transparente sobre os diversos pontos distribuídos pela rede. Um sistema P2P deve lidar com importantes questões, tais como a localização dos dados relevantes e o processamento de consultas, onde o sistema P2P deve ser capaz de descobrir os pontos que podem contribuir com dados relevantes à execução da consulta. Para contribuir para a solução deste problema, esta dissertação apresenta uma metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas em um ambiente P2P, inspirada na metodologia proposta por Figueiredo (2007), que contempla as etapas de decomposição da consulta submetida, incluindo sua representação na álgebra TLC; localização dos dados; otimização global; execução e consolidação dos resultados. Para implementar a metodologia proposta, estendemos a arquitetura proposta por Figueiredo (2007) baseada em um Mediador com Adaptadores. O Mediador realiza as modificações necessárias na consulta original de forma a tornar a distribuição das bases de dados transparente ao usuário. Diferentes técnicas para a etapa de localização dos dados relevantes foram implementadas no protótipo, com o objetivo de avaliar experimentalmente a nossa metodologia de forma a determinar qual técnica melhor se adapta para o processamento de consultas XQuery distribuídas em um ambiente P2P.

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 ON A PEER-TO-PEER APPROACH

Clarissa Netto Vilela

July/2010

Advisors: Marta Lima de Queirós Mattoso
Vanessa Braganholo Murta

Department: System Engineering and Computer Science

Data management in P2P systems is a challenging and difficult problem considering the dynamic nature of P2P networks and the presence of complex data, particularly described in the XML standard. For scalability reasons and in order to preserve the autonomy of a P2P environment, one of the major goals in a P2P system is to enable users to pose queries in a transparent way over the distributed peers in the P2P network. A P2P system should deal with important issues, such as relevant data localization and query processing, where a P2P system should be able to find the peers that can contribute with relevant data to the query execution. To contribute to the solution of this problem, this work presents a methodology for XQuery query processing over distributed XML databases on a P2P approach, inspired by the methodology proposed by Figueiredo (2007), 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. To implement the proposed methodology, we extend the architecture proposed by Figueiredo (2007) based on a Mediator with Wrappers. The Mediator performs all the necessary modifications in the original query in order to make the data distribution transparent to the user. Different techniques for the step of localization of relevant data have been implemented in the prototype, in order to experimentally evaluate our methodology to determine which technique is best suited for XQuery query processing over distributed XML databases on a P2P approach.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Siglas

APT – ***A**nnotaded **P**attern **T**ree*

DHT – ***D**istributed **H**ash **T**able*

LCL – ***L**ogical **C**lass **L**abels*

P2P – ***P**eer-to-**P**eer*

SON – ***S**emantic **O**lervay **N**etwork*

TLC - ***T**ree **L**ogical **C**lasses*

TTL – ***T**ime-**t**o-**L**ive*

URL - ***U**niform **R**esource **L**ocater*

URN - ***U**niform **R**esource **N**ame*

W3C - ***W**orld **W**ide **W**eb Consortium*

XML - *e**X**tensible **M**arkup **L**anguage*

Capítulo 1. Introdução

1.1. Motivação

A distribuição em banco de dados é um assunto bastante estudado na literatura desde a década de 1970, quando surgiram os primeiros protótipos (EPSTEIN et al., 1978; BERNSTEIN et al., 1981; WILLIAMS et al., 1986; STONEBRAKER, 1986). Com o desenvolvimento de novas tecnologias para comunicação e transporte dos dados, o processamento distribuído é mais viável e cada vez mais necessário atualmente, visto que diminui o alto custo dos sistemas centralizados, aumentando a escalabilidade e a disponibilidade dos mesmos, além de promover a integração dos dados (ÖZSU et al., 1999; KOSSMAN, 2000).

No entanto, com o advento da Internet surgem novos desafios na área de processamento de dados distribuídos. As informações se encontram atualmente distribuídas pela *Web* através de múltiplas fontes de dados, principalmente descritas no padrão XML (*eXtensible Markup Language*) (BRAY et al., 2008), o qual é amplamente utilizado para representação e troca de informações na Internet, sendo definido e recomendado pela W3C (*World Wide Web Consortium*) (W3C, 2009). Entretanto, combinar todas estas informações não representa uma tarefa fácil. A grande heterogeneidade e autonomia destas bases de dados distribuídas, assim como a dimensão da Internet, dificultam as atividades de consulta.

Dentro deste contexto, a tecnologia *Peer-to-Peer* (P2P) vem se destacando como uma das grandes topologias de sistemas distribuídos da atualidade. Um sistema P2P inclui uma grande rede heterogênea, autônoma, dinâmica e escalável, no qual pontos (*peers*) podem trocar e compartilhar dados e serviços de maneira descentralizada (ZHUGE et al., 2005).

O paradigma P2P estende as idéias existentes da arquitetura cliente-servidor e do modelo computacional distribuído, eliminando as diferenças entre o cliente e o servidor. A abordagem P2P se baseia nos conceitos de descentralização e compartilhamento de recursos, objetivando evitar gargalos em servidores centrais e distribuir a carga de trabalho. Cada ponto do sistema P2P pode ser considerado tanto servidor quanto

cliente, e fornece parte dos recursos e informações disponíveis no sistema (ABERER et al., 2002).

Sistemas P2P se destacam por diferentes vantagens (SARTIANI et al., 2004; BONIFATI et al., 2008), incluindo (1) escalabilidade em termos do número de pontos e sua distribuição, (2) simples implantação, pois um ponto é facilmente configurável sem precisar de uma infra-estrutura especial para se conectar na rede, (3) tolerância a falha, visto que o sistema continua a funcionar mesmo com algum ponto da rede P2P indisponível, dentre outras.

Por questões de escalabilidade e com o objetivo de preservar a autonomia de um ambiente P2P, um dos maiores objetivos de um sistema P2P é permitir que usuários realizem consultas de forma transparente sobre os diversos pontos da rede. Do ponto de vista de gerenciamento de dados, um sistema P2P deve lidar com importantes questões (SUNG et al., 2005), tais como (1) a localização dos dados, onde os pontos devem ser capazes de referenciar e localizar dados armazenados nos demais pontos da rede P2P; e (2) o processamento de consultas, onde o sistema P2P deve ser capaz de descobrir os pontos que podem contribuir com dados relevantes à execução da consulta a fim de processá-la eficientemente; dentre outras questões.

Embora o problema de otimização e execução de consultas em ambientes com bases de dados distribuídos seja bem conhecido e estudado, novos desafios são apresentados no processamento de consultas distribuídas em ambientes P2P (PAPADIMOS et al., 2003; KOLONIARI et al., 2005; CONFORTI et al., 2007), especialmente na presença de dados complexos e de consultas sofisticadas submetidas (FEGARAS et al., 2006). Entretanto, o processamento de consultas distribuídas sobre as fontes de dados mais estruturadas ainda permanece como um problema complexo, apesar dos esforços de pesquisa acadêmica e industrial nos últimos anos (RIZZOLO et al., 2001; ABITEBOUL et al., 2003; CHEN et al., 2004; FIGUEIREDO, 2007).

Muitos fatores influenciam o desempenho do processamento distribuído de consultas XML. Em particular, a localização eficiente dos dados e o roteamento das consultas têm um impacto significante nas ferramentas típicas para o processamento de consultas distribuídas em ambientes P2P. Além disso, outro ponto importante acerca do desempenho vem da complexidade dos padrões (e recomendações) de linguagens de consulta XML, como o XQuery (BOAG et al., 2007) e o XPath (BERGLUNG et al., 2007), os quais são muito poderosos e requerem algoritmos sofisticados.

Devido à natureza dinâmica da abordagem P2P, não se pode obter uma representação global dos dados distribuídos na rede P2P para o processamento eficiente das consultas submetidas. Ao contrário dos sistemas de bancos de dados distribuídos tradicionais, os quais possuem uma visão global do esquema de dados, no ambiente P2P, o problema consiste em direcionar as consultas distribuídas de um determinado ponto para um ou mais pontos vizinhos baseado no conhecimento parcial do esquema (BRUNKHORST et al., 2003). Além disso, a autonomia dos pontos impõe a construção dinâmica dos planos de execução de uma consulta, visto que o número de pontos participantes e os dados disponíveis em uma rede P2P mudam de acordo com o tempo (YING et al., 2005).

A título de ilustração, considere uma rede de sebos e leitores como o exemplo de um possível sistema P2P. Este cenário é composto por pontos que compartilham o mesmo domínio de conhecimento, expressando suas afinidades específicas para pesquisar, armazenar, replicar ou indexar seus dados. As fontes dos pontos disponíveis podem ter interseções ou complementações dos dados a fim de interagir e produzir resultados de consultas mais amplos e completos, sem intervir na autonomia de cada ponto.

Cada leitor pode comprar ou vender livros usados de sua área de interesse, sem intermediários e sem os papéis pré-definidos de comprador/vendedor (cliente/servidor). Ou seja, cada leitor disponibiliza e gerencia sua própria fonte de dados local que será compartilhada com os demais pontos da rede P2P, a qual pode ser considerada como um fragmento do que seria uma representação global dos dados da rede de sebos.

Para realizar uma transação dentro da rede de sebos, o leitor utiliza o sistema P2P para realizar suas consultas de forma rápida e transparente. Desta forma, o leitor submete suas consultas na rede P2P composta por diversas fontes de livros distribuídas e autônomas, pois seu conjunto local de livros pode não ser suficiente para responder suas consultas. Ou seja, o leitor busca respostas mais amplas, completas e significativas para as suas consultas submetidas.

Além disso, o leitor também pode desejar submeter consultas muito complexas, tal como a consulta que retorna os vendedores disponíveis que possuem os livros mais procurados da rede P2P, que envolvem critérios e condições diferentes combinando várias fontes de dados distintas. No entanto, quando uma consulta engloba fontes de dados distintas, a consulta deve ser submetida em cada fonte de dados localizada como

relevante e processada posteriormente, pois seu resultado final vai combinar os dados retornados dos leitores disponíveis da rede.

Uma possível alternativa para o processamento desta consulta submetida é possuir uma base de dados centralizada contendo réplicas das fontes de dados dos leitores disponíveis da rede de sebos. A alternativa se torna uma opção inviável, visto que os dados podem se tornar obsoletos, pois a consulta não será realizada em tempo real, além de desperdiçar tempo e largura de banda.

Com isso, a solução encontrada se resume em aplicar as técnicas de bancos de dados distribuídos para criar uma representação global dos dados distribuídos da rede de sebos, facilitando assim a consulta sobre os dados requisitados. A representação global virtual das bases de dados disponíveis na rede, chamada de *visão global*, passaria a ser consultada diretamente pelos usuários, tornando transparente a consulta sobre as bases individuais. Assim, as consultas submetidas sobre uma visão global seriam decompostas em sub-consultas e executadas sobre as visões locais das bases remotas dos pontos relevantes.

Para que o processamento distribuído da consulta seja automático e genérico, se torna necessário o uso de uma metodologia que defina etapas que possibilitem o processamento de consultas distribuídas. Neste contexto, Figueiredo (2007) apresenta uma metodologia para o processamento de consultas XQuery (BOAG et al., 2007) sobre bases de dados XML distribuídas e fragmentadas. Semelhante à metodologia utilizada para o processamento de consultas distribuídas sobre o modelo relacional (ÖZSU et al., 1999), a metodologia consiste nas etapas de decomposição da consulta, incluindo a representação da consulta em sua forma algébrica TLC (PAPARIZOS et al., 2004); localização dos dados; otimização global; e execução e consolidação dos resultados.

Para implementar a metodologia, Figueiredo (2007) propôs uma arquitetura baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992) acoplados aos bancos de dados XML remotos. 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 totalmente transparente ao usuário final. O Mediador contém informações sobre a distribuição das bases de dados XML armazenadas em um Catálogo, e realiza todo o processamento de uma consulta sobre a visão XML global dos dados para gerar sub-consultas destinadas aos Adaptadores. Cada Adaptador gerencia a execução de sua sub-consulta sobre a base de dados XML a ele acoplada, retornando o resultado ao Mediador. O Mediador

consolida todos os resultados retornados dos Adaptadores, e envia o resultado final ao usuário.

A abordagem proposta apresenta algumas limitações impostas pelo modelo de dados distribuídos, tais como (1) a topologia estática entre o Mediador e os Adaptadores e (2) a trabalhosa administração do banco para gerenciar as informações contidas no Catálogo. Mas o principal problema identificado na metodologia proposta por Figueiredo (2007) localiza-se no Mediador, o qual possui um ponto único de acesso centralizado para o processamento de consultas distribuídas. Todavia, estes problemas são facilmente solucionados pelo uso de uma abordagem P2P.

1.2. Objetivos

O principal objetivo dessa dissertação é contribuir para uma solução para o processamento de consultas XQuery sobre bases de dados XML distribuídas em um ambiente P2P. Para isso, propomos uma metodologia de processamento de consultas XQuery distribuídas, inspirada na metodologia proposta por Figueiredo (2007), que contempla as etapas de decomposição da consulta submetida, incluindo sua representação na álgebra TLC (PAPARIZOS et al., 2004); localização dos dados; otimização global; criação das sub-consultas e sua execução nas bases XML remotas.

Para implementar a metodologia proposta, estendemos a arquitetura proposta por Figueiredo (2007) baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992). O Mediador realiza todo o processamento de uma consulta XQuery submetida pelo usuário, gerando as sub-consultas destinadas aos Adaptadores. Cada Adaptador gerencia a execução de suas sub-consultas na base de dados XML a ele acoplada, retornando seus resultados ao Mediador, que consolida todos os resultados retornados e envia ao usuário final. Para eliminar o ponto único de acesso centralizado para o processamento de consultas distribuídas, os componentes da arquitetura proposta são distribuídos por todos os pontos disponíveis na rede P2P. Dessa forma, cada ponto da rede deve conter: (1) um Catálogo próprio, incluindo inicialmente somente as informações dos dados disponíveis de sua base local XML; (2) um Mediador, onde o ponto desempenha o papel de servidor para o processamento da consulta submetida; e (3) um Adaptador, onde o ponto é considerado como destino para a execução de sub-consultas.

O Mediador realiza as modificações necessárias na consulta original de forma a tornar a distribuição das bases de dados entre os pontos da rede transparente ao usuário final. A localização dos pontos que possuem dados relevantes para a consulta submetida pode ser realizada a partir de diversas técnicas levantadas: (1) através de uma mensagem de *broadcast* para um número limitado de pontos na rede; (2) através da utilização de um índice distribuído entre os pontos da rede P2P; ou (3) através das informações disponíveis no Catálogo atualizado do ponto solicitante da consulta.

Um protótipo da solução proposta foi desenvolvido, totalmente de acordo com as definições apresentadas, com o objetivo de avaliar experimentalmente a nossa metodologia e arquitetura. Testes e experimentos em laboratório são realizados visando analisar o desempenho da metodologia proposta e o conseqüente ganho em desempenho no processamento das consultas XQuery em relação às diversas técnicas levantadas para o ambiente P2P.

Em particular, uma avaliação é realizada entre as três técnicas de localização de dados a fim de verificar qual técnica melhor se adapta para o processamento de consultas distribuídas em ambientes P2P. Embora tais técnicas já tenham sido muito estudadas em redes P2P tradicionais, não temos conhecimento de estudos voltados para a aplicação de tais técnicas no processamento de consultas XQuery distribuídas sobre bases XML em um ambiente P2P. Com isso, também estamos contribuindo para identificar características das técnicas estudadas e de suas respectivas estratégias utilizadas, que colaboram para o melhor desempenho do sistema, visando descobrir pontos fracos e fortes que podem ajudar na adaptação e melhoria destas técnicas para o processamento de consultas XQuery distribuídas no ambiente P2P.

1.3. Organização dos Capítulos

Esta dissertação está organizada em outros seis capítulos, além deste primeiro capítulo de introdução.

O Capítulo 2 revisa alguns fundamentos teóricos sobre um ambiente P2P incluindo suas principais características, explicando os diferentes tipos de arquitetura, topologia e distribuição de dados. São apresentados diversos cenários P2P e uma comparação entre eles.

O Capítulo 3 apresenta uma revisão bibliográfica sobre as características do processamento de consultas em ambientes distribuídos, indicando os principais trabalhos relacionados ao processamento distribuído de consultas em um ambiente P2P. Neste capítulo ainda é analisada a metodologia e a ferramenta propostas por Figueiredo (2007), explicando detalhadamente a principal referência desta dissertação.

O Capítulo 4 descreve a proposta deste trabalho: uma metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas em um ambiente P2P.

O Capítulo 5 apresenta a arquitetura proposta, detalhando todos os seus principais módulos que viabilizam o processamento distribuído da consulta XQuery em um ambiente P2P. Além disso, a implementação do protótipo da arquitetura é explicada.

O Capítulo 6 apresenta algumas avaliações executadas sobre a abordagem proposta, onde o comportamento do protótipo é verificado sobre diferentes cenários. Os resultados são discutidos e analisados graficamente.

Finalmente, o Capítulo 7 conclui a dissertação, falando sobre as suas principais contribuições, relatando as limitações detectadas tanto na metodologia quanto na implementação e enumerando possíveis trabalhos futuros.

Capítulo 2. Sistemas *Peer-to-Peer*

2.1. Introdução

Atualmente, a tecnologia *Peer-to-Peer* vem atraindo cada vez mais atenção como um meio eficiente de compartilhamento de dados entre um número elevado de usuários dinâmicos e diversificados. Com o crescente aumento do uso da tecnologia P2P, principalmente no contexto de aplicações de compartilhamento de arquivos, como *Napster*, *Gnutella* e *BitTorrent*, o paradigma P2P também pode ser usado para acessar quaisquer tipos de dados distribuídos e está rapidamente se tornando uma das grandes tecnologias para gerenciamento desses dados. Em contrapartida, o tradicional modelo cliente-servidor, devido à sua natureza centralizada, geralmente resulta em altas cargas para as entidades centralizadas, que facilmente se tornam gargalos e pontos únicos de falhas. A abordagem P2P estende as principais idéias existentes da arquitetura cliente-servidor e do modelo computacional distribuído, eliminando assim as diferenças entre o cliente e o servidor.

Neste contexto, o paradigma P2P se refere a uma forma de computação distribuída e descentralizada, envolvendo um grande número de pontos que cooperam para compartilhar recursos e serviços. Cada ponto de um sistema P2P pode ser considerado tanto servidor como cliente, além de fornecer parte dos recursos e informações disponíveis no sistema (ABERER et al., 2002).

Entretanto, a descentralização de sistemas P2P implica em um dos principais desafios desta abordagem: como armazenar e como localizar, de forma eficiente, um determinado dado em um sistema distribuído sem qualquer controle centralizado ou de coordenação (BALAKRISHNAN et al., 2003). Através das classificações existentes para as redes P2P, baseadas principalmente em seu grau de descentralização e no modo de distribuição dos dados entres os pontos, diversas soluções foram propostas a fim de resolver o problema citado e algumas delas são apresentadas nas seções seguintes.

Para melhor entendimento do paradigma P2P, este capítulo está organizado da seguinte forma: na Seção 2.2 são explicadas as principais características distintivas da

abordagem P2P. Já na Seção 2.3 são apresentadas as diversas classificações das redes P2P.

2.2. Características de Sistemas P2P

As seguintes características podem ser consideradas principais ao se trabalhar em uma abordagem P2P (KOLONIARI et al., 2005):

- **Escalabilidade**: Sistemas tradicionais distribuídos geralmente compostos por um elevado número de nós participantes tendem a não ser tão escaláveis quanto um sistema descentralizado ou distribuído. Um sistema P2P deve atingir uma alta escalabilidade com o constante aumento no número de pontos conectados, mas sem que isto prejudique o desempenho do sistema.
- **Descentralização**: O paradigma P2P é uma alternativa para o tradicional modelo cliente-servidor e para o modelo computacional distribuído, os quais são compostos por um único ou limitado número de servidores e vários clientes. Em especial, a descentralização é interessante a fim de evitar ponto único de falhas ou gargalos de desempenho nos servidores do sistema. Em um sistema P2P puro, todos os pontos participantes da rede possuem capacidades idênticas, tanto de servidor como de cliente do sistema. Nos sistemas P2P híbridos, que combinam os sistemas centralizados com os sistemas P2P puros, alguns participantes, chamados de super-pontos (*super-peers*), exercem o papel de servidor para os pontos clientes a eles acoplados.
- **Autonomia**: Uma abordagem P2P possui quatro diferentes tipos de autonomia: armazenamento, execução, tempo de vida e conexão. A autonomia de armazenamento se refere à liberdade que o ponto possui para armazenar seus próprios dados, de acordo com sua área de interesse e necessidade, ao contrário dos tradicionais sistemas de dados distribuídos com administração central que determinam aonde e quais itens de dados serão armazenados em cada nó. Já a autonomia de execução se refere à capacidade de um determinado ponto de responder as consultas submetidas e alterar seus próprios dados. A autonomia de tempo de vida refere-se à liberdade de cada ponto para entrar e sair do sistema a qualquer momento sem restrição. Nos sistemas de dados distribuídos tradicionais, os nós são

adicionados e removidos do sistema de uma forma controlada, necessitando de uma pesada administração para gerenciar os dados distribuídos no sistema. Além disso, um sistema P2P precisa prever mecanismos para lidar com os pontos indisponíveis sem causar problemas significativos durante a sua execução. Por fim, a autonomia de conexão se refere ao modo de ligação entre os pontos de um sistema P2P. Tal autonomia permite que um determinado ponto defina quantos e quais pontos vizinhos serão a si conectados, baseado em algum critério de seleção, como por exemplo, a distância entre os pontos.

2.3. Redes P2P

Os sistemas P2P necessitam de uma rede P2P para seu pleno funcionamento. Esta rede P2P é construída sobre uma rede física (normalmente a Internet) e, portanto, referida como rede de sobreposição. O grau de descentralização e o modo de distribuição dos dados entre os pontos da rede sobreposta provocam um forte impacto nas propriedades não-funcionais de um sistema P2P, tais como tolerância a falhas, desempenho, escalabilidade, segurança, dentre outras.

Quanto ao grau de descentralização, a diferença entre os sistemas P2P reside principalmente no tipo de arquitetura adotada (SCOLLMEIER, 2001; KOLONIARI et al., 2005). Os sistemas P2P variam de uma arquitetura pura, onde todos os pontos exercem o mesmo papel, para uma arquitetura híbrida, onde a pontos específicos são atribuídos diferentes papéis, chegando a uma arquitetura centralizada, na qual os pontos se conectam a um servidor central.

A arquitetura pura apresenta um funcionamento totalmente descentralizado, onde cada ponto é responsável pelo mecanismo de busca e pela manutenção da infra-estrutura, além do compartilhamento de recursos e serviços. O outro tipo de arquitetura chamada híbrida agrega um conjunto de pontos em super-pontos, sendo a comunicação estabelecida somente entre os super-pontos, e destes com seus pontos agregados. Os super-pontos se tornam responsáveis pelo fluxo de pontos na rede, controlando suas entradas e saídas, além de mediar e rotear as tarefas solicitadas. A arquitetura centralizada se assemelha a uma estrela, onde o centro é o servidor central e os pontos

correspondem às pontas da estrela. Em uma rede com esta arquitetura, existe um único ponto central de falha, caso o servidor falhe, toda a rede falha.

Em relação à distribuição dos dados entre os pontos em uma rede P2P, um sistema P2P pode ser classificado como estruturado e não-estruturado (KOLONIARI et al., 2005). As redes P2P estruturadas possuem uma estruturação significativa para controlar a distribuição dos dados entre os pontos da rede. A topologia da rede é controlada e os dados são posicionados de forma eficiente através de diferentes tipos de indexação. As redes P2P não-estruturadas são totalmente distribuídas e, portanto, não possuem um servidor centralizado ou qualquer outra informação sobre a distribuição dos dados na rede. Não existe controle preciso sobre a topologia deste tipo de rede, na qual os pontos se conectam arbitrariamente de modo não estruturado a alguns pontos já presentes na rede.

De acordo com as classificações apresentadas para os sistemas P2P, diversos cenários são formados a partir da arquitetura e tipo de distribuição dos dados adotados. Os principais cenários são apresentados nas próximas seções, juntamente com suas principais características e seus mecanismos de roteamento.

2.3.1. Rede P2P Centralizada e Estruturada

O primeiro cenário identificado se refere a uma rede P2P de arquitetura centralizada e de distribuição de dados estruturada. O modelo consiste em um único ponto principal com a função de servidor central da rede, contendo todas as informações dos dados distribuídos armazenadas através de um índice centralizado no servidor.

No momento da entrada de um novo ponto na rede P2P, o servidor central atualiza seu índice com os novos dados disponíveis para que os demais pontos possam tomar conhecimento de sua entrada. Com este tipo de arquitetura, quando um usuário submete uma consulta a um ponto específico, a mesma é encaminhada para o servidor central que efetua a busca pelas informações desejadas através do índice. Após a busca no servidor central, o ponto solicitante da consulta se comunica diretamente com os pontos relevantes retornados pelo índice para a troca de informações.

Conforme a Figura 1, uma consulta *Q* é submetida no ponto origem (*Ponto 1*), sendo a mesma roteada para o servidor. O processamento da consulta é todo realizado no *Ponto 1*, que reescreve a consulta *Q* nas sub-consultas *Q1* e *Q2* com base nas

informações retornadas pelo índice do servidor (*Pontos Relevantes PR*). Cada ponto destino (*Ponto 2* e *Ponto 3*) executa as sub-consultas e retorna seus resultados (*R1* e *R2*) para o ponto origem. Após, a consolidação dos resultados é realizada, retornando o resultado final *R* ao usuário.

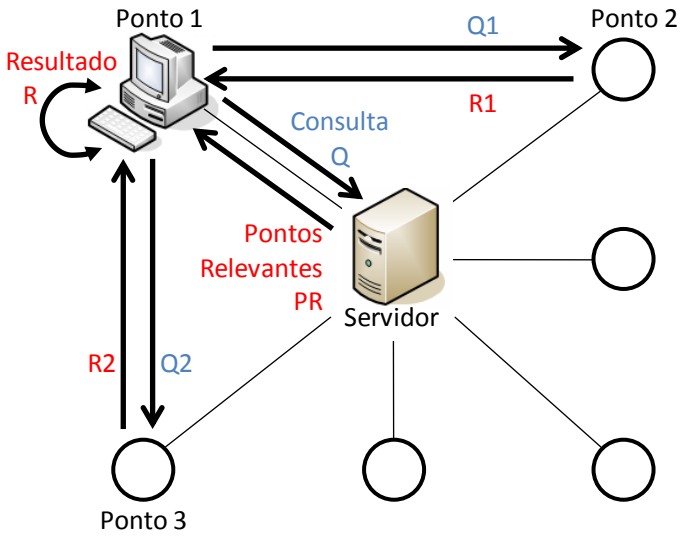


Figura 1 - Cenário 1: Rede P2P Centralizada e Estruturada

O *Napster* (NAPSTER, 2009) pode ser considerado exemplo de um sistema P2P pertencente ao cenário discutido. Quando novos pontos ingressam em sua rede P2P, os mesmos publicam seus dados (principalmente arquivos de músicas) no índice centralizado, o qual é consultado quando uma consulta é submetida. As informações sobre os dados de todos os pontos da rede P2P são armazenadas em um único ponto, considerado o servidor central. O fato principal é que apenas os ponteiros para os pontos descentralizados, ao invés dos dados reais, são armazenados no índice centralizado, ocasionando uma diminuição da carga no servidor. Assim que os dados relevantes são localizados no índice, o ponto solicitante da consulta pode se comunicar diretamente com os pontos fontes dos dados, ignorando completamente o índice centralizado.

Uma das importantes vantagens deste tipo de rede P2P é a garantia de localização dos dados entre os pontos disponíveis da rede. Caso o dado esteja disponível em algum dos pontos da rede P2P, ele será encontrado. Porém, a principal desvantagem deste cenário é a alta suscetibilidade a falhas, pois o servidor central representa um ponto único de falha na rede P2P. Mesmo mantendo réplicas do índice centralizado em outros pontos da rede, a confiabilidade e a escalabilidade da rede P2P pode ser afetada,

além de lidar com o aumento do número de atualizações nos diversos índices, caso algum novo ponto ingresse na rede P2P.

2.3.2. Rede P2P Pura e Não-Estruturada

O segundo cenário consiste em uma rede P2P de arquitetura pura e de distribuição de dados não-estruturada. As redes P2P não-estruturadas são criadas de uma forma não-determinística, não possuindo qualquer servidor central ou qualquer outra forma de conhecimento explícito sobre a localização dos dados dentro da rede. Estas redes são totalmente distribuídas e, portanto, não possuem um ponto central de falha.

Um ponto que deseja entrar na rede simplesmente descobre alguns pontos já presentes na rede através, por exemplo, de inundação (*Flooding Style Networks*), e se conecta arbitrariamente de modo não estruturado. Desta forma, cada ponto conhece apenas seus vizinhos, mas não sabe quais recursos eles provêem. Assim, o processamento da consulta é feito de forma incremental. Por exemplo, quando um usuário submete uma consulta a um determinado ponto, o ponto origem inicia a execução da consulta e a mesma é roteada para seus pontos vizinhos para continuar seu processamento. Deste modo, o plano de execução é incrementado a partir dos resultados parciais obtidos em cada ponto e repassado aos pontos vizinhos conectados.

Como em um ambiente P2P não-estruturado não é utilizado nenhum tipo de indexação, o roteamento da consulta é realizado por mecanismos de inundação (*flooding*) controlada por TTL (*Time-to-Live*). Isto é, cada ponto da rede P2P encaminha recursivamente a consulta submetida para todos os seus pontos vizinhos. Em cada mensagem repassada é atribuído um valor TTL, que define o número máximo de pontos alcançáveis para executar a consulta, e tem como objetivos evitar a propagação infinita da mensagem e controlar o número de mensagens geradas por uma consulta. Cada um dos seus vizinhos avalia a mensagem e verifica se tem o recurso solicitado na consulta. Caso possua, uma mensagem de resposta é enviada ao ponto origem da consulta. Caso não possua, o ponto incrementa o número de saltos da mensagem (*hop count*) e a repassa para todos os seus vizinhos. Quando o número de saltos alcança o TTL, o repasse da mensagem é interrompido.

A Figura 2 ilustra o cenário descrito. Um usuário submete uma consulta *Q* no ponto origem (*Ponto 1*), sendo a busca iniciada com TTL = 2. Dentro de cada um dos pontos alcançados pela busca estão os números que representam o número de saltos da mensagem. Considerando que o único ponto relevante para a resposta da consulta é o *Ponto 7*, tal ponto executa a consulta *Q* localmente ao receber a mensagem e retorna seu resultado *R* para o ponto origem.

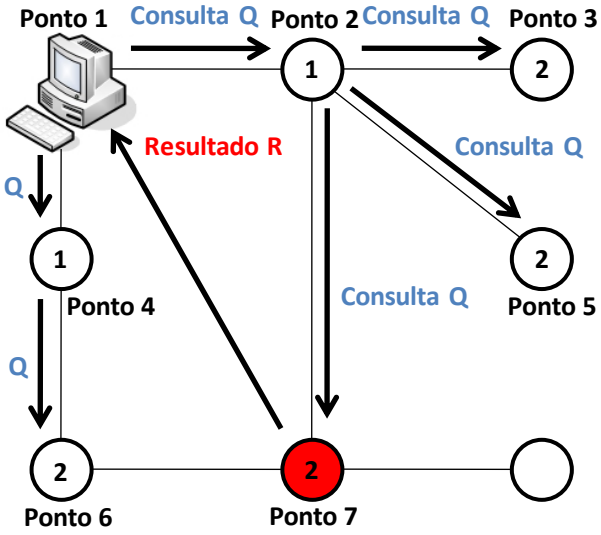


Figura 2 - Cenário 2: Rede P2P Pura e Não-Estruturada

Uma das principais vantagens deste cenário se resume à alta disponibilidade dos pontos em relação à tolerância a falhas na rede P2P, pois todos os pontos fornecem a mesma funcionalidade e são também capazes de replicar dados. Além disso, cada ponto é autônomo para armazenar seus dados e de fácil manutenção, visto que um ponto mantém apenas ponteiros para um número limitado de vizinhos.

No entanto, os principais problemas de redes P2P não-estruturadas são a escalabilidade e a incompletude dos resultados da consulta. O mecanismo de encaminhamento de consultas baseado na técnica de inundação não escala para um elevado número de pontos, pois necessita de um grande poder de processamento e gera um alto tráfego na rede, ocasionado pelas consultas reenviadas e pelos resultados retornados de cada ponto. Além disso, a incompletude dos resultados pode ser elevada, uma vez que alguns pontos com dados relevantes podem não ser consultados por estarem distantes do ponto solicitante da consulta.

Exemplos de sistemas populares em redes P2P puras e não-estruturadas incluem *KaZaA* (KAZAA, 2009) e *BitTorrent* (BITTORRENT, 2009).

2.3.3. Rede P2P Pura e Estruturada

O terceiro cenário se refere a uma rede P2P de arquitetura pura e estruturada. Uma distribuição estruturada de dados significa que a rede sobreposta P2P e a distribuição dos dados são rigidamente controladas. A fim de localizar os dados relevantes em uma rede P2P de forma mais eficiente, os dados, ou seus ponteiros, são armazenados em locais específicos e os mapeamentos entre os dados e sua localização são fornecidos através de uma tabela de roteamento distribuída. Desta forma, as redes P2P estruturadas possuem a finalidade de resolver os problemas da grande suscetibilidade a ataques nas arquiteturas centralizadas e da falta de escalabilidade nas redes P2P não-estruturadas.

Neste cenário, o processamento de consultas pode ser realizado através de um roteamento eficiente do índice distribuído. Por exemplo, um usuário submete uma consulta *Q* ao ponto origem (*Ponto 1*) da rede P2P estruturada, conforme a Figura 3. Com as informações armazenadas no índice no ponto origem, a sub-consulta *Q1* é enviada ao *Ponto 2* e a sub-consulta *Q2* é encaminhada para os pontos *3* e *4*. A execução local é realizada em cada ponto destino e seus resultados (*R1*, *R2* e *R3*) retornam para o ponto origem, aonde a consolidação do resultado *R* é realizada e exibida ao usuário.

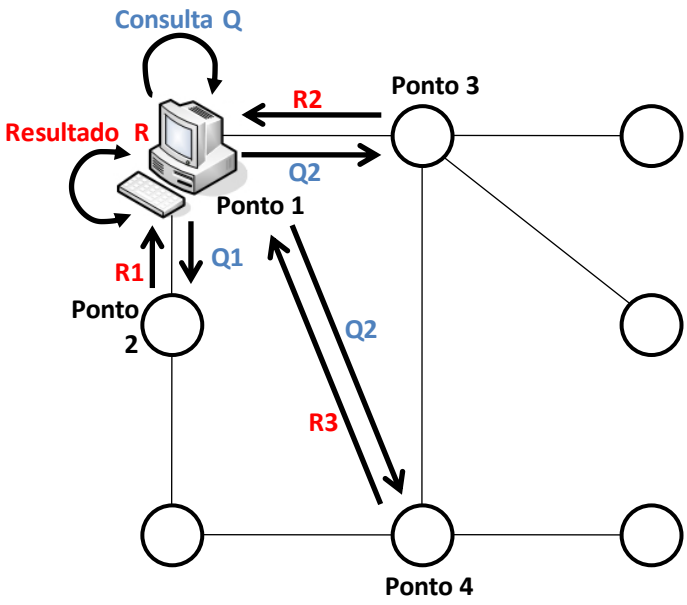


Figura 3 - Cenário 3: Rede P2P Pura e Estruturada

As redes P2P estruturadas são principalmente representadas através de tabelas *hash* distribuídas (DHT – *Distributed Hash Table*). O requisito principal de DHTs é que os valores armazenados possam ser endereçados por chaves numéricas únicas, sendo tais valores compostos pelo conteúdo do dado ou por um ponteiro para o local onde o dado realmente se encontra. Cada ponto conhece apenas um número limitado de pontos vizinhos e possui uma tabela de roteamento que associa os identificadores dos vizinhos com o endereço correspondente. Desta forma, qualquer ponto da rede P2P estruturada pode eficientemente recuperar o valor associado a uma dada chave. A responsabilidade de manter o mapeamento de chaves para os valores é distribuída entre os pontos, de modo que alterações nos pontos da rede P2P, tais como chegadas, saídas e falhas contínuas dos mesmos, ocasionem o menor impacto possível.

O grande desafio em uma arquitetura estruturada baseada em tabelas *hash* distribuídas consiste em mapear unicamente, de forma eficiente e determínistica, a chave de um dado requisitado para o identificador do ponto responsável pelo dado. A maioria dos acessos a dados nas DHTs são operações de consulta para localizar o endereço de um determinado ponto que armazena os dados requisitados, permitindo que ele seja contactado diretamente pelo ponto solicitante da consulta.

Exemplos dos principais sistemas em redes P2P estruturadas incluem *Chord* (STOICA et al., 2001), *CAN* (RATNASAMY et al., 2001) e *Pastry* (ROWSTRON et al., 2001b).

2.3.4. Rede P2P Híbrida e Estruturada

Finalmente, o quarto cenário consiste em uma rede P2P de distribuição estruturada de dados e de arquitetura híbrida, onde pontos especiais, chamados de super-pontos, possuem um papel diferenciado na rede. Em contraste com a arquitetura P2P pura, onde todos os pontos da rede fornecem a mesma funcionalidade, os super-pontos pertencem a uma arquitetura híbrida, pois ela combina funcionalidades das redes P2P com arquitetura pura e com arquitetura centralizada. Ou seja, uma arquitetura híbrida é composta por um grupo limitado de super-pontos que atuam como servidores dedicados para os pontos nele acoplados, além de assumir responsabilidades específicas na rede, como por exemplo, indexação, funções de roteamento, processamento de consultas,

controle de acesso, dentre outras. Conceitualmente, somente os super-pontos formam uma rede P2P de arquitetura pura.

Em um modelo P2P com arquitetura de super-pontos, o roteamento de uma consulta é realizado em duas fases. O roteamento é primeiramente executado nos super-pontos para somente após ser distribuído para seus respectivos pontos conectados. Com isso, todo o processamento da consulta distribuída é realizado apenas nos super-pontos. Quando um usuário submete uma consulta a um determinado ponto da rede P2P, a mesma é enviada para seu respectivo super-ponto. Com isso, o roteamento da consulta é realizado no super-ponto através de seu índice, indicando em quais pontos e/ou super-pontos a consulta deve ser processada. A Figura 4 ilustra o roteamento de uma consulta no cenário estudado.

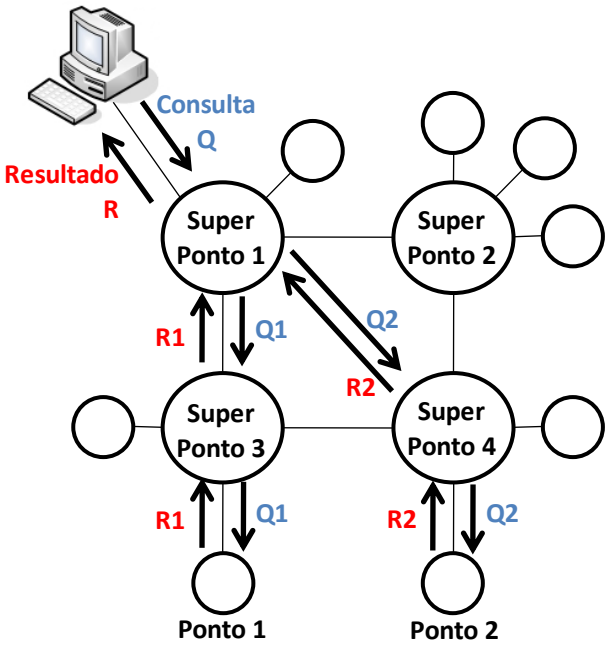


Figura 4 – Cenário 4: Rede P2P Híbrida e Estruturada

Uma das principais vantagens da arquitetura de super-pontos é a eficiência para localizar os dados relevantes através de seus índices, visto que o tempo despendido é muito inferior quando comparado com a técnica de inundação utilizada nas redes P2P não-estruturadas, embora os super-pontos limitem a capacidade de auto-organização de uma rede. No entanto, a autonomia de tempo de vida de um ponto é restrita dado que um ponto não pode se conectar livremente em qualquer super-ponto. Além disso, a arquitetura híbrida normalmente implica em uma baixa tolerância a falhas, visto que um super-ponto pode se tornar um ponto único de falha para os pontos a ele conectados. Porém, a falha de um super-ponto pode ser tolerada quando se permite eleger

dinamicamente outro super-ponto, baseado nas informações em termos de poder de processamento e largura de banda, por exemplo.

Exemplos de sistemas em redes P2P híbridas e estruturadas incluem *Edutella* (NEJDL et al., 2003), *JXTA* (JXTA, 2009), *Skype* (SKYPE, 2009) e *Publius* (WALDMAN et al., 2000).

2.4. Comparação de Redes P2P

A seguir, apresentamos uma breve comparação entre as redes P2P estudadas anteriormente. A comparação foi baseada nas principais características da abordagem P2P, como escalabilidade e autonomia, e nos critérios de tolerância a falhas, eficiência, segurança e busca, descritos a seguir.

- Tolerância a falhas: A eficiência e a qualidade de serviços devem ser prestadas apesar da ocorrência de falhas e ataques nos pontos de uma rede P2P.
- Eficiência: A utilização eficiente dos recursos de um ambiente P2P deve resultar em um menor custo e maior aproveitamento das buscas requisitadas. Com isso, um maior número de consultas pode ser processado pelo sistema P2P em um dado momento.
- Segurança: A questão sobre o controle de acesso surge como o principal problema de segurança em ambientes P2P, quando o controle total das responsabilidades de pesquisa e roteamento é repassado para os pontos da rede.
- Busca: Um problema muito comum em sistemas P2P consiste em localizar eficientemente um ponto que possua dados relevantes para a busca solicitada, levando em consideração a expressividade da consulta, o número de pontos visitados e o tempo de resposta.

A Tabela 1 mostra de forma resumida como os requisitos discutidos acima são definidos em cada cenário discutido anteriormente: rede P2P centralizada e estruturada (Cenário 1); rede P2P pura e não-estruturada (Cenário 2); rede P2P pura e estruturada (Cenário 3); e rede P2P híbrida e estruturada (Cenário 4).

Tabela 1 - Comparativo entre os cenários P2P

Requisitos	**Cenário 1**	**Cenário 2**	**Cenário 3**	**Cenário 4**
Escalabilidade	Ruim	Boa	Boa	Razoável
Autonomia	Ruim	Boa	Razoável	Razoável
Tolerância a Falhas	Ruim	Boa	Boa	Razoável
Eficiência	Boa	Ruim	Boa	Razoável
Segurança	Boa	Ruim	Ruim	Boa
Busca				
Expressividade	Boa	Boa	Ruim	Boa
Alcançabilidade	Boa	Ruim	Boa	Boa
Tempo de Resposta	Bom	Ruim	Bom	Bom

Os dados apresentados na Tabela 1 demonstram que não podemos concluir qual o melhor cenário para a construção de um sistema P2P, pois cada modelo oferece diversas vantagens e desvantagens. Para a escolha do cenário ideal deve-se avaliar o tipo de aplicação, a complexidade, o tamanho e a segurança do sistema que se deseja criar e ponderar estas avaliações em conjunto com as características de cada arquitetura P2P.

Neste capítulo apresentamos as características proporcionadas pela abordagem P2P, além de explicar detalhadamente os principais cenários de redes P2P existentes, fornecendo maior embasamento teórico antes de entrarmos nos detalhes do ambiente proposto nesta dissertação.

Capítulo 3. Processamento de Consultas em Ambientes Distribuídos

3.1. Introdução

Nos últimos anos, o paradigma computacional P2P vem se destacando através de seu rápido crescimento como uma das grandes topologias de sistemas distribuídos da atualidade e tem recebido a atenção da comunidade científica. No entanto, redes P2P trazem à tona uma grande quantidade de problemas interessantes para a pesquisa no campo de sistemas distribuídos, especialmente na presença de dados complexos e de consultas sofisticadas submetidas em um sistema P2P (FEGARAS ET al., 2006).

Para entrar em funcionamento, um sistema P2P precisa resolver alguns problemas fundamentais. Os dois problemas mais importantes identificados são os de busca e roteamento dentro da rede. O problema de busca resume-se em definir qual método utilizar para localizar um recurso dentro da rede P2P de forma escalável e sem o auxílio de servidores centrais, seja a busca por um dado ou por outro ponto na rede. O segundo problema consiste em enviar mensagens de forma eficiente entre os pontos da rede P2P. Diversas soluções foram propostas e freqüentemente as soluções adotadas para a busca e para o roteamento se inter-relacionam de tal forma que o funcionamento de uma depende totalmente da outra.

O objetivo deste capítulo é analisar os principais sistemas P2P presentes na literatura para o processamento distribuído de consultas, focalizando em suas características, arquiteturas e mecanismos de roteamento. Além disso, este capítulo também apresenta a metodologia proposta e o protótipo desenvolvido por Figueiredo (2007), explicando detalhadamente a principal referência desta dissertação.

O capítulo está organizado da seguinte forma, além desta primeira seção de introdução. A Seção 3.2 discute os principais trabalhos relacionados ao processamento de consultas distribuídas em ambientes P2P, sumarizando suas principais características e diferenciais em um quadro comparativo. Na Seção 3.3 é discutida a referência

principal desta dissertação. Por fim, a Seção 3.4 apresenta as considerações finais do capítulo.

3.2. Processamento de Consultas em Ambientes P2P

Sistemas P2P já são uma realidade entre os usuários da Internet. As informações disponíveis na *Web* se encontram atualmente distribuídas através de múltiplas fontes de dados, principalmente descritas no padrão XML. Entretanto, o processamento distribuído de consultas sobre estas fontes de dados mais estruturadas ainda permanece como um problema complexo, apesar dos esforços de pesquisa acadêmica e industrial nos últimos anos (RIZZOLO et al., 2001; ABITEBOUL et al., 2003; CHEN et al., 2004; FIGUEIREDO, 2007).

Muitos fatores influenciam o desempenho do processamento distribuído de consultas XML. Em particular, a localização eficiente dos dados e o roteamento das consultas têm um impacto significante nas ferramentas típicas para o processamento de consultas XML distribuídas em ambientes P2P. Além disso, outro ponto importante acerca do desempenho vem da complexidade dos padrões (e recomendações) de linguagens de consulta XML, como o XQuery (BOAG et al., 2007) e o XPath (BERGLUNG et al., 2007), os quais são muito poderosos e requerem algoritmos sofisticados. Considerando estes dois importantes aspectos, uma análise das ferramentas para o processamento distribuído de consultas XML é então realizada.

Galanis et al. (2003) e Fegaras et al. (2006) propõem ferramentas que implementam uma rede P2P estruturada, baseada em tabelas *hash* distribuídas, para realizar buscas utilizando a XPath como linguagem XML de consulta. O objetivo do índice DHT distribuído usado em ambos os trabalhos é a localização das fontes que contêm os dados relevantes para a resposta da consulta submetida, ao invés de retornar os fragmentos XML reais como resultado. Estruturas de dados chamadas de resumos estruturais (*structural summaries*) são utilizadas para influenciar na busca por pontos relevantes. Nas arquiteturas propostas, tal estrutura de dados consiste na indexação de um documento XML tanto por sua composição estrutural como por seu conteúdo textual, sendo um resumo conciso de todas as expressões de caminhos válidas para os dados do documento XML. No trabalho de Fegaras et al. (2006), o conteúdo textual de um documento XML é resumido em sinopses de dados (*data synopses*), que relacionam

o conteúdo do texto com informações sobre a posição de todas as expressões de caminhos válidas do documento XML. Enquanto Fegaras et al. (2006) usam a expressão de caminho de uma estrutura de sinopse de dados como chave para o roteamento da consulta, Galanis et al. (2003) utilizam os elementos do documento XML como chaves de entrada no índice DHT.

Similarmente às abordagens apresentadas nos trabalhos relacionados acima, o *XPeer* (CONFORTI et al., 2007) é um sistema de compartilhamento e consulta de dados XML. Uma consulta XML submetida no sistema proposto é traduzida em expressões algébricas, as quais são decompostas, validadas através de estruturas *treeguides* (GOLDMAN et al., 1997) e enviadas para os pontos relevantes, permitindo desta forma um *broadcast* para toda a rede P2P. No entanto, *XPeer* é um sistema baseado em uma arquitetura P2P híbrida, permitindo a eleição de super-pontos, e não utiliza uma abordagem estruturada através de tabelas *hash* distribuídas.

Entre os trabalhos relacionados, *Pier* (HUEBSCH et al., 2005) é o mais próximo do sistema *XPeer* (CONFORTI et al., 2007) citado anteriormente. *Pier* utiliza uma arquitetura estruturada através de tabelas *hash* distribuídas, onde dados homogêneos são armazenados. As consultas submetidas nesta arquitetura são executadas através da adaptação de técnicas de bancos de dados paralelos, explorando assim o paralelismo interno das DHTs. Desta forma, o *Pier* está focado principalmente em dados homogêneos e em consultas realizadas paralelamente, enquanto o *XPeer* concentra-se na heterogeneidade dos dados. A única limitação significativa entre os dois sistemas é o roteamento da consulta, pelo fato do *XPeer* exigir um *broadcast* da consulta submetida para toda a rede P2P.

PIERSearch (LOO et al., 2004) apresenta uma abordagem híbrida entre os sistemas *Gnutella* (GNUTELLA, 2009) e *Pier* (CONFORTI et al., 2007). A arquitetura proposta por este trabalho utiliza uma abordagem estruturada para consultas raras, com o uso de tabelas *hash* distribuídas, juntamente com uma abordagem não-estruturada para consultas populares, através de mecanismos de busca por inundação.

Papadimos et al. (2003) propõem uma arquitetura baseada em planos mutantes de consulta (*Mutant Query Plans* - MQPs). O MQP é um grafo do plano algébrico da consulta codificado em XML, podendo incluir também as referências aos nomes dos recursos abstratos, chamadas de URNs (*Uniform Resource Name*), e as referências à localização desses recursos, chamadas de URLs (*Uniform Resource Locater*). Com isso, uma referência URN define a identidade de um dado, enquanto uma referência URL

indica um método de como o localizar. Quando um ponto da rede P2P recebe um MQP, o ponto pode resolver as referências URNs a uma ou mais referências URLs, materializar as referências URLs para seus dados XML correspondentes, validar e/ou otimizar os sub-planos MQPs gerados, ou simplesmente encaminhar o plano de consulta para um outro ponto da rede. No momento em que o plano de consulta é reduzido para apenas códigos XML, a resposta da consulta é enviada ao ponto solicitante da consulta. Como conseqüência, o plano de consulta atravessa toda a rede P2P, carregando resultados parciais e sub-planos de execução não validados, até que o plano de consulta esteja totalmente referenciado por um fragmento de dados XML. O sistema proposto explora fortemente a homogeneidade semântica dos dados fornecidos pelos pontos da rede P2P, não sendo adequado para dados heterogêneos. Além disso, a abordagem adotada aumenta o custo do roteamento e processamento da consulta por utilizar o mecanismo de inundação pelos pontos da rede P2P.

Bonifati et al. (2006) apresentam o sistema *XP2P*, que armazena e recupera dados XML em uma rede P2P estruturada baseada em DHT para realizar buscas utilizando o XPath como linguagem de consulta. Para este fim, a arquitetura se baseia em um modelo de fragmentação de documentos XML para a camada de dados da rede P2P.

Na proposta do sistema *XP2P*, um fragmento XML é um sub-documento extraído do documento XML original, identificado por uma única expressão de caminho, a qual é considerada a raiz do sub-documento XML gerado. Por exemplo, um fragmento XML definido através da expressão de caminho */S1/S2/S3[1]* carrega diferentes informações: (1) os dados locais armazenados correspondem ao elemento *S3*, o qual é considerado como o primeiro fragmento na ordem do documento; (2) em outros pontos externos podem existir fragmentos XML definidos através da expressão de caminho */S1/S2/S3[i]*, com $i >= 1$, parecidos com o fragmento atual; e (3) os elementos *S1* e *S2* também podem existir em qualquer outro ponto da rede. Além disso, um fragmento XML do sistema *XP2P* também pode manter ligações para outros fragmentos distribuídos entre os pontos da rede, possuindo possivelmente um conjunto de expressões de caminho relacionados aos seus fragmentos filhos (*child-fragment*) e/ou relacionado ao seu fragmento pai (*super-document*).

Desta forma, as expressões de caminho definidas pelos fragmentos XML são codificadas para serem consideradas como entrada do índice distribuído, criando um catálogo descentralizado. De fato, as expressões de caminho que servem para armazenar

os fragmentos no índice distribuído são as mesmas utilizadas na validação das consultas submetidas e também são usadas para rotear corretamente a busca solicitada para os pontos relevantes.

A abordagem em XP2P abrange três tipos de consulta: (1) busca exata (*exact-match*), onde a consulta é comparada exatamente com as expressões de caminho codificadas na DHT; (2) busca parcial (*partial-match*), a qual é realizada quando não existe a correspondência exata entre o padrão da árvore da consulta e o fragmento indexado na DHT, ocasionando uma reformulação no padrão de árvore da consulta original para a re-execução da busca; e (3) busca descendente (*descendant*), quando não pode ser realizada uma busca exata e nem uma busca parcial pelos prefixos da consulta solicitada, causando uma busca exaustiva nos pontos considerados como mais relevantes.

Além disso, diversos sistemas P2P (TATARINOV et al. 2003; BRITO, 2005; FERNANDES, 2007; PIRES, 2009) oferecem funcionalidades relacionadas à integração dos dados. Mesmo quando fontes de dados compartilhadas entre os pontos disponíveis da rede P2P seguem diferentes esquemas ou representações, estes sistemas P2P são ainda capazes de acessar tais dados, integrá-los e retornar os resultados.

Neste contexto, o *Piazza* (TATARINOV et al. 2003) é um sistema P2P para gerenciar dados heterogêneos de forma distribuída e escalável. O sistema P2P assume que usuários participantes da rede P2P estão interessados em compartilhar dados, além de estarem dispostos a definir mapeamentos entre seus esquemas. Desta forma, usuários formulam consultas sobre seu esquema de preferência e o sistema responde à consulta expandindo recursivamente todos os mapeamentos relevantes à mesma, recuperando os dados importantes que serão retornados aos usuários. Para identificar os pontos relevantes ao usuário, o sistema P2P constrói um índice para explorar informações sobre os reais dados de um ponto. Diferentemente de outros sistemas que adotam uma estratégia descentralizada baseada em DHT, a indexação dos dados é centralizada, e, portanto, a escalabilidade do sistema é limitada. Além disso, a abordagem *Piazza* ainda requer intervenção humana para a definição de mapeamentos dos esquemas.

O sistema *Rosa-P2P* (*Repository of Objects with Semantic Acess*) apresenta uma solução baseada na topologia de super-pontos e utiliza uma estrutura semântica complexa, denominada de vocabulário controlado, que auxilia na resolução de conflitos semânticos (BRITO, 2005). Estas estruturas são utilizadas em quatro propósitos básicos: (1) na localização dos pontos relevantes da consulta; (2) na reescrita da

consulta pelos pontos receptores; (3) na solução de possíveis conflitos; e (4) na sugestão de opções e caminhos associados com a pesquisa. O processamento da consulta é dividido em duas fases principais, onde a primeira fase consiste na localização dos pontos e/ou super-pontos relevantes ao domínio da consulta. Para tal, os principais termos que fazem parte da consulta solicitada são comparados aos termos existentes no vocabulário controlado de palavras-chaves. Uma vez encontrados, os respectivos assuntos são retornados e a consulta é reescrita de acordo com os termos dos vocabulários identificados e reenviada aos pontos que tratam daquele domínio, englobando a segunda fase do processamento da consulta. O processamento final da consulta é então realizado no ponto ou super-ponto solicitador da consulta.

Finalmente, o *SPEED* (***S****emantic* ***Pee****r-to-Peer* ***D****ata Management System*) é um sistema P2P que utiliza semântica baseada em ontologias para resolver os problemas críticos de gerenciamento de dados em ambientes P2P, tais como conectividade, mapeamentos e processamento de consultas (FERNANDES, 2007; PIRES, 2009). Este sistema P2P adota uma topologia mista, empregando os conceitos de super-pontos e DHT. Nesta arquitetura, pontos disponíveis que possuem fontes de dados associadas são agrupados de acordo com seu domínio em *clusters* semânticos, através de uma rede DHT. *Clusters* semânticos são então agrupados em comunidades semânticas e organizados numa topologia de super-pontos, onde um ponto de dados especial, chamado de ponto de integração, atua como gerenciador da comunidade, oferecendo uma ontologia de domínio e organizando serviços.

Além disso, o ponto de integração associado ao *cluster* semântico é responsável pelo processamento das consultas submetidas. Neste contexto, quando um ponto de integração recebe uma consulta, este ponto identifica os principais termos da consulta e os compara com os termos da ontologia de domínio do *cluster*. A partir de mapeamentos semânticos e de um índice de consultas, o ponto de integração identifica os pontos relevantes que são capazes de responder a consulta submetida. Após a identificação, o ponto de integração reformula a consulta submetida de acordo com os mapeamentos identificados, reenvia para os pontos relevantes, consolida os resultados retornados e envia o resultado final para o usuário solicitante da consulta.

Portanto, muitos fatores influenciam o desempenho do processamento distribuído de consultas XML em ambientes P2P, em especial a localização eficiente dos dados e o roteamento das consultas. Além disso, outro ponto importante acerca do desempenho vem da complexidade dos padrões (e recomendações) de linguagens de

consulta XML, como o XQuery (BOAG et al., 2007) e o XPath (BERGLUNG et al., 2007), os quais são muito poderosos e requerem algoritmos sofisticados.

Considerando principalmente estes dois importantes aspectos, os quadros comparativos, apresentados pela Tabela 2 e pela Tabela 3, foram gerados como resultado da análise sobre os sistemas P2P para o processamento distribuído de consultas XML, onde as principais características e diferenciais são descritos. Os trabalhos analisados que tratam de integração de dados não são comparados pois estão fora do escopo desta dissertação.

Tabela 2 - Quadro Comparativo (Parte 1)

Sistemas P2P	**Galanis et al. (2003)**	**Fegaras et al. (2006)**	**XPeer**
Modelo de Arquitetura	Arquitetura estruturada (DHT)	Arquitetura estruturada (DHT)	Arquitetura Híbrida (*Broadcast*)
Linguagem de Consulta	XPath	XPath	XQuery
Processamento de Consultas	Consulta enviada para os pontos competentes identificados através de um serviço de catálogo descentralizado que armazena a localização dos dados relevantes.	Consulta submetida migra de ponto em ponto de forma a validar partes da consulta baseada em sinopses de dados, coletando documentos XML relevantes ao longo do processamento da consulta.	Consulta submetida é decomposta em expressões algébricas para serem validadas através de estruturas *treeguides* pelos *super-peers* e enviadas para os demais pontos através de um *broadcast* na rede.
Características Particulares	Elementos do documento XML como chaves de entrada no índice DHT.	Expressões de caminho das estruturas de sinopses de dados como chaves no índice DHT.	Esquemas heterogêneos dos pontos devem se reportar ao esquema do super-pontos.

Tabela 3 - Quadro Comparativo (Parte 2)

Sistemas P2P	**Mutant Query Plans**	**XP2P**
Modelo de Arquitetura	Arquitetura não-estruturada (*Broadcast*)	Arquitetura estruturada (DHT)
Linguagem de Consulta	Plano algébrico da consulta codificado em dados literais XML.	XPath
Processamento de Consultas	Consultas roteadas por catálogos hierárquicos distribuídos. Plano de consulta inunda a rede P2P acumulando resultados parciais e sub-planos de execução não validados até que o plano de consulta esteja referenciado por um fragmento de dados XML para então ser retornado para o ponto solicitante.	A abordagem em XP2P abrange três tipos de consultas: busca exata, busca parcial e busca descendente, nas quais as consultas são validadas através das expressões de caminho que servem para armazenar os fragmentos no índice DHT.
Características Particulares	Sistema explora fortemente a homogeneidade semântica dos dados fornecidos pelos pontos.	Expressões de caminho definidas pelos fragmentos XML são codificadas como chaves de entrada no índice DHT.

3.3. Processamento de Consultas XQuery em Ambientes Distribuídos

O trabalho de Figueiredo (2007) apresenta uma metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas e fragmentadas. Essa metodologia serviu como base para o desenvolvimento deste trabalho. Nessa seção, apresentamos a metodologia de Figueiredo (2007), além de também apresentar a arquitetura sobre o qual a nossa proposta é implementada. Para tal, as etapas do processamento de uma consulta XQuery distribuída e a arquitetura elaborada são apresentadas em mais detalhes nas próximas seções.

3.3.1. Metodologia de Processamento de Consultas XQuery Distribuídas

Esta seção apresenta a metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas proposta por Figueiredo (2007). O processo consiste na decomposição da consulta principal em sub-consultas que serão executadas em bases de dados XML remotas. A metodologia proposta é definida através de uma adaptação das quatro etapas básicas da metodologia proposta para o modelo relacional (OZSU, VALDURIEZ, 1999): decomposição da consulta, localização dos dados, otimização global e local, conforme ilustrado na Figura 5. Essa metodologia pode ser aplicada tanto em 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. É importante ressaltar que ambos os casos utilizam um Catálogo centralizado, que armazena informações sobre a distribuição das bases de dados, tais como os esquemas das visões globais, as definições e os esquemas dos fragmentos, estatísticas dos fragmentos, dentre outras. Considerando o contexto de consultas XQuery distribuídas, cada uma das etapas é descrita a seguir.

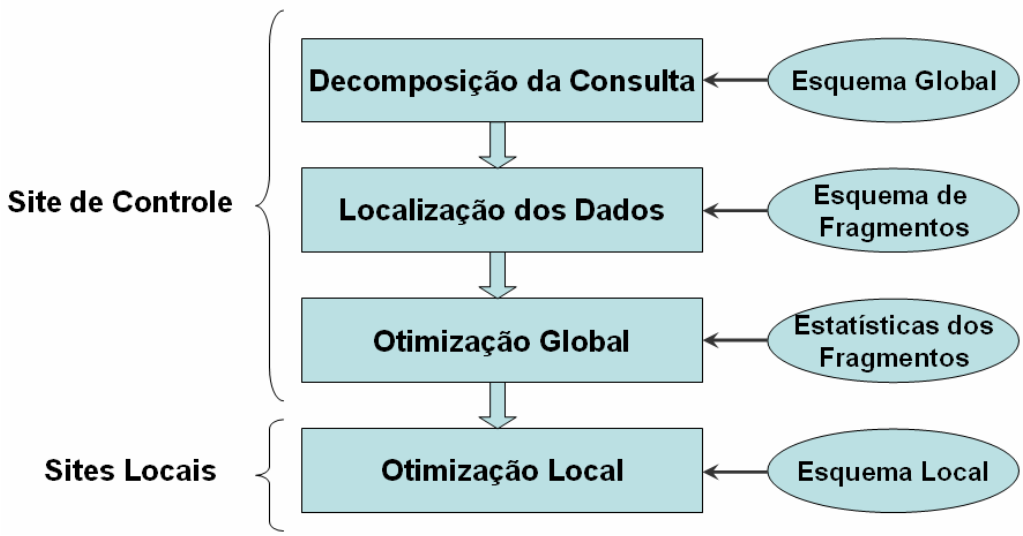


Figura 5 - Metodologia do Processamento de Consultas Distribuídas sobre o Modelo Relacional (ÖSZU et al., 1999)

3.3.1.1. Decomposição da Consulta

A primeira etapa consiste na decomposição da consulta XQuery em uma expressão algébrica sobre as coleções XML globais. Esta etapa consiste nas fases de validação sintática e semântica da consulta; simplificação da consulta; e representação da consulta em sua forma algébrica.

A análise sintática verifica se a consulta original, expressa através da linguagem XQuery, está escrita de acordo com a gramática suportada. Palavras-chave da linguagem escritas ou posicionadas de forma errada na consulta são considerados exemplos de erros de sintaxe. Já a validação semântica verifica se a consulta escrita está de acordo com as especificações presentes no catálogo central. No entanto, as informações sobre a distribuição dos dados ainda não são utilizadas, apenas as informações das coleções XML globais são consultadas para validar as coleções expressas na consulta. Nomes de visões globais errados ou inexistentes são exemplos de erros semânticos. Após a validação sintática e semântica da consulta XQuery, inicia-se a fase de simplificação da consulta que consiste na verificação dos predicados de seleção e na eliminação dos predicados redundantes.

O produto final da etapa de decomposição da consulta é sua representação através de expressões algébricas sobre as coleções XML globais. Como discutido na proposta de Figueiredo (2007), a utilização de uma álgebra formal é fundamental para o processamento e a otimização de consultas. Sendo assim, a álgebra TLC (PAPARIZOS

et al., 2004) é a álgebra utilizada para a decomposição das consultas XQuery na metodologia proposta. Para tal, a representação de uma consulta em sua forma algébrica consiste na execução de um algoritmo que analisa sintaticamente a consulta e monta as operações algébricas, os padrões de árvores e as classes lógicas da TLC.

A álgebra TLC utiliza padrões de árvores anotados (*Annotated Pattern Trees*, APT) para o acesso a conjuntos heterogêneos de árvores. Em especial, o conceito de padrão de árvore é estendido por anotações em suas arestas e generaliza a semântica da verificação dos padrões de árvore para produzir conjuntos heterogêneos de árvores como saída. 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. Para facilitar o acesso às classes lógicas, rótulos chamados *logical class labels* (LCLs) são associados para cada classe lógica do APT. Os LCLs são utilizados no processamento da consulta como uma referência para os nodos dos APTs dentro das operações algébricas.

Em (PAPARIZOS et al., 2004; FIGUEIREDO, 2007) encontramos maiores informações sobre o algoritmo para a reescrita de uma consulta XQuery em uma expressão da álgebra TLC. A reescrita de uma consulta XQuery em sua representação algébrica em TLC compreende a transformação de funções da XQuery em operações da álgebra TLC, como é exemplificado a seguir.

A consulta apresentada na Figura 6 pode ser dividida em quatro partes diferentes: (1) a primeira parte da consulta (expressão *for*) produz uma operação algébrica de seleção com um APT formado inicialmente pela expressão de caminho sobre a coleção global especificada; (2) a segunda parte da consulta (expressão *where*) adiciona um nó com o predicado de seleção da consulta submetida ao APT resultante da operação algébrica de seleção; (3) a terceira parte (expressão *order by*) produz uma operação de ordenação sobre os campos indicados na consulta submetida; e finalmente (4) a última parte da consulta (expressão *return*) produz duas operações distintas, sendo uma operação de seleção para a definição dos LCLs apresentados no resultado final e uma operação de construção para a composição da árvore de saída da consulta através dos LCLs definidos.

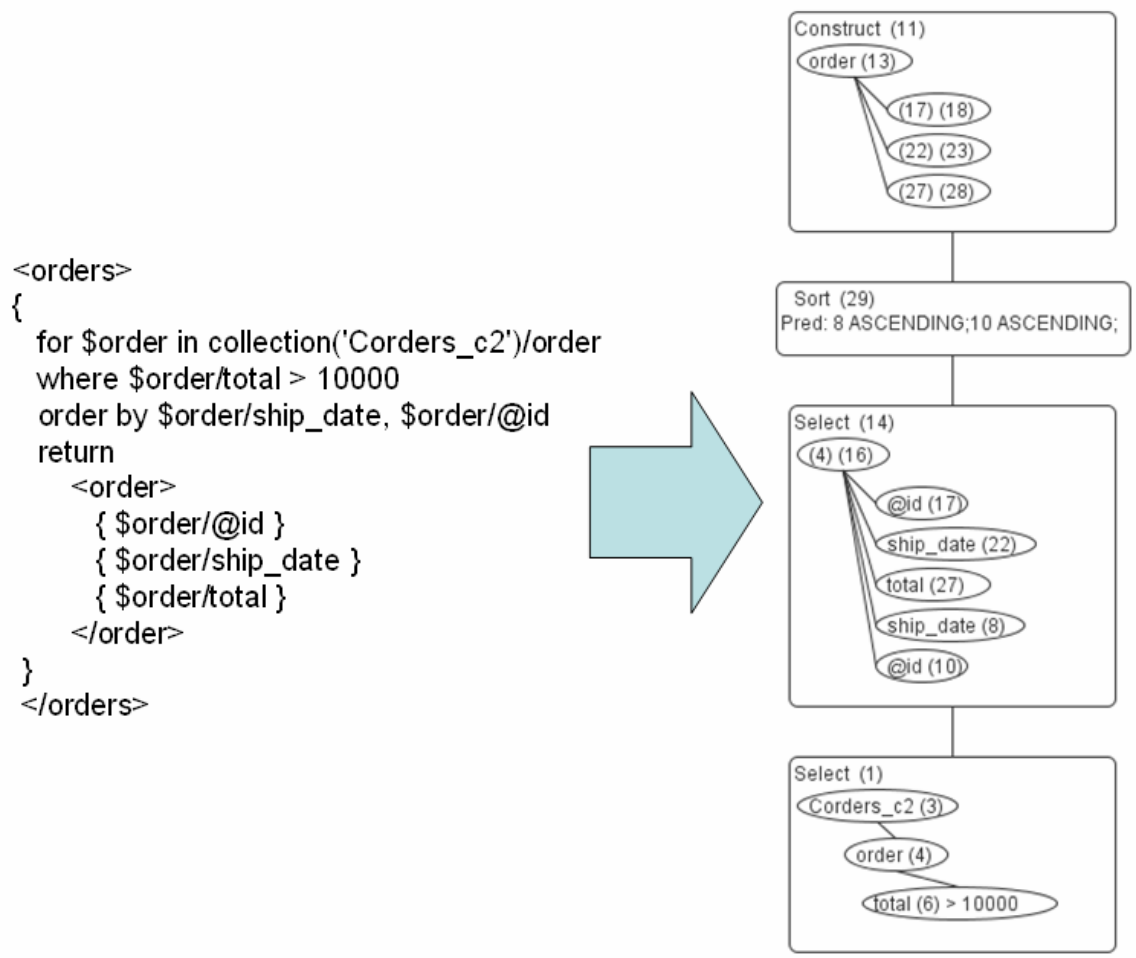


Figura 6 - Representação da consulta em sua forma algébrica TLC (FIGUEIREDO, 2007)

Os APTs são ilustrados como árvores das classes lógicas associados com o seu valor LCL. Cada valor LCL é representado graficamente pelo número especificado entre parênteses dentro da elipse de cada classe lógica. Como exemplo, podemos citar a classe lógica *order* da primeira operação de seleção contendo um valor LCL 4. Na segunda operação de seleção, este LCL é substituído por um novo valor LCL (16), o qual será utilizado na operação final de construção, através do algoritmo de conversão da consulta XQuery para TLC. Desta forma, o valor LCL 4 ou 16 corresponde a representação da variável *$order* da consulta XQuery submetida.

Esta representação algébrica inicial sobre as coleções globais é pré-requisito para a próxima etapa de localização dos dados no processamento distribuído da consulta.

3.3.1.2. Localização dos Dados

A segunda etapa no processamento distribuído de uma consulta XQuery é a localização dos dados da consulta através das informações disponíveis no Catálogo sobre a distribuição dos dados, incluindo as definições da fragmentação de uma base de dados e a alocação dos fragmentos no ambiente distribuído.

Esta etapa engloba duas atividades principais, (1) substituição das referências a coleções globais do plano algébrico por referências a fragmentos destas coleções e (2) eliminação dos fragmentos irrelevantes ao resultado da consulta, utilizando a abordagem de redução do plano algébrico, de maneira semelhante à metodologia proposta em (ÖZSU et al., 1999). Com isso, a etapa de localização dos dados é responsável pelo maior benefício da fragmentação de uma base de dados, pelo fato de ter seu desempenho melhorado devido à diminuição do volume de dados consultados, através da eliminação dos fragmentos desnecessários.

Para realizar a substituição de uma coleção global na representação algébrica da consulta, é necessário conhecer os fragmentos envolvidos na consulta e seus predicados de formação. Essa substituição pode ser feita de modo automático quando o projeto de fragmentação obedece a regras de correção de fragmentação XML (ANDRADE et al., 2005), como a reconstrução da coleção global a partir de seus fragmentos através de uma operação da própria TLC. Esta operação é definida de acordo com o tipo de fragmentação, sendo união para fragmentos horizontais e junção para fragmentos verticais. Em (FIGUEIREDO, 2007) são apresentadas as regras para localização dos dados para os diferentes tipos de fragmentação.

3.3.1.3. Otimização Global

A etapa de otimização global da consulta distribuída procura encontrar um plano de execução da consulta global que seja próximo do plano de execução ótimo. A otimização global de uma consulta começa com a criação de variações equivalentes do plano algébrico, obtido na etapa de localização dos dados da consulta, com o objetivo de encontrar um plano algébrico de execução de custo mínimo.

A busca do plano de execução próximo ao ótimo é realizada através de transformações algébricas, tais como a troca de posição de operações dentro do plano algébrico, a substituição da localização de fragmentos quando existirem réplicas destes fragmentos em diferentes nós, dentre outras.

Para cada plano algébrico de execução equivalente produzido é associado o seu custo estimado a partir de uma função de custo. A função de custo deve calcular o custo de cada operação do plano algébrico através de uma combinação ponderada de diversos parâmetros, tais como o custo de comunicação entre os nós remotos, o custo de acesso ao disco, o custo de transmissão dos dados pela rede, as estimativas do volume de dados processados por cada operação, dentre outros parâmetros.

O produto final da etapa de otimização global de uma consulta distribuída é um plano algébrico de execução próximo do ótimo, visto que a busca do plano ótimo é muito cara para o processamento da consulta, podendo reduzir os ganhos obtidos com a otimização do plano algébrico. Este plano algébrico é utilizado posteriormente para a montagem das sub-consultas destinadas aos nós remotos.

3.3.1.4. Otimização Local

Após a otimização global da consulta, são geradas sub-consultas destinadas aos nós remotos do ambiente distribuído. Cada sub-consulta executada em um nó é otimizada com o uso de seu esquema local. Desta forma, a otimização local da consulta é realizada pelos nós remotos através do próprio banco de dados que armazena o fragmento XML. Após, os resultados das sub-consultas remotas são processados pelo Mediador de acordo com a estratégia de execução para a composição do resultado final da consulta.

3.3.2. Arquitetura

Com o objetivo de avaliar a viabilidade técnica da metodologia proposta e analisar as questões de desempenho para o processamento de consultas XQuery distribuídas, uma arquitetura baseada em camadas foi implementada para a metodologia descrita anteriormente. Para compor uma visão global e servir de ponto único de acesso, Figueiredo (2007) utiliza uma arquitetura baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992) acoplados aos bancos de dados XML remotos, conforme ilustra a Figura 7. A arquitetura apresentada contempla os seguintes componentes: Mediador, Catálogo e Adaptadores, que serão detalhados a seguir.

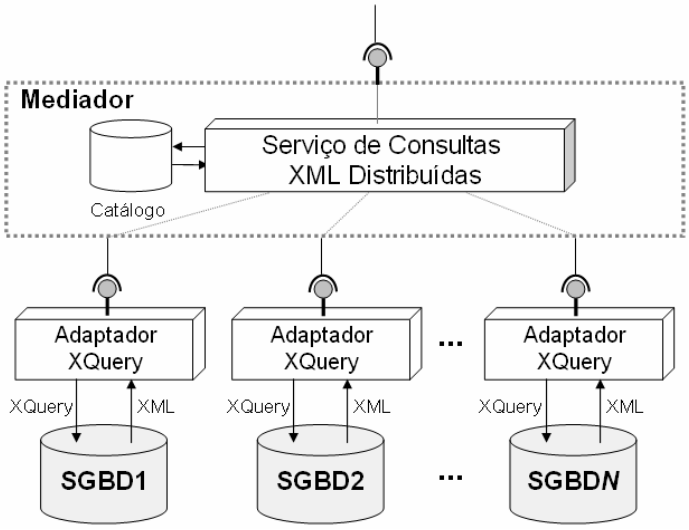


Figura 7 - Arquitetura Mediador – Adaptadores (FIGUEIREDO, 2007)

O **Mediador** é o principal componente da arquitetura proposta por Figueiredo (2007). Este componente é responsável pelo processamento de consultas XQuery distribuídas, realizando as etapas de decomposição, localização e otimização global das consultas, de acordo com a metodologia discutida anteriormente. As consultas submetidas sobre uma visão XML global são decompostas em um conjunto de sub-consultas, que são executadas pelos nós remotos sobre os fragmentos XML. Os resultados de cada sub-consulta retornam ao Mediador para a reconstrução do resultado final. Com isso, a função do Mediador é fornecer uma visão global dos dados distribuídos consultados de forma transparente pelos usuários, ocultando dos usuários finais os detalhes da localização e fragmentação da base de dados e provendo um ponto único de acesso para os dados das fontes distribuídas.

O diagrama ilustrado pela Figura 8 apresenta a implementação do Mediador, baseada em uma arquitetura básica para processamento de consultas, discutida em (KOSSMAN, 2000). Esta arquitetura possui uma série de módulos responsáveis por partes isoladas do processamento da consulta e maiores detalhes podem ser encontrados em (FIGUEIREDO, 2007).

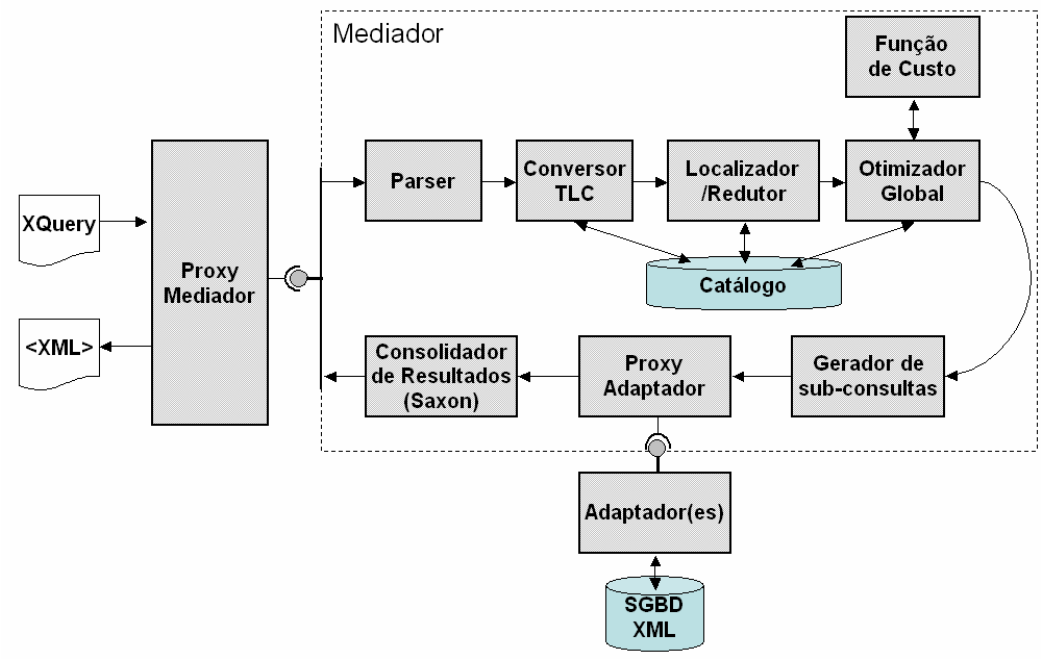


Figura 8 - Diagrama de módulos do Mediador (FIGUEIREDO, 2007)

O **Catálogo** armazena as configurações do ambiente distribuído, possuindo todas as informações necessárias para o processamento da consulta distribuída, em especial para a etapa de localização dos dados. Informações como o nome e o esquema das visões globais, os fragmentos que formam a visão global da coleção XML distribuída, a referência de cada Adaptador remoto que possui uma cópia do fragmento e estatísticas dos fragmentos são exemplos de dados armazenados no Catálogo.

As consultas XQuery submetidas sobre uma visão XML global dos dados são decompostas em um conjunto de sub-consultas destinadas aos **Adaptadores**. Desta forma, os Adaptadores são responsáveis pelo gerenciamento de execução das sub-consultas nos bancos de dados XML a eles acoplados. O resultado de cada sub-consulta é retornado ao Mediador para composição do resultado final. Para finalizar, o Mediador consolida todos os resultados dos Adaptadores e envia ao usuário o resultado final.

3.4. Considerações Finais

Em nosso trabalho, o objetivo é elaborar uma metodologia para o processamento de consultas XML distribuídas utilizando uma abordagem P2P. Uma parte fundamental desta dissertação foi a análise dos principais sistemas P2P para o processamento de consultas XML distribuídas, como vimos na Seção 3.2. Diante dos quadros

comparativos apresentados pela Tabela 2 e pela Tabela 3, os sistemas P2P sumarizados não apresentam nenhum formalismo quanto à definição de etapas para o processamento da consulta XML submetida assim como não utilizam uma álgebra XML para a reescrita da consulta em expressões algébricas, a fim de tornar o processamento da consulta XML correto e automático no ambiente P2P. Além disso, a questão da heterogeneidade abordada em alguns sistemas P2P citados (TATARINOV et al. 2003; BRITO, 2005; FERNANDES, 2007; PIRES, 2009) está fora do escopo apresentado nesta dissertação, conforme discutido no capítulo seguinte.

Dentro deste contexto, a proposta elaborada por Figueiredo (2007) apresenta uma metodologia que define etapas para o processamento de consultas XQuery distribuídas em conjunto com o formalismo da álgebra TLC (PAPARIZOS et al., 2004), conforme detalhada na Seção 3.3. Desta forma, é possível obter uma solução para o processamento de consultas XQuery distribuídas que é facilmente adaptada para o ambiente P2P. Além disso, a alta expressividade da linguagem XQuery (BOAG et al., 2007) também representa um fator positivo para a adoção da metodologia proposta por Figueiredo (2007) em uma abordagem P2P.

Neste escopo, a metodologia adaptada para o ambiente P2P engloba todas as etapas definidas para o processamento de consultas XQuery em um ambiente distribuído (FIGUEIREDO, 2007). Porém, a metodologia em uma abordagem P2P focaliza principalmente nos aspectos relacionados à reformulação da consulta, em especial para a etapa de localização dos dados relevantes entre os pontos da rede P2P que podem contribuir para o resultado da consulta. Esta metodologia é apresentada no próximo capítulo.

Capítulo 4.Metodologia para Processamento de Consultas XQuery Distribuídas em Ambientes P2P

4.1. Introdução

Este capítulo apresenta uma metodologia para o processamento de consultas XQuery sobre bases de dados XML distribuídas em um ambiente P2P. Sobre a consulta principal, são executadas as etapas de decomposição da consulta; localização dos pontos da rede P2P que possuem dados relevantes para o resultado da consulta; otimização global; geração de sub-consultas e a sua execução nas bases XML dos pontos remotos relevantes; e consolidação do resultado final no ponto solicitante da consulta.

Uma característica importante da metodologia proposta é ser genérica o suficiente para ser aplicada tanto em um banco de dados XML distribuído que permita a fragmentação de suas bases, quanto em um sistema que proporcione uma visão integrada de bancos de dados XML semi-autônomos homogêneos. Um ponto da rede P2P não possui liberdade para definir seu próprio esquema de dados, portanto a metodologia não contempla esquemas heterogêneos de dados.

As etapas do processamento de uma consulta XQuery distribuída em um ambiente P2P são apresentadas na Seção 4.3, incluindo as etapas finais de execução das sub-consultas geradas nos pontos remotos e consolidação do resultado final no ponto solicitante da consulta. A importante etapa de localização dos dados, que engloba a pesquisa por pontos da rede P2P com dados relevantes para o resultado da consulta principal, é apresentada com mais detalhes na Seção 4.3.2. Antes disso, na Seção 4.2 é explicado como os fragmentos XML distribuídos pela rede P2P são definidos e alocados nos pontos disponíveis.

4.1.1. Exemplos Utilizados neste Capítulo

Em (YAO et al., 2004), os autores definem uma classificação para repositórios XML. Um repositório é o local onde estão armazenados os documentos XML. Com isso, um repositório XML pode ser composto por múltiplas instâncias (vários documentos XML), chamado de repositório de Múltiplos Documentos (*Multiple Documents*, MD); ou por um único grande documento XML que armazena todas as informações necessárias, chamado de repositório de Documento Único (*Single Document*, SD).

Para exemplificar o processamento de consultas XQuery sobre bases de dados XML em um ambiente P2P, iremos utilizar uma base de dados XML de múltipos documentos (MD). A coleção de documentos XML de pedidos de compras, chamada *COrders,* do *benchmark* XBENCH (YAO et al., 2002), ilustra os exemplos utilizados ao longo desta dissertação. A estrutura da base de dados XML *COrders* é apresentada na Figura 9.

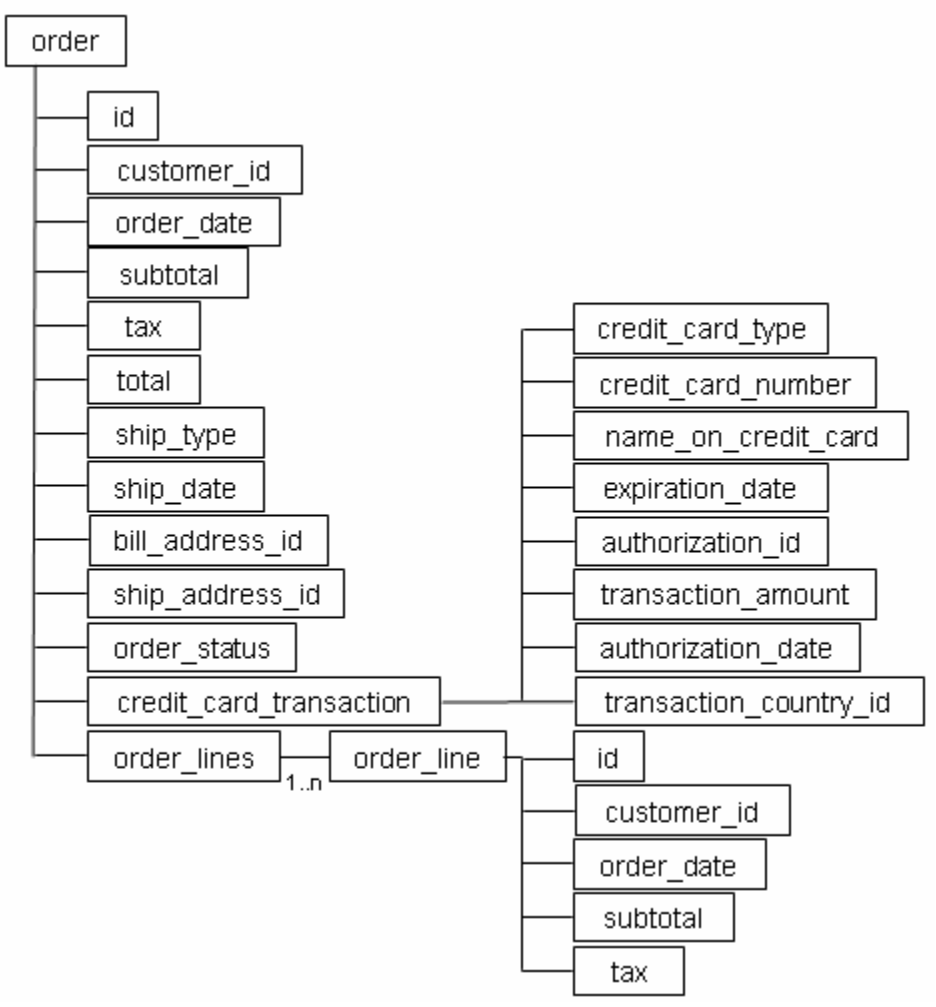


Figura 9 - Estrutura da coleção de documentos XML de pedidos de compra *COrders*

A título de ilustração, considere um cenário P2P composto inicialmente por um número limitado de pontos distribuídos, onde cada ponto possui um banco XML nativo para o armazenamento dos documentos XML que ele gerencia. Vamos supor que este cenário P2P é a rede de sebos e leitores, discutida na introdução, e o banco XML nativo acoplado em cada ponto armazena os documentos XML com os pedidos de compra já efetuados pelo ponto disponível. Caso um usuário queira consultar quais pontos efetuaram uma grande venda de livros com um alto valor total de pedidos, o usuário pode submeter esta consulta em qualquer ponto disponível na rede P2P. Após a submissão da consulta, o ponto origem analisa a consulta, identifica os pontos distribuídos capazes de respondê-la, reformula a consulta de acordo com os dados XML identificados, executa as sub-consultas geradas nos pontos relevantes e finalmente, integra os resultados retornados pelos pontos remotos e apresenta o resultado final ao usuário.

4.2. Especificação dos Fragmentos XML

A fragmentação de uma base de dados XML (ANDRADE et al., 2005) é definida sobre uma coleção homogênea de documentos XML (repositórios MD) ou sobre uma coleção de um único documento XML (repositórios SD) e contempla os três tipos tradicionais de fragmentação: (1) horizontal, onde os dados são agrupados por predicados de seleção; (2) vertical, que divide a estrutura de dados através de projeções; e (3) híbrida, que combina operações de seleção e projeção na sua definição; semelhante à definição de fragmentos proposta para o modelo relacional (ÖZSU et al., 1999).

Um fragmento F é definido em (ANDRADE et al., 2005) como $F := \langle C,\gamma \rangle$, onde C representa uma coleção homogênea de documentos XML (MD) ou uma coleção de um único documento XML (SD) e γ denota uma operação definida sobre C. O fragmento F é horizontal se γ denota uma seleção, vertical se γ é uma projeção ou híbrido, caso a operação γ seja a combinação de operadores de seleção e projeção.

A fragmentação horizontal agrupa os dados que são freqüentemente acessados por um conjunto de consultas com um determinado predicado de seleção. Esta fragmentação implica que todos os fragmentos que compõem a coleção de documentos de um repositório XML possuam apenas predicados de seleção, contendo o mesmo esquema da coleção XML original. O fragmento *COrders_fh1* exemplificado na Figura

10 é definido sobre uma coleção XML *COrders* que contém apenas os documentos XML de *COrders* que satisfazem à condição de */order/total <= 2000*.

$$COrders\_fh1 := \langle C_{orders}, \sigma_{/order/total <= 2000} \rangle$$

Figura 10 - Exemplo de um fragmento XML horizontal

Um detalhe importante observado por (ANDRADE et al., 2005) é o fato de que uma fragmentação horizontal não pode ser aplicada a coleções do tipo SD. Nesses casos, a fragmentação horizontal é impedida pela existência de um único documento XML, pois uma fragmentação horizontal é definida sobre documentos inteiros, ao invés de nodos.

No caso de um repositório XML MD, a fragmentação horizontal somente pode ser aplicada em coleções homogêneas. Isso garante que ao fragmentar o esquema, todos os documentos da coleção XML são fragmentados da mesma maneira, não sendo necessário um tipo de fragmentação para cada documento distinto da coleção.

É importante ressaltar que a metodologia proposta para o processamento de consultas distribuídas em ambientes P2P limita-se apenas à fragmentação horizontal das bases XML. Portanto, a fragmentação vertical, e conseqüentemente a fragmentação híbrida, não são exploradas pela metodologia proposta.

A fragmentação vertical tem como objetivo dividir a estrutura de dados da árvore XML em partes menores de forma a melhorar o desempenho das consultas que acessem somente um sub-conjunto de elementos desta coleção (ANDRADE et al., 2005). Em repositórios do tipo MD, os quais são compostos por múltiplos documentos XML, os fragmentos XML resultantes de uma fragmentação vertical necessitam de uma chave para identificar as sub-árvores geradas do mesmo documento que foram espalhadas pelos fragmentos. Porém, a distinção entre documentos em repositórios do tipo SD não se faz necessária, visto que cada coleção possui apenas um único documento XML associado.

Dentro do contexto P2P, a fragmentação vertical de um repositório MD torna-se inviável devido à natureza dinâmica P2P. Como não se pode obter a representação global dos fragmentos XML distribuídos pela rede P2P, um fragmento XML resultante desta fragmentação vertical não possui informações para localizar os demais fragmentos obtidos pelo mesmo documento, comprometendo, desta forma, a integridade dos dados XML.

Diante desta limitação, a abordagem proposta permite apenas a fragmentação horizontal em repositórios XML de múltiplos documentos (MD). Conforme discutido anteriormente, a fragmentação horizontal não pode ser aplicada em repositórios SD, já que esta fragmentação é definida sobre uma coleção homogênea de documentos XML e não somente sobre um único documento XML.

Dentro do contexto P2P proposto, os fragmentos XML resultantes da fragmentação horizontal em repositórios do tipo MD são distribuídos pela rede P2P e armazenados nos pontos disponíveis, em uma etapa anterior à inicialização do sistema. Conforme definido em Andrade et al. (2005), um fragmento horizontal XML é definido através de seus predicados de seleção. Logo, um fragmento XML contido em um ponto da rede P2P é identificado por um conjunto de expressões de caminho, as quais são definidas através dos predicados de seleção que caracterizam o fragmento XML horizontal.

As definições dos fragmentos XML são armazenadas pelo Catálogo de cada ponto da rede P2P. Desta forma, o Catálogo armazena as informações sobre as bases XML distribuídas conhecidas por um ponto da rede P2P, tais como a referência dos Adaptadores remotos já descobertos e os fragmentos por eles disponibilizados, as definições dos operadores de seleção e as estatísticas disponíveis dos fragmentos XML, dentre outras. Quando um novo ponto ingressa na rede P2P, seu Catálogo irá conter inicialmente somente as informações dos fragmentos existentes na base de dados XML a ele acoplado.

4.3. Metodologia para Processamento de Consultas XQuery Distribuídas em Ambientes P2P

A metodologia de processamento de consultas XQuery distribuídas em ambientes P2P é baseada na proposta elaborada por Figueiredo (2007). Dentro deste contexto, a metodologia proposta sobre ambientes P2P consiste nas etapas de decomposição da consulta; localização dos dados; otimização global e local, conforme ilustrado na Figura 5; além da geração das sub-consultas e sua execução nos pontos remotos; e consolidação dos resultados.

Um dos problemas mais críticos no processamento de consultas distribuídas em um ambiente P2P está relacionado à reformulação da consulta submetida. No contexto

P2P, a consulta não é mais submetida a uma coleção XML específica e, portanto, torna-se necessária a busca por pontos da rede P2P que possuem dados relevantes para a resposta da consulta submetida.

Logo, as etapas de decomposição da consulta e de localização dos dados no processamento de consultas distribuídas, propostas por Figueiredo (2007), precisaram ser adaptadas para a abordagem P2P, as quais são apresentadas em mais detalhes a seguir. Em seguida, as demais etapas do processamento e a etapa final de execução e composição do resultado final também são explicadas.

4.3.1. Decomposição da Consulta

A primeira etapa no processamento de uma consulta distribuída XQuery em ambientes P2P consiste na decomposição da consulta XQuery em uma expressão algébrica, definida através de operadores da álgebra TLC (PAPARIZOS et al., 2004), além da extração das principais expressões de caminho da consulta submetida, para sua posterior utilização na etapa de localização dos dados. O processo de decomposição da consulta XQuery na metodologia proposta é definido através de uma adaptação do processamento de consultas XQuery em ambientes distribuídos (FIGUEIREDO, 2007) e também compreende as seguintes fases, (1) validação semântica e sintática; (2) simplificação da consulta; e (3) representação da consulta em sua forma algébrica. Além disso, a fase (4) de extração das expressões de caminho da consulta foi adicionada na etapa de decomposição da consulta, a fim de auxiliar seu processamento na abordagem P2P. As fases mencionadas da etapa de decomposição da consulta são explicadas a seguir.

Validação Sintática e Semântica. A abordagem proposta utiliza a XQuery (BOAG et al., 2007) como linguagem de consulta nas bases de dados XML, pelo fato de definir consultas com alto grau de expressividade.

Devido à complexidade da linguagem XQuery e para que possamos nos concentrar em aspectos específicos da metodologia proposta para o processamento de consultas distribuídas sobre bases XML em ambientes P2P, iremos utilizar um sub-conjunto suficientemente poderoso da gramática da XQuery original (FIGUEIREDO, 2007), com base principalmente nas expressões *FLWOR* (*FOR*, *LET*, *WHERE*, *ORDER*,

RETURN) da XQuery. As principais operações utilizadas da linguagem XQuery são apresentadas na Tabela 4.

Tabela 4 - Sub-conjunto da gramática XQuery considerada

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)*
ExprSearch	::=	"(some document)"
ExprSingle	::=	FLWORExpr | AndExpr
FLWORExpr	::=	(ForClause | LetClause) + WhereClause? OrderByClause? "return" ExprSingle
ForClause	::=	"for" "$" VarName "in" ExprSearch + ExprSingle
LetClause	::=	"let" "$" 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)*)? ")"

As expressões *FLWOR* fornecem uma sintaxe para a extração de dados de um documento XML simples ou de coleções de documentos XML, executando seleção sobre os dados extraídos e retornando o resultado numa estrutura escolhida pelo usuário. Para realizar a extração dos dados, a linguagem XQuery utiliza um conjunto definido de funções para selecionar os dados requisitados sobre uma coleção ou um documento XML especificado. Por exemplo, considere a expressão *FLWOR* contida na consulta apresentada pela Figura 11.

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

Figura 11 - Exemplo de uma consulta XQuery com a gramática da linguagem suportada

A cláusula *FOR* atua sobre a seqüência de pontos retornados da coleção de documentos XML especificada na consulta (*COrders.xml*), através do uso da expressão de caminho solicitada (*/order*). Porém, indicar uma coleção de documentos XML ou um único documento XML a ser consultado é ineficaz no contexto P2P, já que os usuários não conhecem os dados armazenados nos demais pontos da rede P2P.

Para contornar esta limitação da abordagem P2P, a solução consiste em estender a gramática da linguagem XQuery utilizada para que o documento ou a coleção XML não seja especificado na consulta submetida. Para que a localização dos dados relevantes entre os pontos da rede P2P seja independente do documento especificado na consulta, a expressão *some document* foi adicionada na gramática, conforme já apresentado na Tabela 4. Desta forma, a consulta submetida não fica atrelada a um documento. A Figura 12 ilustra o uso desta expressão na consulta XQuery exemplificada pela Figura 11.

```
<results>
{
        for $order in (some document)/order
        where $order/total > 7000
        order by $order/ship_date, $order/@id
        return
                <order>
                        { $order/@id }
                        { $order/ship_date }
                        { $order/total }
                </order>
}
</results>
```

Figura 12 - Exemplo de uma consulta XQuery com a gramática da linguagem estendida

Note que a consulta submetida não é realizada sobre o documento ou as coleções globais XML. Ao contrário da proposta do Figueiredo (2007), não utilizaremos o Catálogo para validar o documento ou as coleções globais, visto que não sabemos como as bases XML distribuídas estão definidas nos pontos da rede P2P. Desta forma, a consulta não possui uma referência ao documento XML ou coleções XML, pois tais detalhes devem permanecer transparentes para os usuários ao submeter uma consulta no ambiente P2P.

A fim de validar as expressões descritas na linguagem de consulta XQuery, as consultas submetidas são analisadas sintaticamente para verificar se foram escritas de acordo com a gramática suportada, apresentada na Tabela 4. Por exemplo, palavras-chave da linguagem escritas ou posicionadas de forma errada na consulta são consideradas erros de sintaxe. Já a análise semântica verifica se a consulta submetida está de acordo com a estrutura da base de dados XML, especificada pelo esquema apresentado pela Figura 9. Nomes de metadados errados ou inexistentes especificados na consulta são exemplos de erros semânticos.

Durante a realização desta etapa, as consultas analisadas sintática e semanticamente que apresentarem algum tipo de erro em seu processamento são descartadas pelo mediador e retornadas para o cliente para sua possível correção.

Simplificação da Consulta. Após a fase de validação sintática e semântica da consulta submetida, inicia-se a fase de simplificação da consulta. Durante esta fase, a consulta é analisada através de seus predicados de seleção a fim de eliminar os predicados redundantes da consulta.

De forma semelhante à simplificação de consultas sobre o modelo relacional (ÖZSU et al., 1999) e sobre o modelo distribuído (FIGUEIREDO, 2007), a consulta é simplificada através da utilização de um conjunto de regras formais, a fim de eliminar os predicados redundantes pela simplificação da lógica booleana dos predicados de seleção da consulta (cláusula *where* da consulta XQuery), como nos exemplos ilustrado na Tabela 5.

Tabela 5 – Regras para simplificação de predicados de seleção (FIGUEIREDO, 2007)

$p \wedge p \Leftrightarrow p$
$p \vee p \Leftrightarrow p$
$p \wedge true \Leftrightarrow p$
$p \vee false \Leftrightarrow p$
$p \wedge false \Leftrightarrow false$
$p \vee true \Leftrightarrow true$
$p \wedge \neg p \Leftrightarrow false$
$p \vee \neg p \Leftrightarrow true$
$p1 \wedge (p1 \vee p2) \Leftrightarrow p1$
$p1 \vee (p1 \wedge p2) \Leftrightarrow p1$

Representação da Consulta em sua Forma Algébrica. A próxima fase da etapa de decomposição da consulta submetida consiste na transformação de sua representação textual em uma expressão algébrica consistente, definida através da álgebra TLC (PAPARIZOS et al., 2004). Para tal, a representação de uma consulta em sua forma algébrica consiste na execução de um algoritmo, que analisa sintaticamente a consulta e monta as operações algébricas, os padrões de árvores e as classes lógicas da TLC, conforme explicado em (FIGUEIREDO, 2007).

Ao invés da representação algébrica da consulta sobre as coleções XML globais, a representação da consulta submetida em sua forma algébrica no ambiente P2P é definida sobre o elemento *some document*, a fim de construir uma expressão algébrica genérica o suficiente para ser submetida em um ambiente P2P. Este fato pode ser explicado pela vantagem de autonomia de armazenamento da abordagem P2P, que se

refere à liberdade que o ponto possui para armazenar seus próprios dados de interesse, sem nenhuma especificação de como catalogar seus documentos XML armazenados.

Logo, o produto final desta fase é a representação algébrica inicial sobre um elemento genérico, que será utilizada como entrada para a próxima etapa de localização dos dados no processamento distribuído da consulta. A Figura 13 ilustra a representação algébrica TLC inicial da consulta XQuery especificada.

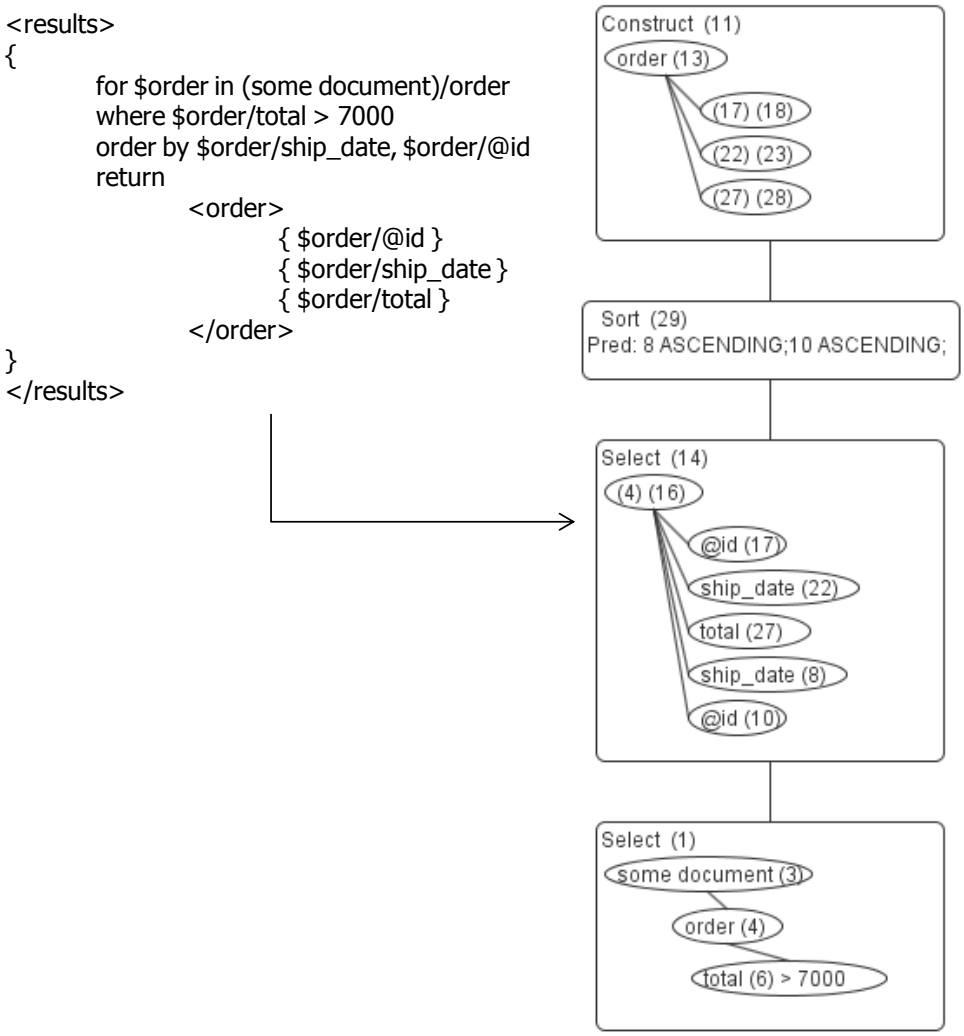


Figura 13 - Representação da consulta XQuery em sua forma algébrica inicial

Extração das Expressões de Caminho da Consulta. As expressões de caminho da linguagem de consulta XQuery são baseadas nas expressões de caminho da linguagem XPath (BERGLUNG et al., 2007). Uma expressão de caminho fornece um meio de endereçar partes específicas de um documento XML através de um caminho para o conteúdo de interesse na árvore do documento XML.

Em uma consulta XQuery, as expressões de caminho são usadas para extrair informações dos dados XML de entrada. As condições para os relacionamentos entre os elementos da consulta também são combinadas a fim de formar uma consulta de caminhos (expressão de caminho) dentro do contexto de pesquisa. Além disso, os operadores de retorno são usados para selecionarem pontos específicos das expressões de caminho, os quais são retornados pela consulta. Ou seja, as expressões de caminho são encontradas nas cláusulas de entrada, predicados e cláusulas de retorno de uma consulta XQuery. Logo, uma consulta XQuery possui um conjunto de expressões de caminho em XPath a ela associada.

Em virtude da definição da nova expressão *some document*, a qual substitui a especificação do documento ou coleção XML a ser consultada, a localização dos dados relevantes para o resultado da consulta no ambiente P2P será realizada baseada nas principais expressões de caminho resultantes da consulta XQuery submetida.

Através de um analisador (*parser*) na consulta XQuery, uma lista de expressões de caminho é extraída da consulta XQuery submetida. Para tal, a metodologia proposta utiliza dois tipos de expressões de caminho descritas na linguagem XPath para a tradução da consulta XQuery submetida. Os tipos de expressões de caminho são:

- **Expressão de caminho simples**: As expressões de caminho simples de uma XQuery são extraídas automaticamente das cláusulas de entrada e saída da consulta submetida. Como neste tipo de expressão de caminho não são definidos nenhum critério de seleção específico sobre determinado grupo de elementos, a expressão de caminho simples é usada apenas para percorrer o documento XML e extrair um conjunto de nodos XML especificado pelo último passo do caminho. Para a consulta exemplifica pela Figura 12, as expressões de caminho simples geradas neste caso estão representadas na Tabela 6.

Tabela 6 - Expressões de caminho simples resultantes da consulta XQuery

Cláusula	Expressão de Caminho
Entrada	*/order*
Saída	*/order/@id* */order/ship_date* */order/total*

- **Expressão de caminho com filtro**: As expressões de caminho com filtro são extraídas dos predicados de seleção definidos na consulta XQuery submetida. Como neste tipo de expressão são definidos critérios de seleção específicos sobre determinado elemento ou grupo de elementos, cada predicado de seleção encontrado na consulta XQuery é incluso em um colchete e associado ao ultimo nó da expressão de caminho. Para a consulta ilustrada pela Figura 12, a expressão de caminho com filtro gerada está contida na Tabela 7.

Tabela 7 - Expressão de caminho com filtro resultante da consulta XQuery

Cláusula	Expressão de Caminho
Predicado	*/order[total > 7000]*

Além disso, as expressões de caminho são classificadas quanto a sua relevância para localizar os dados entre os pontos da rede P2P. Dentre as expressões de caminho obtidas de uma consulta XQuery, uma prioridade é definida apenas nas expressões de caminho que apresentam um alto grau de importância para a localização de dados relevantes.

Como um fragmento XML é definido através de seus predicados de seleção, as expressões de caminho obtidas a partir dos critérios de seleção da consulta submetida são mais relevantes para a etapa de localização dos dados (expressões de caminho com filtros), quando comparadas às expressões de caminho resultantes das cláusulas de entrada e saída da consulta (expressões de caminho simples). Caso a consulta submetida não possua nenhum critério de seleção, as expressões de caminho obtidas das cláusulas de entrada serão consideradas as mais relevantes para a etapa de localização dos dados.

O Algoritmo 1, representado pela Figura 14, ilustra o procedimento realizado para especificar as prioridades das expressões de caminho extraídas da consulta XQuery submetida. A fim de aumentar o domínio da busca para a etapa de localização dos dados, cada expressão de consulta considerada como relevante é decomposta em expressões de caminho simples e classificada de acordo com a sua prioridade para localizar os dados relevantes. Mais precisamente, as expressões de caminho simples são extraídas de cada expressão de caminho da consulta considerada como relevante de forma a decompor a expressão de caminho da consulta em apenas um nó resultante.

Desta forma, as expressões de caminho decompostas possuem uma prioridade menor para a localização dos dados quando comparadas à expressão de caminho original.

Algoritmo 1 – Procedimento para especificar as prioridades das expressões de caminho

```
 1: procedure setPrioritiesPaths (queryPaths)
 2:     priority ← 0;
 3:     prioritiesPaths ← null;
 4:     paths ← queryPaths.getQueryPaths(predicatesPaths);
 5:     if paths is empty then
 6:         paths ← queryPaths.getQueryPaths(inoutPaths);
 7:     end if
 8:     for path in paths then
 9:         relevantPaths ← returnPathsByPriorities(path);
10:         simplePath ← path.getSimplePath();
11:         while simplePath is not null do
12:             prioritiesPaths.add(simplePath, priority);
13:             simplePath ← path.getSimplePath();
14:             priority ← priority + 1;
15:         end while
16:     end for
17:     return prioritiesPaths;
18: end procedure
```

Figura 14 – Algoritmo 1: Procedimento para especificar as prioridades das expressões de caminho

Para exemplificar o Algoritmo 1 descrito, a Figura 15 apresenta as prioridades definidas para as expressões de caminho relevantes da consulta exibida pela Figura 12. A expressão de caminho */order[total > 7000]* extraída da consulta é considerada como relevante para a localização dos dados entre os pontos da rede P2P, de acordo com a Tabela 7, e classificada com a maior prioridade para localizar os dados relevantes (*Prioridade 0*). Após a classificação, esta expressão de caminho é então decomposta na expressão de caminho simples */order/total*, através da eliminação dos colchetes e da associação do nó eliminado à expressão de caminho recortada (ocorre apenas quando uma expressão de consulta contém filtros), além de ser classificada com uma prioridade inferior para localizar os dados relevantes (*Prioridade 1*). Como este processo é realizado até a expressão de caminho resultante ser composta por apenas um nó, a expressão de caminho resultante é finalmente decomposta na expressão de caminho simples */order*, através da eliminação do último nó associado à expressão de caminho

(ocorre apenas em expressão de consulta simples), e é classificada com a menor prioridade (*Prioridade 2*).

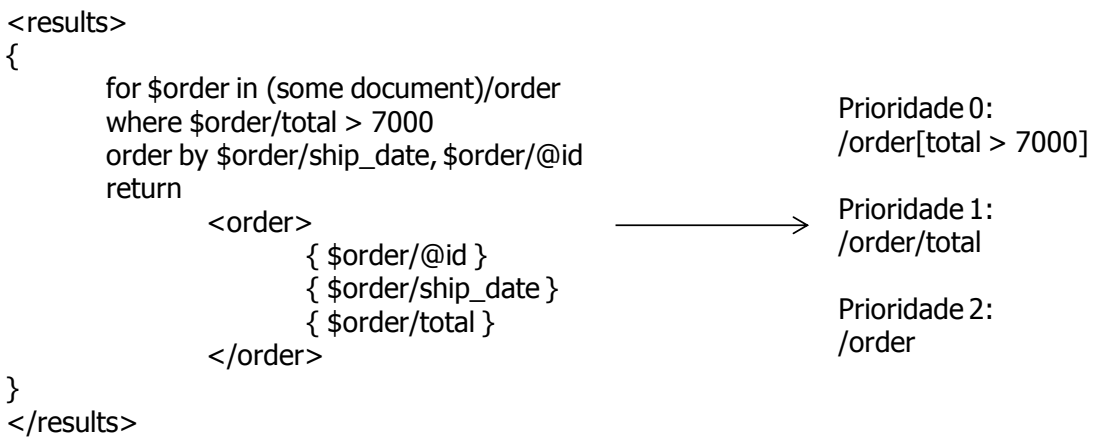


Figura 15 - Prioridade das principais expressões de caminho da consulta XQuery

4.3.2. Localização dos Dados

Os produtos finais obtidos da primeira etapa do processamento distribuído em ambientes P2P compreendem a representação da consulta em sua forma algébrica TLC e as principais expressões de caminho extraídas da consulta submetida, em conjunto com suas prioridades definidas para a localização dos dados relevantes.

Levando em consideração os produtos finais obtidos, a etapa de localização dos dados no processamento de uma consulta XQuery distribuída em um ambiente P2P engloba as seguintes fases, (1) localização dos pontos da rede P2P que possuem dados relevantes para o resultado da consulta, através das prioridades obtidas pelas principais expressões de caminho identificadas; (2) inclusão das referências aos fragmentos XML encontrados nos pontos relevantes da rede P2P, dentro do plano algébrico da consulta; e (3) redução dos fragmentos XML redundantes de acordo com os predicados de seleção da consulta original. As informações contidas no Catálogo de cada ponto da rede P2P serão necessárias durante a execução destas fases para a localização dos dados.

Em relação à primeira fase da etapa em questão, a localização dos pontos relevantes da rede P2P para a consulta submetida pode ser realizada a partir de diversas técnicas levantadas: (1) através de uma mensagem de *broadcast* para um número limitado de pontos na rede P2P; (2) através da utilização de um índice distribuído entre

os pontos da rede P2P; ou (3) através das informações disponíveis do Catálogo atualizado do ponto solicitante da consulta.

As técnicas para a localização dos dados na rede P2P são apresentadas em mais detalhes nas próximas subseções, enquanto que as demais fases desta etapa finalizam a seção.

4.3.2.1. Técnica I de Localização dos Dados: *Broadcast*

A primeira técnica identificada para localizar dados relevantes da rede P2P propaga suas requisições através de mecanismos de inundação (*flooding*) controlada por um parâmetro TTL (*time-to-live*) contido nas mensagens de *broadcast*, que define o número máximo de pontos alcançáveis dentro da rede P2P para receber uma requisição.

No contexto de consultas XQuery submetidas, as expressões de caminho extraídas da consulta submetida, consideradas como principais para localizar os dados relevantes entre os pontos da rede P2P, são classificadas quanto a uma prioridade para a localização dos dados, conforme explicado na Seção 4.3.1.

Em relação à recuperação dos fragmentos XML distribuídos pela rede P2P, a pesquisa por pontos relevantes consiste em verificar se as expressões de caminho extraídas da consulta submetida satisfazem às expressões de caminho obtidas a partir dos predicados de seleção resultantes dos fragmentos XML armazenados no Catálogo.

Através das prioridades obtidas pelas principais expressões de caminho identificadas na consulta submetida, a busca por dados relevantes é executada para o conjunto de expressões de caminho que compõem a prioridade em questão. Desta forma, uma mensagem de *broadcast*, contendo todas as expressões de caminho da prioridade vigente, é roteada para os pontos vizinhos do ponto solicitante da requisição. O Algoritmo 2, representado pela Figura 16, ilustra o procedimento executado pelo ponto solicitante da consulta para localizar os dados relevantes, através de uma mensagem de *broadcast* para seus pontos vizinhos da rede P2P.

Enquanto a mensagem de *broadcast* é propagada pelos pontos da rede P2P, o ponto solicitante torna-se disponível para receber os fragmentos XML, os quais são enviados como resposta dos pontos relevantes da rede P2P. Caso nenhum fragmento XML seja identificado pela pesquisa executada com a prioridade máxima definida, a prioridade é reduzida e a busca é então reiniciada para as expressões de caminho

associadas com a nova prioridade em questão. A busca somente é finalizada caso algum fragmento XML seja identificado pelas expressões de caminho ou se nenhum fragmento XML for localizado pela prioridade mínima possível.

Caso algum fragmento XML seja retornado por um ponto relevante da rede P2P, o Catálogo do ponto solicitante da consulta é alterado através da inserção ou da atualização dos fragmentos XML retornados pela busca. Mas caso nenhum fragmento XML seja identificado como relevante para o resultado da consulta, o usuário final receberá uma mensagem de aviso informando a limitação encontrada para a consulta submetida.

Algoritmo 2 – Procedimento para localizar os dados relevantes pela Técnica I (executado pelo ponto solicitante da consulta)

```
1: procedure findFragmentsByBroadcast (prioritiesPaths)
2:     priority ← 0;
3:     relevantFragments ← null;
4:     while priority <= prioritiesPaths.size() do
5:         relevantPaths ← prioritiesPaths.get(priority);
6:         broadcastPathsNeighborsPeers(relevantPaths);
7:         while waitResponseNeighborsPeers() do
8:             relevantFragments ← relevantPeers.getFragmentsXML();
9:         end while
10:        if relevantFragments is not empty then
11:            break;
12:        end if
13:        priority ← priority + 1;
14:    end while
15:    if relevantFragments is empty then
16:        return null;
17:    end if
18:    newCatalogPeer ← reloadCatalogPeerByBroadcast(relevantFragments);
19:    return newCatalogPeer;
20: end procedure
```

Figura 16 - Algoritmo 2: Procedimento da Técnica I executado pelo ponto solicitante da consulta

Cada ponto descoberto pela mensagem de *broadcast* propagada na rede P2P processa a mensagem recebida através da extração das expressões de caminho nela contida, além de decrementar o parâmetro TTL da mensagem. O Algoritmo 3, ilustrado

pela Figura 17, apresenta o procedimento executado por cada ponto alcançado pela mensagem de *broadcast*.

Algoritmo 3 – Procedimento para localizar os dados relevantes pela Técnica I (executado por um ponto da rede P2P)

```
1: procedure retrieveFragmentsByBroadcast (relevantPaths)
2:     relevantFragments ← false;
3:     timeToLive ← timeToLive - 1;
4:     if timeToLive >= 0 then
5:         fragmentsXML ← catalogPeer.getFragmentsXML();
6:         for fragment in fragmentsXML then
7:             relevantPeer ← checkPathsByPredicates(relevantPaths, fragment);
8:             if relevantPeer then
9:                 relevantFragments ← true;
10:                routeDirectFragmentXML(fragment);
11:            end if
12:        end for
13:    end if
14:    if (not relevantFragments) and (timeToLive > 0) then
15:        broadcastPathsNeighborsPeers(relevantPaths);
16:    end if
17: end procedure
```

Figura 17 - Algoritmo 3: Procedimento da Técnica I executado por um ponto da rede P2P

Para cada fragmento XML pertencente ao Catálogo do ponto atingido pela requisição, uma comparação é feita entre as expressões de caminho obtidas da mensagem recebida e os predicados de seleção do fragmento XML. Se alguma expressão de caminho for respondida pelo fragmento XML, o ponto responsável pelo Catálogo é considerado como relevante e uma mensagem de resposta é enviada diretamente ao ponto origem da consulta, contendo a especificação do fragmento XML associado. Ou seja, o catálogo do ponto solicitante da consulta é atualizado com a nova referência proveniente do fragmento XML relevante. Caso contrário, o ponto redireciona recursivamente a mensagem de *broadcast* para todos os pontos de sua lista de vizinhos, de acordo com o valor do parâmetro TTL. Porém, as mensagens recebidas pelos pontos da rede P2P que apresentam valor TTL igual a zero não são encaminhas para seus pontos vizinhos.

4.3.2.2. Técnica II de Localização dos Dados: *DHT*

A segunda técnica identificada para localizar dados relevantes da rede P2P é realizada através de tabelas *hash* distribuídas. Conforme apresentada anteriormente (ver Seção 2.3.3), uma tabela *hash* distribuída é um conjunto de tabelas distribuídas entre os pontos da rede P2P, compostas por chaves numéricas únicas e valores armazenados, sendo os valores compostos pelo conteúdo do dado ou por um ponteiro para o local onde o dado realmente se encontra. Os valores armazenados devem ser endereçados por chaves numéricas únicas geradas através do uso de uma função *hash*. Desta forma, qualquer ponto da rede P2P estruturada pode usar a DHT para recuperar eficientemente o valor armazenado pela chave numérica única.

Conforme apresentado na Seção 4.1.2, um fragmento XML é definido através dos predicados de seleção obtidos pela fragmentação horizontal da base XML. Com isso, expressões de caminho podem ser extraídas a partir dos critérios de seleção definidos no fragmento XML. Logo, um fragmento XML pode ser identificado através das expressões de caminho resultantes de seus predicados de seleção.

Com o objetivo de aumentar o domínio na busca por dados relevantes da consulta submetida, as expressões de caminho que caracterizam um fragmento XML são decompostas em expressões menores. Cada expressão de caminho resultante da decomposição em conjunto com as expressões de caminho originais de um fragmento XML é representada por uma seqüência de caracteres que pode ser transformada em uma única chave numérica. Assim, a expressão de caminho convertida em uma chave numérica única representa uma entrada na tabela *hash* distribuída.

Então, a entrada na DHT é composta pela expressão de caminho identificadora, convertida em uma chave numérica única, em conjunto com os fragmentos XML associados à expressão de caminho inserida. Caso o fragmento XML seja definido por mais de uma expressão de caminho, todas as entradas inseridas na DHT serão compostas pelo mesmo fragmento XML. Considerando o fragmento XML representado pelo Catálogo da Figura 10, as entradas na DHT correspondentes às expressões de caminho resultantes do fragmento XML são apresentadas na Tabela 8.

Tabela 8 - Tabela de entradas na DHT de um fragmento XML

Expressão de Caminho	Fragmento XML
/order/total <= 2000	* Visão Local [0]: Corders_fh1.xml - Predicados de Seleção: - /order/total <= 2000 - Site do Adaptador: - http://peer:9091/WrapperExist/services/XQueryWrapper
/order/total	* Visão Local [0]: Corders_fh1.xml - Predicados de Seleção: - /order/total <= 2000 - Site do Adaptador: - http://peer:9091/WrapperExist/services/XQueryWrapper
/order	* Visão Local [0]: Corders_fh1.xml - Predicados de Seleção: - /order/total <= 2000 - Site do Adaptador: - http://peer:9091/WrapperExist/services/XQueryWrapper

A tabela *hash* distribuída disponibiliza duas operações básicas, (1) inserir e (2) recuperar, respectivamente exemplificados através dos algoritmos (1) *insert* e (2) *retrieve* descritos a seguir. É importante notar que ambos os algoritmos são executados localmente em cada ponto.

Dentro do contexto de fragmentos XML, uma entrada pode ser inserida na DHT através de uma expressão de caminho associada ao seu fragmento XML; e um fragmento XML pode ser recuperado de uma entrada DHT a partir de uma expressão de caminho especificada.

Algoritmo 4 – Procedimento para inserir uma expressão de caminho na DHT

```
1: procedure insert (path, fragmentXML)
2:     hashPath ← buildHashPath(path);
3:     existingHashPath ← lookup(hashPath);
4:     waitResponse();
5:     if existingHashPath is empty then
6:         insert(hashPath, fragmentXML);
7:     else
8:         existingHashPath.update(fragmentXML);
9:     end if
10: end procedure
```

Figura 18 - Algoritmo 4: Procedimento da Técnica II para inserir dados na DHT

A inserção de entradas na tabela *hash* distribuída é realizada no momento em que um ponto entra na rede P2P. Assim como é realizada a tradução da consulta XQuery em expressões de caminho (ver Seção 4.3.1), os predicados de seleção do fragmento XML também são representados por expressões de caminho simples e expressões de caminho com filtros. Através de um analisador (*parser*) no Catálogo do ponto adicionado, as expressões de caminho resultantes do fragmento XML são inseridas na DHT. Cada expressão de caminho inserida na DHT é associada à especificação do fragmento XML analisado, o qual é armazenado como o valor associado da expressão de caminho.

Desta forma, o Algoritmo 4, representado pela Figura 18, é executado para cada expressão de caminho decomposta do fragmento XML. Caso a expressão de caminho selecionada já exista como entrada na DHT, o seu valor associado é atualizado com a adição do novo fragmento XML analisado.

Em relação à recuperação dos fragmentos XML da tabela *hash* distribuída, a busca realizada consiste em analisar se as expressões de caminho, em conjunto com suas prioridades para localização de dados definidas, são utilizadas na definição dos fragmentos XML.

Então, dada uma lista de expressões de caminho extraídas de uma consulta XQuery com suas prioridades definidas, a busca por dados relevantes é executada para cada expressão de caminho que compõe a prioridade em questão. Caso nenhum fragmento XML seja retornado pela busca por dados relevantes com a prioridade máxima definida, a prioridade é reduzida e a busca é novamente realizada para cada expressão de caminho definida com essa nova prioridade.

Considerando que a busca é executada para todas as expressões de caminho associadas a cada prioridade, o Algoritmo 6 (Figura 20) em conjunto com Algoritmo 5 (Figura 19), são executados até que a prioridade mínima seja atingida ou até o momento em que fragmentos XML relevantes sejam identificados para a prioridade utilizada. Se algum fragmento XML relevante for encontrado, o Catálogo do ponto solicitante da consulta será alterado através da inserção dos fragmentos XML retornados pela busca, caso não existam; ou através da atualização das informações dos fragmentos XML retornados pela busca, caso já existam no Catálogo. Caso nenhum fragmento XML seja identificado na etapa de localização de dados, o usuário final receberá uma mensagem de aviso para que ele possa reformular a consulta submetida.

Algoritmo 5 – Procedimento para recuperar fragmentos XML da DHT

```
 1: procedure retrieve (path)
 2:     hashPath  ← buildHashPath(path);
 3:     existHashPath ← lookup(hashPath);
 4:     waitResponse();
 5:     if existHashPath is empty then
 6:         return null;
 7:     end if
 8:     fragmentsXML ← existHashPath.getFragmentsXML();
 9:     return fragmentsXML;
10: end procedure
```

Figura 19 - Algoritmo 5: Procedimento da Técnica II para recuperar dados da DHT

Algoritmo 6 – Procedimento para localizar os dados relevantes pela Técnica II

```
 1: procedure findFragmentsByDHT (prioritiesPaths)
 2:     priority ← 0;
 3:     relevantFragments  ← null;
 4:     while priority  <= prioritiesPaths.size() do
 5:         relevantPaths ← prioritiesPaths.get(priority);
 6:         for path in revelantPaths do
 7:             existPath ← retrieve(path);
 8:             if existPath is not empty then
 9:                 revelantFragments ← existPath.getFragmentsXML();
10:             end if
11:         end for
12:         if relevantFragments is not empty then
13:             break;
14:         end if
15:         priority ← priority + 1;
16:     end while
17:     if relevantFragments is empty then
18:         return null;
19:     end if
20:     newCatalogPeer ← reloadCatalogPeerByDHT(relevantFragments);
21:     return newCatalogPeer;
22: end procedure
```

Figura 20 - Algoritmo 6: Procedimento da Técnica II para localizar os dados relevantes

Uma das limitações encontradas na arquitetura estruturada baseada em tabelas *hash* distribuídas proposta é a baixa flexibilidade na recuperação dos fragmentos XML, uma vez que para recuperar um fragmento XML armazenado, é necessário fornecer

exatamente a chave numérica única utilizada para gerar a entrada na tabela *hash* distribuída. Esta limitação explica o fato pelo qual tanto a consulta XQuery submetida como os fragmentos XML distribuídos são decompostos em várias expressões de consultas, incluindo as expressões de caminho contendo apenas um prefixo inicial até a expressão de caminho completa.

4.3.2.3. Técnica III de Localização dos Dados: Catálogo

A terceira técnica identificada para localizar dados relevantes da rede P2P é realizada através dos fragmentos XML contidos no Catálogo do ponto origem da consulta. De forma semelhante ao que é realizado nas técnicas identificadas já citadas, a busca é realizada pelo ponto origem da consulta, porém apenas seu Catálogo é consultado para a localização dos dados relevantes, conforme descrito no Algoritmo 7 da Figura 21.

Algoritmo 7 – Procedimento para localizar os dados relevantes pela Técnica III

```
1: procedure findFragmentsByCatalog (prioritiesPaths)
2:     priority ← 0;
3:     relevantFragment  ← true;
4:     fragmentsXML ← catalogPeer.getFragmentsXML();
5:     while priority  <= prioritiesPaths.size() do
6:         relevantPaths ← prioritiesPaths.get(priority);
7:         for fragment in fragmentsXML then
8:             relevantPeer ← checkPathsByPredicates(relevantPaths, fragment);
9:             if relevantPeer then
10:                relevantFragment ← true;
11:                reloadCatalogPeer();
12:            end if
13:        end for
14:        if relevantFragment then
15:            break;
16:        end if
17:        priority ← priority + 1;
18:    end while
19: end procedure
```

Figura 21 - Algoritmo 7: Procedimento da Técnica III para localizar os dados relevantes

Dada uma lista de expressões de caminho extraídos da consulta XQuery submetida com suas prioridades definidas, a busca é executada para o conjunto de expressões de caminho da prioridade vigente. Se qualquer expressão de caminho da prioridade em questão for respondida por algum fragmento XML do Catálogo no ponto origem da consulta, o seu Catálogo é utilizado para responder à consulta submetida. Caso nenhum fragmento XML seja identificado pela prioridade máxima definida, a prioridade é reduzida e reiniciada para o novo conjunto de expressões de caminho.

Desta forma, a pesquisa é realizada para todos os fragmentos XML do Catálogo do ponto solicitante da consulta até identificar os fragmentos XML relevantes para a prioridade utilizada ou até que a prioridade mínima seja atingida. Caso nenhum fragmento XML seja identificado como relevante para o resultado da consulta, o usuário final receberá uma mensagem de aviso para que ele possa reformular a consulta submetida.

4.3.2.4. Inclusão e Redução dos Fragmentos XML

A representação algébrica da consulta submetida no ambiente P2P é definida sobre um elemento genérico (ver Seção 4.3.1), a fim de construir uma expressão algébrica abrangente o suficiente para ser submetida em um ambiente P2P. Através da aplicação de uma das técnicas de localização de dados anteriormente explicadas, as referências aos fragmentos XML identificados são inseridas no plano algébrico da consulta. Desta forma, as coleções XML globais são especificadas na forma algébrica da consulta submetida.

A Tabela 9 apresenta a especificação de seis fragmentos XML obtidos através da fragmentação horizontal da coleção XML *COrders*. Estas especificações de fragmentos XML foram retornadas por pontos distribuídos na rede P2P identificados como relevantes para o resultado da consulta mostrada pela Figura 12.

Logo, podemos concluir que a localização por fragmentos XML relevantes não foi realizada com a prioridade máxima dentre as expressões de caminho extraídas da consulta submetida, visto que o critério de seleção da consulta (*/order[total > 7000]*) não corresponde a nenhum predicado de seleção dos fragmentos XML descobertos na rede P2P. Desta forma, todos os fragmentos XML representados pela Tabela 9

identificados como relevantes foram localizados através de uma expressão de caminho com menor prioridade (*/order/total*).

Tabela 9 - Exemplos de fragmentos XML distribuídos pela rede P2P

Pontos P2P	Fragmentos XML
Ponto 1	COrders_fh1 := $\langle C_{orders}, \sigma_{/order/total <= 2000} \rangle$
Ponto 2	COrders_fh2 := $\langle C_{orders}, \sigma_{/order/total > 2000 \wedge /order/total <= 4000} \rangle$
Ponto 3	COrders_fh3 := $\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total <= 6000} \rangle$
Ponto 4	COrders_fh4 := $\langle C_{orders}, \sigma_{/order/total > 6000 \wedge /order/total <= 8000} \rangle$
Ponto 5	COrders_fh5 := $\langle C_{orders}, \sigma_{/order/total > 8000 \wedge /order/total <= 10000} \rangle$
Ponto 6	COrders_fh6 := $\langle C_{orders}, \sigma_{/order/total < 10000} \rangle$

As referências dos fragmentos XML ilustrados pela Tabela 9 são então inseridos no plano algébrico da consulta solicitada. A Figura 22 apresenta o resultado da localização da operação de seleção da consulta submetida sobre as seis especificações de coleções XML retornadas por cada ponto relevante. Neste exemplo, cinco operações de união da álgebra TLC são criadas para ligar os seis fragmentos XML relevantes.

Após a inserção das coleções XML globais dos pontos relevantes identificados pela primeira fase da etapa de localização dos dados, a próxima fase consiste em eliminar os fragmentos XML irrelevantes para o resultado da consulta original submetida. O processo de redução dos fragmentos XML irrelevantes analisa a operação de seleção aplicada sobre a coleção XML global, a fim de identificar se os critérios de seleção da consulta não contradizem os predicados de seleção do fragmento XML relacionado. Predicados incompatíveis podem ser eliminados do plano algébrico da consulta submetida através da remoção das operações algébricas sobre os fragmentos XML inúteis para o resultado da consulta.

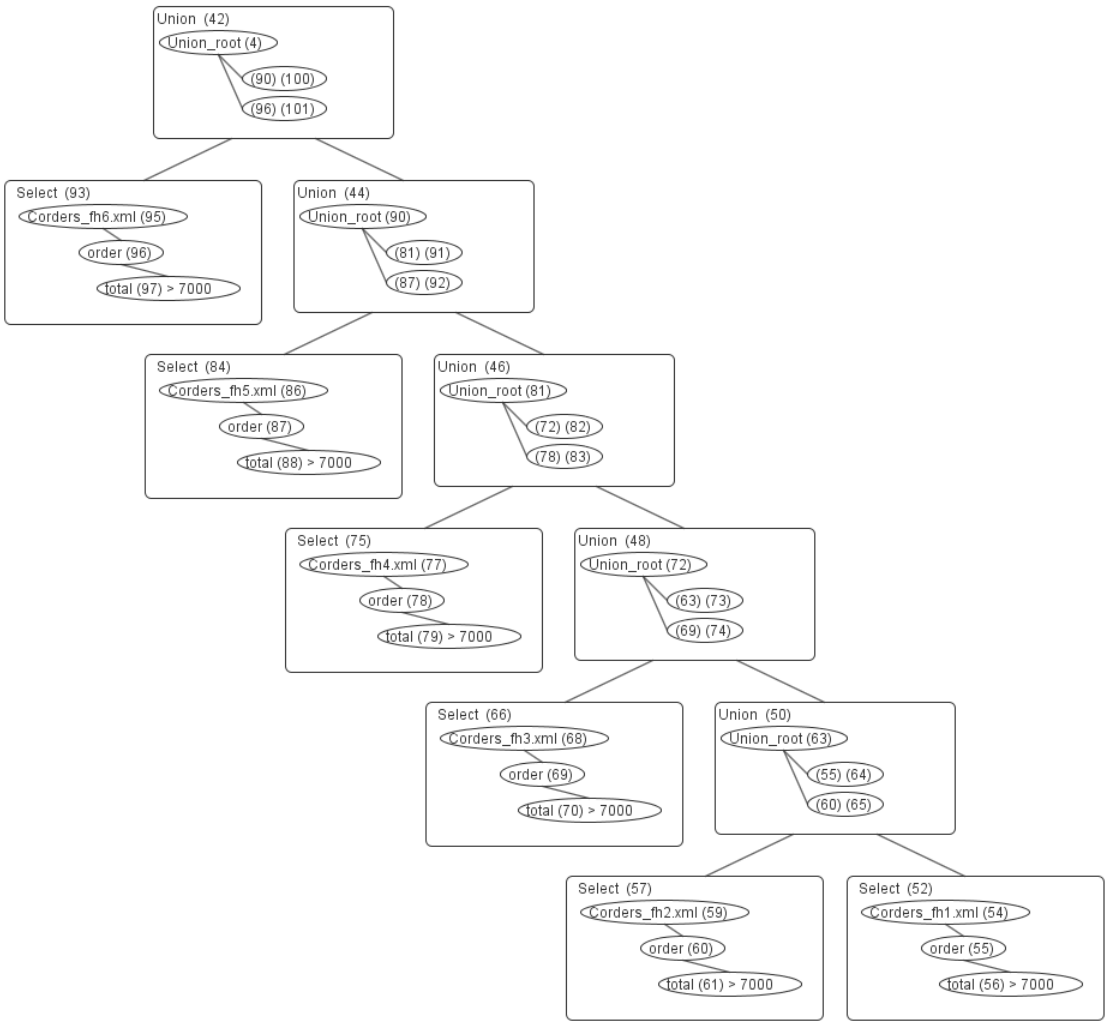
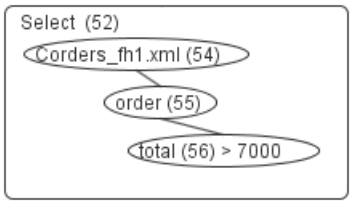
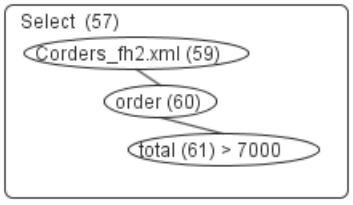
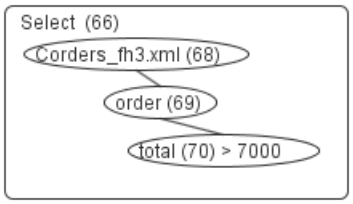
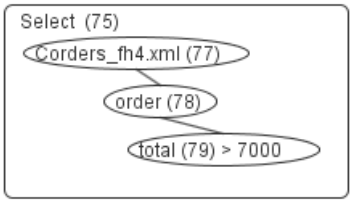
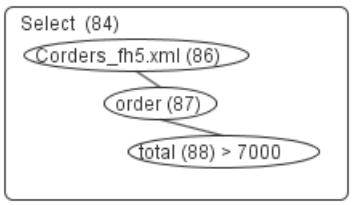
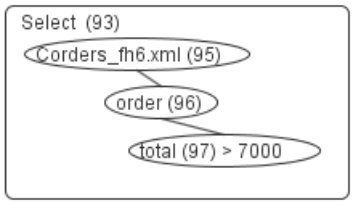


Figura 22 – Localização da coleção XML global de cada ponto relevante da rede P2P

A Tabela 10 exemplifica a redução de três fragmentos XML irrelevantes para o resultado da consulta exibida pela Figura 12, de acordo com as definições dos fragmentos contidas na Tabela 9. Neste exemplo, os fragmentos dos Pontos 1, 2 e 3 não satisfazem o critério de seleção da consulta submetida e por isso são considerados inúteis para o resultado final da consulta, já que a consulta original seleciona pedidos de compra (*orders*) com valor total maior do que 7000.

Tabela 10 - Redução dos Fragmentos XML

Operador de Seleção	Predicado de Seleção do Fragmento XML	Fragmento XML Irrelevante?
	*/order/total <= 2000*	Sim
	*/order/total > 2000* ^ */order/total <= 4000*	Sim
	*/order/total > 4000* ^ */order/total <= 6000*	Sim
	*/order/total > 6000* ^ */order/total <= 8000*	Não
	*/order/total > 8000* ^ */order/total <= 10000*	Não
	*/order/total > 10000*	Não

A remoção de uma operação de seleção sobre um fragmento do plano algébrico implica também na remoção da operação de união superior ao fragmento eliminado, realocando a operação de seleção relacionada ao operador reduzido. Desta forma, a representação da consulta, exibida pela Figura 12, em sua forma algébrica reduzida é mostrada pela Figura 23.

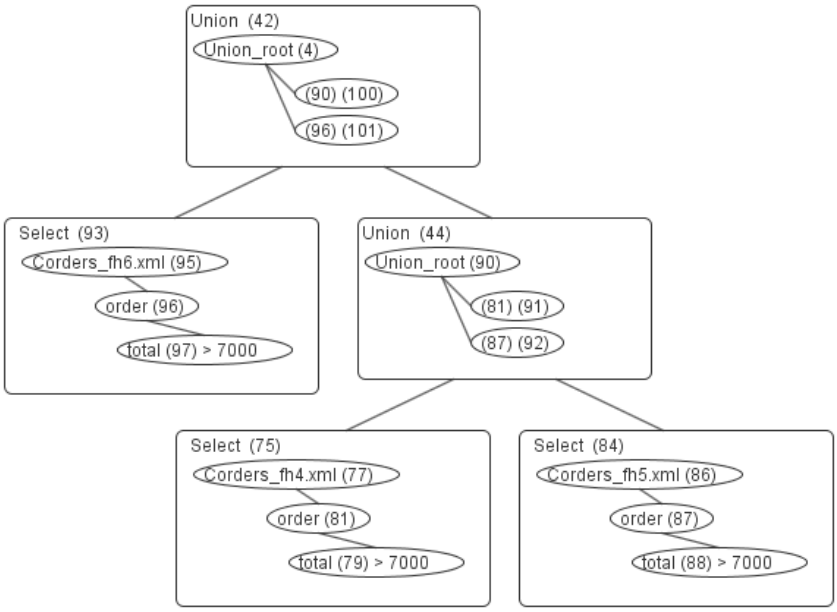


Figura 23 - Representação da consulta XQuery em sua forma algébrica reduzida

4.3.3. Otimização da Consulta

As próximas etapas no processamento de uma consulta XQuery distribuída em ambientes P2P compreendem: (1) a otimização global do plano algébrico reduzido da consulta submetida e (2) a otimização local da sub-consulta executada pelo ponto remoto.

A etapa de otimização global da consulta consiste em obter um plano algébrico de custo mínimo a partir da aplicação de uma função de custo para determinar o melhor plano algébrico, dentre as variações equivalentes do plano algébrico reduzido obtido pela etapa anterior, geradas através de transformações algébricas.

O plano algébrico equivalente gerado com o menor custo será escolhido para execução da consulta distribuída. Esta função de custo deverá utilizar e estimar parâmetros para o cálculo do custo de cada operação do plano, a fim de obter o custo total do plano algébrico, tais como o volume de dados processados pela operação, as estimativas de volume de dados retornados por uma operação, o custo de transmissão de dados pela rede P2P, dentre outros parâmetros.

O objetivo final da etapa de otimização global de consultas é encontrar um plano de execução para a consulta que esteja próximo do plano ótimo. Cada operação do plano algébrico otimizado deverá conhecer o local (*site*) onde será executada, que pode ser o próprio ponto ou um ponto remoto identificado. O plano algébrico final será

utilizado para a montagem das sub-consultas transformadas para a linguagem de consulta XQuery, que executarão todas as operações direcionadas a um mesmo ponto. Caso mais de um ponto seja envolvido na consulta, será necessária a consolidação dos resultados retornados pelos pontos participantes.

Finalmente, a etapa de otimização local da consulta é realizada pelos pontos remotos, os quais contêm os fragmentos XML envolvidos no plano algébrico otimizado da consulta submetida. Cada sub-consulta XQuery enviada ao ponto remoto é localmente otimizada com o uso do esquema local do ponto e é executada através do próprio banco de dados XML nativo que armazena o fragmento XML.

4.3.4. Execução e Composição do Resultado Final

A última etapa no processamento de uma consulta XQuery distribuída em um ambiente P2P corresponde a (1) execução das consultas remotas e a (2) composição do resultado final.

O plano algébrico localizado, reduzido e otimizado obtido pela etapa anterior de otimização da consulta é utilizado para a execução da consulta distribuída, pois cada operação do plano algébrico otimizado é identificada pelo seu local de execução. A partir disso, as operações algébricas são identificadas por grupos, onde cada grupo irá conter operações a serem executadas em um mesmo ponto remoto. Desta forma, cada grupo de operações equivale a uma sub-consulta a ser executada pelo ponto identificado. Os resultados das sub-consultas remotas são processados pelo SGBD de acordo com a estratégia de execução para composição do resultado final.

A Figura 24 apresenta o plano algébrico otimizado da consulta exibida pela Figura 12, de acordo com a localização dos fragmentos na rede P2P definida na Tabela 9. Neste exemplo, o plano algébrico localizado, reduzido e otimizado gera quatro sub-planos algébricos distintos, três para execução em pontos remotos e um para execução no próprio ponto solicitante da consulta a partir dos resultados retornados das sub-consultas executadas remotamente.

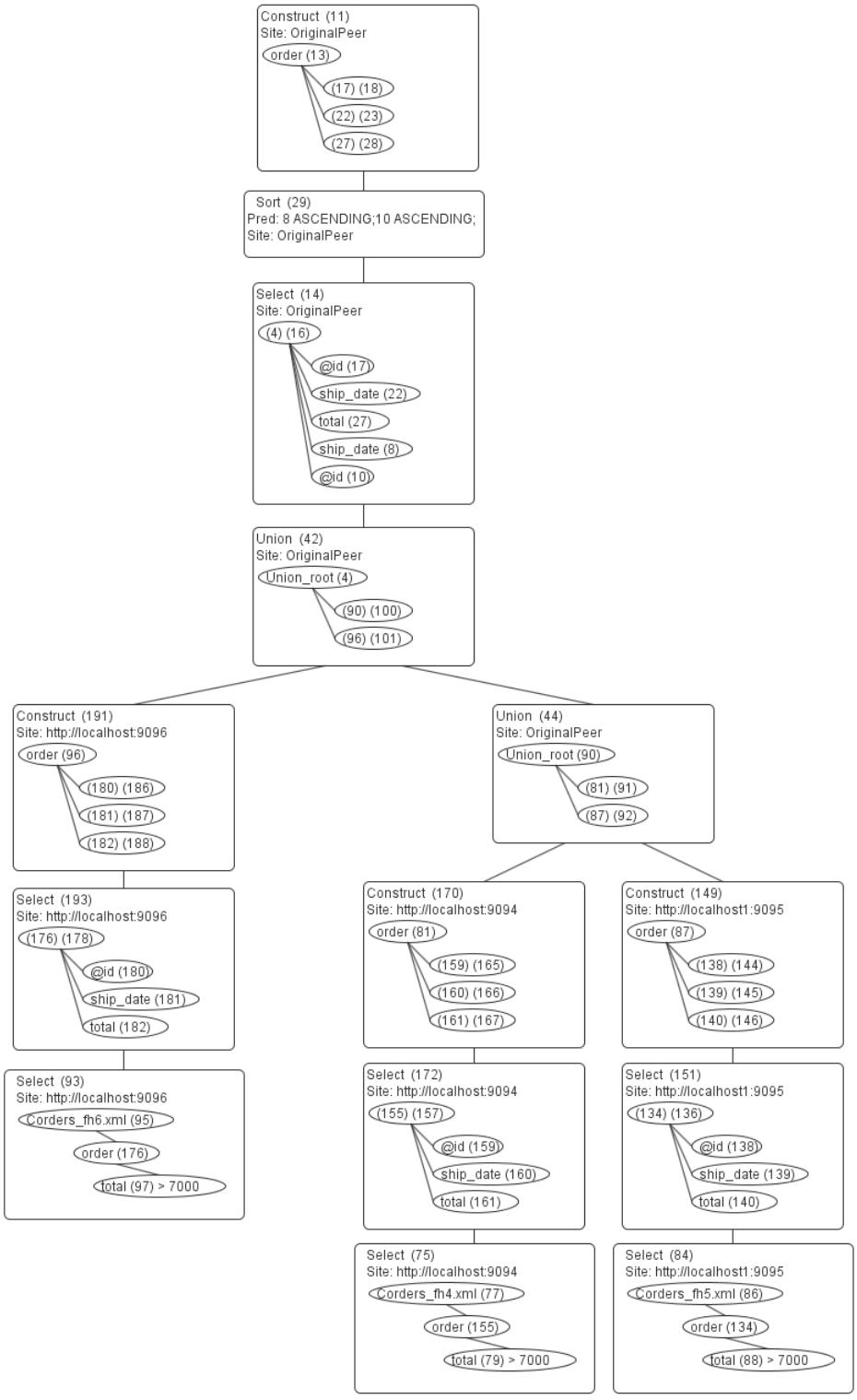


Figura 24 - Representação da consulta XQuery em sua forma algébrica otimizada

As sub-consultas geradas são então enviadas aos bancos de dados XML remotos em paralelo para serem executadas. Após a obtenção de todos os resultados dos pontos remotos, o ponto solicitante da consulta executa as operações de composição do

resultado final sobre os resultados remotos. A Tabela 11 mostra as sub-consultas remotas geradas para a consulta representada pela Figura 12.

Tabela 11 – Sub-consultas destinadas aos pontos remotos

Fragmentos XML	Predicado de Seleção	Sub-Consultas
COrders_fh4	*/order/total > 6000 ∧ /order/total <= 8000*	for $v125 in VIEW('Corders_fh4.xml')/order where $v125/total > 7000 return <order> { $v125/@id } { $v125/ship_date } { $v125/total } </order>
COrders_fh5	*/order/total > 8000 ∧ /order/total <= 10000*	for $v146 in VIEW('Corders_fh5.xml')/order where $v146/total > 7000 return <order> { $v146/@id } { $v146/ship_date } { $v146/total } </order>
COrders_fh6	*/order/total > 10000*	for $v167 in VIEW('Corders_fh6.xml')/order where $v167/total > 7000 return <order> { $v167/@id } { $v167/ship_date } { $v167/total } </order>

Observe que as sub-consultas remotas apresentadas na Tabela 11 utilizam uma expressão (*VIEW*) para se referir a um fragmento. Esta sub-consulta é processada por um adaptador no banco de dados do ponto remoto, onde é feita a substituição desta expressão pela expressão correta da sintaxe do XQuery que represente o endereço local da coleção XML consultada.

No caso onde apenas um fragmento XML seja identificado como relevante para a consulta submetida ou em situações onde os fragmentos XML identificados estejam localizados em um mesmo ponto, a fase de consolidação do resultado final não será necessária, visto que apenas uma sub-consulta remota foi gerada para o ponto. Logo, o resultado retornado pela única sub-consulta gerada irá corresponder ao resultado final da consulta submetida, o qual será repassado diretamente ao usuário final sem necessidade de pós-processamento.

Caso existam fragmentos XML localizados em pontos distintos no plano algébrico da consulta, obrigatoriamente existirá uma sub-consulta que será executada pelo ponto solicitante da consulta para realizar a composição do resultado final. Maiores informações sobre a consolidação do resultado final podem ser encontradas em (FIGUEREIDO, 2007).

4.4. Considerações Finais

Neste capítulo apresentamos em mais detalhes a maior contribuição deste trabalho que consiste na definição de uma metodologia para o processamento de consultas XQuery sobre bases XML distribuídas em um ambiente P2P. As etapas definidas para o processamento de consultas distribuídas em ambientes P2P podem ser sintetizadas nas etapas de decomposição da consulta; localização dos pontos da rede P2P que possuem dados relevantes para o resultado da consulta; otimização; geração de sub-consultas e sua execução nas bases XML remotas dos pontos relevantes; e consolidação do resultado final no ponto solicitante da consulta.

O diferencial entre esta proposta e os outros sistemas P2P para o processamento de consultas distribuídas está relacionado à utilização de uma definição formal para as etapas do processamento das consultas XML distribuídas e do uso de uma álgebra XML para a reescrita da consulta XQuery em expressões algébricas, a fim de tornar o processamento da consulta XML correto e automático no ambiente P2P. Além disso, a alta expressividade utilizada da linguagem XQuery também representa um fator diferencial para a metodologia proposta no ambiente P2P.

Porém, considerando as técnicas estudadas para a etapa de localização dos dados relevantes entre os pontos da rede P2P, a solução proposta por Bonifati et al. (2006) se assemelha à proposta desta dissertação. Conforme detalhado na Seção 3.2, o sistema P2P *XP2P* (BONIFATI et al., 2006) armazena e recupera dados XML em uma rede P2P estruturada baseada em DHT, para realizar buscas utilizando a XPath como linguagem de consulta. Para tal, a arquitetura se baseia em um modelo de fragmentação de documentos XML para a camada de dados da rede P2P.

Na proposta do sistema *XP2P*, um fragmento XML é um sub-documento extraído do documento XML original, identificado por uma única expressão de caminho, a qual é considerada a raiz do sub-documento XML gerado. Além disso, um

fragmento XML do sistema *XP2P* também pode manter ligações para outros fragmentos distribuídos entre os pontos da rede, possuindo possivelmente um conjunto de expressões de caminho relacionadas aos seus fragmentos filhos (*child-fragment*) e/ou relacionado ao seu fragmento pai (*super-document*).

Embora adequada aos requisitos P2P, a solução proposta por Bonifati et al. (2006) não se aplica ao nosso cenário, pois estamos interessados na fragmentação horizontal de coleções XML (ANDRADE et al., 2005). A solução apresentada por Bonifati et al. (2006) focaliza em uma fragmentação vertical, visto que os fragmentos XML definidos pelo *XP2P* dividem a estrutura de dados de um documento XML em partes menores, e se aplica somente a repositórios XML do tipo SD. Maiores detalhes sobre os problemas ocasionados por uma fragmentação vertical em nossa abordagem são apresentados na Seção 4.2.

Apesar das tabelas *hash* distribuídas serem construídas sobre as expressões de caminho que definem o fragmento XML, a semântica empregada é diferente entre as duas abordagens, pois a expressão de caminho do fragmento XML na nossa abordagem é definida através dos predicados de seleção que caracterizam o fragmento XML horizontal.

Os tipos de consultas permitidas pelo sistema *XP2P* possuem características comuns em relação às técnicas especificadas para a localização dos dados relevantes da nossa metodologia. Quando não existe a correspondência exata entre as expressões de caminho resultantes da consulta submetida e qualquer fragmento XML distribuído pela rede P2P, uma reformulação da consulta original é realizada a fim de re-executar a busca por dados relevantes. Porém, os fragmentos XML do sistema *XP2P* são indexados apenas na DHT, ao contrário da nossa abordagem, onde um fragmento XML pode ser indexado pela DHT, identificado através de um *Broadcast* na rede P2P ou localizado no próprio Catálogo do ponto solicitante da consulta.

Por fim, a linguagem de consulta especificada para nossa metodologia é a XQuery, que apresenta um maior grau de complexidade e expressividade quando comparada à linguagem XPath, a qual é utilizada como linguagem de consulta do sistema *XP2P*.

Capítulo 5. Implementação de um Sistema para Processamento de Consultas XQuery Distribuídas em Ambientes P2P

5.1. Introdução

Este capítulo apresenta a arquitetura utilizada para avaliar a metodologia proposta e analisar questões de desempenho para o processamento de consultas XQuery distribuídas em um ambiente P2P. Uma arquitetura baseada em camadas foi implementada de acordo com a metodologia apresentada no Capítulo 4 e os experimentos realizados sobre a arquitetura proposta e seus resultados serão descritos no Capítulo 6.

O capítulo está organizado da seguinte forma, além desta primeira seção de introdução. A Seção 5.2 faz a definição da arquitetura e dos seus componentes. A Seção 5.3 apresenta a rede de sobreposição utilizada, explicando as alterações necessárias para os módulos *Scribe* e DHT implementados. E por fim, na Seção 5.4 é ilustrada a interface do Mediador para a execução de consultas XQuery distribuídas.

5.2. Arquitetura Proposta

Para implementar a metodologia definida no Capítulo 4, a arquitetura proposta para o processamento de consultas XQuery em um cenário distribuído (FIGUEIREDO, 2007), baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992), foi estendida com o objetivo de viabilizar o processamento de consultas em um ambiente P2P.

Conforme a implementação do protótipo para o ambiente distribuído, a arquitetura proposta para a abordagem P2P contempla os seguintes componentes: Mediador,

Catálogo e Adaptador, conforme descritos a seguir. Porém, no contexto da abordagem P2P, os componentes são distribuídos pelos pontos disponíveis da rede P2P, a fim de eliminar o ponto único de acesso ao processamento da consulta em um ambiente distribuído (FIGUEIREDO, 2007). Logo, a arquitetura proposta para cada ponto da rede P2P é apresentada na Figura 25.

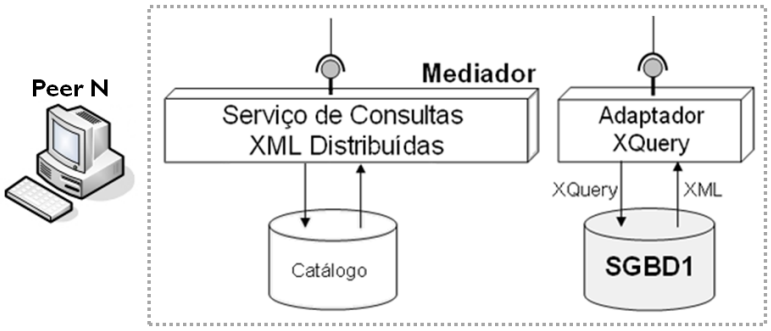


Figura 25 – Componentes da arquitetura proposta para um ponto da rede P2P

O **Mediador** é o principal componente da arquitetura, sendo responsável pelo processamento da consulta XQuery sobre as coleções XML globais relevantes, identificadas através da etapa de localização dos dados; envio das sub-consultas para os Adaptadores remotos; e consolidação do resultado final a partir dos resultados retornados das sub-consultas executadas, caso mais de um ponto seja relevante para a consulta submetida.

A arquitetura para o processamento das consultas distribuídas pelo Mediador no ambiente P2P estende a arquitetura implementada para o processamento de consultas em um cenário distribuído. A arquitetura do Mediador, apresentada pelo diagrama da Figura 8 (ver Seção 3.3.2), é composta por uma série de módulos responsáveis por partes isoladas do processamento da consulta. Os módulos encarregados pela etapa de decomposição da consulta e pela etapa de localização dos dados foram alterados para permitir o processamento da consulta submetida em um ambiente P2P, conforme a metodologia descrita na Seção 4.3. Maiores informações sobre os demais módulos responsáveis pelo processamento da consulta podem ser encontradas em (FIGUEIREDO, 2007).

O **Catálogo** armazena as configurações dos pontos distribuídos, identificados através da etapa de localização dos dados, tais como: a referência de cada Adaptador remoto já descoberto e os fragmentos XML por ele disponibilizados; a definição do predicado de seleção de cada fragmento XML; dentre outras informações. O Catálogo

no ambiente P2P é definido conforme a estrutura e implementação do ambiente distribuído, sendo composto por um conjunto de objetos em Java que pode ser serializado e desserializados em um documento XML para edição manual. Porém, cada ponto da rede P2P possui um Catálogo associado ao Mediador, ao contrário do Catálogo centralizado do ambiente distribuído (FIGUEREIDO, 2007). A Figura 26 exemplifica a configuração inicial de um Catálogo, além de apresentar a estrutura de seu documento XML.

```
<catalog>
   <catalogName>Catalogo_Peer_COrders</catalogName>
   <globalViews>
      <localViews>
         <wrapperLocation>
            <uri>http://peer:9091/WrapperExist/services/XQueryWrapper</uri>
            <communicationWeight>10.0</communicationWeight>
         </wrapperLocation>
         <referenceCollection>COrders</referenceCollection>
         <selectionPredicates>
            <string>/order[total <= 4000]</string>
         </selectionPredicates>
         <viewName>COrders_fh1.xml</viewName>
         <totalNodes>100</totalNodes>
      </localViews>
      <viewName>COrders_Peer</viewName>
      <xsdFile>COrders.xsd</xsdFile>
       <queryView>false</queryView>
   </globalViews>
</catalog>
```

Figura 26 - Exemplo de um Catálogo e sua estrutura XML

O Catálogo, exemplificado na Figura 26, é caracterizado através do nome *Catalogo_Peer_COrders* (*catalogName*), contendo um fragmento da coleção XML global (*globalViews*) de esquema *COrders.xsd* (*xsdFile*). O fragmento XML é definido através de uma visão local (*localViews*) sobre a coleção XML *COrders* (*referenceCollection*) e possui pedidos de compras com valor total menor ou igual a 4000 (*selectionPredicates*). Além disso, a referência de seu Adaptador (*wrapperLocation*) acoplado à base de dados do fragmento XML também é disponibilizada, a qual indica o local (*uri*) em que a consulta requisitada será executada.

O **Adaptador** é responsável pelo gerenciamento de execução da sub-consulta no banco de dados XML a ele acoplado. A sub-consulta enviada pelo Mediador já está praticamente pronta para execução, visto que o Adaptador precisa apenas atualizar a

localização da coleção XML sendo consultada para o seu endereço na base de dados XML local do ponto remoto, através de um arquivo de configuração contido no ponto da rede P2P. O resultado da sub-consulta é retornado ao Mediador para a composição do resultado final. Na nossa implementação utilizamos o banco de dados XML nativo eXist (EXIST, 2009).

É importante ressaltar que a interface de comunicação implementada entre os componentes é baseada em serviços *Web* (BOOTH et al., 2004). A interface projetada possui a mesma operação *executeXQuery* implementada pelos componentes Mediador e Adaptador, permitindo a comunicação entre o cliente e o Mediador e entre o Mediador e os Adaptadores através do mesmo protocolo de comunicação, cujos parâmetros de entrada e saída são apresentados na Tabela 12 (FIGUEIREDO, 2007).

Tabela 12 - Interface de comunicação (FIGUEIREDO, 2007)

Parâmetros de Entrada	**Descrição**
query (*String*)	Consulta a ser executada pelo Mediador ou Adaptador.
Parâmetros de Saída	**Descrição**
queriesExecuted (*int*)	Número de consultas executadas pelo Mediador ou Adaptador.
result (*String*)	Resultado da consulta.
sucess (*Boolean*)	Indicador de sucesso ou falha de execução da consulta.
timeMsComRemote (*long*)	Tempo gasto em milissegundos com a comunicação com o ponto remoto.
timeMsCompile (*long*)	Tempo gasto em milissegundos com a compilação da consulta pelo Mediador ou Adaptador.
timeMsLocal (*long*)	Tempo gasto em milissegundos com a execução da consulta pelo Mediador ou Adaptador.
timeMsRemote (*long*)	Tempo gasto em milissegundos com a execução remota da consulta.
totalBytes (*long*)	Quantidade em *bytes* do resultado da consulta.

5.3. Rede de Sobreposição Utilizada

Um problema muito comum em aplicações P2P consiste em localizar eficientemente um ponto da rede P2P que possua um dado relevante. No contexto da metodologia proposta, descrita na Seção 4.3, a localização dos pontos que possuem dados relevantes para a consulta submetida pode ser realizada a partir de diversas técnicas levantadas: (1) através de uma mensagem de *broadcast* para um número limitado de pontos na rede; (2) através da utilização de um índice distribuído entre os pontos da rede P2P; ou (3) através das informações disponíveis do Catálogo atualizado do ponto solicitante da consulta. O objetivo é avaliar o comportamento do protótipo de acordo com os protocolos desenvolvidos, indicando as vantagens e desvantagens de cada técnica no processamento de consultas distribuídas no ambiente P2P.

A partir das técnicas levantadas para a localização eficiente dos dados relevantes, uma rede de sobreposição estruturada (*structured overlay network*) foi então selecionada como a rede P2P utilizada para o desenvolvimento do protótipo. Dentre as redes de sobreposição estruturadas encontradas na literatura, tais como *CAN* (RATNASAMY et al., 2001) e *Chord* (STOICA et al., 2001), o *Pastry* (ROWSTRON et al., 2001b) foi escolhido para a aplicação da metodologia proposta pelas razões explicadas em mais detalhes a seguir.

O *Pastry* visa construir uma rede P2P estruturada, descentralizada, auto-organizável e tolerante a falhas para aplicações P2P. Cada ponto do sistema é definido através de um identificador único (*nodeId*) de 128 *bits* gerado aleatoriamente. Quando um ponto entra no sistema P2P, o ponto também recebe um identificador de objeto (*objId*) com o mesmo número de *bits* e é mapeado para o ponto que possui o identificar *nodeId* numericamente mais próximo. Desta forma, cada ponto da rede P2P mantém informações sobre os pontos que possuem identificadores mais próximos numericamente.

Em comparação às redes P2P não-estruturadas, o *Pastry* oferece um roteamento P2P eficiente e escalável pela utilização de índices distribuídos entre os pontos da rede. O roteamento de mensagens e requisições é bem eficiente, possuindo complexidade *O(log N)*, onde *N* é o número de pontos ativos no sistema em um dado momento, e o tamanho da tabela de roteamento que cada ponto deve manter também tem complexidade *O(log N)*.

Dentro deste contexto, o *FreePastry* (FREEPASTRY, 2009), implementação livre do *Pastry*, foi utilizado como a plataforma P2P do protótipo desenvolvido. O *FreePastry* disponibiliza a API proposta por Darek *et al*. (2003), especificando serviços de roteamento de mensagens, notificação de eventos, busca e armazenamento distribuído na rede P2P.

Para tal, as aplicações fornecidas pela plataforma *FreePastry* que utilizamos na implementação da metodologia proposta compreendem, (1) *SCRIBE* (CASTRO et al., 2002), um sistema de comunicação em grupo; e (2) *PAST* (ROWSTRON et al., 2001a), um sistema de armazenamento distribuído (DHT do *FreePastry*); as quais serão analisadas em mais detalhes nas seções a seguir.

5.3.1. Módulo Scribe

A primeira técnica identificada para a etapa de localização dos dados relevantes entre os pontos da rede P2P (ver Seção 4.2.2.1) propaga suas requisições através de mensagens de *broadcast* para um número limitado de pontos. Esta técnica utiliza mecanismos de inundação controlada por um parâmetro (TTL), que define o número máximo de pontos alcançáveis dentro da rede P2P.

As redes P2P baseadas em inundação são classificadas como redes P2P não-estruturadas, visto que o modo pelo qual os pontos se ligam à rede não é estruturado. Um ponto que deseja entrar na rede simplesmente descobre alguns pontos já presentes na rede P2P por inundação (*flooding*) e se conecta arbitrariamente de modo não-estruturado a alguns deles. A pesquisa por recursos é realizada através de uma mensagem de *broadcast* para todos os pontos aos quais o ponto solicitante esteja conectado.

Logo, o roteamento de requisições empregado pela primeira técnica da etapa de localização dos dados é proveniente de redes P2P não-estruturadas, porém a plataforma P2P selecionada para a implementação do protótipo é totalmente baseada em uma rede P2P estruturada. Então, a solução para esta limitação encontrada consiste em simular o *broadcast* em uma rede P2P estruturada, a partir da implementação de um módulo baseado no serviço de comunicação em grupo, o *Scribe*, da plataforma P2P *FreePastry*. O serviço de comunicação em grupo pode ser usado para algo simples, como a distribuição de eventos para um grupo de interessados, ou para algo mais complexo,

como a transmissão otimizada de dados a um grupo de pontos. Embora seja possível implementar a comunicação em grupo usando apenas o envio de mensagens diretas entre dois pontos, isso não é uma tarefa trivial.

Desta forma, a implementação do protótipo para a primeira técnica da etapa de localização dos dados utiliza o serviço de comunicação em grupo oferecida pelo *FreePastry* para criar uma rede semântica de sobreposição (CRESPO et al., 2002). Uma rede semântica de sobreposição (*Semantic Overlay Network*, SON) é uma rede onde os pontos com recursos ou interesses semelhantes são aglomerados (*clustered*) em conjuntos lógicos de conteúdos ou interesses, sendo que cada ponto pode participar de mais de um conjunto. Cada ponto da rede P2P pode criar um grupo, no qual outros pontos que compartilhem recursos ou interesses semelhantes podem participar. Assim, pontos podem assinar um grupo específico e passar a receber as mensagens que são enviadas a ele.

No contexto do módulo implementado, um grupo específico é criado a fim de localizar os pontos que possuem dados relevantes para o resultado da consulta. Os pontos da rede P2P que possuem o mesmo objetivo de localizar os dados relevantes da consulta submetida se inscrevem neste grupo específico. Uma vez que o grupo tenha sido criado, qualquer ponto participante pode enviar consultas para todos os pontos inscritos no grupo específico, que serão distribuídas através de mensagens *multicast* para os demais pontos.

Assim, a primeira técnica para a localização dos dados implementada propaga suas requisições através de mensagens *multicast* para os pontos participantes do grupo específico para a localização dos dados. Porém, como todos os pontos da rede P2P proposta possuem o mesmo interesse em localizar os dados relevantes para o resultado da consulta submetida, as requisições serão sempre enviadas para todos os pontos da rede P2P. Desta forma, o protótipo simula o mecanismo de inundação em um ambiente P2P estruturado. Assim que um ponto recebe uma requisição, ele tenta resolvê-la. Caso consiga, uma mensagem de resposta é então enviada diretamente ao ponto solicitante.

5.3.2. Módulo DHT

A segunda técnica identificada para localizar dados relevantes da rede P2P é realizada através de tabelas *hash* distribuídas (ver Seção 4.2.2.2). Uma tabela *hash*

distribuída é um conjunto de tabelas distribuídas entre os pontos da rede P2P, compostas por chaves numéricas únicas e valores armazenados. Os valores armazenados devem ser endereçados por chaves numéricas únicas geradas através do uso de uma função *hash*. Desta forma, qualquer ponto da rede P2P estruturada pode usar a DHT para recuperar eficientemente o valor armazenado pela chave numérica única.

A implementação do protótipo para a segunda técnica da etapa de localização dos dados é baseada em um sistema de armazenamento distribuído, o *PAST*, da plataforma P2P *FreePastry*. O *PAST* é na verdade a DHT fornecida pelo *FreePastry* e, portanto, oferece o serviço de mapeamento de chaves em valores.

A DHT do *FreePastry* não permite a utilização da mesma chave numérica para duas publicações distintas de valores.Para tal, o *PAST* possui um tratamento para colisões de chaves numéricas nas operações de inserção na sua DHT, definindo, desta forma, que uma chave numérica deve apontar para apenas um valor armazenado.

Então, a solução para a limitação encontrada consiste no desenvolvimento de um objeto lista que contém os objetos com a mesma chave numérica e publicar a lista de objetos ao invés do único objeto associado. Assim, o módulo DHT implementado permite que um conjunto de valores diferentes seja publicado com a mesma chave numérica associada.

No contexto do módulo implementado, a tabela *hash* distribuída permite que um conjunto de visões locais definidas por fragmentos XML distintos seja publicado em uma chave numérica única, gerada através da expressão de caminho associada. Neste caso, uma expressão de caminho pode ser localizada através de diferentes fragmentos XML a ela associada.

No momento da publicação de uma expressão de caminho na DHT, a tabela *hash* distribuída é analisada a fim de verificar se algum fragmento XML já existe com a mesma chave numérica gerada pela expressão de caminho solicitada. Caso exista, o fragmento XML da expressão de caminho já publicada é adicionado em sua lista de fragmentos XML. Como a DHT do *FreePastry* não possui uma operação de remoções de publicações, o módulo DHT implementado em nossa arquitetura não faz qualquer suposição a esse respeito.

Entretanto, esta abordagem implica em outras complicações, tal como o problema de publicações concorrentes. Visto isso, a tabela *hash* distribuída do módulo DHT implementado também possui um tratamento para a concorrência de publicações, a fim de evitar que um ponto sobrescreva as publicações de outros pontos da rede P2P.

5.4. Interface para Execução de Consultas

A aplicação desenvolvida para demonstrar as etapas do processamento das consultas XQuery distribuídas possui dois estágios: configuração e execução. Durante o primeiro estágio são carregados os arquivos de configuração que especificam a infra-estrutura P2P a ser utilizada. Após a configuração da rede P2P, os pontos propostos, em conjunto com seus módulos *Scribe* e DHT, são inicializados na rede P2P a fim de permitir a localização eficiente dos dados relevantes entre os pontos da rede. Cada ponto inicializado possui seu Catálogo analisado e é carregado em sua tabela *hash* distribuída a partir das expressões de caminhos decompostas do predicado de seleção de seus fragmentos XML.

Em seguida, a aplicação entra no estágio de execução. Nesse estágio é possível efetuar consultas e trocar mensagens com os outros pontos da rede P2P, utilizando uma das técnicas de localização dos dados implementadas. Durante o estágio de execução, consultas XQuery podem ser submetidas no protótipo através de uma interface desenvolvida para demonstrar as etapas do processamento das consultas XQuery distribuídas no ambiente P2P (FIGUEIREDO et al., 2007).

A interface desenvolvida no Mediador permite selecionar uma das técnicas implementadas para localizar os dados relevantes da consulta submetida, além de exibir uma lista de pontos já identificados na rede P2P, conforme ilustrado na Figura 27. Desta forma, o usuário pode submeter sua consulta XQuery em qualquer ponto já previamente descoberto da rede P2P, permitindo selecionar a forma mais apropriada para a localização dos dados relevantes da consulta. Além disso, também é possível visualizar o Catálogo correspondente a cada ponto da rede P2P. A representação algébrica da consulta XQuery submetida durante as diferentes etapas de processamento da consulta pode ser visualizada de forma escrita ou gráfica. Através da interface, a evolução no processamento da consulta XQuery submetida pode ser visualizada de maneira mais clara, o que facilita o entendimento da metodologia proposta (ver Seção 4.3). A Figura 28 exemplifica a interface gráfica para execução de consultas no Mediador, ilustrando a visualização gráfica do plano algébrico inicial da consulta XQuery submetida no Mediador, representada pela Figura 27.

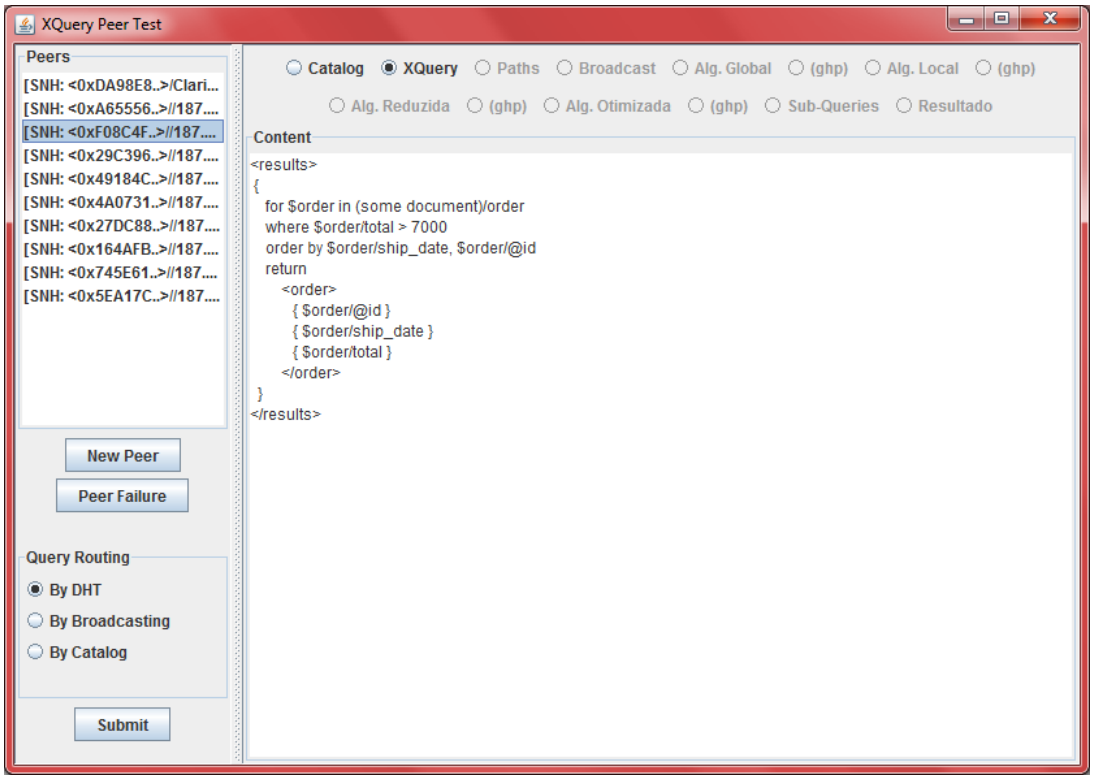


Figura 27 - Interface para execução de consultas

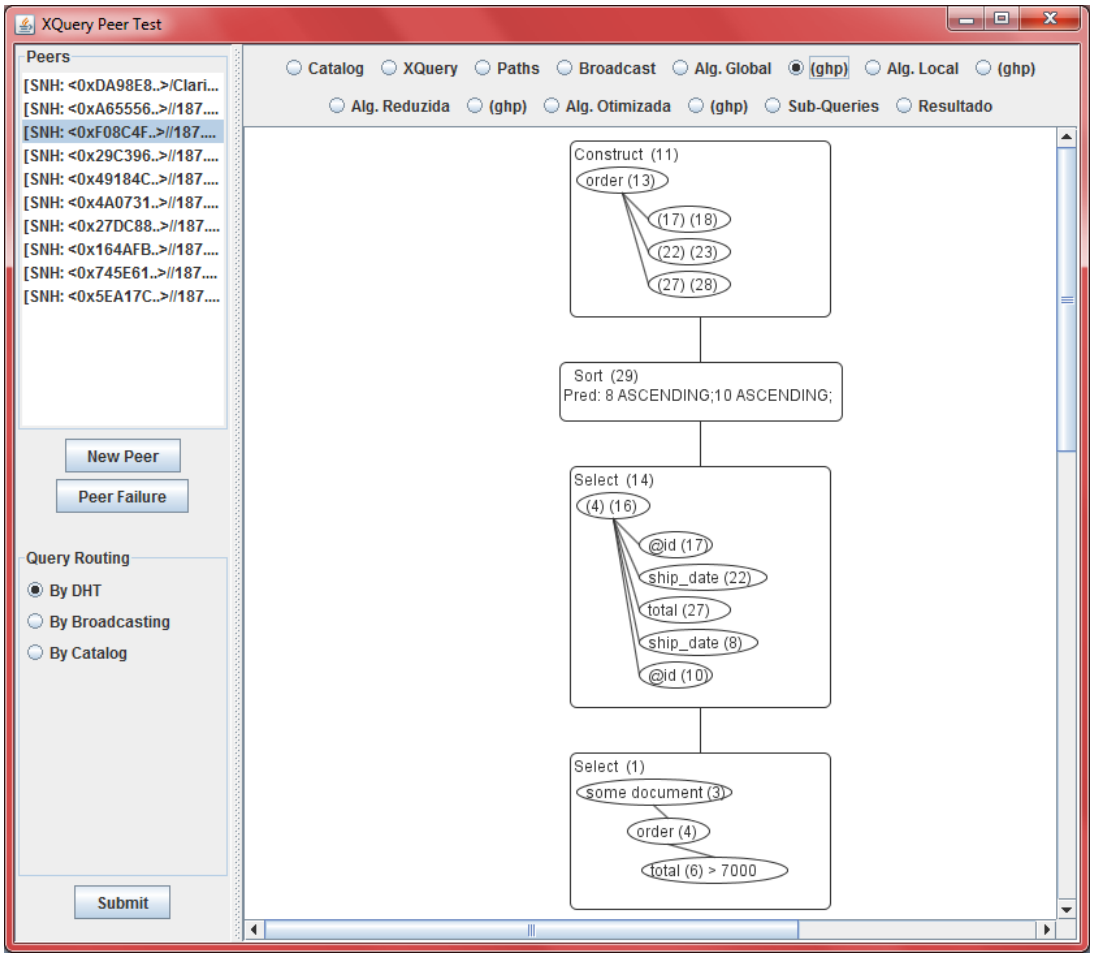


Figura 28 - Interface gráfica para execução de consultas

5.5. Considerações Finais

Neste capítulo apresentamos a arquitetura proposta para o processamento de consultas XQuery distribuídas sobre bases XML em ambientes P2P. A partir do desenvolvimento dos componentes da arquitetura, Mediador, Catálogo e Adaptador, foram executados testes para avaliar todos os módulos desenvolvidos.

Além disso, também explicamos as duas principais técnicas utilizadas para a localização dos dados relevantes na rede P2P, uma não-estruturada e outra estruturada. Para tal, empregamos dois protocolos fornecidos pela plataforma P2P *FreePastry*: (1) *Scribe*, o qual simula um ambiente não-estruturado; e (2) *Past*, protocolo correspondente a DHT do ambiente estruturado do Pastry.

Para finalizar, iniciamos o planejamento de uma série de experimentos a serem executados para análise dos resultados, que são apresentados no Capítulo 6, indicando as vantagens e desvantagens de cada técnica empregada para o processamento de consultas distribuídas em um ambiente P2P.

Capítulo 6. Avaliação Experimental

6.1. Introdução

Neste capítulo apresentamos os resultados obtidos através dos experimentos realizados no protótipo desenvolvido de acordo com a arquitetura proposta no Capítulo 5. O objetivo dos experimentos é avaliar a adequação da metodologia de processamento de consultas XQuery distribuídas em um ambiente P2P descrita no Capítulo 4, em especial para as etapas de decomposição da consulta submetida e da localização dos dados relevantes no ambiente P2P.

O capítulo está organizado da seguinte forma, além desta primeira seção de introdução. Na Seção 6.2 apresentamos a preparação dos experimentos em detalhes, incluindo os objetivos dos experimentos, a base de dados utilizada e seus fragmentos XML gerados, a alocação dos fragmentos XML nos pontos do ambiente P2P e a metodologia de execução dos experimentos. Por fim, a Seção 6.3 é dedicada à avaliação dos resultados obtidos com os experimentos.

6.2. Preparação dos Experimentos

Para a execução dos experimentos com o protótipo implementado, é necessário seguir as seguintes diretrizes: (1) definir os objetivos dos experimentos; (2) planejar a sua execução, incluindo a preparação do ambiente onde serão executados os testes, a definição das bases de dados e fragmentos XML que serão utilizados, a alocação destes fragmentos no ambiente proposto e a descrição das consultas submetidas durante os testes propostos; e (3) definir uma metodologia de execução dos experimentos para permitir uma análise conclusiva dos dados obtidos.

6.2.1. Objetivos

No contexto da metodologia apresentada nesta dissertação, os experimentos visam avaliar a adequação da metodologia de processamento de consultas XQuery

distribuídas em um ambiente P2P, em especial para as etapas de decomposição da consulta e localização dos dados. Em particular, uma avaliação é realizada para o processamento de consultas XQuery distribuídas entre o ambiente distribuído (FIGUEIREDO, 2007) e o ambiente P2P, a fim de comparar o comportamento dos protótipos quando submetidos a uma série de consultas XQuery.

Além disso, as principais técnicas identificadas para a etapa de localização dos dados são também analisadas a fim de verificar qual técnica melhor se adapta para o processamento de consultas distribuídas em ambientes P2P. Com isso, podemos identificar importantes características das técnicas estudadas, e de suas respectivas estratégias aplicadas, que colaboram para o melhor desempenho do sistema. Através da identificação de vantagens e desvantagens na utilização de cada técnica empregada, podemos ajudar na adaptação e melhoria destas técnicas para o processamento de consultas XQuery distribuídas no ambiente P2P.

Porém, podemos observar que o tempo gasto durante o processamento de uma consulta XQuery sobre um cenário P2P é maior quando comparado ao tempo gasto no processamento de uma consulta XQuery em um cenário distribuído. Tal fato pode ser explicado pelo aumento no tempo despendido principalmente para a etapa de localização dos dados relevantes entre os pontos da rede P2P, visto que um ponto possui apenas um conhecimento parcial das informações sobre os dados distribuídos na rede P2P; além do tempo gasto no processo inicial de decomposição da consulta XQuery em expressões de caminho.

O aumento no tempo do processamento de consultas XQuery submetidas no cenário P2P poderá ser compensado através da escalabilidade proporcionada pelo uso de uma abordagem P2P. O objetivo é comprovar que o tempo total gasto com o processamento de consultas submetidas de forma seqüencial no protótipo desenvolvido para o cenário distribuído (FIGUEIREDO, 2007) é superior ao tempo total gasto com o processamento das mesmas consultas em um cenário P2P, utilizando a escalabilidade oferecida pela abordagem P2P.

Logo, o objetivo principal é demonstrar até qual momento o tempo total gasto durante o processamento de um conjunto de consultas em um cenário distribuído é menor que o tempo total de processamento das mesmas consultas submetidas em um cenário P2P, determinando assim o ponto limiar no qual o cenário P2P começa a se destacar sobre o cenário distribuído com ponto único de acesso.

6.2.2. Ambiente Experimental

O ambiente experimental é constituído por dez computadores em rede local totalmente dedicados para os testes, que exercem o papel de pontos distribuídos em uma rede P2P. Para tal, utilizamos parte do ambiente computacional disponível no Laboratório de Uso Geral (LUG) do Programa de Engenharia de Sistemas e Computação (PESC/COPPE) da Universidade Federal do Rio de Janeiro (UFRJ). Cada computador possui processador Pentium 4 HT de 1,8 GHz com 512 Mb de memória RAM e sistema operacional Windows XP.

Cada ponto do ambiente experimental simulado possui os seguintes componentes instalados: Mediador, Catálogo e Adaptador, conforme ilustrado na Figura 29. Além do serviço *Web* do Mediador implantado, cada ponto também possui um servidor *Web* Tomcat 5.5 (APACHE TOMCAT, 2009), no qual foi implantado o serviço *Web* do Adaptador desenvolvido para o eXist e o próprio banco de dados XML nativo eXist (EXIST, 2009).

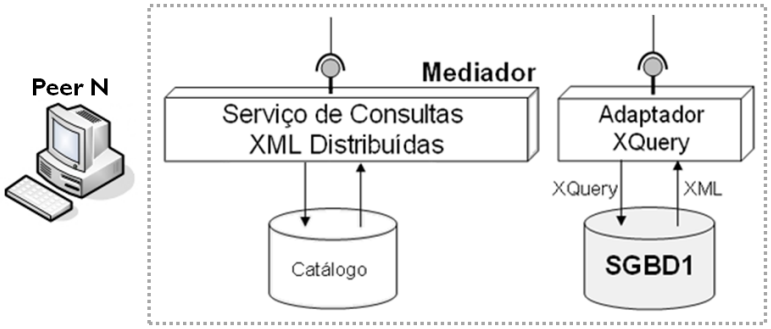


Figura 29 - Componentes implantados em cada ponto da rede P2P

A base de dados utilizada nos experimentos é uma coleção de documentos XML de pedidos de compra (ver Seção 4.1.1), chamada *COrders* do *benchmark* XBENCH (YAO et al., 2002), gerada através do gerador de bases ToXgene (BARBOSA et al., 2002) e carregada na banco de dados XML nativo eXist pela sua ferramenta de administração. A base de dados *COrders* apresentada pela Figura 9, de múltiplos documentos, foi fragmentada horizontalmente em diversos fragmentos XML definidos sobre as faixas de valores do total do pedido.

As consultas utilizadas sobre a base de dados XML foram desenvolvidas levando-se em consideração as definições das fragmentações. Foram criadas consultas que se beneficiavam e consultas que não se beneficiavam da fragmentação da base, para

que fosse possível avaliar o desempenho nesses dois tipos de consultas. Como iremos considerar o trabalho proposto por Figueiredo (2007) como base de comparação, a base de dados e as consultas definidas sobre a base XML foram reaproveitadas e utilizadas nos experimentos do protótipo desenvolvido para o ambiente P2P. A relação das consultas XQuery é apresentada no Anexo I.

Analisando os objetivos apresentados na Seção 6.2.1, projetamos quatro cenários para a execução dos experimentos. Os cenários foram construídos com um número crescente de pontos distribuídos pela rede experimental: apenas um ponto participante (Cenário 1), três pontos participantes (Cenário 2), seis pontos (Cenário 3) e dez pontos participantes na rede (Cenário 4). Além disso, a distribuição dos fragmentos XML entre os pontos foi realizada de forma que a consulta submetida seja sempre respondida. Desta forma, a distribuição dos fragmentos XML entre os pontos disponíveis de cada cenário proposto é apresentada pela Tabela 13, juntamente com sua definição e tamanho.

6.2.3. Metodologia de Execução

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

A metodologia de execução dos experimentos neste trabalho consiste na execução de consultas XQuery sobre as bases de dados XML nos quatro cenários apresentados (ver Seção 6.2.2). Desta forma, o experimento visa avaliar os cenários propostos a partir da comparação dos tempos de resposta para o processamento de consultas XQuery submetidas sobre um ambiente distribuído e um ambiente P2P.

Tabela 13 – Definição dos cenários para a execução dos experimentos

Fragmentos	Definição dos Fragmentos XML	Tamanho	Local
Cenário 1			
COrders_C1	$\langle C_{orders} \rangle$	10,1 MB	Ponto 1
Cenário 2			
COrders_C2_F1	$\langle C_{orders}, \sigma_{/order/total <= 4000} \rangle$	3,24 MB	Ponto 1
COrders_C2_F2	$\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total <= 8000} \rangle$	3,71 MB	Ponto 2
COrders_C2_F3	$\langle C_{orders}, \sigma_{/order/total > 8000} \rangle$	3,17 MB	Ponto 3
Cenário 3			
COrders_C3_F1	$\langle C_{orders}, \sigma_{/order/total <= 2000} \rangle$	1,37 MB	Ponto 1
COrders_C3_F2	$\langle C_{orders}, \sigma_{/order/total > 2000 \wedge /order/total <= 4000} \rangle$	1,87 MB	Ponto 2
COrders_C3_F3	$\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total <= 6000} \rangle$	1,87 MB	Ponto 3
COrders_C3_F4	$\langle C_{orders}, \sigma_{/order/total > 6000 \wedge /order/total <= 8000} \rangle$	1,83 MB	Ponto 4
COrders_C3_F5	$\langle C_{orders}, \sigma_{/order/total > 8000 \wedge /order/total <= 10000} \rangle$	1,87 MB	Ponto 5
COrders_C3_F6	$\langle C_{orders}, \sigma_{/order/total > 10000} \rangle$	1,29 MB	Ponto 6
Cenário 4			
COrders_C4_F1	$\langle C_{orders}, \sigma_{/order/total <= 2000} \rangle$	1,37 MB	Ponto 1
COrders_C4_F2	$\langle C_{orders}, \sigma_{/order/total > 2000 \wedge /order/total <= 3000} \rangle$	944 KB	Ponto 2
COrders_C4_F3	$\langle C_{orders}, \sigma_{/order/total > 3000 \wedge /order/total <= 4000} \rangle$	972 KB	Ponto 3
COrders_C4_F4	$\langle C_{orders}, \sigma_{/order/total > 4000 \wedge /order/total <= 5000} \rangle$	996 KB	Ponto 4
COrders_C4_F5	$\langle C_{orders}, \sigma_{/order/total > 5000 \wedge /order/total <= 6000} \rangle$	924 KB	Ponto 5
COrders_C4_F6	$\langle C_{orders}, \sigma_{/order/total > 6000 \wedge /order/total <= 7000} \rangle$	928 KB	Ponto 6
COrders_C4_F7	$\langle C_{orders}, \sigma_{/order/total > 7000 \wedge /order/total <= 8000} \rangle$	952 KB	Ponto 7
COrders_C4_F8	$\langle C_{orders}, \sigma_{/order/total > 8000 \wedge /order/total <= 9000} \rangle$	892 KB	Ponto 8
COrders_C4_F9	$\langle C_{orders}, \sigma_{/order/total > 9000 \wedge /order/total <= 10000} \rangle$	1,00 MB	Ponto 9
COrders_C4_F10	$\langle C_{orders}, \sigma_{/order/total > 10000} \rangle$	1,29 MB	Ponto 10

Os tempos apresentados na Tabela 14 foram definidos levando-se em consideração o tempo total gasto com o processamento de uma consulta XQuery distribuída entre o envio da consulta e a recuperação do resultado final, do ponto de vista do cliente. Em cada cenário, os tempos são coletados, o que permitirá fazer uma comparação dos resultados de forma a avaliar o desempenho dos protótipos em diferentes ambientes e configurações.

Tabela 14 - Tempos coletados para cada consulta XQuery executada

Tempo Coletado	Descrição
Tempo de comunicação com o Mediador (*Tcm*)	Tempo gasto com a comunicação entre o ponto solicitante e o Mediador, considerando os tempos de envio da consulta e o recebimento do resultado final.
Tempo de processamento no Mediador (*Tpm*)	Tempo gasto no processamento da consulta, incluindo os tempos de compilação da consulta e de consolidação de resultados, caso mais de um ponto remoto seja envolvido.
Tempo de comunicação com os pontos remotos (*Tmn*)	Tempo gasto com a comunicação entre o Mediador e os pontos remotos, incluindo os tempos de envio da sub-consulta e o recebimento do resultado final.
Tempo de processamento no ponto remoto (*Tpn*)	Tempo gasto na execução de uma sub-consulta por um ponto remoto, considerando os tempos de execução do Adaptador e do banco de dados XML acoplado.

Para a execução dos experimentos, um total de dez rodadas foram executadas para cada consulta XQuery descrita no Anexo 1. As rodadas com os dois melhores resultados e com os dois piores resultados são desconsideradas para o cálculo dos tempos médios. Para tal, as consultas são submetidas nos protótipos estudados por uma aplicação cliente que gera um arquivo de saída no formato CSV (*comma-separated values*) com os tempos especificados pela Tabela 14. Desta forma, o arquivo gerado contém os tempos coletados de todas as execuções das consultas XQuery para todos os cenários projetados. Este formato pode ser importado para uma planilha eletrônica para análise dos dados e geração dos gráficos dos resultados, como foi realizado para estes experimentos.

Dentro deste contexto, a metodologia de execução dos experimentos consiste em duas fases principais descritas a seguir.

6.2.3.1. Fase 1 da Metodologia dos Experimentos

A primeira fase consiste na execução das consultas XQuery no ambiente distribuído (FIGUEIREDO, 2007) e no ambiente P2P proposto, a fim de comparar os tempos médios gastos para o processamento destas consultas distribuídas. O objetivo principal nesta fase é comprovar a hipótese de que o tempo gasto para as etapas adaptadas do processamento de consultas distribuídas é maior em ambientes P2P, além de analisar os tempos gastos na execução destas consultas em cada protocolo implementado para as técnicas de localização de dados propostas.

As consultas XQuery são submetidas ao mesmo ponto disponível do ambiente experimental proposto. Desta forma, a mesma consulta XQuery é executada em três ambientes distintos: (1) no protótipo desenvolvido para o ambiente distribuído, utilizando as informações disponíveis sobre a distribuição dos dados entre os pontos; (2) no protótipo implementado para o ambiente P2P, empregando a técnica estruturada (DHT) para a etapa de localização dos dados; e (3) no mesmo protótipo implementado para o ambiente P2P, porém utilizando a técnica não-estruturada (*Broadcast*) para a etapa de localização dos dados. A técnica de localização dos dados relevantes realizada através das informações disponíveis no Catálogo atualizado do próprio ponto solicitante da consulta não foi testada pelo fato de ter uma abordagem semelhante ao ambiente distribuído (FIGUEIREDO, 2007).

Desta forma, os tempos coletados para a execução de uma consulta XQuery no ambiente distribuído e no ambiente P2P estão especificados de acordo com o tempo total de resposta para o processamento desta consulta nos dois ambientes distintos, considerando os tempos definidos na Tabela 14.

O tempo total de resposta para o processamento de uma consulta XQuery no cenário distribuído (*Tdist*), utilizando o protótipo desenvolvido por Figueiredo (2007), é dado pela soma do tempo de comunicação entre o cliente e o Mediador (*Tcm*), do tempo de processamento da consulta submetida no Mediador (*Tpm*) e o maior valor obtido individualmente dentre os tempos de execução da sub-consulta em cada Adaptador (*Tpn*) adicionado ao tempo de envio da resposta dos Adaptadores para o Mediador (*Tmn*), conforme a Fórmula 1.

$$T_{dist} = T_{cm} + T_{pm} + \quad (T_{mn_i} + T_{pn_i}) \qquad (1)$$

De forma semelhante ao que ocorre no cenário distribuído, o tempo de resposta para o processamento de uma consulta XQuery no cenário P2P (*Tp2p*) também é obtido pela soma do processamento da consulta no Mediador do ponto solicitante da consulta (*Tpm'*) com o tempo de comunicação entre o cliente e o Mediador (*Tcm*), adicionando com o maior valor obtido individualmente dentre os tempos de execução da sub-consulta no ponto destino (*Tpn*) junto com o tempo de comunicação entre o Mediador e o ponto destino (*Tmn*), conforme especificado na Fórmula 2.

É importante relembrar que partimos da hipótese de que o tempo de processamento de uma consulta XQuery no cenário P2P é maior devido ao tempo gasto para a localização dos pontos da rede P2P com dados relevantes para o resultado da consulta. Isso não ocorre na abordagem distribuída, onde todas as informações sobre a distribuição das bases de dados XML utilizadas na etapa de localização dos dados estão previamente armazenadas no Catálogo.

$$T_{p2p} = T_{cm} + T_{pm'} + \quad (T_{mn_i} + T_{pn_i}) \qquad (2)$$

6.2.3.2. Fase 2 da Metodologia dos Experimentos

A segunda fase consiste na execução seqüencial das consultas XQuery no ambiente distribuído (FIGUEIREDO, 2007) e nos dois ambientes P2P propostos (DHT e *Broadcast*), a fim de comparar os tempos gastos para o processamento destas consultas distribuídas quando uma série de consultas são submetidas nos protótipos correspondentes. O objetivo principal desta fase é demonstrar que o aumento no tempo do processamento de consultas XQuery submetidas no ambiente P2P pode ser compensado pela escalabilidade proporcionada pelo uso de uma abordagem P2P.

Levando em consideração as fórmulas e seus respectivos tempos definidos pela Tabela 14, o tempo total gasto durante a execução de um conjunto de consultas no cenário distribuído (*Ttd*) é obtido através da soma do tempo total de resposta para o

processamento de uma única consulta XQuery (*Tdist*). A Fórmula 3 especifica o tempo total de espera para a execução de uma série de consultas no cenário distribuído.

$$Ttd = \qquad (Tdist_i) \qquad (3)$$

Nesta segunda fase, consideramos apenas o cenário quatro como sendo o mais crítico dentre os cenários projetados para a avaliação do processamento de consultas XQuery distribuídas sobre o ambiente distribuído e os ambientes P2P propostos. O cenário quatro é composto por dez pontos distribuídos pelo ambiente experimental e possui a fragmentação especificada na Tabela 13. Portanto, o tempo total de resposta para o processamento de uma única consulta distribuída (*Tdist*) no cenário distribuído corresponde ao tempo total de resposta para o processamento desta mesma consulta na Fase 1 da metodologia proposta para a execução dos experimentos.

Em um cenário P2P, o tempo total gasto durante a execução de uma série de consultas submetidas (*Ttp*) é dado pelo maior valor obtido individualmente dentre os tempos de resposta no processamento das consultas XQuery (*Tp2p*), conforme a Fórmula 4. É importante ressaltar que, neste caso, cada consulta XQuery foi submetida a um ponto diferente da rede P2P e o número de consultas submetidas é menor que o número de pontos disponíveis no ambiente experimental.

$$Ttp = \qquad (Tp2p_i) \qquad (4)$$

Para alcançar o objetivo principal do experimento, é necessário demonstrar os casos nos quais os tempos coletados no ambiente distribuído são superiores aos tempos encontrados no ambiente P2P quando submetidos a uma série de consultas seqüenciais, conforme a Fórmula 5. A hipótese que queremos comprovar é que o aumento no tempo para o processamento de consultas submetidas no ambiente P2P pode ser compensado pela escalabilidade proporcionada pelo uso de uma abordagem P2P.

$$Ttd > Ttp \qquad (5)$$

Os tempos coletados para a execução de um conjunto de consultas submetidas no ambiente distribuído estão de acordo com a Fórmula 3 especificada, assim como os tempos gastos para a execução de uma série de consultas seqüenciais nos ambientes P2P são definidos através da Fórmula 4.

6.3. Avaliação dos Resultados

Nesta seção, apresentamos uma avaliação dos resultados coletados de cada fase especificada pela metodologia proposta para a execução dos experimentos. Além disso, analisamos também os dados coletados em cada fase dos protocolos utilizados pelas principais técnicas de localização dos dados relevantes entre os pontos da rede P2P (DHT e *Broadcast*).

6.3.1. Análise dos Resultados da Fase 1

A primeira fase especificada pela metodologia proposta para a execução dos experimentos consiste na execução das consultas XQuery no ambiente distribuído (FIGUEIREDO, 2007) e nos dois ambientes P2P propostos (DHT e *Broadcast*). Neste contexto, um dos objetivos principais desta fase é comprovar o aumento no tempo gasto para a execução destas consultas com as etapas adaptadas para o processamento das consultas distribuídas em ambientes P2P.

Dentre as etapas estudadas para o processamento de consultas XQuery distribuídas sobre bases XML (FIGUEIREDO, 2007), podemos destacar as etapas de decomposição de consultas e de localização dos dados que representam uma sobrecarga causada pela adoção de uma abordagem P2P, conforme descrito pela metodologia proposta no Capítulo 4. Portanto, a partir destas duas etapas é que podemos provar o aumento no tempo despendido para o processamento destas consultas distribuídas no ambiente P2P, uma vez que as outras etapas são equivalentes à metodologia proposta por Figueiredo (2007).

A fim de confirmar o aumento no tempo do processamento das consultas XQuery em ambientes P2P, iremos considerar os tempos médios gastos com a compilação destas consultas distribuídas pelo Mediador, incluindo os tempos gastos

pelas etapas de decomposição da consulta, localização dos dados, otimização global e criação das sub-consultas.

A análise dos resultados foi então realizada a partir da comparação dos tempos médios de compilação pelo Mediador para cada consulta nos cenários projetados e ambientes propostos. A importância do tempo de compilação se justifica para refletir o trabalho adicional introduzido no Mediador pelo uso de uma abordagem P2P, em especial para a etapa de decomposição da consulta e de localização dos dados relevantes entre os pontos da rede P2P.

Os resultados apresentados nas Figura 30, Figura 31, Figura 32 e Figura 33 correspondem aos gráficos do tempo médio de compilação pelo Mediador nos cenários projetados e nos ambientes propostos para cada consulta XQuery.

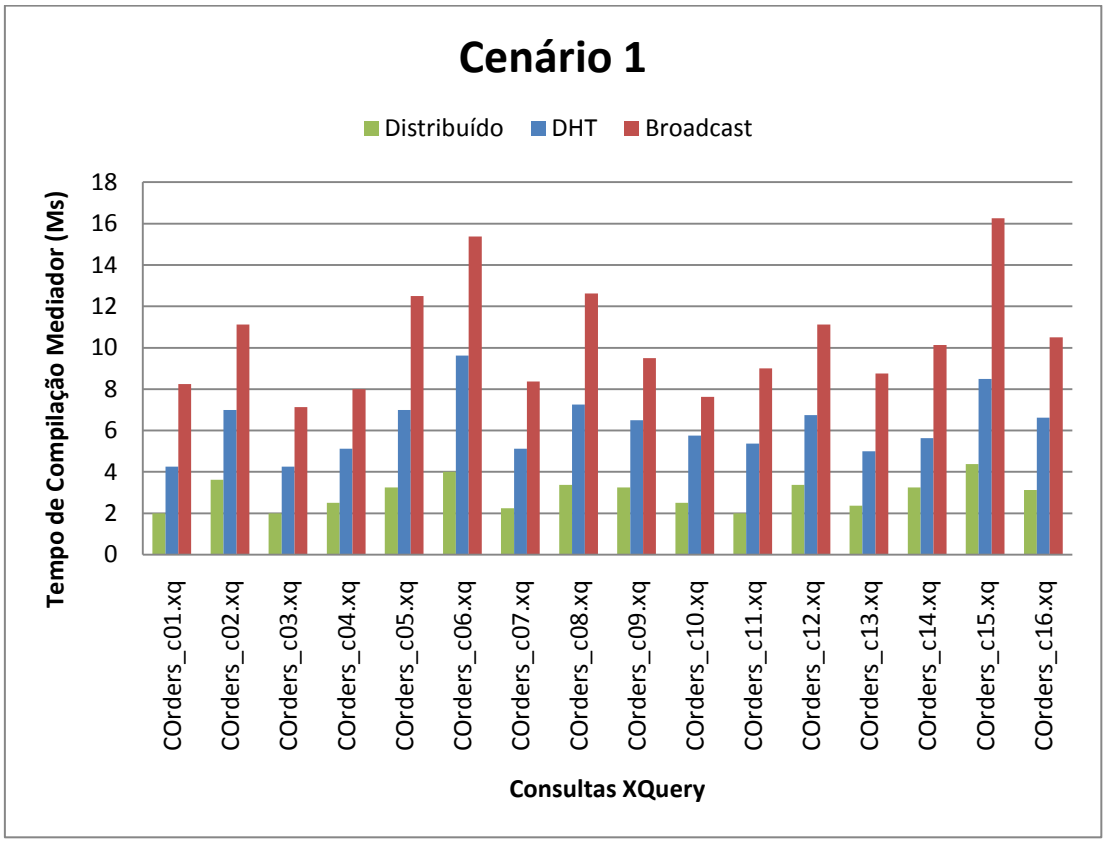


Figura 30 - Tempos de compilação das consultas distribuídas pelo Mediador no Cenário 1

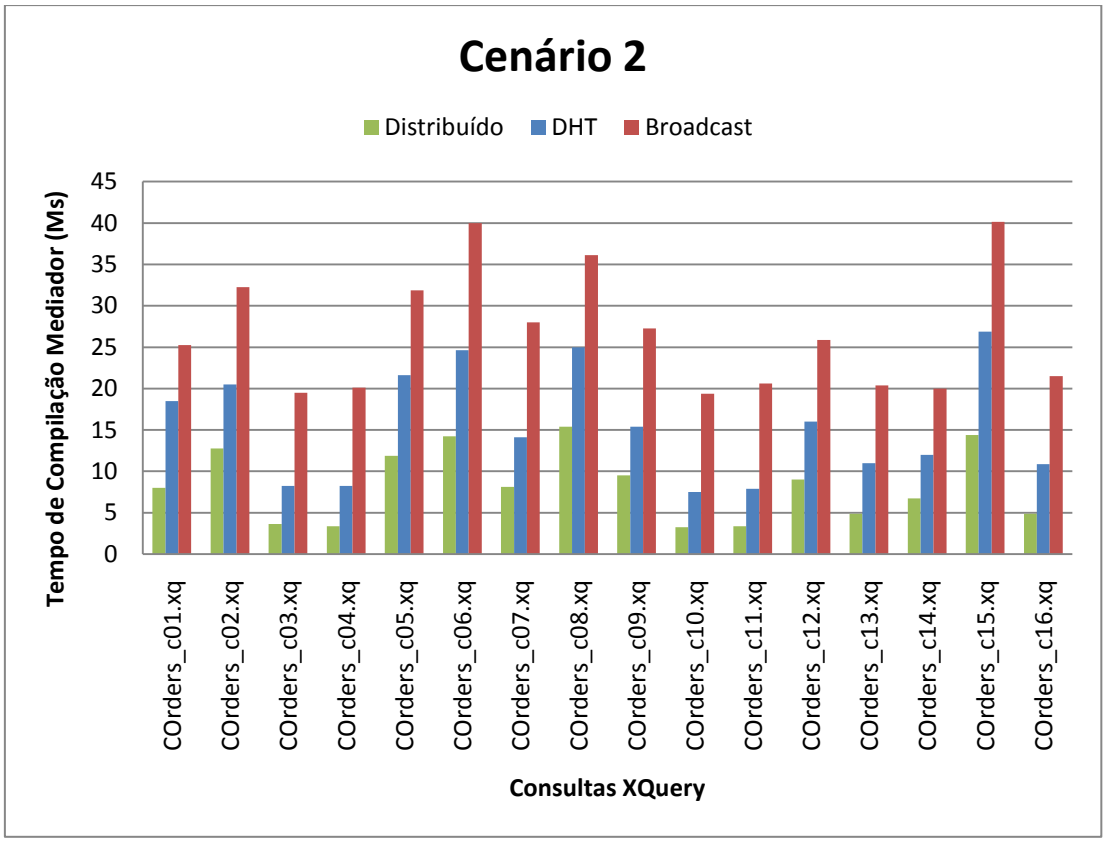


Figura 31 - Tempos de compilação das consultas distribuídas pelo Mediador no Cenário 2

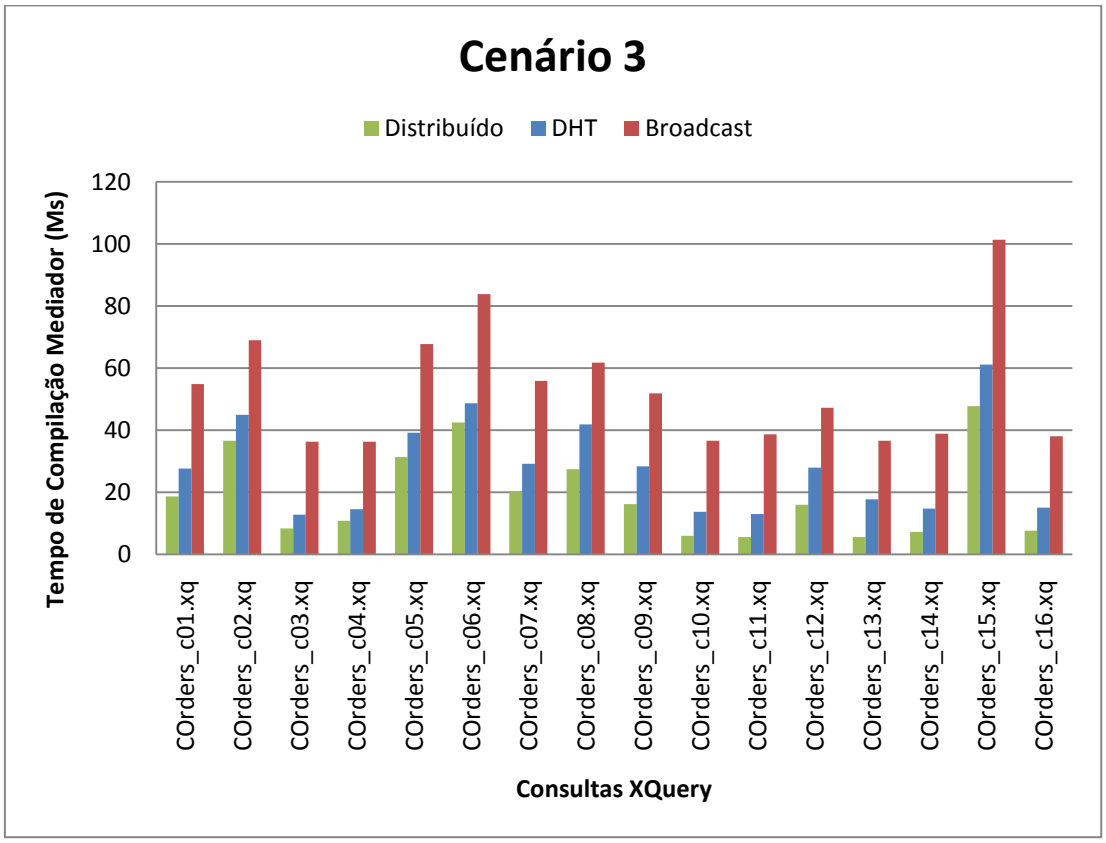


Figura 32 - Tempos de compilação das consultas distribuídas pelo Mediador no Cenário 3

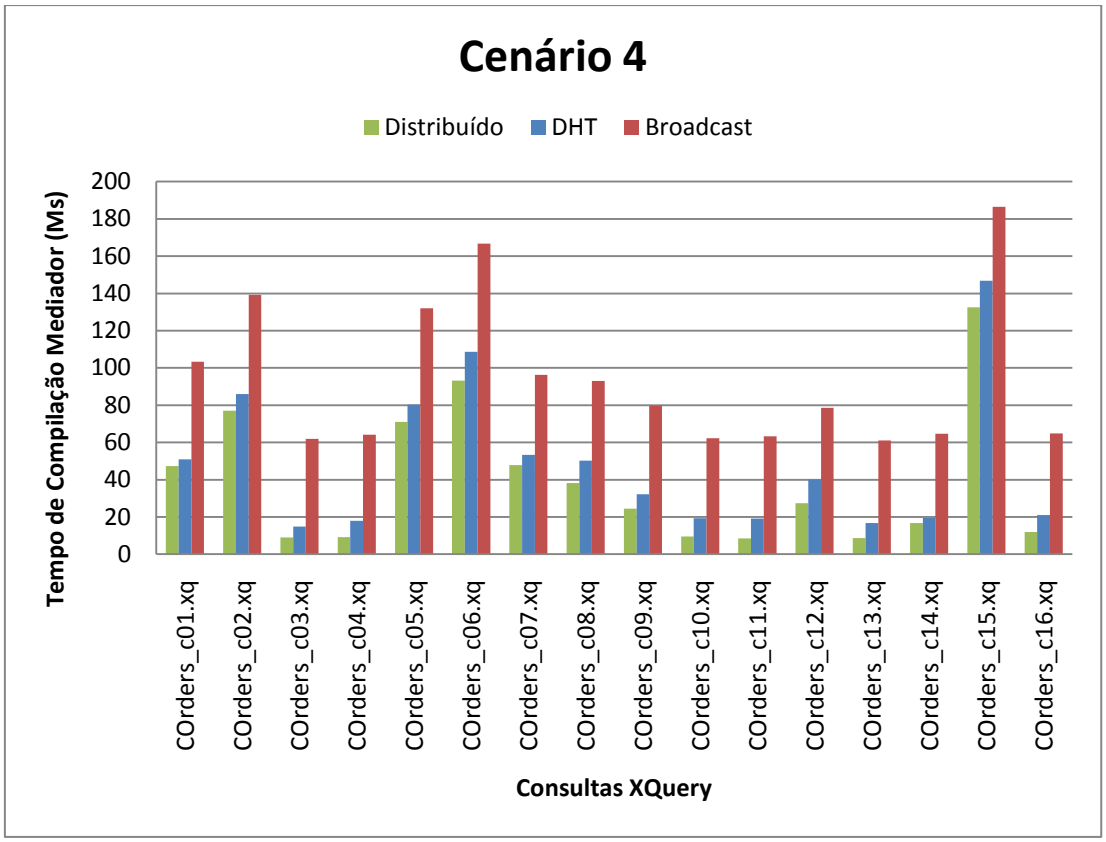


Figura 33 - Tempos de compilação das consultas distribuídas pelo Mediador no Cenário 4

Como podemos observar nas figuras apresentadas nesta seção, o tempo de compilação das consultas pelo Mediador foi aumentando à medida que foram introduzidos mais pontos no ambiente experimental. Além disso, os ambientes P2P estruturado e não-estruturado apresentam um aumento significativo em relação ao ambiente distribuído, chegando ao aumento da ordem de 82% no protocolo DHT e ao aumento da ordem de 276% no protocolo *Broadcast*, considerando a média de todos os cenários projetados. Os resultados também mostram que o protocolo não-estruturado apresenta menor desempenho em relação ao protocolo estruturado por utilizar redundância de mensagens para a localização dos dados relevantes entre os pontos do ambiente experimental.

Consultas que não se beneficiam da fragmentação distribuída pelos pontos do ambiente experimental exigem um maior processamento por parte do Mediador, tanto no ambiente distribuído como no ambiente P2P proposto, como o caso das consultas *COrders_c06* e *COrders_c15*. As consultas XQuery sem benefício da fragmentação no ambiente distribuído geram um maior número de sub-consultas, ocasionando um aumento no tempo de compilação pelo Mediador. Porém, estas consultas no ambiente P2P apresentam um aumento ainda maior no tempo gasto de compilação pelo Mediador

quando comparado ao ambiente distribuído. Além do maior processamento ocasionado pelo aumento do número de sub-consultas geradas assim como no ambiente distribuído, o aumento do tempo de compilação é principalmente explicado pela re-execução da busca por pontos relevantes na etapa de localização dos dados, visto que os fragmentos XML não são identificados com a prioridade máxima pelas consultas sem benefício da fragmentação proposta.

No entanto, as consultas que se beneficiam da fragmentação distribuída pelos pontos do ambiente experimental reduzem a necessidade de processamento pelo Mediador nos ambientes distribuídos e P2P. No ambiente distribuído, as consultas com benefício da fragmentação eliminam operações na etapa de localização dos dados de forma a diminuir o tempo gasto de compilação pelo Mediador, visto que as informações sobre os dados distribuídos pela rede estão previamente armazenadas no Catálogo. No ambiente P2P, o tempo despendido com a compilação destas consultas pelo Mediador é maior em relação ao processamento destas consultas em um ambiente distribuído devido à identificação dos fragmentos XML na rede experimental. Mesmo com os fragmentos XML identificados com prioridade máxima pelas consultas com benefício da fragmentação proposta, não necessitando a re-execução da busca para a etapa de localização dos dados, o aumento do tempo gasto na compilação destas consultas pelo Mediador no ambiente P2P é significativo em relação ao ambiente distribuído.

Dentro deste contexto, podemos comprovar o significativo aumento no tempo gasto para as etapas adaptadas do processamento de consultas distribuídas em ambientes P2P, de acordo com o objetivo principal proposto pela Fase 1 (ver Seção 6.2.3.1) da metodologia de execução dos experimentos.

6.3.2. Análise dos Resultados da Fase 2

A segunda fase especificada pela metodologia proposta para a execução dos experimentos consiste na execução seqüencial de consultas XQuery no ambiente distribuído (FIGUEIREDO, 2007) e nos dois ambientes P2P propostos (DHT e *Broadcast*). Neste contexto, o objetivo principal desta fase é demonstrar que o aumento no tempo do processamento de consultas XQuery submetidas no ambiente P2P pode ser compensado pela escalabilidade proporcionada pelo uso de uma abordagem P2P.

Para tal, selecionamos duas consultas XQuery dentre as consultas disponíveis que apresentam características distintas para o processamento distribuído destas consultas no ambiente P2P. A consulta *COrders_c06* não se beneficia da fragmentação distribuída pelos pontos do ambiente experimental. O aumento do tempo total gasto para o processamento distribuído desta consulta em um ambiente P2P é principalmente devido à re-execução da busca por pontos relevantes na etapa de localização dos dados, visto que os fragmentos XML relevantes são identificados com a prioridade mínima na rede P2P.

Já a consulta *COrders_c11* é totalmente beneficiada pela fragmentação distribuída pelos pontos disponíveis do ambiente experimental. Mesmo com os fragmentos XML identificados com prioridade máxima na etapa de localização de dados para seu processamento distribuído no ambiente P2P, o aumento do tempo total gasto na compilação desta consulta pelo Mediador no ambiente P2P é significativo em relação ao ambiente distribuído, chegando ao aumento da ordem de 140% no ambiente P2P estruturado (DHT) e ao aumento da ordem de 520% no ambiente P2P não-estruturado (*Broadcast*), considerando a média desta consulta em todos os cenários projetados.

Para a execução dos experimentos da segunda fase da metodologia proposta, consideramos o cenário quatro como sendo o mais crítico dentre os cenários projetados para a avaliação do processamento destas consultas distribuídas sobre os protocolos P2P e por fim, analisamos o tempo total gasto para a execução de um conjunto de consultas submetidas sobre o cenário quatro proposto nos ambientes distribuído e P2P.

A comparação dos tempos médios de execução destas consultas XQuery a partir da variação do número de requisições efetuadas nos diferentes ambientes foi feita através de um gráfico que apresenta os tempos de execução das consultas normalizados pelo tempo de execução da mesma consulta no ambiente distribuído. Portanto, calculamos, para cada consulta descrita nesta seção, a relação entre o seu tempo médio de execução sobre os ambientes P2P propostos com o tempo médio da sua execução no ambiente distribuído, de forma a apresentar um gráfico normalizado com o tempo de execução no ambiente distribuído sempre igual a um. Os resultados obtidos, apresentados em função do número de requisições efetuadas, são apresentados na Figura 34 para a consulta *COrders_c06* e na Figura 35 para a consulta *COrders_c11*.

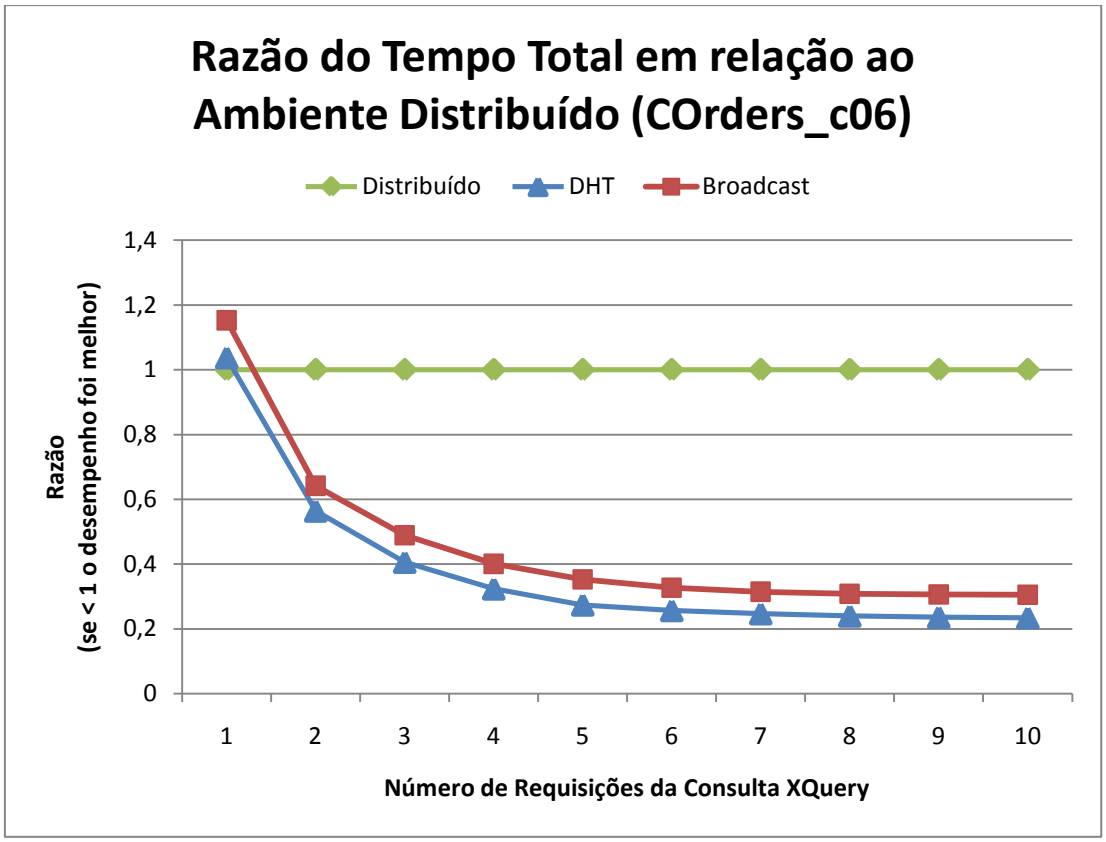


Figura 34 - Comparação do tempo total de execução de uma série de consultas XQuery *COrders_c06* no ambiente distribuído (normalizado) e nos ambientes P2P

Para as submissões realizadas com a consulta XQuery *COrder_c06*, foi observado um significativo ganho de desempenho nos ambientes P2P propostos, chegando a reduções da ordem de 77% no ambiente P2P estruturado (DHT) e a reduções da ordem de 70% no ambiente P2P não-estruturado (*Broadcast*), em relação ao tempo de execução com o mesmo número de requisições desta consulta no ambiente distribuído.

As requisições submetidas com a consulta XQuery *COrders_c11* também apresentaram ganho de desempenho nos ambientes P2P propostos, indicando reduções em relação ao tempo total de execução com o mesmo número de requisições da mesma consulta no ambiente distribuído, da ordem de 50% no protocolo estruturado (DHT) e da ordem de 42% no protocolo não-estruturado (*Broadcast*).

Verificamos facilmente pelos resultados que protocolos P2P estruturados (DHT) são mais eficientes do que os protocolos P2P não-estruturados (*Broadcast*). Tal fato pode ser explicado pelo maior número de mensagens necessárias para a localização dos dados relevantes no protocolo P2P não- estruturado.

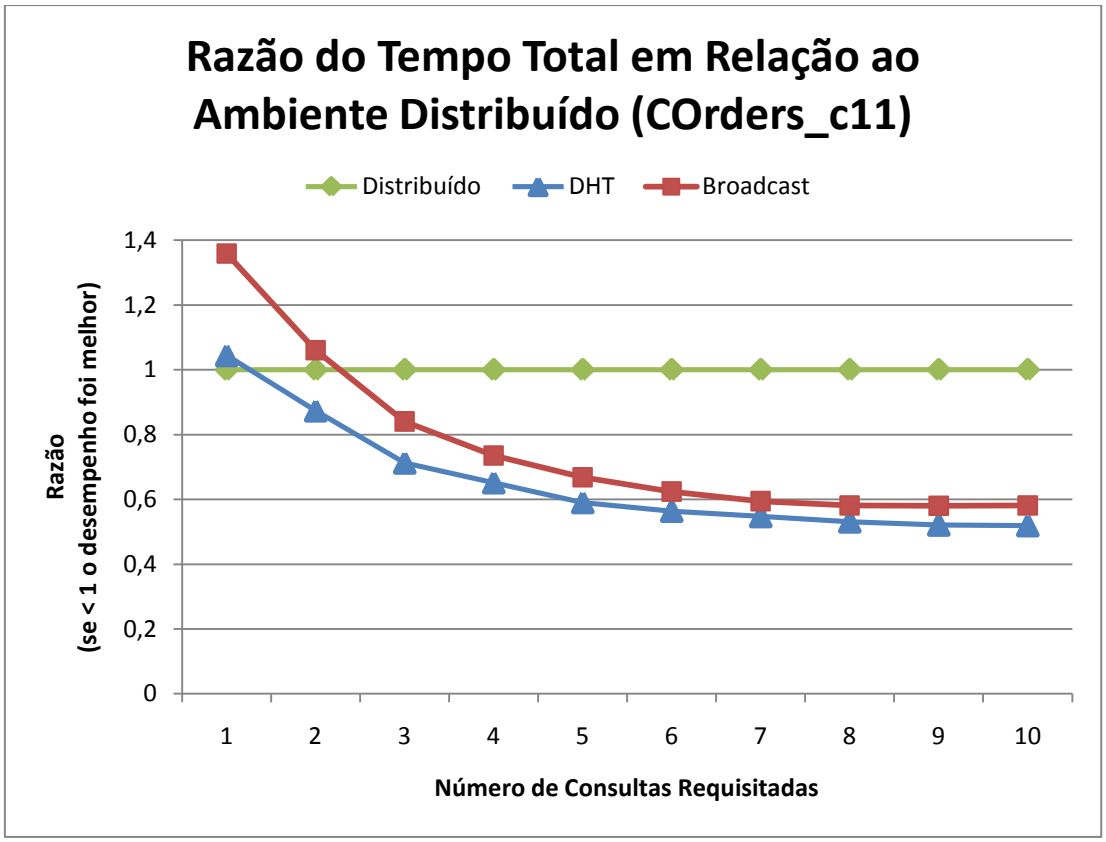


Figura 35 - Comparação do tempo total de execução de uma série de consultas XQuery *COrders_c11* no ambiente distribuído (normalizado) e nos ambientes P2P

Com esses resultados, podemos concluir que consultas XQuery que se beneficiam e que não se beneficiam da fragmentação distribuída pelos pontos de um ambiente P2P apresentaram melhores resultados quando comparadas ao ambiente distribuído, quando submetidos a uma série de requisições solicitadas. Porém, os maiores ganhos foram observados para as consultas que não se beneficiam da fragmentação distribuída pelos pontos do ambiente experimental, visto que o processamento distribuído deste tipo de consulta no ambiente distribuído é altamente custoso, tornando o desempenho no ambiente P2P bastante superior.

Por fim, podemos comprovar a escalabilidade proporcionada pelo uso de uma abordagem P2P, de acordo com o objetivo principal proposto pela Fase 2 (ver Seção 6.2.3.2) da metodologia de execução dos experimentos. Apesar do significativo aumento no tempo gasto para as etapas adaptadas do processamento de consultas distribuídas em ambientes P2P, de acordo com os resultados obtidos pela Fase 1 (ver Seção 6.3.1) da metodologia proposta, a escalabilidade de um ambiente P2P compensa este tempo adicional despendido.

6.4. Considerações Finais

Podemos concluir com base nos resultados dos experimentos que o processamento de consultas XQuery distribuídas em um ambiente P2P é possível a partir de uma fragmentação de bases XML distribuídas pelos pontos da rede P2P, como proposto neste trabalho. Os resultados obtidos nos mostram que se pode reduzir o tempo total de execução de consultas XQuery em até 77% quando submetidas em pontos disponíveis de uma rede P2P, dependendo da consulta e do protocolo P2P utilizado.

Através da execução dos experimentos, também contribuímos para a avaliação das principais técnicas de localização dos dados no ambiente P2P proposto. O protocolo P2P estruturado apresenta melhores resultados para o processamento de consultas distribuídas em ambientes P2P. Levando em consideração que os experimentos foram realizados em uma rede P2P estática, verificamos que o DHT sobressaiu-se em todos os cenários, sendo mais eficiente quando comparado ao protocolo P2P não-estruturado (*Broadcast*).

Porém, não podemos garantir que a estratégia estruturada também se mostre mais eficiente em ambientes P2P de topologia dinâmica. Nota-se que, embora a maioria das aplicações P2P possua alto índice de desconexões e de rotatividade de pontos na rede, isso não é uma regra. Existem aplicações P2P que são executadas em ambientes de maior confiabilidade e por um número restrito de pontos, e, por conseguinte, são mais "bem comportadas", conforme o ambiente P2P proposto nesta dissertação. Então, a análise da propriedade de dinâmica da rede mostrou que em contextos menos dinâmicos, o protocolo P2P estruturado (DHT) apresenta maiores ganhos de desempenho do que o protocolo P2P não-estruturado (*Broadcast*).

Capítulo 7. Conclusões e Trabalhos Futuros

7.1. Considerações Finais

Esta dissertação apresentou uma solução para o processamento de consultas XQuery distribuídas em um ambiente P2P. O objetivo foi atingido a partir da definição de uma metodologia inspirada na metodologia proposta por Figueiredo (2007), que contempla as etapas de decomposição da consulta submetida, incluindo sua representação na álgebra TLC (PAPARIZOS et al., 2004); localização dos dados; otimização global e local; além da criação das sub-consultas e sua execução nas bases XML dos pontos remotos; e consolidação dos resultados.

Um dos problemas mais críticos no processamento de consultas distribuídas em um ambiente P2P está relacionado à reformulação da consulta submetida. No contexto P2P, a consulta não é mais submetida a uma coleção XML específica e, portanto, torna-se necessária a identificação dos pontos da rede P2P que possuem dados relevantes para a resposta da consulta submetida. Logo, as etapas de decomposição da consulta e de localização dos dados no processamento de consultas distribuídas (FIGUEIREDO, 2007) precisaram ser adaptadas para a abordagem P2P.

O diferencial entre esta proposta e os outros principais sistemas P2P para o processamento de consultas distribuídas (GALANIS et al., 2003; PAPADIMOS et al., 2003; BONIFATI et al., 2006; FEGARAS et al., 2006; CONFORTI et al., 2007) está relacionado à utilização de uma definição formal para as etapas do processamento das consultas XML distribuídas e do uso de uma álgebra XML para a reescrita da consulta XQuery em expressões algébricas, a fim de tornar o processamento da consulta XML correto e automático no ambiente P2P. Além disso, a alta expressividade utilizada da linguagem XQuery também representa um fator diferencial para a metodologia proposta no ambiente P2P. Desta forma, a metodologia proposta depende da linguagem XML de consulta.

Para implementar a metodologia proposta, estendemos a arquitetura proposta por Figueiredo (2007) baseada em um Mediador com Adaptadores (WIEDERHOLD, 1992). Os componentes da arquitetura proposta são distribuídos por todos os pontos disponíveis na rede P2P: Mediador, Catálogo e Adaptador. O Mediador realiza as modificações necessárias na consulta original de forma a tornar a distribuição das bases XML entre os pontos relevantes da rede P2P transparente ao usuário final. É importante ressaltar que o Mediador utiliza um esquema global de dados para a reformulação da consulta. A localização dos pontos que possuem dados relevantes para a consulta submetida pode ser realizada a partir de diversas técnicas levantadas: (1) através de uma mensagem de *broadcast* para um número limitado de pontos na rede; (2) através da utilização de um índice distribuído entre os pontos da rede P2P; ou (3) através das informações disponíveis no Catálogo atualizado do ponto solicitante da consulta.

Os experimentos realizados em laboratório mostram que a solução proposta pode apresentar ganhos de até 77% no tempo de execução de consultas XQuery submetidas em pontos disponíveis de uma rede P2P em relação ao tempo de execução das mesmas consultas no ambiente distribuído (FIGUEIREDO, 2007). Através da execução dos experimentos, também contribuímos para a avaliação das principais técnicas de localização dos dados no ambiente P2P proposto. O protocolo P2P estruturado apresenta melhores resultados para o processamento de consultas distribuídas em ambientes P2P quando comparado ao protocolo P2P não-estruturado. A análise da propriedade de dinâmica da rede mostrou que em contextos menos dinâmicos, o protocolo P2P estruturado apresenta maiores ganhos de desempenho do que o protocolo P2P não-estruturado.

7.2. Trabalhos Futuros

O processamento de consultas distribuídas é um assunto bastante complexo, principalmente quando aplicado ao ambiente P2P. Portanto, as diferentes etapas do processamento de uma consulta em um ambiente P2P podem ser continuamente evoluídas, em especial a etapa de localização dos dados relevantes entre os pontos disponíveis do ambiente P2P.

Desta forma, temos a intenção de enriquecer a metodologia proposta de forma a obter uma metodologia para integração de dados em um ambiente P2P, solucionando

questões relacionadas aos fatores de heterogeneidade, distribuição e autonomia das fontes de dados. Esta metodologia busca desenvolver soluções para o problema de reformulação da consulta submetida em função dos dados localizados entre os pontos relevantes da rede P2P, utilizando mapeamentos semânticos entre os esquemas heterogêneos como forma de enriquecer toda a metodologia e conseqüentemente otimizá-la. Os mapeamentos semânticos visam descrever relacionamentos entre os termos usados em dois ou mais esquemas. Isto significa que soluções para o problema de reformulação da consulta devem incluir linguagens para a especificação dos mapeamentos e algoritmos que usem estes mapeamentos para resolver e responder adequadamente às consultas.

Do ponto de vista da arquitetura proposta, testes com ambientes P2P em contextos mais dinâmicos seriam interessantes, para comparar com os resultados obtidos nos protocolos P2P (DHT e *Broadcast*) em um ambiente P2P estático.

Referências Bibliográficas

ABERER, K., HAUSWIRTH, M., 2002, "An Overview on Peer-To-Peer Information Systems". *Workshops on Distributed Data and Structures (WDAS)*, v.36, n. 4, pp. 1-14.

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, USA.

ABITEBOUL, S., BONIFATI, A., COBENA, G., 2003, "Dynamic XML documents with distribution and replication". In: *Proceedings of the 2003 SIGMOD international conference on Management of data*, pp. 527-538, San Diego, California, USA.

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, "Efficiently processing XML queries over fragmented repositories with PartiX". In: *DATAX - EDBT Workshop Proceedings*, pp. 150-163, Munich, Germany.

APACHE TOMCAT, 2009, "The Apache Software Foundation – Apache Tomcat 5.5". Disponível em: http://tomcat.apache.org/download-55.cgi/. Acesso em: 08 dez. 2009.

BALAKRISHNAN, H., KAASHOEK, M., KARGER, D., et al., 2003, "Looking up data in P2P systems". *Communications of the ACM*, v. 46, n. 2, pp. 43-48.

BARBOSA, D., MENDELZON, A., KEENLEYSIDE, J., et al., 2002, "ToXgene: a template-based data generator for XML". In: *Proceedings of 5th International WebBD Workshop*, pp. 49-54, Wisconsin, USA.

BERGLUNG, A., BOAG, S., CHAMBERLIN, D., et al., 2007, "XML Path Language (XPath) 2.0 – W3C Recommendation 23 January 2007". Disponível em: http://www.w3.org/TR/xpath20/. Acesso em: 11 dez. 2009.

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, n. 4, pp. 602-625.

BITTORRENT, 2009, "BitTorrent". Disponível em: http://www.bittorrent.com/. Acesso em: 05 set. 2009.

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/. Acesso em: 09 jan. 2010.

BONIFATI, A., CHRYSANTHIS, P., OUKSEL, A., SATTLER, K., 2008, "Distributed Databases and Peer-to-Peer Databases: Past and Present". *ACM SIGMOD Record*, v. 37, n. 1, pp. 5–11.

BONIFATI, A., CUZZOCREA, A., 2006, "Storing and retrieving XPath fragments in structured P2P networks". *Data & Knowledge Engineering*, v. 59, n. 2, pp. 247-269.

BOOTH, D., HAAS, H., MCCABE, F., et al., 2004, "Web Services Architecture". Disponível em: http://www.w3.org/TR/ws-arch/. Acesso em: 10 jan. 2010.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, et al., 2008, "Extensible Markup Language (XML) 1.0 – W3C Recommendation 26 November 2008". Disponível em: http://www.w3.org/TR/xml/. Acesso em: 09 jan. 2010.

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

BRITO, G., 2005, *Integração de Objetos de Aprendizagem no Sistema ROSA-P2P*. Dissertação de M.Sc., Instituto Militar de Engenharia (IME), Rio de Janeiro, RJ, Brasil.

BRUNKHORST, I., DHRAIEF, H., KEMPER, A., NEJDL, W., WIESNER, C., 2003, "Distributed Queries and Query Optimization in Schema-Based P2P-Systems". In: *Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, pp. 184-199, Berlin, Germany.

CASTRO, M., DRUSCHEL, P., KERMARREC, A-M., et al., 2002, "Scribe: A large-scale and decentralized application-level multicast infrastructure". *IEEE Journal on Selected Areas in Communication (JSAC)*, v. 20, n. 8, pp. 100-110.

CHEN, Y., DAVIDSON, S., ZHENG, Y., 2004, "BLAS: An Efficient XPath Processing System". In: *Proceedings of the 2004 SIGMOD international conference on Management of data*, pp. 47-58, Paris, France.

CONFORTI, G., GHELL, G., MANGUI, P., SARTIANI, C., 2007, "Scalable Query Dissemination in XPeer". In: *Database Engineering and Applications Symposium International*, pp. 199-207, Los Alamitos, California, USA.

CRESPO, A., GARCIA-MOLINA, H., 2002, *Semantic overlay networks for P2P systems*. In: Technical report, Computer Science Department, Stanford University, California, USA.

DAREK, F., ZHAO, B., DRUSCHEL, P., et al., 2003, "Towards a common API for structured peer-to-peer overlay". In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA.

EPSTEIN, R., STONEBRAKERM, M., WONG, E., 1978, "Distributed query processing in relational database system". In: *Proceedings of 1978 ACM SIGMOD international conference on Management of data*, pp. 169-180, Austin, Texas, USA.

EXIST, 2009, "eXist-db: Open Source Native XML Database", v. 1.2.5. Disponível em: http://exist.sourceforge.net/. Acesso em: 12 dez. 2009.

FEGARAS, L., HE, W., DAS, G., LEVINE, D., 2006, "XML Query Routing in Structured P2P System". In: *DBISP2P*, pp. 273-284, Seoul, Korea.

FERNANDES, D. Y. S., 2007, *Reformulação de Consulta Baseada em Semântica para PDMS*. Exame de Qualificação e Proposta de Tese, UFPE, Recife, PE, Brasil.

FIGUEIREDO, G., 2007, *Processamento de Consultas sobre Bases XML Distribuídas*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FIGUEIREDO, G., BRAGANHOLO, V., MATTOSO, M. L. Q., 2007, "Um Mediador para o Processamento de Consultas sobre Bases XML Distribuídas". In: *Sessão de Demos do Simpósio Brasileiro de Banco de Dados (SBBD)*, pp. 21-26, João Pessoa, PB, Brasil.

FREEPASTRY, 2009, "FreePastry", v. 2.1. Disponível em: http://www.freepastry.org/. Acesso em: 08 dez. 2010.

GALANIS, L., WANG, Y., JEFFERY, S.R., et al., 2003, "Locating data sources in large distributed systems". In: *Proceedings of the 29th international conference on very large data bases (VLDB)*, pp. 874-885, Berlin, Germany.

GOLDMAN, R., WIDOM, J., 1997, "DataGuides: Enabling query formulation and optimization in semistructured databases". In: *Proceedings of the 23rd international conference on very large data bases (VLDB)*, pp. 436-445, Athens, Greece.

HUEBSCH, R., CHUN, B. N., HELLERSTEIN, J. M., et al., 2005, "The architecture of Pier: An internet-scale query processor". In: *Proceedings of the 2005 CIDR Conference*, pp. 28-43, Asilomar, California, USA.

JXTA, 2009, "JXTA". Disponível em: http://www.jxta.org/. Acesso em: 05 dez. 2009.

KAZAA, 2009, "KaZaA". Disponível em: http://www.kazaa.com/. Acesso em: 05 dez. 2009.

KOLONIARI, G., PITOURA, E., 2005, "Peer-to-Peer Management of XML Data: Issues and Research Challenges". *ACM SIGMOD Record*, v. 34, n. 2, pp. 6-17.

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

LOO, B. T., HELLERSTEIN, J. M., HUEBSCH, R., et al., 2004, "Enhancing P2P file-sharing with an internet-scale query processor". In: *Proceedings of the Thirtieth international conference on very large data bases (VLDB)*, pp. 432-443, Toronto, Canada.

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

NAPSTER, 2009, "Napster". Disponível em: http://www.napster.com/. Acesso em: 05 dez. 2009.

NEJDL, W., SIBERSKI, W., SINTEK, M., 2003, "Design issues and challenges for RDF and schema-based peer-to-peer systems." *ACM SIGMOD Record*, v. 32, n. 3, pp. 41-46.

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

PAPADIMOS, V., MAIER, D., TUFTE, K., 2003, "Distributed Query Processing and Catalogs for Peer-to-Peer Systems". In: *Proceedings of the 2003 CIDR Conference*, pp. 5-8, Asilomar, California, USA.

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

PIRES, C. E. S., 2009, *Ontology-based Clustering in a Peer Data Management System*. Tese de D.Sc, UFPE, Recife, PE, Brasil.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., et al., 2001, "A scalable content-addressable network". In: *Proceedings of the 2001 ACM SIGCOMM conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 161–172, San Diego, CA, USA.

RIZZOLO, F., MENDELZON, A., 2001, "Indexing XML Data with ToXin". In: *Proceedings of the Fourth International Workshop on the Web and Databases (WebDB)*, pp. 49–54, Santa Barbara, California, USA.

ROWSTRON, A., DRUSCHEL, P., 2001a, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility". *ACM SIGOPS Operating Systems Review*, v. 35, n. 5, pp. 188-201.

ROWSTRON, A., DRUSCHEL, P., 2001b, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems". In: *Proceedings of the IFIP/ACM international conference on distributed systems platforms Heidelberg*, pp. 329-350, Springer-Verlag.

SARTIANI, C., MANGHI, P., GHELL, G., CONFORTI, G., 2004, "XPeer: A Self-organizing XML P2P Database System". In: *Proceedings of the First EDBT Workshop on P2P and Databases*, Crete, Greece.

SCOLLMEIER, R., 2001, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architetures and Applications". In: *Proceedings of the first international conference on Peer-to-Peer Computing*, pp. 27-29, Linkoping, Sweden.

SKYPE, 2009, "SKYPE". Disponível em: http://www.skype.com/. Acesso em: 05 dez. 2009.

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

STOICA, I., MORRIS, R., KARGER, D., et al., 2001, "Chord: A scalable peer-to-peer lookup service for internet applications". In: *Proceedings of the 2001 ACM SIGCOMM conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 149-160, San Diego, California, USA.

STONEBRAKER, M., 1986, "The design and implementation of distributed INGRES", *The INGRES papers: anatomy of a relational database system*, pp. 187-196, Boston, MA, USA, Addison-Wesley Longman Publishing Co.,Inc.

TATARINOV, I., IVES, Z., MADHAVAN, J., et al., 2003, "The Piazza Peer Data Management Project", *ACM SIGMOD Record*, v. 32, n. 3, pp. 47-52.

TSOUMAKOS, D., ROUSSOPOULOS, N., 2003, "A comparison of Peer-to-Peer search methods". In: *Proceedings of the Ninth International Workshop on the Web and Databases (WebDB)*, pp. 61-66, Chicago, Illinois, USA.

W3C, W. W. W. C., 2009, "World Wide Web Consortium". Disponível em: http://www.w3.org. Acesso em: 12 dez. 2009.

WALDMAN, M., RUBIN, A., CRANOR, L., 2000, "Publius: A robust, tamper-evident, censorship-resistant web publishing system". In: *Proceedings of 9th USENIX Security Symposium*, pp. 59-72, Denver, Colorado, USA.

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

WILLIAMS, R., DANIELS, D., HAAS, L., et al., 1986, "R*: an overview of the architecture", In: *Readings in database systems*, v. 2, *Distributed systems, Vol. II: distributed data base systems*, Artech House, Inc, pp. 196-218.

YAO, B., ÖZSU, M. T., KHANDELWAL, N., 2004, "XBench Benchmark and Performance Testing of XML DBMSs". In: *Proceedings of the 20th international conference on Data Engineering (ICDE)*, pp. 621 – 632, Boston, MA, USA.

YING, Y., LE, J., 2005, "Catalog Search for XML Data Sources in Peer-to-Peer Systems". In: *Proceedings of Information Networking, Convergence in Broadband and Mobile Networking International Conference (ICOIN)*, pp. 600-608, Jeju Island, Korea.

ZHUGE, H., LIU, J., FENG, L., et al., 2005, "Query Routing in a Peer-to-peer Semantic Link Network", *Computational Intelligence*, v. 21, n. 2, pp. 197-216.

Anexo 1. Consultas XQuery

Consultas XQuery	XQuery	Benefício	Complex.

COrders_c01.xq	<results>
	{
	for $order in (some document)/order
	where $order/@id = "1"
	return
	<order>
	{ $order }
	</order>
	}
	</results>
		Nenhum	Não

COrders_c02.xq	<results>
	{
	for $a in (some document)/order
	where $a/@id = "3"
	return
	<items>
	{ $a//order_line/item_id }
	</items>
	}
	</results>
		Nenhum	Não

COrders_c03.xq	<results>
	{
	for $a in (some document)/order
	where $a/total > 11000
	order by $a/ship_type, $a/@id
	return
	<Output>
	{$a/@id}
	{$a/order_date}
	{$a/ship_type}
	</Output>
	}
	</results>
		Parcial	Não

COrders_c04.xq	<results>
	{
	for $a in (some document)/order
	where $a/total > 11000
	order by $a/total descending, $a/@id
	return
	<Output>
	{$a/@id}
	{$a/order_date}
	{$a/total}
	</Output>
	}
	</results>
		Parcial	Não

COrders_c05.xq	<results>
	{
	for $a in (some document)/order
	where $a/@id = "5"
	return
	<Output>
	{$a/order_lines}
	</Output>
	}
	</results>
		Nenhum	Não

COrders_c06.xq	<results>
	{
	for $a in (some document)/order
	where count($a/order_lines/order_line) = 1
	order by $a/@id
	return
	<Output>
	{$a/@id}
	</Output>
	}
	</results>
		Nenhum	Não

COrders_c07.xq	<results>
	{
	for $a in (some document)/order
	where $a/@id = "6"
	return
	<Output>
	{$a}
	</Output>
	}
	</results>
		Nenhum	Não

COrders_c08.xq	<results>
	{
	for $order in (some document)/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>
		Parcial	Sim

COrders_c09.xq	<results>
	{
	for $order in (some document)/order
	where $order/total > 7000
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>
		Parcial	Não

COrders_c10.xq	<results>
	{
	for $order in (some document)/order
	where $order/total > 10000
	order by $order/ship_date, $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>
		Total	Não

COrders_c11.xq	<results>
	{
	for $order in (some document)/order
	where $order/total > 10000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>
		Total	Não

COrders_c12.xq	<results>
	{
	for $order in (some document)/order
	where $order/total > 7000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>
		Parcial	Não

COrders_c13.xq	<results>
	{
	for $order in (some document)/order
	where $order/total > 7000
	and $order/total < 8000
	order by $order/@id
	return
	<order>
	{ $order/@id }
	{ $order/ship_date }
	{ $order/total }
	</order>
	}
	</results>
		Parcial	Não

COrders_c14.xq	<results>
	{
	for $order in (some document)/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>
		Total	Sim

COrders_c15.xq	<results>
	{
	for $order in (some document)/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>
		Nenhum	Sim

COrders_c16.xq	<results>
	{
	for $order in (some document)/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>
		Parcial	Sim