UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA PROPOSTA DE PERSONALIZAÇÃO DE CONSULTAS EM DOCUMENTOS XML BASEADA EM PREFERÊNCIAS CONDICIONAIS

ANGÉLICA FELIX MEDEIROS

JOÃO PESSOA-PB
Março-2015

ANGÉLICA FELIX MEDEIROS

UMA PROPOSTA DE PERSONALIZAÇÃO DE CONSULTAS EM DOCUMENTOS XML BASEADA EM PREFERÊNCIAS CONDICIONAIS

DISSERTAÇÃO APRESENTADA AO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA, COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE EM INFORMÁTICA (SISTEMAS DE COMPUTAÇÃO).

Orientador: Prof. Dr. Valéria Gonçalves Soares

JOÃO PESSOA-PB
Março-2015

M488u Medeiros, Angélica Felix.
Uma proposta de personalização de consultas em documentos XML baseada em preferências condicionais / Angélica Felix Medeiros.-- João Pessoa, 2015.
89f. : il.
Orientadora: Valéria Gonçalves Soares
Dissertação (Mestrado) – UFPB/CI
1. Informática. 2. Ciência da computação. 3. Computação distribuída. 4. Personalização de consultas. 5. XQuery-Pref. 6. XQPref.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **Angélica Felix Medeiros**, candidata ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 27 de março de 2015.

Aos vinte e sete dias do mês de março do ano de dois mil e quinze, às dez horas, no laboratório 2 da Escola Superior de Redes - Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar a candidata ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", a **Sra. Angélica Felix Medeiros**. A comissão examinadora foi composta pelos professores doutores: **Valéria Gonçalves Soares (PPGI-UFPB)** orientadora e presidente da Banca, **Alexandre Nóbrega Duarte (UFPB)**, examinador interno, **Danielle Rousy Dias da Silva (UFPB),** examinadora externa ao Programa, **Ana Carolina Brandão Salgado (UFPE)** examinadora externa à Instituição. Dando início aos trabalhos, a professora Valéria Gonçalves Soares cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra à candidata para que a mesma fizesse, oralmente, a exposição do trabalho de dissertação intitulado *"Uma Proposta de Personalização de Consultas em Documentos XML Baseada em Preferências Condicionais"*. Concluída a exposição, a candidata foi arguida pela Banca Examinadora que emitiu o seguinte parecer: "*aprovada*". Eu, Nadja Rayssa Soares de Almeida, Secretária do Programa de Pós Graduação em Informática - PPGI, lavrei a presente ata que vai assinada por mim e pelos membros da Banca Examinadora. João Pessoa, 27 de março de 2015.

Nadja Rayssa Soares de Almeida

Profª Valéria Gonçalves Soares
Orientadora (PPGI-UFPB)

Profº Alexandre Nóbrega Duarte
Examinador Interno (PPGI-UFPB)

Profª Danielle Rousy Dias da Silva
Examinadora Externa ao Programa (UFPB)

Profª Ana Carolina Brandão Salgado
Examinadora Externa à Instituição (UFPE)

Resumo

O presente trabalho apresenta uma proposta de extensão da linguagem de consulta XQuery, denominada XQuery-Pref, para suporte a preferências condicionais. Com objetivo de tornar transparente para o usuário a execução de consultas escritas na linguagem XQuery-Pref, utiliza-se um sistema denominado XQPref, que é responsável pela elicitação de preferências dinâmicas e pelo processamento destas consultas personalizadas. Restringimos o escopo deste trabalho a documentos XML, tecnologia essencial para a gestão do conhecimento e divulgação de dados pela web, que diante da sobrecarga da informação, tem despertado a preocupação em personalizar os resultados de consultas de acordo com as necessidades de cada usuário.

Palavras-chave: Personalização de consultas, XQuery-Pref, XQPref.

Abstract

This work presents an extension of the XQuery query language called XQuery-Pref, to support conditional preferences. In order to make transparent to the user executing queries written in language XQuery-Pref, we use a system called XQPref, which is responsible for elicitation of preferences dynamics and the processing of these custom queries. Restrict the scope of this work to XML documents, essential technology for knowledge management and dissemination of data on the Web, that given the information overload, has aroused concern in personalizing query results according to the needs of each user.

Keywords: Personalization of queries, XQuery-Pref, XQPref.

Agradecimentos

A Deus, por ter me proporcionado esta incrível experiência, por ter iluminado em cada decisão a ser tomada, clareando o meu caminho durante esta jornada e me guiando sempre pelo melhor caminho.

Aos meus pais, avós, irmãos e demais familiares por terem compartilhado os meus ideais, me incentivando a prosseguir na jornada, fossem quais fossem os obstáculos.

A minha orientadora Valéria Gonçalves Soares pelo tempo, compreensão e incentivo dedicados que tornaram possível a conclusão desta dissertação.

A minha Cúmplice Intelectual e amiga Ayslânya, juntas, encontramos a força e o incentivo necessário para que continuássemos caminhando.

Por fim, agradeço em especial ao meu amigo, companheiro e noivo Vinicius, pela compreensão nos momentos difíceis, pelo ombro amigo para ouvir meus desabafos e silêncios, pelos momentos de ausência e extremos de alegria e de tristeza. O meu "muito obrigada" é tão sincero e tão intenso quanto o nosso amor.

Conteúdo

Lista de Símbolos

CP-nets - *Conditional Preference Networks*

CSV - *Comma-Separated Values*

JSON - *JavaScript Object Notation*

ODF - *Open Document Spreadsheet*

OQL- *Object Query Language*

RDF - *Resource Description Framework*

SQL - *Structured Query Language.*

TCP-nets - *tradeoffs-enhanced CP-net*

W3C – *World Wide Web*

XML- *eXtensible Markup Language*

XML-QL - *XML Query Language*

XQL- *XML Query Language*

XQuery - *XML Query Language*

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Códigos Fonte

Capítulo 1

Introdução

1.1 Motivação

O crescimento exponencial de informação disponível em diversos domínios tem despertado a preocupação dos pesquisadores em personalizar os resultados de consultas de acordo com o perfil do utilizador [Amo et al. 2012], abrindo assim, possibilidades de pesquisa em áreas como Banco de Dados, Inteligência Artificial e Recuperação de Informação.

Dentre estes domínios, o XML é uma tecnologia essencial para a gestão do conhecimento e divulgação de dados pela web. No entanto com a sobrecarga de informações, o usuário pode ser confrontado com uma quantidade muito grande de resultados, o que tem impulsionado a procura de técnicas para resolver problemas associados com a busca em documentos XML [Santos et al, 2012].

Um exemplo recente de domínio que possui grande quantidade de informação em formato XML são os Dados Abertos, que consiste da publicação e disseminação de dados e informações públicas na Web, respeitando critérios que possibilitam sua reutilização e o desenvolvimento de aplicativos por toda a sociedade.

A inclusão digital aliada à informatização dos procedimentos governamentais e a integração entre os diversos repositórios de dados públicos provocam crescentes demandas da população por mais transparência e participação através de meios tecnológicos. Nessa direção, o governo brasileiro tem definido políticas e desenvolvido plataformas tecnológicas na intenção de promover a disseminação das informações públicas.

O paradigma de dados abertos está fundamentado na constatação de que o dado, quando compartilhado abertamente, tem seu valor e seu uso potencializados [Cartilha, 2011], mas para tanto é necessário facilitar a análise dos dados por parte dos usuários, para que assim esses dados se transformem em informação.

Como solução para realizar consultas em documentos XML, foram propostas algumas linguagens que devido às diferentes origens, possuem distinções consideráveis em termos de sintaxe e abrangência. Dentre elas, a linguagem XML-QL [Deutsch et al. 1998] é baseada numa sintaxe XML textual, já XML Query Language -XQL [Robie, 2009], que foi proposta pela comunidade de processamento de documentos, explora expressões similares às expressões de caminho em diretórios. Enquanto a XQuery [Chamberlin et al. 2001] é uma linguagem flexível, originada da mistura de ambos os tipos de linguagem.

Por outro lado, mesmo a linguagem mais completa para consulta em XML não possui base para personalização de consultas e assim realizar buscas em dados XML. Por exemplo, a busca de informações em dados abertos, tem se tornado maçante devido à sobrecarga da informação, o que tem dificultado ao usuário o acesso a essas informações.

Se por exemplo um usuário desejar fazer uma consulta a dados abertos governamentais utilizando a linguagem XQuery diretamente na base de dados governamentais uma grande quantidade de elementos do documento XML vai ser retornado como resultado desta consulta, dificultando a análise desse resultado, o entendimento. O que de fato interessa ao usuário vai estar no resultado da consulta, mas o usuário vai ter dificuldades no acesso deste conteúdo devido a quantidade de informações retornadas. Sendo assim, este fato é uma das primeiras preocupações deste trabalho o que direciona a pesquisa aos conceitos da personalização de consultas.

A personalização de consultas consiste no processo de dinamicamente melhorar uma consulta de acordo com preferências relatadas pelos usuários, produzindo respostas distintas para cada usuário [Koutrika e Ioannidis, 2005]. Deste modo, o objetivo da personalização é garantir que “as pessoas certas recebam as informações certas no momento certo”, levando em consideração que as pessoas podem ter diferentes preferências e prioridades quando se trata de suas necessidades de informação.

Dessa forma, tem se tornado necessárias técnicas que facilitem este processo de consulta e, com isso, diversos estudos procuram auxiliar na busca de conteúdo em domínios de grande quantidade de informação, visando identificar os itens que podem ser atraentes para cada usuário levando em consideração as suas preferências pessoais.

O presente trabalho busca personalizar as consultas em documentos XML através da mineração de preferências. Com isso, pode-se reduzir os esforços do usuário na busca por

conteúdos específicos. Para tanto, é proposta uma extensão da linguagem XQuery com suporte a preferências condicionais, que foi denominada como XQuery-Pref. Essa extensão utiliza um sistema denominado XQPref que será responsável pela elicitação de preferências dinâmicas e pelo processamento destas consultas personalizadas. Essa escolha permite-nos tornar as consultas escritas na linguagem XQuery-Pref transparentes para o usuário.

1.2 Objetivos

O objetivo do trabalho é especificar uma extensão, com suporte a preferências condicionais, para uma linguagem de consultas a documentos XML. O modo escolhido para estender a linguagem base foi à definição de novas funções. Dessa forma, não é modificada a especificação original da linguagem de consultas.

Para atingir tal objetivo faz-se necessário definir a linguagem de consulta a documentos XML que receberá a extensão, identificar o formalismo lógico necessário para o suporte a consultas contendo preferências condicionais, especificar e implementar as funções da XQuery-Pref e o sistema XQPref necessário para utilizar a extensão da linguagem de consulta e verificar a acurácia da proposta através de consultas em dados reais.

1.3 Organização

Este documento está organizado sete capítulos. No Capítulo 2, é apresentada a fundamentação teórica, para tanto o capítulo em três seções para abranger as três grandes áreas envolvidas nesta pesquisa.

Deste modo, no Capítulo 2 são apresentados os conceitos que envolvem Dados Abertos Governamentais, dando ênfase a sua importância para a sociedade e destacando a utilização do formato XML. Em seguida, são expostos os conceitos sobre XML e suas linguagens de consulta. E abrangendo ainda os principais conceitos e formalismos inerentes à modelagem e representação de preferências, bem como uma visão geral sobre o tema personalização de consultas.

O Capítulo 3 apresenta uma revisão bibliográfica, destacando os principais trabalhos envolvendo dados abertos governamentais, a personalização de consultas e/ou a extensão de linguagens de consulta que fundamentaram esta pesquisa.

A proposta desta pesquisa é apresentada no Capítulo 4, onde inicialmente são especificadas as definições iniciais e formalizado o problema da modelagem de preferências.

Em seguida é apresentada a especificação da extensão da linguagem XQuery. Enquanto no Capítulo 5 é apresentado o sistema que viabiliza a utilização da XQuery-Pref para realizar consultas personalizadas, destacando tal sistema como contribuição secundária do trabalho.

O estudo de caso é apresentado no Capítulo 5, onde estão descritos e discutidos detalhes da avaliação realizada para verificar a consistência da XQuery-Pref. E por fim, no Capítulo 6 são apresentadas as considerações finais do trabalho, destacando suas principais contribuições, limitações e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Personalização de Consultas

Neste capítulo é apresentado o referencial teórico sobre personalização de consultas, relacionando o tema com a mineração de preferências. Desta forma, o capítulo foi dividido em três partes: conceitos, representação e modelos de preferências e na última seção serão enaltecidas as considerações finais do capítulo, relacionando com a presente pesquisa.

2.1.1 Conceitos

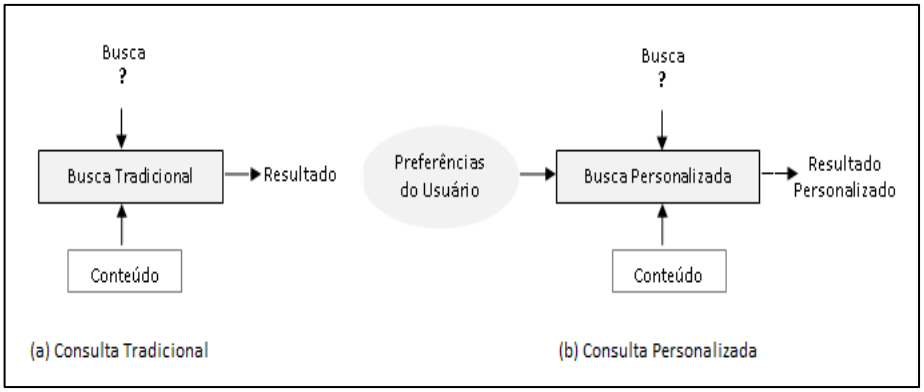
A personalização de consultas consiste no processo de melhorar dinamicamente uma consulta de acordo com preferências relatadas pelos usuários, produzindo respostas distintas para cada usuário [Koutrika e Ioannidis, 2005]. O objetivo é produzir resultados de consultas mais objetivos e de acordo com as necessidades dos usuários, reduzindo, portanto, os esforços empregados para a localização de conteúdo relevante.

O ponto chave desse estudo é que usuários distintos poderão obter resultados diferentes mesmo que efetuem a mesma consulta. Dessa forma, as preferências podem ser utilizadas para personalizar as buscas e realizar a filtragem da informação, reduzindo o volume de dados que devem ser apresentados para o usuário.

Observando a figura 2.1, é interessante analisar que na consulta tradicional a busca é realizada simplesmente consultando o conteúdo armazenado na base de dados, o que retornaria o mesmo resultado para qualquer usuário que realizasse determinada consulta. Enquanto que na consulta personalizada, além da busca no conteúdo da base de dados também é realizada a busca nas preferências do usuário, de modo que o resultado respeitaria estas preferências ao retornar o resultado da consulta, destacando o fato que usuários distintos ao realizarem a mesma consulta, caso possuam preferências diferentes, receberiam resultados diferentes.

Na figura 2.1, Koutrika [2010] destaca como deveriam ser a estrutura das consultas, levando em consideração as preferências do usuário.

Figura 2.1: Consulta tradicional versus Consulta personalizada

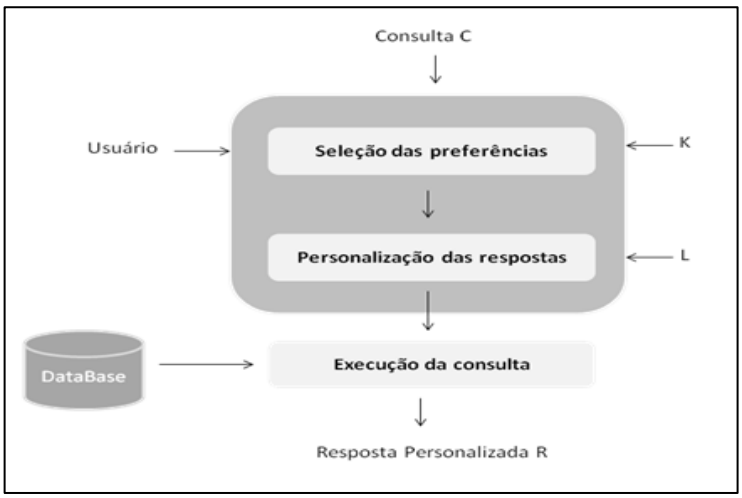


Fonte: Koutrika, "Query Personalization based on User Preferences". V. 35, Abril, New York, USA. 2010.

Jung et al. [2005] observa que o problema na maioria das representações de preferências existentes nem sempre correspondem ao senso comum de preferência. Porém, uma representação de preferência adequada é decisiva para a precisão do processamento e análise das características da preferência.

Para Koutrika [2010] o primeiro passo do processo de personalização de consulta lida com a extração do topo K preferências relacionadas a uma consulta. A questão básica a ser respondida por um sistema é como as preferências são consideradas e relacionadas a uma consulta. Na Figura 2.2, é mostrada como autor interpreta este processo.

Figura 2.2: Arquitetura para Personalização de Consultas



Fonte: Koutrika, "Query Personalization based on User Preferences". V. 35, Abril, New York, USA. 2010.

É interessante analisar a Figura 2.2 destacando que o usuário deve fornecer suas preferências e o sistema precisa ser capaz de, ao processar uma consulta, selecionar as

preferências que dizem respeito àquela busca. De modo que ao executar a busca na base de dados o sistema já saiba quais são as preferências que devem ser priorizadas e isto esteja refletido na resposta que o usuário receberá.

Otimizar este processo de seleção de preferências envolve o modelo de preferências utilizado pela arquitetura e os algoritmos de mineração. Os trabalhos em mineração de preferências normalmente partem do pressuposto que os usuários fornecem algumas informações iniciais sobre as suas preferências e estas funcionarão como entrada para os algoritmos de mineração de preferências. Estes temas serão abordados nas seções a seguir.

2.1.2 Representação e Modelos de Preferências

Entender as preferências do usuário e encontrar representações apropriadas para representá-las é o maior desafio para o tratamento de preferências. Com isso, existem algumas abordagens de modelos de preferências na literatura que lidam com representação de preferências e sua composição, para tentar alcançar significativas conclusões sobre as respostas desejadas de uma consulta a uma base dados de diferentes perspectivas.

A elicitação das preferências do usuário pode ser realizada de forma quantitativa ou qualitativa. Na abordagem quantitativa, as preferências são expressas através da atribuição de pontuações numéricas para tuplas do banco de dados, o que é infactível para bases de dados muito grande, como é o caso dos Dados Abertos Governamentais.

Nesta abordagem, uma tupla t_i é preferida sobre uma tupla t_j, se e apenas se a sua pontuação é maior do que a pontuação do t_j. Scores podem ser atribuídos através de funções de preferência [Agrawal e Wimmers, 2000] ou como graus de juros associada a condições específicas que devem ser satisfeitas [Koutrika e Ioannidis. 2005].

Para exemplificar a elicitação de preferências de forma quantitativa, considere uma base de dados governamental onde se deseja identificar as informações que são preferidas por um usuário. Para isto pode-se solicitar ao mesmo que dê uma nota a cada tupla da base e seleciona as que obtiverem maiores notas. É notável que tal método se torna inviável para bases de dados muito grande. Daí, outra forma de realizar a mesma tarefa seria a qualitativa, obtendo do usuário regras genéricas que refletem suas preferências. Por exemplo, se o usuário diz que prefere informações da região Nordeste a Sudeste, já se consegue obter uma classe de informações preferidas sem que o usuário avalie cada uma individualmente.

Na abordagem qualitativa, as preferências são especificadas tipicamente usando relações de preferência binárias, ou seja, o usuário especifica um conjunto de regras de preferência que devem ser respeitadas no resultado da consulta. Estas relações preferenciais podem ser especificadas usando fórmulas lógicas [Chomicki, 2003], ou construtores de preferências especiais [Kießling, 2002].

Nas preferências expressas por um usuário sobre um conjunto de objetos, mesmo que implicitamente, existe uma relação de ordem imposta pelas preferências entre os elementos do conjunto. Sendo assim, é importante compreender as possíveis relações de ordem que podem estar diretamente ligadas às preferências entre objetos, de modo que se a é preferido a b implica em a > b.

Formalmente, uma relação de preferências sobre um conjunto finito de objetos $A=\{a_1,a_2,...,a_n\}$ é uma relação de ordem sobre A, onde $R \subseteq C\ A \times A$ e satisfaz as seguintes propriedades:

- Irreflexiva: $\forall a \in A: (a, a) \notin R$;
- Transitiva: $\forall a, b, c \in A: (a, b) \in R$ e $(b, c) \in R$, então $(a,c) \in R$.

Assim, denotamos a < b se (a; b) $\in$ a relação de ordem denominada pelo símbolo <. Se < é uma relação de preferência entende-se por a < b o fato de b ser preferido em relação ao objeto a. E assim, uma relação de ordem é considerada total se $\forall a$, $\forall b \in A$, (a, b) $\in$ R ou (b, a) $\in$ R e *parcial* caso contrário.

Os formalismos para especificação de preferências abordam a modelagem das mesmas, descrevendo um padrão formal para tradução de uma linguagem natural para uma linguagem de especificação de preferências e quanto aos modelos de preferência, estes podem ser condicionais ou não condicionais.

Nas preferências não condicionais, um atributo X é preferencialmente independente de um atributo Y se e somente se para qualquer valor assumido por Y as preferências sobre o atributo X permaneçam as mesmas. Trazendo para o cenário da base de dados abertos governamentais, um usuário pode manifestar que sua preferência sobre os dados da região nordeste independe do conteúdo tratado na base (economia, violência, etc.) neste caso para informações de qualquer gênero, época da informação ou quaisquer outros atributos existentes na base, a preferência do usuário em relação à região do país da qual ele quer informação não vai mudar.

As preferências são ditas condicionais quando o valor de um atributo possui influência na preferência sobre os valores de outro atributo [Ribeiro, 2008] e apenas a elicitação qualitativa possibilita o uso de preferências condicionais. Com esta modelagem é possível que um usuário expresse a influência que um determinado atributo exerce sobre outro, ou seja, um atributo X é condicionalmente dependente de um atributo Y se para cada valor assumido por Y as preferências sobre X são alteradas. Por exemplo, o usuário pode expor que prefere informações sobre a região sul a da nordeste se o gênero tratado for economia, mas prefere a região nordeste se o gênero tratado for violência. Assim, o atributo região depende do atributo gênero, ou seja, o atributo região é condicionalmente dependente do atributo gênero.

Diante destes conceitos e unindo aos já revisados nas demais seções já é possível caracterizar que neste trabalho a elicitação das preferências será realizada de forma qualitativa e a modelagem destas preferências será a condicional, tendo em vista ainda que com a sobrecarga da informação também já mencionada são necessários métodos que facilitem cada vez mais o acesso à informação por parte dos usuários, prezando ainda a redução de esforço ao acessar dados cada vez mais relevantes. Com isso, serão aprofundados apenas os formalismos referentes às representações utilizadas na realização deste trabalho.

Dentre as principais propostas de modelagem e raciocínio de preferências condicionais, estão as CP-nets e TCP-nets [Boutilier et al. 2004, Brafman et al. 2006] e as preferências condicionais propostas pelo Wilson [2004] e estas são abordadas a seguir.

2.1.2.1 CP-nets e TCP-nets

CP-net é um formalismo que pode ser usado para definir relações de preferências de forma compacta, intuitiva e estruturada, usando definições *ceteris paribus* condicionais, sendo esta uma expressão do latim que pode ser traduzida como "todo o resto é igual". No contexto de preferência significa que para comparar dois objetos a partir de uma dada preferência, todos os demais atributos que não estão envolvidos na preferência devem coincidir em seus valores e neste formalismo as relações de ordem podem ser obtidas por transitividade. Esta técnica de inferência pretende abordar como trabalhar comparações entre preferências distintas e como encontrar a melhor escolha a partir de definições parciais.

Uma CP-net é um grafo dirigido sobre um conjunto de atributos onde cada nó representa um atributo e as arestas representam as dependências condicionais. Quando há uma aresta do atributo X para o atributo Y significa que as preferências sobre o atributo Y são

condicionadas pelo valor do atributo X. Cada atributo X possui vinculado ao seu respectivo nó uma tabela de preferência condicional (TCP) que descreve as preferências sobre o atributo X levando em conta suas dependências. A representação e a semântica apresentada nesse modelo objetiva usar o gráfico para capturar definições qualitativas sobre a independência condicional das preferências.

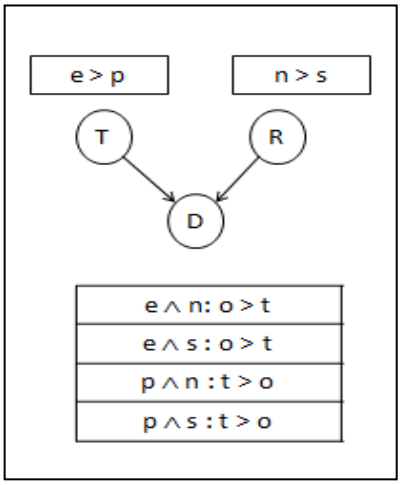
A principal vantagem da representação de instruções de preferências qualitativas por meio de CP-nets está na forma compacta com que as preferências são reproduzidas permitindo o uso de dependência preferencial condicional, em que a preferência sobre os valores de um atributo é influenciada pelos valores de outros atributos.

No exemplo a seguir, são expostas algumas preferências e criada uma CP-net que está representada na Figura 2.3:

Exemplo. Suponha as seguintes preferências de um usuário diante de opções de acesso a dados abertos governamentais:

- Informações sobre o tema (T) economia (e) são mais interessantes do que política (p);
- Prefiro informações sobre a região (R) nordeste (n) a sudeste (s);
- Para informações sobre o tema (T) política (p), prefiro Dados (D) orçamentais (o) a estatísticos (t). Caso contrário, são preferidas as informações estatísticas (t).

Figura 2.3: CP-net para acesso a Dados Abertos Governamentais



Observando a Figura 2.3 encontramos uma CP-net para as preferências especificadas no exemplo supracitado onde é possível aferir algumas interpretações, por exemplo, sugere que objetos com Tema = economia (e) são melhores que objetos com T = política (p), para todo o restante igual. Logo, o objeto o_1 = (e; n; t) é preferido a o_2 = (p; n; t), já que e > p e os

outros atributos assumem o mesmo valor (nordeste e Paraíba), mesmo que outra preferência sugira que dados estatísticos são preferidos a dados orçamentais.

Essa interpretação se dá pelo fato de que a preferência sobre o Tema que deve ser retornado é independente, enquanto as demais preferências são condicionadas a outros atributos.

Em Brafman et al. (2006) é descrita uma extensão da CP-net, chamada de TCP-net, que são grafos que possuem as mesmas características das CP-nets, com o adicional de mais dois tipos de arestas para representar as importâncias absoluta e relativa entre atributos. A Importância absoluta acontece quando, dados dois atributos X e Y preferencialmente independentes, X é mais importante que Y. Isto faz com que as preferências sobre os valores de X sobreponham as preferências sobre os valores de Y. Já na importância relativa, um atributo é mais importante do que outro de acordo com os valores de um terceiro atributo ou um conjunto de atributos. Isto é, dados três atributos X, Y e Z, X é mais importante que Y dependendo do valor assumido por Z.

Deste modo, retomando o exemplo anterior percebe-se que a preferência relacionada ao tema é manifestada pelo usuário como importância absoluta, e com isso torna-se preferível em relação às demais preferências.

2.1.2.2 Regras de preferências condicionais

Em Wilson [2004] foi especificada uma linguagem de preferências condicionais L que permite expressar preferências mais genéricas do que as tratadas pelas CP-nets e TCP-nets. A representação de preferências nesta linguagem considera um conjunto de atributos A = $\{X_1,...,X_n\}$, onde $Dom(X_i)$ denota o domínio de X_i, ou seja, os possíveis valores que podem ser assumidos por X_i. O conjunto $Dom(X_1) \times ... \times Dom(X_n)$ é denotado por O e representa o conjunto de objetos formados por todas as combinações possíveis de valores para os atributos de A. Se $o = (x_1,...,x_n)$ é um objeto então $o[X_i] = x$ denota valor x para o atributo X_i no objeto o.

A representação de preferências na linguagem é realizada através de regras de preferências condicionais, esta regra é uma sentença no formato $\varphi: u \rightarrow (X = x) > (X = x')[W]$, onde a condição u é uma fórmula no formato $(X_{i1} = x_1) \wedge ... \wedge (X_{ik} = x_k)$ onde $X_{ik} \in A$ e $x_j \in Dom(X_{ij})$ para todo $j \in \{1,...,k\}$. Sendo que $W \subseteq (A - \{X, X_{ij}, ..., X_{ik}\})$ é o conjunto de atributos desconsiderados pela regra de preferência. Quando se têm um conjunto

finito de regras de preferências condicionais, diz-se tratar uma Teoria de Preferência Condicional (Teoria-pc).

O exemplo a seguir, apresenta como as preferências do usuário são expressas através de uma teoria-pc.

Exemplo. Suponha que o cliente expresse as seguintes preferências sobre uma base de dados governamentais com os elementos (Tema, Região, Dado):

- Para qualquer tipo de dados (D), prefiro informações sobre o tema (T) economia (e) a violência (v);
- Se o tema for violência (v), prefiro me informar sobre violência doméstica (d) a infantil (i), para qualquer região (R) do país;

Tais preferências podem ser representadas pela teoria-pc $\Gamma = \{\varphi_1, \varphi_2\}$, onde:

$$\Gamma = \begin{Bmatrix} \varphi 1 : C = i > C = n\ [\{R\}], \\ \varphi 2 : C = n \rightarrow R = u > R = e\ [\{T\}]. \end{Bmatrix}$$

Onde:

- φ: $teoria\ de\ preferência\ condicional$;
- C: $Condição\ imposta\ pelo\ usuário$;
- i, n: $possíveis\ valores\ de\ C$;
- R: $Elementos\ que\ não\ foi\ considerado\ pelo\ usuário\ na\ preferência\ fornecida$;
- u, e: $Possíveis\ valores\ de\ R$.

O conjunto de regras de preferências definidas pelo usuário (que formam a teoria-pc) é passível de possuir inconsistências, podendo gerar contradições. Por exemplo, se num momento o usuário diz que prefere informações sobre o assunto violência à política, pode acontecer que ele expresse: assuntos relacionados à violência, fazendo com que o motor de inferência deduza que um assunto é melhor que ele mesmo.

Para evitar teorias-pc inconsistentes, foram propostas duas condições suficientes para que, dada uma teoria-pc Γ, consiga-se garantir que ela é consistente. Tais condições referem-se ao grafo de dependência preferencial e à consistência local de Γ, ou seja, um grafo de dependência preferencial representa a relação de dependência entre os atributos que aparecem nas regras-pc de Γ.

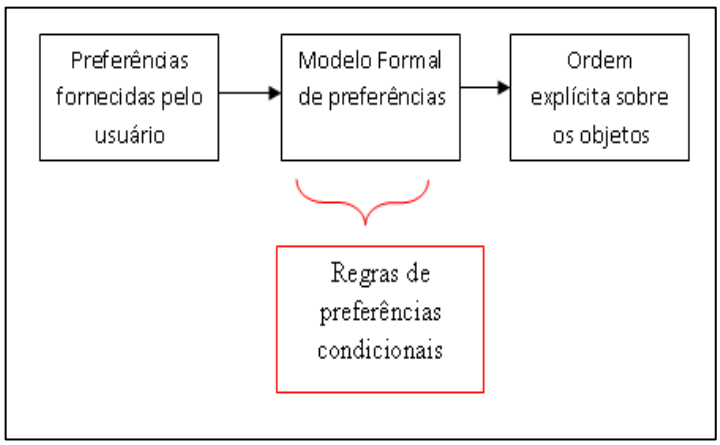
A consistência local, por sua vez, refere-se à ordem induzida sobre o domínio dos atributos. Uma teoria-pc é localmente consistente se as ordens locais produzidas em cada atributo do lado direito das regras são irreflexivas. Formalmente, tem-se:

Seja Γ uma teoria-pc sobre um conjunto de atributos A. Seja $U_\Gamma = \{u_{\varphi i} \,|\, \varphi_i \in \Gamma\}$. Para cada $u \in \Gamma$ e para cada X ∈ A, $R^*_{u,x} = \{(x_{\varphi i}, x'_{\varphi i}) \,|\, X_{\varphi i} = X \; e \; u \; satisfaz \; u_{\varphi i}\}$. Seja $\succ^X_u$ o fecho transitivo sobre $R^*_{u,x}$. Uma teoria-pc Γ é localmente consistente se e somente se para todo X ∈ A e para todo $u \in U_\Gamma$, a relação $\succ^X_u$ é irreflexiva.

Outra importante contribuição do trabalho de Wilson [2004] foi a especificação de um método para inferir os objetos ótimos a partir de uma Teoria de Preferência Condicional. O método consiste em: dada uma Teoria de Preferência Condicional $\Gamma = \{\varphi_1, \varphi_2, ..., \varphi_n\}$, seguir os passos:

- Obtém-se todos os pares de tuplas que pode-se ordenar diretamente utilizando cada uma das regras φ_i, onde $\{\{i = 1, \ldots, n\}$;
- Faz-se a união dos conjuntos de pares inferidos por cada regra φ_i;
- Inclui todos os pares de tuplas obtidos por transitividade.

Figura 2.4: Contribuição do trabalho de Wilson [2004]



Fonte: Wilson, N. (2004). Extending CP-Nets with Stronger Conditional Preference Statements. In 19th National Conference on Arti cial Intelligence (AAAI), pp.735-741, San Jose, California, USA.

Em resumo, no trabalho de Wilson [2004] é abordado um formalismo para especificação das preferências do usuário por meio de regras de preferências condicionais e a partir das regras obtidas é proposto um método para inferir uma ordem sobre os objetos de uma base de dados, como está sintetizado na Figura 2.4.

2.1.3 Considerações Finais

Diante destes conceitos é possível caracterizar que neste trabalho a elicitação das preferências será realizada de forma qualitativa e a modelagem destas preferências será a condicional, tendo em vista ainda que com a sobrecarga da informação também já mencionada são necessários métodos que facilitem cada vez mais o acesso à informação por parte dos usuários e prezando ainda a redução de esforço ao acessar dados cada vez mais relevantes.

Neste trabalho serão utilizados os formalismos propostos por Wilson [2004], mas também serão úteis conceitos especificados com a CP-net [Boutilier et al., 2004], para melhor compreender como serão tratadas as preferências na personalização de consultas proposta nesta pesquisa.

Muito se tem falado em relação ao tratamento de preferências, personalização de consultas em bancos de dados relacionais e aplicações comerciais, mas esta pesquisa estende essa problemática também para as consultas em dados XML que é uma tecnologia essencial para a gestão do conhecimento e divulgação de dados pela web, No entanto com a sobrecarga de informações o usuário, pode ser confrontado com uma quantidade muito grande de resultados, o que tem impulsionado a procura de técnicas sensíveis para resolver problemas associados com a busca em documentos XML.

Diante disto, na próxima seção o tema tratado será XML, abordando os principais conceitos e destacando suas vantagens, além da apresentação sobre suas linguagens de consultas.

2.2 XML e Linguagens de Consulta

Esta seção está dividida em três subseções. Na primeira, são apresentados conceitos de da linguagem XML, enaltecendo as características e benefícios de sua utilização. Na segunda, são apresentados os conceitos que envolvem as Linguagens de Consultas para XML, bem como suas principais características e práticas, destacando ainda três representantes deste tipo de linguagem: XQL, XML-QL e XQuery. Na terceira e última parte deste capítulo, serão apresentadas as considerações em relação ao capítulo, apresentando os principais pontos relacionados a este tema.

2.2.1 Conceitos Básicos

A linguagem XML (eXtensible Markup Language) foi desenvolvida pelo XML Working Group formado sobre o patrocínio do World Wide Web Consortium (W3C) em 1996 e em fevereiro de 1998 tornou-se uma tecnologia recomendada pelo W3C [Bray et al., 2004]. Segundo a especificação, um objeto de dados é um documento XML se este está bem formatado e se cumpre certas restrições adicionais. Podemos dizer que um documento está bem formatado se:

- Observando-o como um todo, corresponde a um documento rotulado;
- Atende a todas as restrições de boa formatação presente na especificação;
- Cada uma das entidades referenciadas no documento esteja bem formatada.

Para Harold [1999], o formato XML pode ser conceituado como um conjunto de regras para a definição de marcadores semânticos, que dividem um documento em partes identificáveis. O autor destaca que ela é uma metalinguagem que define uma sintaxe para ser utilizada na criação de outras linguagens de marcação para um domínio específico, com estrutura e semântica próprias.

Como já citado no capítulo anterior, XML é uma metalinguagem, pois disponibiliza recursos para a definição de gramáticas que caracterizam linguagens para classes de documentos específicos, com conjunto de elementos, atributos e regras de composição bem determinados.

Em relação ao armazenamento de dados, vale ressaltar que documentos XML são, geralmente, armazenados em arquivos texto com a extensão .xml, embora isto não seja um requisito, além disso, qualquer editor de texto pode ser usado para criar um documento XML.

Um criador de um documento XML pode definir novas marcações para indicar a estrutura que melhor representa as informações que serão disponibilizadas. Por exemplo, a estrutura <musica> ... </musica> pode ser usada para descrever os dados referentes a uma música. A descrição destas informações é feita de maneira estruturada e hierárquica. A hierarquia de um documento XML é semelhante a uma árvore, ou seja, os documentos possuem um único elemento raiz que contém todos os outros elementos, chamados de elementos filhos [Deitel et al., 2003].

XML permite que os usuários definam a sua própria linguagem de marcação adaptada aos seus requisitos. O Código 1, mostra um exemplo de um documento XML.

Código Fonte 2.1: Trecho de código em XML

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <lista_de_musicas>
3    <musica> <artista>ROBERTO CARLOS</artista>
4    <titulo>AMIGO</titulo> </musica>
5    <musica> <artista>ERASMO CARLOS </artista>
6    <titulo>ALEM DO HORIZONTE</titulo> </musica>
7    <musica> <artista>LUIZ GONZAGA</artista>
8    <titulo>ASA BRANCA</titulo> </musica>
9    </lista_de_musicas>
```

A linha 1 do Código é obrigatória em todos os documentos, pois especifica a versão da XML que está sendo usada e a codificação dos caracteres [Bray et al., 2004]. Dois conceitos principais de estruturação que são usados para construir um documento XML são os elementos e atributos [Elmasri e Navathe, 2005], onde um elemento é um pedaço de texto intercalado pelos sinais “<” e “>”, criados pelo autor do documento, neste exemplo os elementos são lista_de_musicas, artista e titulo. As marcações, juntamente com o texto entre elas formam o elemento. O conteúdo de um elemento é delimitado pelo marcador de início (<nome>) e o marcador de fim (</nome>), sendo necessário que para cada marcador de início deve existir um marcador de fim com o mesmo nome.

O conteúdo de um elemento pode ser um texto simples e/ou outros elementos, por exemplo, os conteúdos dos elementos ‘titulo’ são textos simples, como “ASA BRANCA”. Já o conteúdo do elemento musica é formado pelos elementos titulo e artista e ainda, em alguns casos um elemento pode não possuir conteúdo.

Um elemento sem conteúdo pode ser representado por <musica> </musica> ou por <musica/>. Os elementos refletem uma estrutura particular associada a este contexto, e os marcadores permitem que qualquer pessoa interprete o seu conteúdo, pois é possível interpretar claramente que o conteúdo do marcador é o título de alguma coisa. Como os elementos artista e titulo estão aninhados dentro do elemento musica, se percebe que uma musica está sendo descrita, logo “ALEM DO HORIZONTE” é o título de uma música cujo artista que a toca é “ERASMO CARLOS”.

Em alguns casos, um marcador de início pode ter informações adicionais sobre um elemento, chamadas de atributos. Um elemento pode ter um ou vários atributos. Um atributo descreve características ou propriedades dos elementos e consiste de um nome, um sinal de igual e um valor entre aspas, estes atributos devem ser descritos no marcador de início.

Um software chamado de analisador sintático XML é requerido para processar um documento XML, e este utilitário permite a obtenção de elementos pela relação de pais e filhos [Rambhia, 2002]. Um analisador sintático XML lê um documento XML, verifica sua sintaxe, relata possíveis erros e permite acessar o conteúdo do documento [Deitel et al., 2003].

O analisador sintático XML envolve pelo menos três componentes: um documento XML (com as marcações), um processador XML (que divide o documento em "pedaços" de marcação e de dados de caracteres) e uma aplicação XML (que utiliza as informações enviadas pelo analisador sintático).

Um documento XML pode, opcionalmente, incluir outro documento, também chamado de esquema, que define uma gramática, a qual define regras para a estrutura, o conteúdo e a semântica de um documento XML Um documento XML é dito válido se ele segue uma gramática ou esquema. O uso de um documento esquema é opcional e existem dois tipos de esquemas XML: DTD e Esquema XML [Thompson et al., 2004].

Silva et al. [2008] ressalta como benefício da linguagem o fato dela ser baseada em texto, sendo possível assim, criar um documento XML apenas com uma ferramenta de texto simples. Além disso, ela não possui limitação para descrever somente dados de texto, podendo também descrever imagens, gráficos, animações ou qualquer outro tipo de dado.

Com todas estas facilidades, a XML está sendo cada vez mais utilizada, seja para a construção de arquivos de configuração, seja para o intercâmbio de dados entre aplicações na Web, ou estruturação e armazenamento de dados. Tal fato fez com que o volume de informações disponíveis nesse formato crescesse a um ritmo acelerado, despertando o interesse na pesquisa de estratégias de consultas eficientes.

Diante disto, a representação dos dados no formato XML abriu uma infinidade de possibilidades para sua manipulação. Dentre estas, a necessidade de análise destes dados XML e para tanto, diversas linguagens de consulta foram propostas, dentre as quais XML-QL, XQL e XQuery, como veremos na seção a seguir.

2.2.2 Linguagens de Consulta

Como citado anteriormente, com o grande aumento das informações armazenadas em XML, torna-se necessário algum mecanismo eficiente para fazer buscas sobre estas base de dados e com este objetivo foram surgindo linguagens de consultas próprias para este fim, dentre as quais foram citadas XML-QL, XQL e XQuery, como veremos a seguir.

2.2.2.1 XML - QL

A linguagem XML-QL [Deutsch et al, 1998] combina a sintaxe da linguagem XML com técnicas de linguagens de consulta a dados semiestruturados, tais como expressões de caminho. XML-QL é baseada em uma sintaxe do tipo *where/construct* ao invés da familiar *select/from/where* de SQL ou OQL. Nesta linguagem, a cláusula *construct* corresponde a *select*, enquanto que *where* combina as partes *from* e *where* da consulta, isto é, as faixas de variáveis, bem como alguma filtragem.

A sintaxe básica da linguagem combina, portanto, elementos da sintaxe de XML com elementos de sintaxe tradicionais de linguagens de consulta de sistemas de banco de dados. Sua forma geral é:

where <argumentos de seleção>
construct <resultado>

O trecho "*argumentos de seleção*" representa uma construção, no formato XML, com os dados que devem ser encontrados e variáveis (iniciadas por "*$*"). Nos argumentos da cláusula *where* deve aparecer, em pelo menos um deles, a palavra "*in*", indicando o documento que deve ser consultado. "*Resultado*" é a especificação do documento XML que deve ser construído como saída da consulta.

Um exemplo simples é apresentado no Código 2, onde a consulta recupera os títulos e álbuns das músicas lançadas pela banda ROBERTO CARLOS:

Código Fonte 2.2: Trecho de código em XML-QL

```
1    Where
2    <musica>
3    <artista><nome>ROBERTO CARLOS</nome></artista>
4    <titulo> $T </titulo>
5    <album> $A </album>
6    </musica> in "www.listademusicas.com/bib.xml"
7    construct <titulo> $T </titulo>
8    <album> $A </album>
```

Padrão

Neste exemplo, *$T* e *$A* são variáveis, enquanto a estrutura compreendida entre <musica>...</musica> é o padrão, e com isso o processador combina o padrão de todas as formas possíveis com os dados e instancia as variáveis $R e $R. Para cada instância, produz-se o resultado.

XML-QL também consegue realizar consultas aninhadas com partes opcionais, por exemplo, supondo a tag *<tempduracao>* em *<musica>* como opcional, e supondo ainda a necessidade de uma consulta de todos os títulos de musicas, quando disponíveis, seus respectivos tempos de duração.

Código Fonte 2.3: Trecho de código em XML-QL para consultas aninhadas – elementos opcionais

```
1    Where
2    $B in "www.listademusicas.com/bib.xml",
3    <titulo> $T </titulo> in $B
4    construct
5    <resultado>
6    <titulo_musica> $T </titulo_musica>
7    where <tempduracao> $D </tempduracao> in $B
8    construct <tempduracao_musica> $D
9    </tempduracao_musica >
          </resultado>
```

Outra necessidade que o XML-QL atende é quanto à necessidade de realizar consultas aninhadas através de agrupamento. Por exemplo, suponha a necessidade de se encontrar todos os artistas, e para cada um deles encontrar todos os títulos que publicaram. Este tipo de operação pode ser feito em XML-QL também através de consultas aninhadas, denominadas subconsultas, como pode ser visto no exemplo abaixo:

Código Fonte 2.4: Trecho de código em XML-QL para consultas aninhadas – agrupamento

```
1    Where
2    <musica><artista> $A </artista></musica>
3    in "www.listademusicas.com/bib.xml",
4    construct
5    <resultado>
6    <artista> $A </artista>
7    where
8    <musica>
9    <artista> $A </artista>
10   <titulo> $T </titulo>
11
12
13   </musica> in "www.listademusicas.com/bib.xml",
     construct <titulo> $T </titulo>
```

```
    </resultado>
```

Outro ponto a ser destacado da linguagem XML-QL é em relação às variáveis que são instanciadas em nós no modelo de dados semiestruturado. Em termos de XML, isto significa que variáveis são instanciadas no conteúdo do elemento e não no elemento em si. A XML-QL possui uma simplificação sintática que permite a instanciação no elemento propriamente dito.

A consulta em atributos em XML-QL é bastante simples, pois se realiza de forma direta, de modo que o valor de um atributo da entrada se torna o valor de um elemento na saída.

Em relação às junções na linguagem de consulta XML-QL, estas podem ser expressas através da utilização de uma mesma variável em duas combinações. Vamos exemplificar através da necessidade de uma consulta que recupere todos os artistas que publicaram pelo menos dois álbuns:

Código Fonte 2.5: Código XML-QL para junções

```
1    Where
2    <album><artista> $A </artista></album>
3    content_as $B1 in "www.listademusicas.com/bib.xml",
4    <album><artista> $A </artista></album>
5    content_as $B2 in "www.listademusicas.com/bib.xml",
6    B1 != B2
7          construct <resultado> $A </resultado>
```

Como visto anteriormente, esta linguagem de consulta a documentos XML é capaz de realizar as principais tarefas para este fim, tais como: extração de dados de documentos extensos e integração entre dados XML de múltiplas fontes, através da realização de junções e outras operações encontradas em SQL. Todavia, algumas características não foram incluídas nesta linguagem, tais como o suporte a agregados e *semijoins*, entre outras.

2.2.2.2 XQL

A XQL (ROBIE, 1999) é uma linguagem projetada especialmente para XML, fornecendo uma descrição bastante compreensível dos elementos, bem como uma maneira direta de consultá-los. Para navegar na hierarquia de elementos dos documentos, ela utiliza uma notação de caminho semelhante à das URL (Uniform Resource Locator).

As construções básicas de XQL correspondem diretamente às estruturas básicas de XML, onde a consulta é definida sobre um conjunto de nodos de um ou mais documentos e o resultado de uma consulta XQL pode ser um elemento, uma lista deles ou um documento XML completo.

Ainda em relação ao resultado de uma consulta é o conjunto de nodos definidos por um documento XML, sendo que este conjunto pode ser encapsulado por um nodo raiz a fim de que seja definido um documento XML bem estruturado.

Com o exemplo a seguir é possível apresentar estes conceitos de forma concreta, determinando uma consulta XQL simples, para analisar a entrada da consulta, a consulta em si e o seu resultado. No Código Fonte 2.6, a entrada da consulta (conhecida como *contexto da pesquisa*) é um elemento simples <LM>, o qual é a raiz do documento.

Código Fonte 2.6: Trecho de código XML para consultar em XQL

```
1     <LM>
2     <musica> <artista>ROBERTO CARLOS</artista>
3     <titulo>AMIGO</titulo> </musica>
4     <musica> <artista>ERASMO CARLOS </artista>
5     <titulo>ALEM DO HORIZONTE</titulo> </musica>
6     <musica> <artista>LUIZ GONZAGA</artista>
7     <titulo>ASA BRANCA</titulo> </musica>
8     </LM>
```

A consulta mais simples possível na linguagem XQL é uma string que representa o nome de um elemento. Por exemplo, *"LM"* é uma consulta XQL válida e completa para o documento de entrada XML acima descrito. Desta forma a consulta descrita simplesmente pelo comando *"DRF"* seleciona todos os elementos *LM* do contexto de pesquisa, retornando como resultado para o exemplo acima o próprio contexto de pesquisa.

No exemplo do subitem anterior o conjunto resultado da consulta continha apenas um único nodo raiz. No entanto, uma consulta pode retornar mais do que um nodo raiz, o que implicaria numa representação textual sem a característica de ser bem formado para uma representação XML, já que um documento XML deve ter apenas um único elemento raiz. Supondo que o conjunto resultado para uma dada consulta fosse como mostra o Código Fonte 2.7:

Código Fonte 2,7: Trecho de código com conjunto resultado em XQL

```
1      <artista>ROBERTO CARLOS</artista>
```

```
2     <titulo>AMIGO</titulo>
```

Como este resultado contém dois nodos raízes, ele não se constitui como um documento XML válido. No entanto, se os nodos deste resultado forem encapsulados por um elemento raiz comum, o documento XML resultante torna-se válido. Portanto, o documento resultante de uma consulta XQL sempre encapsula os nodos do conjunto resultado em um elemento <XQL:RESULTADO> , como mostra o Código Fonte 2.8.

Código Fonte 2.8: Trecho de código com documento resultante em XQL

```
1     <XQL:RESULTADO>
2      <artista>ROBERTO CARLOS</artista>
3     <titulo>AMIGO</titulo>
4     </XQL:RESULTADO>
```

Como visto anteriormente, uma string simples com o nome de um elemento contido no documento XML que está sendo consultado pode ser considerada como uma consulta XQL completa. A fim de permitir navegabilidade à consulta pelo documento XML que está sendo consultado, a linguagem XQL utiliza o operador "/" para indicar hierarquia de elementos. Por exemplo ‘MUSICA/TITULO’**,** onde esta consulta descrita seleciona todos os elementos filhos <TITULO> do seu elemento pai <MUSICA>. O elemento raiz deve ser precedido pelo operador "/", por exemplo, ‘/DRF/JURISDICAO’.

Na linguagem XQL, a raiz de um documento é diferente do elemento raiz, pois a raiz de um documento se refere à entidade documento, o que é basicamente equivalente ao documento em si. Já o elemento raiz é o elemento que contém o resto dos elementos do documento. A raiz de um documento sempre contém o elemento raiz, mas pode conter também informações relativas ao tipo de documento, instruções de processamento e comentários.

Como a XQL tem por base expressões de caminho, o caminho sempre é descrito do elemento mais externo para o mais interno do documento, ou seja, da raiz para o interior, e a menos que seja especificado, o elemento mais à direita deste caminho é retornado como resultado da consulta. Por exemplo, com a consulta ‘/LM/MUSICA/ARTISTA’ recebe-se como resultado todos os valores contidos no elemento <*ARTISTA*>.

O conteúdo de um elemento ou o valor de um atributo pode ser especificado na consulta utilizando-se o operador de igualdade (“=”). Por exemplo, com a consulta

‘/LM/MUSICA/ARTISTA= ‘ROBERTO CARLOS’’ serão retornados todos os elementos <ARTISTA> com valor igual a “ROBERTO CARLOS”.

Os atributos são tratados como filhos dos elementos que os contêm e os seus nomes devem ser iniciados com o caractere especial “@”. Por exemplo, considerando o atributo *nome* do elemento *LOCAL*, conforme ilustrado abaixo:

Código Fonte 2.9: Trecho de código em XML para consulta em XQL

```
1     <LM>
2     <artista nome = "ROBERTO CARLOS">
3     <titulo>AMIGO</titulo>
4     </artista>
5     </LM>
```

Uma consulta para selecionar os nomes de todos os artistas da LM que sejam iguais a Roberto Carlos poderia ser expressa como segue: /LM/ARTISTA/@nome= ‘ROBERTO CARLOS’.

O operador de descendentes (“//”) indica qualquer número de níveis (elementos) intermediários, ou caso apareça no começo da consulta indica todos os nodos anteriores ao elemento referenciado, por exemplo, ‘/LM//TITULO’.

Quanto aos operadores de retorno e de sequência da linguagem, são básicos para o modelo XQL completo, porém não se fazem necessários para todas as implementações XQL. Operadores de retorno são análogos à estrutura *SELECT* da SQL, permitindo melhor controle sobre o que é retornado no resultado de uma consulta. No entanto, estes operadores não são necessários para todas as aplicações, já que muitas aplicações retornam nodos simples em consultas ou possuem requisitos muitos simples a serem retornados.

Operadores de sequência permitem determinar a ordem em que os dados aparecerão num documento a ser utilizado em condições de consulta, sendo extremamente útil em muitos tipos de documentos.

Os relacionamentos básicos entre os elementos são: Hierarquia (pai/filho, ancestral, descendente); Sequência (antecessor imediato, antecessor) e Posição (absoluta, relativa, intervalo). As condições para nodos e as condições para os relacionamentos entre os elementos são combinadas a fim de formar uma consulta de caminhos (expressão de caminho) dentro do contexto de pesquisa. Operadores de retorno são usados para selecionarem nodos específicos das expressões de caminho, estes nodos são então retornados pela consulta.

Basicamente existem dois tipos de operadores de retorno: Retorno de superfície (“?”) e Retorno de profundidade (“??”). Quando o operador de retorno de superfície é aplicado sobre um elemento, ele não retorna nem os atributos deste elemento e nem seus filhos. Já o operador de retorno de profundidade retorna o nodo e todos os seus filhos. Os operadores de retorno podem simplificar consultas para documentos com estruturas mais complexas.

Supondo que a consulta pretenda retornar todos os títulos de um artista, é possível realizá-la com a expressão ‘ARTISTA//TITULO’, mas dessa forma o resultado não mostra quais músicas são do mesmo artista. No entanto, isto pode ser facilmente consultado com a utilização dos operadores de retorno, refazendo a consulta da forma ‘ARTISTA?//NOME??’. E supondo ainda a necessidade de apresentar o nome do artista juntamente com suas músicas, é possível reformular a consulta para ‘ARTISTA?[NOME??]//NOME?’.

Destaca-se que consultas simples, baseadas em filtragens pouco complexas e agrupamentos, são facilmente expressas. Contudo, a XQL dificulta o processo de expressar consultas mais complexas (com aninhamento, por exemplo). O implemento de joins pode ser saudado como uma interessante inovação, porém é obscurecido pela dificuldade de expressão.

2.2.2.3 XQuery

A linguagem XQuery [W3C, 2010] foi criada para atender aos requisitos identificados pelo W3C XML Query Working Group. Sendo assim, foi projetada para ser uma linguagem pequena e facilmente implementável, na qual as consultas são facilmente compreendidas. A XQuery é flexível o bastante para realizar consultas em uma grande variedade de fontes de informação XML, incluindo tanto bancos de dados como documentos [Walmsley, 2007].

XQuery fornece recursos flexíveis de consulta para extrair dados de documentos já existentes e gerados dinamicamente na Web. O objetivo do XML *Query Working Group* era produzir um modelo de dados para documentos XML, um conjunto de operadores de consulta para esse modelo de dados e uma linguagem de consulta com base nesses operadores [Graves, 2003].

A XQuery é uma linguagem funcional na qual uma consulta é representada por uma expressão, e sua sintaxe suporta muitos tipos de expressões, de modo que as várias formas de expressões XQuery podem ser aninhadas e combinadas de uma forma extremamente completa.

O modelo de dados XQuery fornece a entrada e saída de um processador de consultas XQuery. Uma instância do modelo de dados XQuery pode consistir de três itens principais: nodos, valores simples e sequências. Um nodo pode ser um dos seguintes tipos: Documento, Elemento, Atributo, Namespace3, Comentário, Instrução de Processamento, Texto ou Referência.

O modelo trata documentos XML e fragmentos do documento como uma árvore de nodos. Um documento é uma árvore com o nodo documento sendo a raiz da árvore. Similarmente, um fragmento de documento é uma árvore com um elemento na sua raiz. Nodos do tipo *documento* e *namespace* não possuem pais. Todos os outros nodos podem ter zero ou um pai. Apenas um nodo *documento* ou um nodo *elemento* podem ter filhos.

A sequência da informação para o modelo de dados da XQuery pode ser imaginada como uma lista linear de nodos e/ou valores, o que significa dizer que uma sequência não pode ter uma outra sequência como seu membro. Uma característica importante desta sequência é que o modelo de dados não faz distinção entre um nodo (ou valor) simples e uma sequência contendo este nodo (ou valor) simples.

A linguagem XQuery (*XML Query Language*) é uma tentativa da W3C de disponibilizar uma linguagem de consulta que fornece a mesma funcionalidade que a linguagem SQL apresentada em bancos de dados relacionais, e com isso possui algumas similaridades muito perceptíveis com as linguagens SQL e OQL. O termo módulo XQuery significa uma unidade de consulta. Basicamente um módulo XQuery é composto de três partes: declarações *namespace* e *Schema* (opcional), definição de funções (opcional) e expressões de consulta.

A *XQuery* permite a especificação de consultas mais genéricas em um ou mais documentos XML e para melhor compreender as características da linguagem XQuery, está ilustrado abaixo, através do Código Fonte 2.10, um fragmento de um documento usado nos casos de uso apresentado pelo *XQuery Working Group* [W3C, 2010]:

Código Fonte 2.10: Trecho de código com documento XML

```
1     <bib>
2     <book year="1994">
3     <title>TCP/IP Illustrated</title>
4     <author>
5     <last>Stevens</last>
6     <first>W.</first>
```

```
7     </author>
8     <publisher>Addison-Wesley</publisher>
9     <price> 65.95</price>
10    </book>
11    <book year="2000">
12    <title>Data on the WEB</title>
13    <author>
14    <last>Abiteboul</last>
15    <first>Serge</first>
16    </author>
17    <author>
18    <last>Buneman</last>
19    <first>Peter</first>
20    </author>
21    <author>
22    <last>Suciu</last>
23    <first>Dan</first>
24    </author>
25    <publisher>Morgan Kaufmann Publishers</publisher>
26    <price> 39.95</price>
27    </book>
28    </bib>
```

Fonte: World Wide Web Consortium. XML Query Requirements: W3C Working Draft. 14 December 2010

Considerando que o fragmento acima seja um documento denominado "*books.xml*" e, considerando ainda a necessidade de uma consulta para selecionar todos os livros publicados pela editora *Addison-Wesley* após o ano de *2000*, incluindo no resultado da consulta o *ano* e o *título* do livro. Na linguagem XQuery, tal consulta seria expressa como mostra o código 2.11:

Código Fonte 2.11: Trecho de Consulta na linguagem XQuery

```
1     <bib>
2     {
3     FOR $b IN document("books.xml")/bib/book
4     WHERE $b/publisher = "Addison-Wesley" AND $b/@year >
5     2000
6     RETURN
7     <book year={ $b/@year }>
8     { $b/title }
9     </book>
10    }
11    </bib>
12    O resultado gerado pela consulta seria:
13    <bib>
14    <book year="2000">
15    <title> Data on the WEB </title>
16    </book>
17    </bib>
```

Fonte: World Wide Web Consortium. XML Query Requirements: W3C Working Draft. 14 December 2010

Esta consulta utiliza algumas expressões XQuery simples, dentre elas as expressões de caminho, construtores de elemento e de atributo e as expressões típicas de um consulta XQuery, que é conhecida como FLWR (*FOR-LET-WHERE-RETURN*).

A expressão FLWR fornece uma sintaxe semelhante à sintaxe SQL para extração de dados, e representa as quatro cláusulas principais do *XQuery*, que são divididas em cláusulas *FOR, LET, WHERE e RETURN*, sendo que a cláusula *FOR* são as variáveis ligadas a nodos individuais; a cláusula *LET* são variáveis ligadas a coleção de nodos (elementos); a cláusula *WHERE* são as condições qualificadoras e a cláusula *RETURN* é a especificação do resultado da consulta [Santos, 2007].

Quanto às expressões de caminho da linguagem XQuery, estas são baseadas nas expressões de caminho da linguagem XPath. 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. Por exemplo, considerando o exemplo anterior, para fazer referência a todos os elementos book, os quais são filhos do elemento raiz bib do documento books.xml, basta utilizar a seguinte expressão:‘document("books.xml")/bib/book’.

O resultado da consulta ilustrada acima possui dois elementos: <bib> e <book>. Para construir estes elementos, os elementos <bib>...</bib> e <book>...</book> foram escritos diretamente no corpo da própria consulta. As chaves "{ }" são usadas para separar o conteúdo literal de qualquer subexpressão dentro de um elemento. As subexpressões incluídas nas chaves são analisadas pelo processador de consulta XQuery. De forma similar, os construtores de atributo são especificados pela sua inclusão junto à subexpressão contida dentro das chaves, como ilustrado abaixo: <book year={ $b/@year }>.

Alguns mecanismos análogos à SQL são também fornecidos pela linguagem XQuery a fim de facilitar algumas operações de estruturação dos dados nas consultas. Um destes mecanismos é fornecido pela cláusula SORTBY, a qual é usada para determinar uma ordem sobre a sequência.

Declarações namespace são usadas para declarar endereços de namespaces a serem usados num módulo de consulta XQuery. Ela pode ser usada também para declarar um namespace padrão para um módulo XQuery. Por exemplo, a seguinte consulta retorna todos os elementos <book> no namespace identificado pela URI “http://www.bibliophile.com”:

Namespace booklovers="http://www.bibliophile.com"

document("books.xml")//booklovers:book

A linguagem XQuery possui um conjunto central de biblioteca de funções definida na sua especificação Funções e Operadores. A função document é um exemplo destas funções. O exemplo abaixo demonstra a sintaxe para definição de funções.

Código Fonte 2.12: - Trecho de código criando uma função na linguagem XQuery

```
1    Namespace xsd = "http://www.w3.org/2001/XMLSchema"
2    Namespace booklovers="http://www.bibliophile.com"
3    Schema "http://www.bibliophile.com"
4    "http://bibliophile.com/books.xsd"
5    Define Function getBooksByTitle(xsd:string $title)
6    RETURNS book_seq {
7    RETURN Document("books.xml")/bib/books[title=$title]
8    }
```

Fonte: World Wide Web Consortium. XML Query Requirements: W3C Working Draft. 14 December 2010.

Este exemplo define uma função que pega uma string como argumento e retorna uma sequência de nodos <book> que possuem um determinado título (title = $title). O tipo retornado é o book_seq, o qual deve ter sido previamente definido no esquema books.xsd incluso na cláusula Schema.

Além desta biblioteca central, a XQuery permite que os usuários definam suas próprias funções, com isso a XQuery implanta o conceito de funções extensíveis que atuam sobre documentos, seções ou valores atômicos. A vantagem na definição de funções na XQuery concentra-se no fato que as funções são implementadas na própria linguagem, não sendo necessário especificar funções numa linguagem de programação à parte.

A linguagem possui suporte ainda para expressões condicionais, que são análogas ao IF-THEN-ELSE das linguagens de programação e são uteis quando a estrutura da informação a ser retornada depende de alguma condição, podendo ser aninhadas e utilizadas em qualquer lugar onde um valor é esperado. De modo que, se o teste for verdadeiro, o valor da primeira expressão é retornado. Caso contrário, o resultado da segunda expressão é retornado.

Destaca-se ainda na linguagem XQuery o grande poder de expressão de consultas, por unir os pontos destacados de outras linguagens numa especificação clara e funcional. A flexibilidade proporcionada para a manipulação dos resultados é algo pouco visto nas demais linguagens, bem como a boa integração com a linguagem XPath. A XQuery ainda recebe

complementações e ajustes, entretanto tende a se tornar a principal linguagem de consulta para dados XML.

2.2.3 Considerações Finais

Como visto, XML provê uma representação estruturada dos dados e dentre tantas vantagens que possui, destaca-se o fato de ser autodescritivo, flexível, simples e escalável. Mas apesar da simplicidade, permite criar estruturas bastante complexas e é extensível, possibilitando criar sua própria sintaxe de dados através da criação ilimitada de tags.

Diante da necessidade de análise destes dados XML, diversas linguagens de consulta foram propostas, dentre as quais XML-QL, XQL e XQuery. Observando estas linguagens perceberam-se algumas semelhanças, porém comparando diretamente as funcionalidades, a XQuery apresenta-se como a linguagem de consulta XML mais completa em termos de funcionalidades que abrange e para melhor observar este comparativo, o Quadro 2.1 [Comai, 2001] exibe uma amostra destas diferenças.

Quadro 2.0.1: Comparativo entre linguagens de consulta XML [Adaptado de Comai, 2001].

Funcionalidade	**XML – QL**	**XQL**	**XQuery**
Junção	S	S	S
Documentos múltiplos	S	S	S
Quantificação, negação redução	P	S	S
Modificação de estrutura	S	N	S
Novos elementos/relacionamentos	S	P	S
União, diferença, produto cartesiano	P	S	S
Agrupamento	S	S	S
Funções de agregação	S	P	S
Funções aritméticas	N	N	S
Subconsultas aninhadas	S	P	S
Ordenamento do resultado	S	N	S
Funções Extensíveis	N	N	S
Expressões Condicionais	N	N	S

Onde:

- S = Sim, funcionalidade existente;
- N = Não, funcionalidade não existente;
- P = Parcial, funcionalidade existe parcialmente, pois apesar de não haver o comando direto, a combinação de outros realiza a funcionalidade.

Nesta seção foram explanados os principais conceitos e aplicações do XML, bem como características e funcionalidades de três de suas linguagens de consulta e diante dos

aspectos apresentados sobre as linguagens supracitadas foi escolhida a linguagem XQuery como objeto de estudo deste trabalho, por esta ser a mais completa, além de ser a sugerida atualmente pelo W3C para consultas a dados XML.

Com todas estas vantagens destacadas, o XML vem ganhando cada vez mais espaço em diversas aplicações, principalmente na divulgação de dados pela web e um exemplo recente são os Dados Abertos Governamentais que constituem a publicação e disseminação de dados e informações públicas na Web, respeitando critérios que possibilitam sua reutilização e o desenvolvimento de aplicativos por toda a sociedade. No entanto, estes domínios possuem grande quantidade de informação e necessitam de análise profunda para realmente cumprir seu papel, mais detalhes serão explanados na próxima seção.

2.3 Dados Abertos

Esta seção está dividida em duas partes. Na primeira são apresentados conceitos relacionados a dados abertos, destacando os seus princípios e principalmente, benefícios de sua utilização por parte da população. E na segunda parte serão realizadas as considerações finais sobre o tema, relacionando com o presente trabalho.

2.3.1 Conceitos

Com a redemocratização brasileira, a população tem exigido mais transparência e participação nas decisões de governo, e em resposta a essa demanda foi criado um elo entre governo e sociedade através da Lei de Acesso à Informação Brasileira [Brasil, 2011] que entrou em vigor no dia 16 de maio de 2012, com o objetivo de garantir o acesso dos cidadãos à informação, tornando obrigatórios o registro e a publicação dos dados governamentais.

A Open Knowledge Foundation destaca a definição que dados são abertos quando qualquer pessoa pode livremente usá-los, reutilizá-los e redistribuí-los, estando sujeito, no máximo, à exigência de creditar a sua autoria e compartilhar pela mesma licença, o que geralmente é satisfeito pela publicação dos dados em formato aberto e sob uma licença aberta.

Dados abertos podem ser considerados tanto no setor público quanto no privado. Este trabalho, por uma questão de foco, trata somente de dados governamentais abertos. De acordo com o W3C, dados abertos governamentais são a publicação e a disseminação de informações na internet, compartilhadas em formatos abertos, legíveis por máquinas e que possam ser

livremente reutilizadas de forma automatizada pela sociedade. Estes dados são produzidos pelos governos e devem ser colocados à disposição de qualquer cidadão e para qualquer fim.

De acordo com Eaves [2009], as três leis dos dados abertos governamentais são:

1. Se o dado não pode ser encontrado e indexado na Web, ele não existe;
2. Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado;
3. Se algum dispositivo legal não permitir sua reaplicação, ele não é útil.

Para atender a essas leis, os dados devem ser disponibilizados sem nenhuma restrição quanto ao seu uso e de forma padronizada na internet, seguindo oito princípios dos dados abertos governamentais estabelecidos pelo *OpenGovData* [2007]:

1. Completos, pois todos os dados públicos estão disponíveis.
2. Primários, ou seja, devem ser apresentados tais como os coletados na fonte, com o maior nível possível de granularidade e sem agregação ou modificação.
3. Atuais, os dados são disponibilizados tão rapidamente quanto necessário à preservação do seu valor.
4. Acessíveis, pois devem estar disponibilizados para o maior alcance possível de usuários e para o maior conjunto possível de finalidades.
5. Compreensíveis por máquinas, uma vez que é importante que os dados sejam razoavelmente estruturados de modo a possibilitar processamento automatizado.
6. Não discriminatórios, estando disponíveis para todos, sem exigência de requerimento ou cadastro.
7. Não proprietários, pois os dados são disponíveis em formato sobre o qual nenhuma entidade detenha controle exclusivo.
8. Livres de licenças, ou seja, não estão sujeitos a nenhuma restrição de direito autoral, patente, propriedade intelectual ou segredo industrial. Restrições sensatas relacionadas à privacidade, segurança e privilégios de acesso são permitidas.

Disponibilizar dados abertos governamentais tem a potencialidade de prover mais transparência ao distribuir dados que podem ser reutilizados por qualquer cidadão livremente permitindo uma visão mais ampla das ações de governo. E se os dados governamentais abertos forem analisados comparando-os com dados de outras fontes, será possível obter uma nova visão sobre o desempenho do governo, o que demanda maior responsabilidade dos agentes públicos [Diniz, 2010].

Outro benefício da adoção dos dados abertos governamentais trata-se da possibilidade de criação de aplicativos e, assim, novos entendimentos a partir dos dados governamentais abertos. Nesse contexto, além de haver a promoção da transparência, novos serviços podem ser motivados como fruto da interação entre o governo e sociedade através da exploração desses dados.

Para tanto, a Cartilha Técnica para Publicação de Dados Abertos no Brasil [2011] recomenda que as informações estejam disponíveis em formato passível a modificações e que o acesso às informações não dependa da aquisição de um software proprietário, e propõe os seguintes formatos:

- JSON, é um acrônimo para JavaScript Object Notation que é muito fácil de ler em qualquer linguagem de programação.
- XML, um formato amplamente utilizado para intercâmbio de dados porque proporciona boas oportunidades para guardar a estrutura nos dados e na maneira pela qual os arquivos são construídos, além de permitir que desenvolvedores escrevam parte da documentação juntamente com os dados sem interferir na sua leitura.
- RDF, modelo de dados estruturado em grafos e possui diversos formatos de serialização, tais como RDF/XML,. Dados RDF podem ser armazenados em XML e JSON, além de outras serializações.
- CSV, *Comma-Separated Values*, ou valores separados por vírgula. É compacto e, portanto, adequado para transferir grandes conjuntos de dados com a mesma estrutura. Entretanto, para o formato é particularmente importante para os arquivos CSV que a documentação de cada campo seja precisa.
- ODF, *Open Document Spreadsheet*, é um formato não proprietário de arquivo baseado em XML, padronizado pela ABNT sob a norma NBR ISO/IEC 26300:2006. É comumente chamado de planilha, similar ao XLS do MS Office Excel, porém aberto, e, por isso deve ser utilizado em substituição ao XLS.

Dentre os formatos recomendados pela Cartilha Técnica para Publicação de Dados Abertos no Brasil, este trabalho tem como foco apenas o XML, que é uma metalinguagem, pois dispõe de recursos para a definição de gramáticas que caracterizam linguagens para classes de documentos específicos, com conjunto de elementos, atributos e regras de composição bem determinados.

Duas características importantes de XML são a independência de dados a separação entre conteúdo e apresentação. Com isso, a abertura dos dados nesse formato permite que qualquer interessado possa, ao processar livremente os dados governamentais, criar conteúdo a partir da reutilização dos dados [Vaz et al., 2010].

O foco no cidadão no conceito de dados governamentais abertos é destacado por Silva [2010], quando este ressalta que com a inteligência coletiva é possível criar formas melhores de trabalhar com os dados do que os próprios governos poderiam fazer.

Neste sentido, os governos devem incentivar a sociedade a utilizar os dados abertos disponíveis pelos governos, principalmente porque não existe relevância na disponibilização destes dados se o cidadão não tem interesse em reutilizá-los [Diniz, 2010]. Entretanto, é importante frisar que nem sempre o interesse em obter e reutilizar os dados é suficiente para utilização de todo o poder das informações que neles constam, devido à complexidade em realizar consultas em grandes quantidades de dados. O que recai no foco deste trabalho, que visa reduzir os esforços do usuário na busca por conteúdo relevante de acordo com suas preferências.

2.3.2 Considerações Finais

Os benefícios da utilização dos dados abertos governamentais pela e para a sociedade parecem ser óbvios. No entanto, este trabalho os destaca não apenas com o intuito de motivar sua utilização, mas para justificar pesquisas da área de tecnologia envolvendo este tema, principalmente quando se trata do desenvolvimento de técnicas que facilitem o acesso a estes dados.

Como citado anteriormente, dentre os formatos apresentados na seção anterior como recomendados pela Cartilha Técnica para Publicação de Dados Abertos no Brasil, este trabalho tem como foco apenas o XML, que como será mostrado no próximo Capítulo, além de prover uma representação estruturada dos dados é bastante flexível, extensível e autodescritiva, o que facilita sua manipulação.

Esta abordagem será utilizada como estudo de caso para avaliar a extensão, com suporte a preferências condicionais, proposta para uma linguagem de consultas a documentos XML, com o objetivo de mensurar se os resultados das consultas respeitam as preferências condicionais impostas nos testes.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos relacionados com esta pesquisa, enfatizando pontos em comum e principalmente enfatizando as diferenças entre as abordagens. Para melhor compreensão o capítulo foi dividido em três seções, de modo que cada área de conhecimento abordado na pesquisa fosse contemplada com uma seção. Assim, a primeira seção serão discutidos os trabalhos relacionados com a pesquisa que tratam de personalização de consultas e mineração de preferências, na segunda destacam os trabalhos relacionados com a extensão de linguagens de consulta. E por fim, na terceira seção são abordados os trabalhos da área Dados Abertos Governamentais.

3.1 Personalização De Consultas

A personalização de consultas tem como objetivo fornecer soluções que permitam filtrar o que é mais relevante, ou o que corresponda à expectativa e interesse de cada usuário. Para tanto, técnicas de personalização vêm sendo pesquisadas e esta área de estudo vem crescendo e aplicando seus conceitos em diversas áreas da computação, com o simples objetivo de dinamizar as consultas e produzir respostas personalizadas.

Como principais fundamentos desta área de conhecimento, diversos trabalhos foram tomados como base como Wilson [2004], Boutilier et al. [2004] e tantos outros que desenvolvem muito bem os conceitos relacionados à personalização de consultas e modelagem de preferências. No entanto, os trabalhos relacionados que merecem ressalvas são as pesquisas de Koutrika [2010] e Yan et al. [2011], pois compactuam de objetivos semelhantes tendo assim mais pontos para serem discutidos e mais urgência em diferenciar suas contribuições.

O trabalho de Koutrika [2010] estuda a personalização de consultas em banco de dados, com a finalidade de gerar um resultado personalizado e específico para cada usuário.

Para tanto, o autor apresenta um modelo de preferência que combina a expressividade e concisão, onde as preferências do usuário são armazenadas como graus de interesse em elementos de consulta atômicas, que pode ser utilizado para transformar uma consulta. No trabalho é proposto um framework de personalização de acordo com o qual os resultados personalizados devem satisfazer. Este autor, já possui diversas publicações na área de personalização de consultas, onde foca o aproveitamento das preferências do usuário na realização de consultas em bancos de dados relacionais.

A principal diferença entre o trabalho de Koutrika [2010] e a presente pesquisa se enquadra inicialmente no foco dos trabalhos, pois como mencionado o foco do autor são bancos de dados relacionais, enquanto o XQPref visa personalizar as consultas a documentos XML. No entanto, destaca-se ainda o fato de que o presente trabalho se concentra na modelagem das preferências condicionais, levando em conta que existem elementos que sua preferência influencia na preferência de outros.

O trabalho de Yan et al. [2011] porém, também possui seu foco na utilização de preferências do usuário ao consultar documentos XML. Esta pesquisa propõe um método de consulta baseado em preferência contextual em documento XML com o objetivo de resolver o problema de respostas vazias ao realizar consultas. Para tal, ele propõe o relaxamento e a pontuação de conteúdo. O modelo proposto é chamado de XCP e tem como objetivo permitir aos usuários expressar seus interesses sobre nós da árvore XML, e então o usuário atribuir pontuações para seus nós interessantes para fornecer as melhores respostas.

Comparando a pesquisa de Yan et al. [2011] ao presente trabalho, destaca-se que o autor não utiliza uma linguagem de consulta para XML em sua proposta, de modo que o usuário fica preso ao modelo e não consegue realizar buscas envolvendo artifícios disponíveis nestas linguagens (como agrupamento, consultas aninhadas, dentre outros). Outro ponto a ser destacado é que no presente trabalho o processo de elicitação das preferências ocorre de forma bastante simples para o usuário, pois o mesmo necessita apenas selecionar os elementos que lhe são prediletos, enquanto no modelo de Yan et al. [2011] um usuário leigo pode ter dificuldade ao se deparar com uma árvore para selecionar seus nodos prediletos.

3.2 Extensão De Linguagens De Consultas

Gomes [2002] propôs uma extensão a linguagem de consulta XQuery visando proporcionar recursos para a recuperação de informações temporais e de versão representadas

em documentos XML. Para tanto, são definidas funções temporais e versionadas que são incorporadas à linguagem base.

A importância da dissertação de Gomes [2002] na fundamentação desta pesquisa se deu pela forma em que foi realizada a extensão da linguagem de consulta para documento XML, pois além desta extensão ser implementada na mesma linguagem que é objeto de estudo desta proposta, ainda se assemelha na forma como foi realizada a extensão, ou seja, ambas utilizaram a vantagem das funções extensíveis da linguagem XQuery para realizar o complemento da linguagem.

Ribeiro [2008] e Pereira [2011] são trabalhos complementares que propuseram a CPref-SQL. No primeiro, foi especificada uma extensão a SQL padrão por meio de operadores de seleção de tuplas ótimas Best-E, Best-N e Best-D que selecionam as tuplas de uma relação considerando um conjunto de preferências condicionais especificadas por um usuário. Neste trabalho são propostos algoritmos para cada um dos operadores de seleção de tuplas ótimas e implementado um protótipo para um fragmento da linguagem CPref-SQL.

Em Pereira [2011], foi proposta a implementação da linguagem CPref-SQL, onde o modelo de preferência da CPref-SQL foi tornado mais expressivo e desenvolveram algoritmos que implementam os novos operados de preferência Select-Best e SelectK-Best. Esses operadores são capazes de avaliar consultas top-K com preferências, ou seja, consultas que retornam as K tuplas mais preferidas de acordo com a hierarquia de preferências do usuário. Por fim, implementaram a linguagem no *core* do processador de consultas do Postgre-SQL.

Muitas semelhanças regem a especificação das extensões CPref-SQL e XQuery-Pref, uma vez que ambas permitem expressar preferências condicionais sob a abordagem qualitativa para possibilitar a modelagem das preferências condicionais e se baseiam no formalismo de teorias de preferências condicionais do trabalho de Wilson [2004].

Entre as diferenças nestas extensões podem ser ressaltadas o foco e a forma como foram realizadas, pois a CPref-SQL tem como foco as consulta a banco de dados relacionais e foi implementada através da definição de novos operadores na linguagem base. A XQuery-Pref possui como foco a consulta personalizada em documentos XML, mais especificamente em dados abertos governamentais e esta extensão foi realizada através da definição de funções extensíveis, permitidas na própria XQuery. Outro fator de destaque, é que esta pesquisa foi

mais além ao propor seu próprio sistema (XQPref) onde a elicitação de preferência é viabilizada.

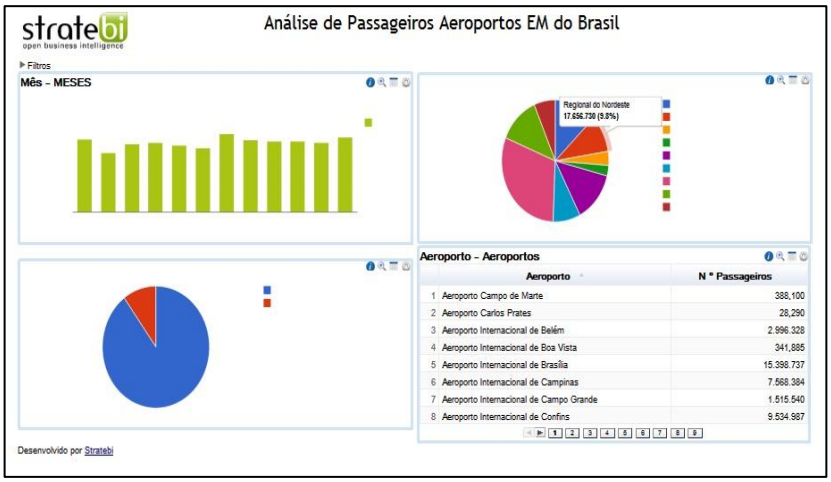
3.3 Dados Abertos Governamentais

A inclusão digital aliada à informatização dos procedimentos governamentais e a integração entre os diversos repositórios de dados públicos provocam crescentes demandas da população por mais transparência e participação através de meios tecnológicos. Nessa direção, o governo brasileiro tem definido políticas e desenvolvido plataformas tecnológicas na intenção de promover a disseminação das informações públicas.

Através do site http://dados.gov.br/aplicativos/ é possível ter acesso a essas ferramentas. Porém, cada uma dessas ferramentas utiliza um conjunto específico de dados abertos governamentais no seu desenvolvimento de modo que, cada ferramenta possibilita o acesso a estes dados especificamente.

A exemplo destas ferramentas, o Aeroportos Brasil [2012] é um aplicativo que mostra o movimento de aeronaves e passageiros nos aeroportos administrados pela Infraero em 2011. Em relação a aeronaves, estão computados pousos e decolagens e aos passageiros estão vinculados embarques e desembarques. Para tanto, são utilizados os dados sobre o Movimento dos Aeroportos Administrados pela Infraero. O aplicativo está disponível em: http://ison.stratebi.es/aerobrasil/ e a Figura 3.1 a seguir mostra a tela de análise do aplicativo em relação aos passageiros.

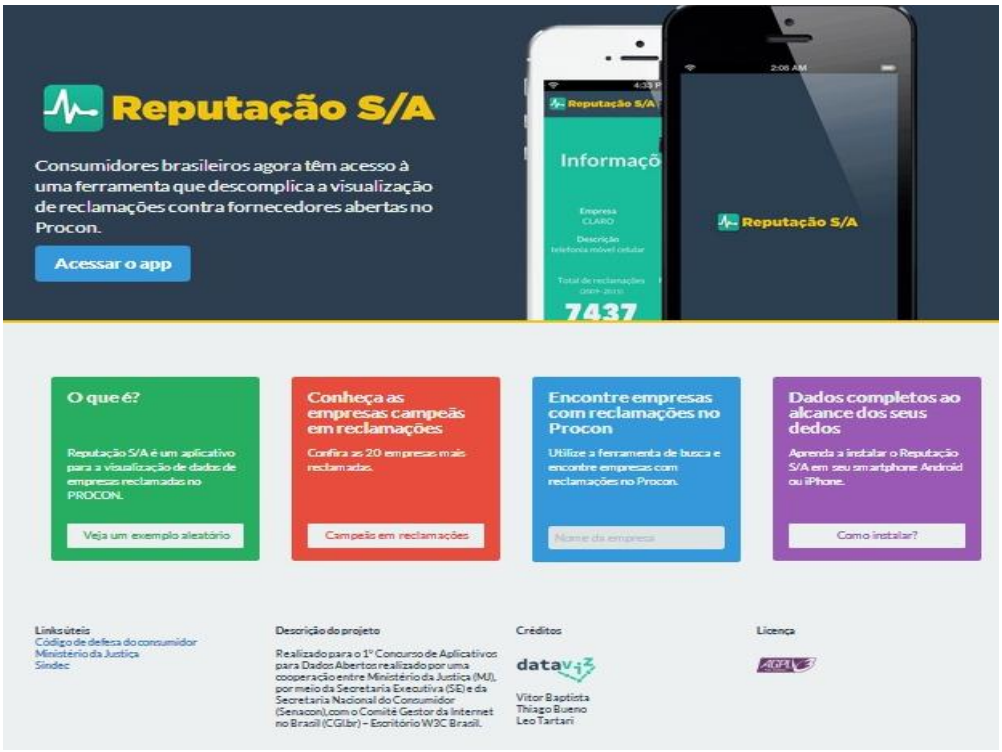
Figura 3.1: Tela de Análise do Aplicativo Aeroportos Brasil



Fonte: Aeroportos Brasil. 2012. Disponível em: http://ison.stratebi.es/aerobrasil/.

Outro exemplo seria o aplicativo Reputação S.A. [2014] que utiliza os dados do Cadastro Nacional de Reclamações Fundamentadas para fornecer diversas informações sobre as empresas em formato ilustrativo e intuitivo, sendo de fácil utilização em *smartphones* e *tablets* devido ao *layout* vertical. Tal aplicativo está disponível também através do endereço: http://reputacao-sa.org/ e a Figura 3.2 mostra a interface do aplicativo.

Figura 3.2: Tela do site Reputação S/A



Fonte: Reputação S.A. 2014. Disponível em: http://reputacao-sa.org/.

Como destacado inicialmente, estas ferramentas foram desenvolvidas a partir de um conjunto específico de dados e, dessa forma, para que fosse possível analisar todos os dados abertos disponíveis pelo governo seria necessário uma ferramenta para cada conjunto de dados, o que notoriamente é pouco viável. Sendo assim, este trabalho percebe a demanda de se criar uma ferramenta universal que possibilite a consulta independente desses dados, de modo que qualquer conjunto de dados possa ser analisado através dela.

Com isso, é justificado o desenvolvimento do XQPref que além de propiciar a elicitação das preferência necessárias para a extensão XQuery-Pref ainda viabiliza a realização de consultas nestes dados abertos governamentais que estejam em formato XML. Uma importante ressalva em relação à limitação pelo formato exigido do sistema XQPref é que os dados que não estejam disponíveis neste formato no site http://dados.gov.br, podem ser solicitados através do mesmo.

3.4 Considerações Finais

Neste capítulo foram apresentados os trabalhos relacionados com esta pesquisa, enfatizando pontos em comum e principalmente enfatizando as diferenças entre as abordagens. A contribuição desta dissertação será diferenciada das demais na seção 5.3, após a avaliação da proposta defendida neste trabalho.

No entanto, já é possível diferenciar, principalmente, o foco de estudo com cada um dos trabalhos relacionados citados neste Capítulo, como ilustra o Quadro 3.1.

Quadro 3.1: Comparativo entre as áreas de pesquisa envolvidas nos trabalhos relacionados

Trabalhos	**Personalização de Consultas**	**Preferências Condicionais**	**XML**	**Banco de dados Relacional**	**Dados Abertos**
Koutrika[2010]	√				
Yan et al.[2001]	√		√		
Gomes[2002]	√		√		
Ribeiro[2008] e Pereira [2011]	√	√		√	
Aeroportos Brasil [2012]					√
Reputação S.A. [2014]					√

Diante do observado, destaca-se que a presente pesquisa se relaciona com pelo menos uma das áreas de pesquisa de cada um dos trabalhos ilustrados no Quadro 3.1 e discutidos neste Capítulo. No entanto, esta pesquisa se beneficia da interseção destas áreas. De modo que, seu objetivo se concentra em aplicar os formalismos da personalização de consultas na elicitação de preferências condicionais, para a realização de consultas em documentos XML visando o acesso otimizado e transparente ao conteúdo disponibilizado pelo governo através dos dados abertos Governamentais.

No próximo Capítulo será apresentada a proposta desta pesquisa, a extensão XQuery-Pref, para tanto, serão definidos os formalismos necessários para a personalização de consultas e as funções extensíveis que dão vida a esta extensão.

Capítulo 4

Extensão da Linguagem XQUERY

Neste capítulo é apresentada a proposta desta pesquisa, descrevendo e formalizando metodologicamente o problema. Para melhor apresentar a proposta o capítulo foi dividido em três seções. Na primeira seção são especificadas as definições preliminares da pesquisa, e na segunda, o problema no tratamento das preferências do escopo deste trabalho é formalizado. A terceira seção apresenta a especificação da extensão para linguagem XQuery, denominada de XQuery-Pref.

4.1 Definições Preliminares

A XQuery-Pref é uma extensão da linguagem XQuery que permite realizar consultas em documentos XML contendo preferências condicionais. A sua grande vantagem é a possibilidade de expressar preferências condicionais de maneira intuitiva através de um conjunto de regras de preferência.

Propor uma extensão à linguagem de consulta XQuery para suporte a preferências envolve desde a definição de um modelo de preferência, até a proposição de operadores algébricos ou funções e sua implementação, sendo importante ainda desenvolver a forma como o usuário vai ter acesso a esta extensão, possibilitando de fato o aproveitamento das vantagens oferecidas.

Com isso, a proposta deste trabalho envolve também o desenvolvimento de um sistema que funcionará como um mini SGBD para XML utilizando a linguagem XQuery + XQuery- Pref, denominado XQPref. A principal motivação para o desenvolvimento do sistema XQPref foi o fato de não encontrar facilmente um sistema que realize consulta em XML utilizando a XQuery que seja gratuito e livre, para possibilitar a inserção das funções da XQuery-Pref no core do sistema. Mais detalhes do Sistema XQPref serão explanados no próximo Capítulo.

O sistema XQPref, através de um processo de tradução, que utiliza informações de metadados, converte a sentença de consulta com as preferências para uma sentença de consulta em XQuery, e realiza a busca efetiva à base de dados XML de forma transparente ao usuário, respeitando as suas preferências no resultado retornado.

Em relação ao modelo de preferência, como visto, este é um formalismo lógico para especificação e raciocínio com preferências. Neste trabalho, o modelo de preferência segue a linha do formalismo de Wilson [2004], mas também serão uteis conceitos especificados com a CP-net [Boutilier et al, 2004], para melhor compreender como serão tratadas as preferências na personalização de consultas, no sentido de representar os desejos do usuário através de regras de preferências condicionais e teorias de preferências condicionais, que, por sua vez, induzem uma ordem de preferência sobre os elementos de um documento.

As expressões condicionais pré-estabelecidas no XQuery são bastante úteis no contexto da personalização de consultas baseado em preferências condicionais, pois estas expressões baseadas em *if, then, else* são favoráveis quando a estrutura da informação a ser retornada depende de alguma condição, podendo ainda ser aninhadas e usadas em qualquer lugar onde um valor é esperado.

Desta forma, quando o usuário fornecer suas preferências estas serão armazenadas em variáveis, onde uma relação de preferências sobre um conjunto finito de objetos $A=\{a_1,a_2,...,a_n\}$ formam uma relação de ordem sobre A, e esta relação de ordem será aplicada à consulta através das expressões condicionais.

A declaração de funções é uma das mais simples maneiras de estender uma linguagem e a XQuery implanta o conceito de funções extensíveis que atuam sobre documentos, seções ou valores atômicos. A grande vantagem na definição de funções na XQuery concentra-se na linguagem em que elas são definidas, pois a própria XQuery, com suas construções, permite que funções sejam escritas, não sendo necessário especificar funções numa linguagem de programação à parte.

Diante do exposto, um bloco de consulta na linguagem XQuery-Pref utiliza o domínio das expressões condicionais já existentes na linguagem XQuery aliada à flexibilidade da linguagem para criar funções customizadas.

Em resumo, a proposta da XQuery-Pref é que o usuário expresse primeiro suas preferências através do sistema XQPref sem precisar se preocupar com manipulação das

variáveis onde estas preferências serão armazenadas e em seguida, as consultas serão realizadas naturalmente utilizando as caixas de seleção do XQPref, e este se encarregará de traduzir as consultas para XQuery e retornar os resultados de acordo com as preferências expressas pelo usuário.

De forma transparente ao usuário, cada preferência expressa se transforma em uma regra que é declarada seguindo a sintaxe IF <a> THEN <b> <relacao_pref > e várias regras são conectadas pela palavra-chave and. Estas regras podem ser criadas com ou sem antecedentes, resguardando a existência de preferências independentes, por exemplo, no caso de um usuário preferir informações sobre determinada região independente do tema tratado no conteúdo.

O parâmetro opcional <relação_pref>, é representado entre colchetes e indica os elementos que são menos importantes ou que não levam em conta a semântica *ceteris paribus*, dependendo das preferências já expressas.

Para nível de explicação, o termo c*eteris paribus* no contexto deste trabalho significa que os elementos que não são citados na preferência do usuário não influenciam no resultado, mas para que duas tuplas possam ser comparadas tais elementos precisam ser iguais.

É interessante destacar que o XQPref cria através das preferências expressas pelo usuário, um conjunto de preferências, denominado <user_pref> sobre as relações especificadas em <lista_prefs> e o armazena em outro documento XML, pois em um segundo acesso ao sistema, o usuário pode recarregar suas preferências já expressas ou expressar novas preferências de acordo com suas necessidades.

4.2 Formalização do Problema

Inicialmente, é importante destacar que no decorrer do trabalho será utilizado o termo tupla a qual é definida matematicamente como uma sequência finita (também chamada de "lista ordenada") de objetos, cada um deles sendo de um tipo específico. Chama-se atenção para o fato de o termo tupla utilizado no trabalho não se relacionar com o contexto do banco de dados relacional.

Uma regra de preferência condicional (regra-pc) sobre um conjunto de elementos E é uma expressão φ da forma:

$$\varphi: u \rightarrow Q_1(X) > Q_2(X)[W]$$

Onde:

- $X \in E, W \subseteq E, X \notin W$;
- $X \in E, W \subseteq E, X \notin W$;
- $Q_i(X), para\ i = 1,2,$ é $um\ predicado\ unário\ sobre\ o\ domínio\ de\ X$ ($\boldsymbol{dom}(X)$);
- $\{x \in \boldsymbol{dom}(X) \mid x \models Q_1(X)\} \cap \{x \in \boldsymbol{dom}(X) \mid x \models Q_2(X)\} = \emptyset$;
- u é $chamado\ de\ condição\ de\ regra - pc\ \varphi$. É uma condição simples de expressões do tipo: $P_1(A_1) \wedge P_2(A_2) \wedge \ldots \wedge P_k(A_k)$, onde cada $P_i(A_i)$ é um predicado unário sobre o atributo $A_i, para\ i = 1, 2, \ldots, k$);
- Seja $E(u)$ o conjunto de atributos que aparecem em u. Então, $(\{X\} \cup W) \cap E(u) = \emptyset$.

Para cada elemento E que está do lado esquerdo de φ, uma tupla u sobre E é dita compatível com uma regra-pc. Tem-se que $u[A]$ satisfaz $P_\varphi(A)$, onde $u[A]$ constitui o valor do atributo A na tupla u e $P_\varphi(A)$ constitui a propriedade P associada ao atributo A na regra-pc φ.

A representação de um conjunto maior de preferências é feita através de uma teoria de preferência condicional (teoria-pc), onde uma teoria-pc sobre E é um conjunto finito de regras-pc sobre E.

Para melhor ilustrar a formalização, tomemos o Apêndice A o qual é um trecho de código XML que será utilizado durante todos os exemplos desta seção, o qual possui a lista de elementos a seguir, com seus respectivos domínios e suas siglas que serão utilizadas nos exemplos abaixo.

- Modalidade educacional (M): educação especial (m_1), educação à distância (m_2) e educação infantil (m_3);
- Região (R): nordeste (r_1), sudeste (r_2), sul (r_3), norte (r_4);
- Dados (D): estatísticos (d_1) e orçamentais (d_2);
- Dados Estatísticos (d_1) : evasão (d_{11}), aprovação (d_{12}), reprovação (d_{13});
- Dados orçamentais (d_2): empenho anual (d_{21}), alimentação (d_{22}), material (d_{23});

Exemplo: Tomando o trecho de um documento XML ilustrado no Apêndice A, que contém informações sobre Educação do Brasil, traz dados abertos como orçamento e dados estatísticos sobre Educação nas modalidades de ensino Educação Especial, Educação à

Distância e Educação Infantil. Diante disto, para o usuário conseguir acessar o conteúdo de seu interesse, ele precisa enfrentar uma gama muito grande de informações, o que o levou a utilizar o sistema XQPref. Para que realize suas consultas ao documento de forma personalizada, o usuário forneceu as seguintes preferências:

- φ_1: Entre informações de mesma modalidade (M), prefiro as relacionadas com a região (R) nordeste (r_1) a norte (r_4), ou seja, (R = r_1) > (R = r4) [{D}];
- φ_2: Para informações da mesma região (R), prefiro informações sobre educação especial (m_1) à educação à distância (m_2), ou seja, (M = m_1) > (M = m_2) [{D}];
- φ_3: Para informações sobre a educação especial (m_1) da mesma região (R), prefiro dados estatísticos (d_1), ou seja, (M = m_1) → (D = d_1) > (D = d_2);

Observando a regra φ_2, por exemplo, dadas duas tuplas $x_1 \, e \, x_2$, se elas tiverem informações sobre a mesma região, e x_1 abordar educação profissional enquanto x_2 a educação à distância, então $x_1 \, \text{é} \, preferida \, a \, x_2$, logo $x_1 > x_2$. Observe ainda que os atributos entre colchetes representam aqueles atributos cujos valores não interferem na escolha das preferidas. Como o atributo Tipo de dado (D) não é antecedente, nem consequente e nem está entre colchetes, para que duas tuplas possam ser comparadas através dessa regra φ_2, elas têm que conter o mesmo tipo de dado (D) (*ceteris paribus*).

As regras de preferências obtidas seguem o formalismo das Cp-nets [Boutilier et al., 2004], abordado na Seção 4.2.1 e, em conformidade com estas regras de preferências é possível inferir uma ordem sobre um conjunto de tuplas que pertence ao domínio do problema. De fato, a semântica de uma regra-pc está relacionada com a ordem de preferência induzida pela teoria-pc sobre as tuplas da relação. De modo que:

Seja A(A_1, A_2, ..., A_n) um documento XML e X um conjunto de tuplas sobre A, sabendo que X_i e X_j são tuplas no conjunto X, um elemento de $X_i \times X_j \times \{1,0\}$, com $X_i \neq X_j$ é denominado uma amostra de preferência. Ou seja, $[(a, b), 1]$ com a,b $\in$ X, significa que a tupla a é preferida à tupla b e assim, a < b, enquanto $[(a, b), 0]$ significa que b > a.

Uma ordem de preferência sobre X(A) é uma relação binária Pref sobre X(R), isto é, um subconjunto Pref $\subseteq$ X(R) × X (R) desde que seja irreflexiva e transitiva. Logo, uma ordem de preferência é uma relação binária irreflexiva e transitiva sobre duas tuplas X_i e X_j, que permite estabelecer se X_i é preferida a X_j, se X_j é preferida a X_i ou se X_i e X_j são

incomparáveis. No contexto da presente proposta, uma teoria-pc Γ sobre uma relação A induz uma ordem de preferência sobre as tuplas de A.

Em um primeiro momento, este trabalho propõe para a consulta XQuery-Pref a obtenção de uma ordem de preferência forte sobre um conjunto de tuplas, inspirada no formalismo de Wilson [2004], discutido no Capítulo 2. A ordem forte é baseada na semântica *ceteris paribus*, mais conservativa e intuitiva. Com os estudos sobre a ordem forte espera-se que ela seja suficiente para resultados satisfatórios no contexto da busca em documentos XML, tendo em vista a representação estruturada dos dados. Onde:

Seja φ uma regra-pc sobre os elementos de A, E(A). Denotamos $\succ_{\varphi}$ o conjunto de pares de tuplas $(x_1,x_2) \in X(A)$ que satisfazem as seguintes condições:

- Para cada elemento $A \in E(A)$ que aparece do lado esquerdo de φ, temos $x_1[A] = x_2[A]$ e $x_1[A] \models P_{\varphi}[A]$ (predicado P associado ao atributo A em φ).
- Para cada atributo $B \in E(A)$ que não aparece em φ, temos $x_1[B] = x_2[B]$;
- Para cada atributo X que aparece do lado direito de φ, $t_1[X] \models Q_1(X)$ e $t_2[X] \models Q_2[X]$.

Seja Γ uma teoria-pc sobre E(A). Denotamos por $\succ_{\Gamma}$ o fecho transitivo de $\cup_{\varphi \in \Gamma} \succ_{\varphi}$. Γ induz uma ordem de preferência forte sobre X(A). Logo, para todo par de tuplas $(x_1, x_2) \in \succ_{\varphi}$, dizemos que x_1 é preferido a x_2 de acordo com uma de preferência forte e denotamos $x_1 \succ_{\Gamma} x_2$.

Tomando outro exemplo, ainda utilizando o trecho do documento XML no Apêndice A, onde cada preferência expressa sobre os elementos do documento compõe uma tupla ordenada, de modo que, seria representado da seguinte forma: A (R, M, D) e para dom (R) = $\{r_1, r_2, r_3, r_4\}$, dom(M) = $\{m_1, m_2, m_3\}$ e dom(D) = $\{d_1, d_2\}$. Seja a teoria-pc $\Gamma = \{\varphi_1, \varphi_2\}$ sobre E(A), onde:

- φ_1: $(R = r_1) \rightarrow (M = m_2) > (M = m_1)$ [{D}];
- φ_2: $(M = m_2) > (M = m_3)$;

De acordo com a ordem de preferência forte induzida por Γ, temos que:

- $\succ_{\varphi 1}$= *{((r1,m2,d1), (r1,m1,d1)), ((r1,m2,d2), (r1,m1,d2)), ((r1,m2,d2), (r1,m1,d1)), ((r1,m2,d1), (r1,m1,d2)},*
- $\succ_{\varphi 2}$= *{((r1,m2,d1), (r1,m3,d1)), ((r1,m2,d2), (r1,m3,d2))} e,*

$>_\Gamma = \{((r_1, m_1, d_1),\ (r_1, m_3, d_1)),\ ((r_1, m_1, d_1),\ (r_1, m_3, d_2)),((r_1, m_1, d_2),\ (r_1, m_3, d_1)),$
$((r_1, m_1, d_2), (r_1, m_3, d_2))\} \cup >_{\varphi 1} \cup >_{\varphi 2}$.

Desta forma, a tupla $x_1(r_1, m_1, d_1)$ é preferida a $x_2 = (r_1, m_3, d_1)$, assim sendo, $x_1 >_\Gamma x_2$.

Com a definição da Ordem de Preferência Forte dois pontos merecem destaque:

1. Cada regra φ individualmente induz uma ordem sobre as tuplas. Nessa ordem, as preferências locais sobre os valores dos atributos das tuplas não são absolutas, ou seja, só é possível inferir que uma tupla é preferida a outra em relação ao atributo consequente de φ, dependendo de como os outros atributos estão instanciados;
2. Para testar se duas tuplas $x = (x_1, x_2, \ldots, x_n)$ e $x' = (z_1, z_2, \ldots, z_n)$ são comparáveis de acordo com a teoria-pc $\Gamma = \{\varphi_1, \ldots, \varphi_k)$, isto é, $x >_\Gamma x'$, deve-se verificar se existe um subconjunto de $\Gamma' = \{\delta_1, \ldots, \delta_m)$ para$\Gamma' \subseteq \Gamma$, e tuplas $y_1, \ldots, y_{m-1}$tal que $x >_{\delta 1} s_1, s_1 >_{\delta 2} s_2, \ldots, s_{m-1} >_{\delta m} x'$.

É importante frisar que para cada regra-pc φ, a relação $\succ_\varphi$ mantém a semântica *ceteris paribus* em relação aos atributos que não pertencem à regra φ. Ou seja, tuplas com valores diferentes para cada elemento Z $\in$ E(A), que não estão em φ, são incomparáveis de acordo com $\succ_\varphi$. Em resumo, a ordem forte de preferência do XQuery-Pref não induz uma preferência absoluta sobre os atributos (daí preferências condicionais) e é inferida a partir do fecho transitivo da união das ordens de preferência de cada regra-pc. Mais detalhes de como a extensão será especificada serão abordados na próxima seção.

4.3 Extensão da Linguagem XQUERY

Nesta seção, é apresentada a extensão à linguagem XQuery em especial, as funções definidas para que a linguagem suportasse a personalização de consultas. A esta extensão foi dado o nome XQuery-Pref.

4.3.1 XQuery-Pref

A linguagem XQuery é formada por vários tipos de expressões, dentre elas as expressões FLWR, que funcionam semelhante ao conjunto de instruções SELECT-FROM-WHERE do SQL, para a XQuery-Pref o comando Where merece atenção especial, pois é utilizado para especificar um ou mais critérios para o resultado. E esta semelhança ocorre

também no plano de execução de consultas XQuery-Pref, sendo executados após a seleção e antes da projeção.

A linguagem atua sobre a estrutura lógica e abstrata dos documentos, de modo que o modelo de dados representa os documentos como árvores onde os nós podem corresponder a documentos, elementos, atributos, textos, espaços de nomes, instruções de processamento ou comentários, e cada nó tem um identificador único.

Um diferencial importante da linguagem XQuery, e que merece destaque nesta seção, é o fato desta possuir o conceito de funções extensíveis que atuam sobre documentos, seções ou valores atômicos. A grande vantagem na definição de funções na XQuery concentra-se no fato de não ser necessário especificar funções numa linguagem de programação à parte, pois a própria XQuery permite que estas funções sejam criadas.

A definição de funções pelo utilizador permite que os fragmentos de consulta possam ser reutilizados, e que as bibliotecas de código desenvolvido sejam reutilizadas por outras partes, tornando ainda o algoritmo passível a mudança de forma simples, pois é possível alterar a lógica do algoritmo ou corrigir *bug* em apenas um lugar.

Com esta flexibilidade oferecida pela XQuery, o modo escolhido para efetuar a extensão foi a definição de novas funções, visando facilitar a realização de consultas baseadas nas preferências expressas pelo usuário e não sendo necessário, dessa forma, a modificação na especificação original da linguagem de consultas.

O objetivo elementar desta extensão é que ela seja um mecanismo para consultar partes ou subconjuntos de documentos XML, fornecendo uma solução para a necessidade de localizar elementos, atributos ou outros nodos de documentos XML de modo personalizado.

Para melhor compreender o funcionamento da função criada, destaca-se que a partir das preferências expressas por um usuário o sistema XQPref, que será abordado mais profundamente na próxima seção, realizará o tratamento das preferências de modo que terá como entrada e saída:

- Entrada: Um conjunto de amostras de preferências consistentes sobre um conjunto X de tuplas, onde: $A \subseteq X \; x \; X \; x \; \{0,1\}$;
- Saída: Uma ordenação que seja compatível com as amostras de preferências A, com boa acurácia, onde: $O: X \; x \; X \; \rightarrow \{0,1\}$;

Para cada preferência expressa pelo usuário será atribuída uma variável denominada *relevance* que armazenará a relevância associada a uma expressão, e esta variável fica associada a um valor de tipo xs:float no intervalo [0,1], onde um valor maior indica uma maior relevância. É importante ressaltar aqui, que o valor desta relevância será calculado de acordo com a teoria-pc explicada na seção anterior.

As expressões condicionais são análogas ao IF-THEN-ELSE das linguagens de programação e são uteis quando a estrutura da informação a ser retornada depende de alguma condição. Podendo ser aninhadas e utilizadas em qualquer lugar onde um valor é esperado, de modo que, se o teste for verdadeiro, o valor da primeira expressão é retornado. Caso contrário, o resultado da segunda expressão é retornado. Este artifício será bastante útil para a XQuery-Pref, uma vez que será necessário condicionar o resultado da consulta personalizada de acordo com as preferências expressas pelo usuário e ordenadas pela regra-pc.

A função *fn:createpref* é responsável pela criação das preferências do usuário, sendo a partir disto atribuída a relevância da tag no conjunto de preferências do usuário. Como mostrado abaixo:

Declaração da função extensível *createpref*

```
1   declare function fn:createpref ($i as string) as return String
2       {for $i in (doc("...")/elemento1/elemento2)
3        return if ($i/@elemento = 'pref')
4               then <elemento>{pref($pref2/info)}</elemento>
5               else
6                   if ($elemento2 = 'pref3')
7                   then<elemento2>{pref3($pref4/info)}</elemento2>
8                   else
9                       if ($elemento3 = 'pref5')
10                      then <elemento3>{pref5($pref6/info)}</elemento3>
11                      else <elemento3>{pref6($pref7/info)}</elemento3>
12              [...]};
```

Para outro exemplo, tomemos novamente o trecho do documento XML (Apêndice A), e um trecho do exemplo citado no início da seção, onde o usuário queira expressar a seguinte preferência:

- φ_1: Entre informações de mesma modalidade, prefiro as relacionadas com a região nordeste;

```
1   declare function fn:createpref ($i as string) as return String
2     {for $i in (doc("educacao_brasileira"))
3      return if ($i/@modalidade_educacao = 'educacao_especial')
```

```
4                then <regiao>{$nordeste)}</regiao>
5               else
6                  if ($modalidade_educacao = 'educacao_a_distancia)
7                  then<regiao>{$nordeste)}</regiao>
8                  else <educacao_infantil>{$nordeste)}</educacao_infantil>};
```

Neste caso, o usuário está expressando que ao acessar informações desta base de dados e realizar consultas sobre a modalidade educacional ele prefere receber informações da região nordeste. Assim, entre uma tupla t_1(educacao_especial, nordeste, evasao) e t_2(educacao_especial, norte, evasao), o usuário prefere receber a tupla t_1 como resultado.

Outra função definida foi a *fn:preference*, tal função pode ser usada sempre que uma comparação é possível, e esta inclui um caminho para especificar os nós onde a função será aplicada, e a expressão com a palavra-chave que está sendo procurada no conteúdo dos nós. Onde:

Declaração da função extensível *preference*

```
1   declare function fn:preference($i as string) as return String
2          { Let $i relevance $r in doc ("...") /elemento1/elemento2
3          [tema fnpreferences "tema"]
4          Order by $r
5          Return $i };
6                 [...]};
```

Por exemplo, em uma consulta para retornar informações relacionadas à evasão sobre a educação da região nordeste ordenado de acordo com as preferências do usuário:

```
1   For $i in doc("...") /regiões/nordeste
2          Where $i/tema fnpreferences "dados_estatisticos"
3          Return $i;
```

Levando em consideração que o usuário já expressou suas preferências e nelas prevaleceu que para a região nordeste prefere informações sobre o índice de evasão na educação especial a demais modalidades, então esta consulta terá como resultado os índices de evasão desta modalidade no topo dos resultados, tendo em vista que a função ordenará o resultado de acordo com esta preferência.

Como a consulta realizada solicita dados estatísticos da região nordeste, e o usuário informou que prefere dados da educação especial, o resultado da consulta trará todas as informações estatísticas da região citada, no entanto, colocará no topo dos resultados os dados da educação especial. Deste modo, o resultado obtido pelo usuário seria o seguinte:

```
1  <educacao_especial>
2  <regiao>
3  <nordeste>
4  <dados_estatisticos>
5  <evasao>5.3 </evasao>
6  <aprovacao>9.1</aprovacao>
7  <reprovacao>0.9</reprovacao>
8  </dados_estatisticos>
9  </educacao_especial>
10 </regiao>
   </nordeste>
   [...]
```

É importante relembrar que ao realizar consultas na linguagem XQuery o resultado é retornado em XML, como foi mostrado na Seção 2.2.2.3, e esta forma de retorno é mantida ao se utilizar as funções definidas através da XQuery-Pref. A grande diferença em relação ao resultado é a quantidade de informação retornada e a ordem estabelecida neste resultado.

Por fim, destaca-se o fato que uma execução transparente de consultas escritas na linguagem XQuery utilizando os recursos da extensão XQuery-Pref, requer um processo automático de tradução de consultas. Esta tarefa além de não ser trivial, pois é necessário que o tradutor conheça as estruturas do esquema de representação utilizado para armazenar os documentos XML, pode também não ser eficiente principalmente em consultas recursivas.

Assim, optou-se pela implementação de um sistema denominado XQPref, que servirá como camada de tradução, que utiliza informações de metadados do documento XML e proporciona a conversão da consulta realizada pelo usuário por uma sentença de consulta XQuery, e realiza a busca efetiva no documento de forma transparente ao usuário. Mais informações sobre o XQPref serão explanadas no próximo capítulo.

No entanto, ressalta-se que qualquer pessoa pode utilizar-se da extensão aqui apresentada e criar seu próprio mecanismo de consulta da maneira que lhe convir, tendo em vista que a extensão é realizada na própria linguagem XQuery através de funções extensíveis e independe de linguagem de programação.

4.4 Considerações Finais

Neste capítulo foi proposta a extensão da linguagem de consulta XQuery, denominada no trabalho como XQuery-Pref. O mecanismo utilizado para realizar a extensão foi as funções extensíveis, suportadas pela própria XQuery.

A função fn:createpref é responsável pela criação das preferências do usuário, sendo a partir disto atribuída a relevância da tag no conjunto de preferências do usuário. Outra função definida foi a fn:preference, a qual pode ser usada sempre que uma comparação é possível, e esta inclui um caminho para especificar os nós onde a função será aplicada, e a expressão com a palavra-chave que está sendo procurada no conteúdo dos nós.

Tendo em vista que, propor uma extensão à linguagem de consulta XQuery para suporte a preferências envolve também desenvolver a forma como o usuário vai ter acesso a esta extensão, possibilitando de fato o aproveitamento das vantagens oferecidas.

Deste modo, como dito anteriormente, a proposta deste trabalho envolve também o desenvolvimento de um sistema que funcionará como um mini SGBD para XML utilizando a linguagem XQuery + XQuery- Pref, denominado XQPref. A principal motivação para o desenvolvimento do sistema XQPref foi o fato de não encontrar facilmente um sistema que realize consulta em XML utilizando a XQuery que seja gratuito e livre, para possibilitar a inserção do operador XQuery-Pref no core do sistema. Mais detalhes do Sistema XQPref serão explanados no próximo Capítulo, seguido da avaliação da proposta através de um estudo de caso.

Capítulo 5

Estudo de Caso

Para realização do estudo de caso desta proposta foram realizados testes utilizando o sistema XQPref, apresentado como contribuição secundária deste trabalho, com o intuito de analisar a consistência e acurácia dos resultados na realização de consultas em dados abertos governamentais através da extensão XQuery-Pref. Mais detalhes serão apresentados e discutidos nas subseções abaixo.

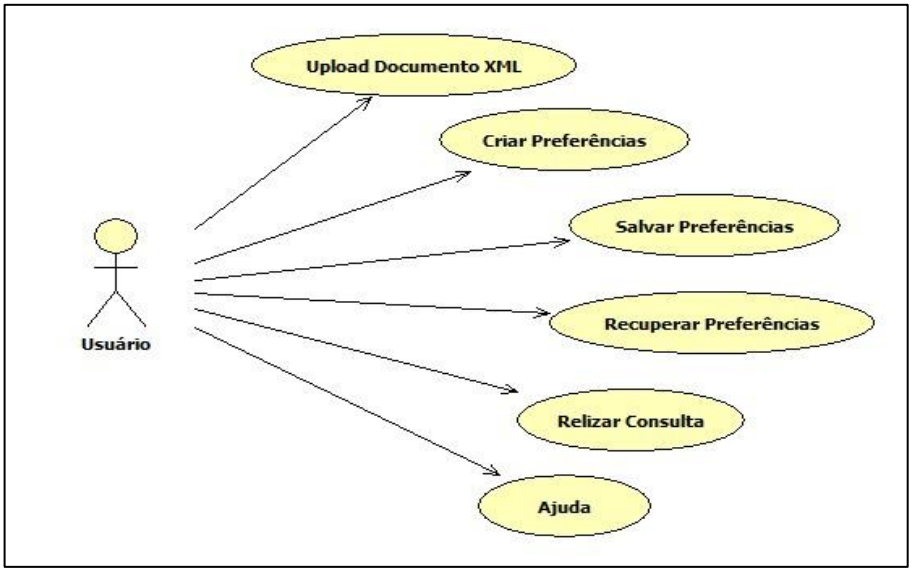
5.1 XQPref

O sistema XQPref tem como objetivo suprir a necessidade de um mecanismo de consulta personalizada baseada em preferências condicionais em base de dados XML. Para tanto, o sistema permite a execução de consultas XQuery em que funções de preferências estão embutidas na linguagem e o uso destas funções possibilita a otimização da recuperação de informações nessas bases de dados, reduzindo ainda o esforço do usuário para encontrar conteúdo relevante, de acordo com suas necessidades.

O XQPref torna o processo de fornecimento de preferências mais simples, de modo que a função *fn:createpref* é mapeada para o sistema através da seleção das tags pelo usuário, de modo que o próprio sistema atribui a relevância da tag no conjunto de preferências do usuário.

O XQPref proporciona uma interface de usuário para a biblioteca de execução XQuery. As funções propostas para realizar as consultas respeitando as preferências expressas pelo usuário são disponibilizadas através de um *namespace* na linguagem XQuery e permitem atender a propósitos de busca variados, simplificando o processo de consulta para o usuário. As funcionalidades incluídas na interface de usuário podem ser vistas na Figura 5.1, a qual apresenta o diagrama de casos de uso da ferramenta.

Figura 5.1: Diagrama de casos de uso com as funcionalidades do sistema XQPref



A primeira funcionalidade do sistema é o upload do documento XML, esta função visa facilitar o processo de consulta para o usuário, pois ao alimentar o sistema com o documento que será a fonte das consultas, o usuário ficará isento de fornecer o caminho deste sempre que realizar buscas, além de possibilitar que o sistema realize uma varredura sobre as tags, de modo a mapeá-las de forma dinâmica para opções que serão oferecidas para o usuário selecionar suas preferências.

As três próximas funcionalidades referem-se à manipulação das preferências, as quais podem ser criadas, salvas em formato XML e recuperadas. Assim, em um segundo acesso ao sistema, o usuário pode reutilizar preferências já expressas ou expressar novas, de acordo com suas necessidades.

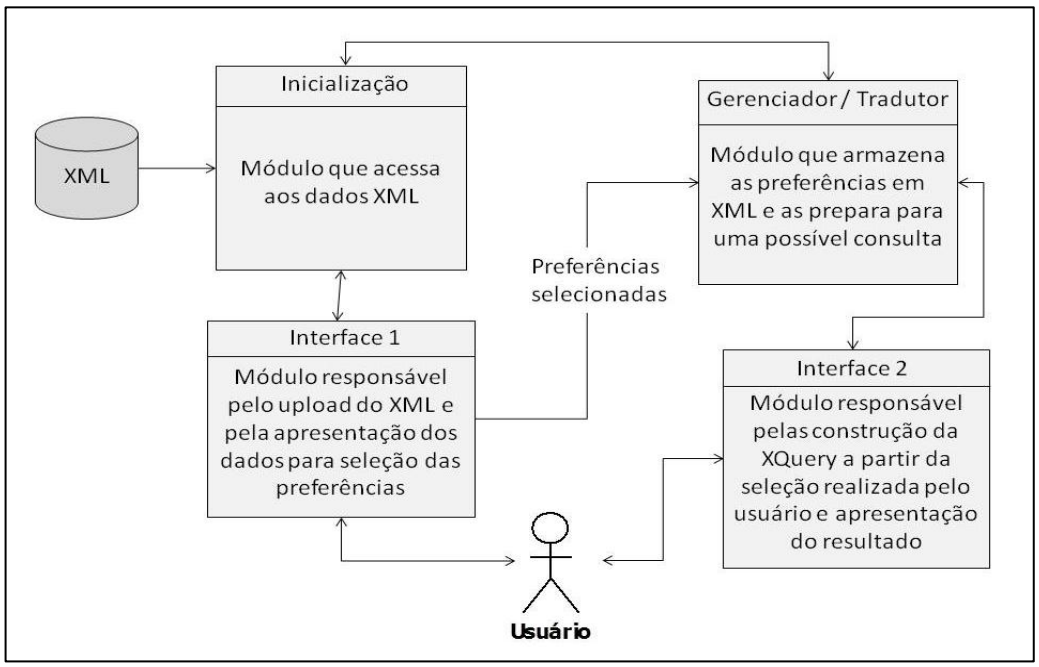
Em seguida, a funcionalidade de realizar a consulta permite que usuário busque as informações de seu interesse através da seleção de opções referentes ao documento fornecido ao sistema e o resultado é retornado na própria ferramenta. É importante destacar que nesta funcionalidade não é preciso nenhum esforço do usuário para que o sistema retorne suas preferências, uma vez que a extensão com as funções de relevância já estão acopladas a linguagem.

Outro destaque em relação ao XQPref, é que o usuário não precisa saber nenhum comando da linguagem XQuery para realizar suas consultas utilizando o sistema, pois uma das preocupações desta pesquisa é que qualquer cidadão possa ter livre acesso às informações disponíveis com os dados abertos governamentais.

Por fim, a última funcionalidade se refere à ajuda disponível para que o usuário possa tirar alguma dúvida que surja na utilização do sistema. Nesta ajuda, haverá uma simples documentação explicando o passo a passo para um bom aproveitamento de suas funcionalidades.

Na Figura 5.2, é apresentada de forma gráfica a arquitetura do sistema XQPref, onde estão ilustrados os módulos necessários para a implementação do sistema e suas relações.

Figura 5.2: Arquitetura do sistema XQPref



Inicialmente, percebeu-se a necessidade de criar um repositório onde serão armazenados os documentos XML. Esse repositório está representado na Figura 5.2 como um cilindro que deve ser acessado pelo módulo da inicialização. A partir desse repositório será possível realizar o mapeamento de forma dinâmica dos elementos presentes no documento.

a) Inicialização

O módulo da inicialização responsabiliza-se em implementar mecanismos de leitura da lista de elementos do documento XML e enviar para o módulo de Interface 1, disponibilizando-as em forma de seleção para que o usuário expresse suas preferências. Quando a interface 1 solicita conteúdo do XML, este módulo deve, portanto, ser capaz de ler o conteúdo do XML a partir do repositório e devolvê-lo à Interface Inicial.

Este módulo também interage com o módulo Gerenciador/Tradutor, recebendo as queries, buscando no repositório e retornando o resultado para que o Gerenciador as envie através da Interface 2 para o usuário.

b) Interface 1

O usuário fornece o documento no qual quer realizar as consultas e a Interface 1 pede uma análise deste XML ao módulo de Inicialização, que deve analisar a árvore da estrutura do documento XML para ter a base das consultas possíveis. Depois de ter analisado o documento, o módulo de Inicialização deve implementar um mecanismo que permita percorrer o conteúdo do XML como uma árvore, de nodo em nodo, e a cada nodo é preciso atribuir um elemento selecionável, para uma possível preferência do usuário.

A Interface 1, além de fornecer o upload do documento XML para inicialização, é o módulo que trata da apresentação da listagem de elementos que deve ser selecionado ainda neste módulo pelo usuário. O módulo da Interface Inicial deve saber pedir informação ao módulo da Inicialização sempre que o usuário seleciona um elemento. Assim, quando a seleção é iniciada, o módulo Interface 1 pede a listagem do nome dos elementos descendentes ao módulo de Inicialização e as mostra ao usuário. Este escolhe um destes descendentes e então é feito um novo pedido ao módulo de Inicialização que devolve dessa vez a listagem de todos os descendentes deste novo elemento selecionado.

Por fim, quando o usuário tiver finalizado o processo de construção de suas preferências este módulo deverá enviar todas as seleções para o módulo Gerenciador ser capaz de concluir o processo de construção das funções adequando às preferências submetidas.

c) Gerenciador / Tradutor

É neste módulo que o sistema calcula a relevância das preferências através da teoria-pc, com base nos nodos selecionados pelo usuário. Portanto, é necessário que também essa referência, com a ordem de seleção, seja passada do módulo Interface 1 para que este módulo possa atribuir a variável relevance explicada na seção anterior a cada preferência, tornando possível assim a comparação de relevância entre elas.

Este módulo também é responsável pela construção das consultas, de modo que deve interpretar as seleções realizadas na Interface 2 e traduzir para linguagem XQuery, ou

seja, a partir de cada nodo selecionado deve ser construída uma consulta de acordo com as preferências expressas da Interface 1. Assim, o resultado deve abranger apenas os elementos que respeitem estas preferências, para tanto as funções de seleção de preferências propostas aqui devem ser adequadas de acordo com a opção submetida no módulo Interface 1. Depois de calculada a consulta, este módulo deve retornar o resultado para o módulo Interface 2.

d) Interface 2

Nesta interface serão apresentados os elementos presentes no documento XML, fornecidos pelo módulo de Inicialização para que o usuário selecione apenas o objeto da sua consulta, e o objeto selecionado será enviado para que o módulo Gerenciador prepare a consulta, busque no repositório através do módulo de Inicialização e retorne para esta Interface o resultado esperado pelo usuário.

Um conjunto de dados reais e fictícios de Dados Abertos Governamentais foi selecionado para a realização de consultas testes utilizando o XQPref, com o objetivo de verificar a consistência e a acurácia das funções extensíveis definidas neste trabalho, o detalhamento desta avaliação experimental se encontra no próximo capítulo.

5.2 Estudo de Caso

5.2.1 Descrição

O estudo de caso tem como proposta realizar consultas personalizadas em dados abertos governamentais utilizando a extensão proposta XQuery-Pref, com o objetivo de verificar o índice de acerto dos resultados recebidos pelos usuários. Para tanto, o mecanismo de consulta utilizado foi o sistema XQPref, onde através dele serão fornecidas as preferências necessárias para que a função *Preference* realize a ordenação das preferências.

Neste momento, é importante destacar que o desenvolvimento do XQPref ocorreu de maneira bastante superficial, apenas com o intuito de viabilizar a realização dos testes utilizando a extensão proposta para a linguagem XQuery. Sendo assim, não foi foco desta avaliação experimental o sistema em si, mas sim o emprego das funções extensíveis definidas neste trabalho, com o objetivo de fornecer uma solução para a personalização de consultas em dados abertos governamentais.

O requisito básico deste estudo de caso está obviamente relacionado ao formato do documento utilizado para realizar consultas ser XML. No entanto, também se destaca a importância de que os documentos utilizados nos testes tenham padrões definidos para estabelecer as tags para que assim a escolha do usuário através destas tags seja facilitada.

5.2.2 Planejamento

Inicialmente, foi realizada uma demonstração do sistema XQPref utilizando dados XML fictícios ou sintéticos (criados pelo próprio autor), para que viabilizasse uma primeira impressão do sistema e, assim, fosse possível detectar falhas simples.

Em seguida foram utilizados um conjunto de dados disponibilizados pelo governo brasileiro através do site: http://dados.gov.br/, com o objetivo de avaliar a acurácia dos resultados.

Todo o processo de avaliação experimental se deu de maneira manual devido à falta de tempo para implementar softwares de testes. Tanto nos testes com dados fictícios quanto reais, foram fornecidas preferências e, em seguida, consultas foram realizadas com o objetivo de verificar as taxas de acerto, ou seja, a média de vezes em que o resultado correspondia ao esperado.

5.2.3 Execução

A realização deste primeiro estudo de caso com documentos menores facilitou a adequação do sistema para os testes com dados abertos reais. Tendo em vista que, o foco inicial é avaliar se os resultados das consultas estão respeitando as preferências fornecidas pelos usuários.

Nesse sentido, os resultados obtidos com essa avaliação inicial serviram apenas verificar o funcionamento do XQPref. Por isso, optou-se por usar documentos menores, o que facilitou facilitou a visualização das opções fornecidas para o usuário. Um exemplo de documento utilizado, segue no trecho de Código Fonte 5.1 como mostra a seguir.

Código Fonte 5.13: Trecho do documento XML utilizado para pesquisar

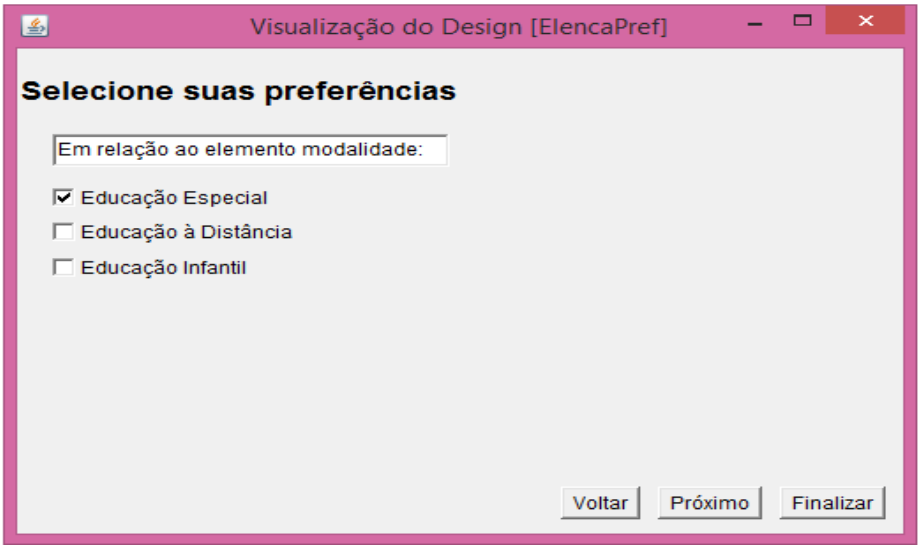
```
1       ...
2       <educacao_especial>
3       <dados_gerais>
4       <evasaobrasil>5.7 </evasaobrasil>
5       <orcamentoanual2013>R$ 5.200.000,00 </orcamentoanual2013>
6       </dados_gerais>
```

```
7       <dados_nordeste>
8       <evasaobrasil>5.3 </evasaobrasil>
9       <orcamentoanual2013>R$ 1.500.000,00 </orcamentoanual2013>
10      </dados_nordeste>
11      <dados_sudeste>
12      <evasaobrasil>6.2 </evasaobrasil>
13      <orcamentoanual2013>R$ 1.200.000,00 </orcamentoanual2013>
14      </dados_sudeste>
15      <dados_sul>
16      <evasaobrasil>4.5 </evasaobrasil>
17      <orcamentoanual2013>R$ 1.350.000,00 </orcamentoanual2013>
18      </dados_sul>
19      <dados_norte>
20      <evasaobrasil>6.7 </evasaobrasil>
21      <orcamentoanual2013>R$ 1.150.000,00 </orcamentoanual2013>
22      </dados_norte>
23      </educacao_especial>
24      <educacao_a_distancia>
25      <dados_gerais>
26      <evasaobrasil>4.2 </evasaobrasil>
27      <orcamentoanual2013>R$ 4.900.000,00 </orcamentoanual2013>
28      </dados_gerais>
29      <dados_nordeste>
30      <evasaobrasil>4.8 </evasaobrasil>
31      <orcamentoanual2013>R$ 1.100.000,00 </orcamentoanual2013>
32      </dados_nordeste>
33      <dados_sudeste>
34      <evasaobrasil>4.2 </evasaobrasil>
35      <orcamentoanual2013>R$ 1.400.000,00 </orcamentoanual2013>
36      </dados_sudeste>
37      <dados_sul>
38      <evasaobrasil>3.9 </evasaobrasil>
39      <orcamentoanual2013>R$ 1.000.000,00 </orcamentoanual2013>
40      </dados_sul>
41      <dados_norte>
42      <evasaobrasil>3.7 </evasaobrasil>
43      <orcamentoanual2013>R$ 1.400.000,00 </orcamentoanual2013>
44      </dados_norte>
45      </educacao_a_distancia>
46      <educacao_infantil>
47      <dados_gerais>
48      <evasaobrasil>2.3 </evasaobrasil>
49      <orcamentoanual2013>R$ 7.700.000,00 </orcamentoanual2013>
50      </dados_gerais>
51      <dados_nordeste>
52      <evasaobrasil>2.3</evasaobrasil>
53      <orcamentoanual2013>R$ 2.000.000,00 </orcamentoanual2013>
54      </dados_nordeste>
55      <dados_sudeste>
56      <evasaobrasil>1.8 </evasaobrasil>
57      <orcamentoanual2013>R$ 1.800.000,00 </orcamentoanual2013>
58      </dados_sudeste>
59      <dados_sul>
60      <evasaobrasil>1.5 </evasaobrasil>
61      <orcamentoanual2013>R$ 1.900.000,00 </orcamentoanual2013>
62      </dados_sul>
63      <dados_norte>
64      <evasaobrasil>3.1 </evasaobrasil>
65      <orcamentoanual2013>R$ 2.000.000,00 </orcamentoanual2013>
66      </dados_norte>
```

```
67      </educacao_infantil>
68      ...
```

No Código Fonte 5.1 está sendo apresentado um trecho de um dos documentos XML fictícios utilizados na primeira etapa da execução dos testes, uma dos pontos que merecem destaque é que todos os documentos utilizados durante esta etapa da execução foram padronizados, ou seja, todas as tags são bem nomeadas e seguem certa rigidez no decorrer do documento. A Figura 5.3 abaixo mostra o processo de consulta.

Figura 5.3: Tela do XQPref para fornecer preferências



Neste primeiro exemplo o usuário selecionou como sua preferência apenas a modalidade educacao_especial e na hora de fazer a consulta também selecionou educacao_especial e desta forma obteve como resultado todas as informações contidas no documento dentro da tag <educacao_especial>, como ilustra a Figura 5.4.

Figura 5.4: Tela do XQPref para realizar a consulta e visualizar resultado

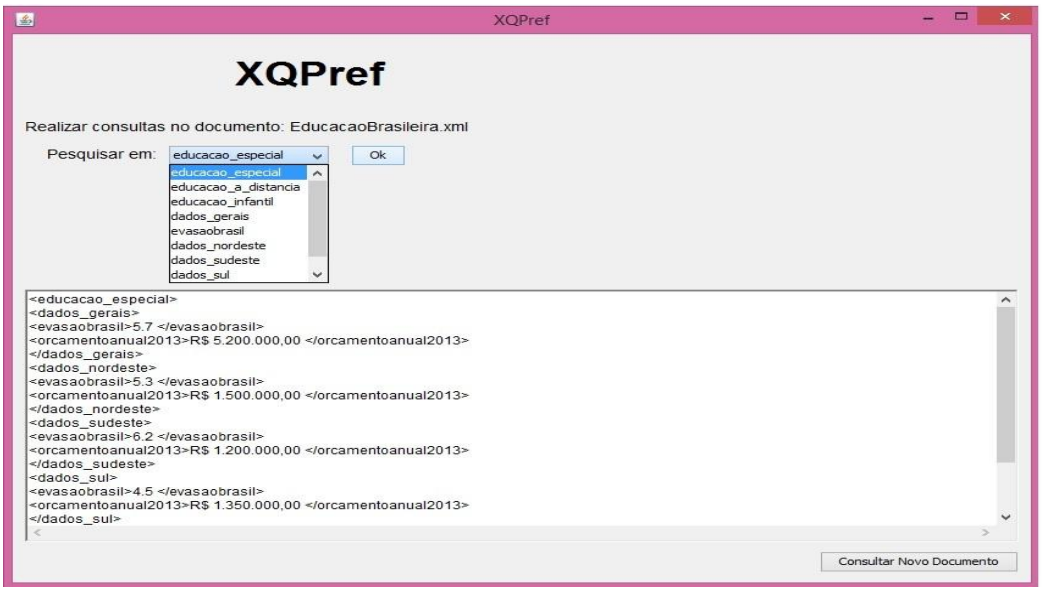
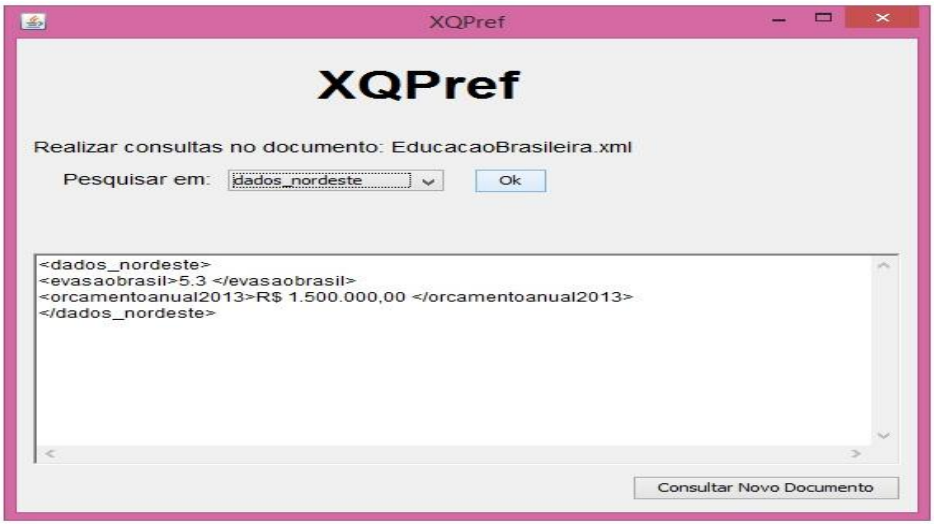


Figura 5.5: Demonstração de Consulta Personalizada XQuery-Pref



Na Figura 5.5, o mesmo usuário que selecionou como preferência informações sobre educacao_especial, vai seleciona a tag <dados_nordeste> para consulta, com isso recebe os resultados referentes à educacao_especial que pertencem ao nordeste

Seguindo o plano definido para o estudo de caso, iniciaram-se os testes com dados abertos governamentais disponibilizados no site do governo federal do Brasil. Da mesma forma que aconteceu com os dados fictícios, nestes testes foram fornecidas preferências de um usuário e consultas eram realizadas, de modo que as taxas de acertos eram computadas pelo próprio usuário (testador).

A taxa de acertos para a avaliação do estudo de caso foi definida como sendo o percentual entre a quantidade de resultados que respeitam as preferências do usuário e o total de consultas realizadas.

$$Taxa\ de\ Acerto = \frac{Resultados\ corretos}{Total\ de\ consultas\ realizadas}$$

Na próxima seção, serão apresentados os resultados experimentais obtidos durante a execução dos testes, seguidos por uma análise geral dos resultados obtidos.

5.2.4 Análise dos Resultados

A análise realizada nesta seção considera dois resultados. O primeiro é o resultado recebido para o conjunto de consultas realizadas utilizando dados fictícios. O segundo é o resultado obtido, para o conjunto de consultas realizadas utilizando os dados abertos governamentais reais.

O primeiro resultado serve basicamente para verificar o funcionamento das funções extensíveis da linguagem XQuery-Pref embutidas no sistema para realização de consultas personalizadas XQPref. Neste sentido, observou-se que ao realizar as consultas em documentos fictícios os resultados recebidos correspondiam à expectativa, ou seja, o usuário recebia o resultado esperado com 100% de acerto. Tal resultado serve principalmente para mostrar que o funcionamento das funções definidas no trabalho realiza o tratamento das preferências da maneira esperada.

O testes realizados com os dados abertos governamentais reais, além de servirem para avaliação das funções definidas no trabalho também objetivou a verificação do funcionamento destas no sistema XQPref, analisando se as funções foram devidamente mapeadas para a ferramenta.

Como dito na seção anterior, para análise dos resultados obtidos com as consultas em dados abertos governamentais, a taxa de acertos para avaliação do estudo foi definida como sendo o percentual entre a quantidade de resultados que respeitam as preferências do usuário e o total de consultas realizadas.

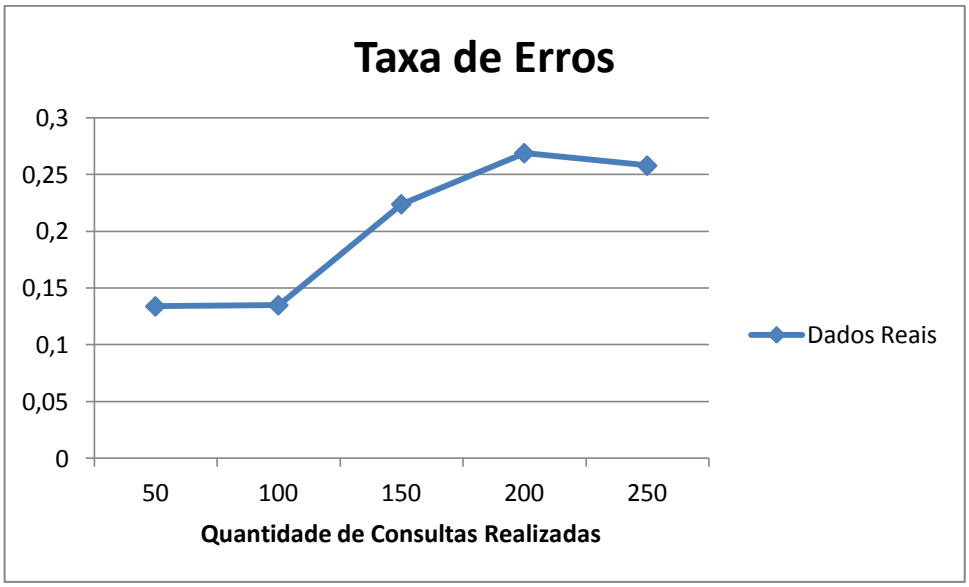
Para melhor comparação, optou-se aplicar a fórmula para taxa de acerto também para as consultas em dados fictícios. Todos os números obtidos estão apresentados na Tabela 6.1 abaixo.

Tabela 6.1 - Taxa de Acertos

	Dados reais	Dados Fictícios
Resultados corretos	194	250
Total de consultas realizadas	250	250
Taxa de acerto	0,776	1

A taxa de acertos mostrada acima revela que ao realizar as consultas em dados abertos governamentais reais ainda existe uma taxa de erro de aproximadamente 22%, com isso foi realizada uma análise aprofundada para investigar as causas destes erros. O gráfico abaixo mostra a taxa de erros de acordo com o aumento da amostra.

Gráfico 6.1 - Taxa de erros ao realizar consultas em dados abertos governamentais reais



Um ponto crucial da análise de resultados foi buscar a compreensão do motivo pelo qual em algumas consultas realizadas em dados abertos governamentais reais os resultados recebidos não correspondiam aos resultados esperados. Para tal entendimento, foi necessária uma avaliação nos dados XML que estavam sendo consultados, tendo em vista que nas consultas realizadas nos dados sintéticos estavam retornando o resultado esperado em todas as buscas.

Diante desta análise, percebeu-se que a causa da taxa de erros nas consultas a dados abertos governamentais reais estava no próprio documento XML, de modo que a causa deste problema estava na falta de padronização das tags. A Figura abaixo nos mostra um trecho de documento XML onde estão sendo mostradas apenas algumas tags.

Trecho Dado Aberto Governamental com as tags bem padronizadas

```
<resource>
...
<identificador>38093150000011014</identificador>
<uasg>380931</uasg>
<modalidade_licitacao>05</modalidade_licitacao>
<numero_licitacao>000232013</numero_licitacao>
<tipo_contrato>50</tipo_contrato>
<licitacao_associada>38093105000232013</licitacao_associada>
<origem_licitacao>SISPP</origem_licitacao>
<numero>000011014</numero>
<objeto>
Prestação de Serviços de Vigilância Desarmada para atender a
sede da SRTE/AL e suas un idades Descentralizadas.
</objeto>
<numero_aditivo>0</numero_aditivo>
<numero_processo>46201006411201376</numero_processo>
```

```
<cnpj_contratada>11179264000766</cnpj_contratada>
<data_assinatura>2014-03-10</data_assinatura>
<fundamento_legal>Lei 8.666/93, Dec 2.271/97 e IN SLTI n¨
02/08.</fundamento_legal>
<data_inicio_vigencia>2014-03-18</data_inicio_vigencia>
<data_termino_vigencia>2015-03-17</data_termino_vigencia>
<valor_inicial>481514.00</valor_inicial>
</resource>
<resource>
...
<identificador>15303254000011978</identificador>
<uasg>153032</uasg>
<modalidade_licitacao>03</modalidade_licitacao>
<numero_licitacao>000011978</numero_licitacao>
<tipo_contrato>54</tipo_contrato>
<licitacao_associada>15303203000011978</licitacao_associada>
<origem_licitacao>SISPP</origem_licitacao>
<numero>000011978</numero>
<objeto>Concessão de prédio no Campus da UFLA</objeto>
<numero_aditivo>3</numero_aditivo>
<numero_processo>0001/78</numero_processo>
<cnpj_contratada>20372967000101</cnpj_contratada>
<data_assinatura>1978-06-30</data_assinatura>
<fundamento_legal>Lei 8.666 Modalidade
Concorrência</fundamento_legal>
<data_inicio_vigencia>1978-06-30</data_inicio_vigencia>
<data_termino_vigencia>1979-12-31</data_termino_vigencia>
<valor_inicial>111132.00</valor_inicial>
</resource>
....
```

Disponível em: http://compras.dados.gov.br/contratos/v1/contratos.xml

Neste trecho do documento acima é possível constatar que elementos de uma mesma categoria possuem o mesmo conjunto de tags disponíveis para consulta, o que facilita na comparação entre as preferências expressas pelo usuário e as tags existentes em cada objeto. Observe o trecho a seguir, no qual a padronização citada não foi bem empregada.

Trecho Dado Aberto Governamental com as tags bem padronizadas

```
...
<habilitacao_area_atuacao href="http://api.convenios.gov.br/sicon
v/id/habilitacao_area_atuacao/1" id="1">
<id>1</id>
<subarea>
<SubAreaAtuacaoProponente href="http://api.convenios.gov.br/sicon
v/id/subarea_atuacao_proponente/25"/>
</subarea>
<proponente>
<Proponente href="http://api.convenios.gov.br/siconv/id/proponent
e/3637022000155">APB ASSOCIACAO POSITIVA DE BRASILIA</Proponente>
</proponente>
<orgao>
<Orgao href="http://api.convenios.gov.br/siconv/id/orgao/20408">F
UNDACAO CULTURAL PALMARES</Orgao>
```

```
</orgao>
<pessoa_responsavel>
<PessoaResponsavel href="http://api.convenios.gov.br/siconv/id/pe
ssoa_responsavel/CC9D4FF2EEBD607DD82F849E878C89CAF5E1C359">SISTEM
A</PessoaResponsavel>
</pessoa_responsavel>
<cpf_responsavel>***003552**</cpf_responsavel>
<situacao>NAO_APROVADA</situacao>
<data_inicio>2015-01-03</data_inicio>
</habilitacao_area_atuacao>
<habilitacao_area_atuacao href="http://api.convenios.gov.br/sicon
v/id/habilitacao_area_atuacao/2" id="2">
<id>2</id>
<subarea>
<SubAreaAtuacaoProponente href="http://api.convenios.gov.br/sicon
v/id/subarea_atuacao_proponente/26"/>
</subarea>
<proponente>
<Proponente href="http://api.convenios.gov.br/siconv/id/proponent
e/3637022000155">APB ASSOCIACAO POSITIVA DE BRASILIA</Proponente>
</proponente>
<orgao>
<Orgao href="http://api.convenios.gov.br/siconv/id/orgao/20411">I
NSTITUTO DO PATRIMONIO HIST. E ART. NACIONAL</Orgao>
</orgao>
<pessoa_responsavel>
<PessoaResponsavel href="http://api.convenios.gov.br/siconv/id/pe
ssoa_responsavel/F06BF5BBF79CF01417C8B639D5D419A69AC413AC">NAO
INFORMADO</PessoaResponsavel>
</pessoa_responsavel>
<cpf_responsavel>***999999**</cpf_responsavel>
<situacao>NAO_APROVADA</situacao>
```

Não existe a tag <data_inicio>

```
....
<habilitacao_area_atuacao href="http://api.convenios.gov.br/sicon
v/id/habilitacao_area_atuacao/4" id="4">
<id>4</id>
<subarea>
<SubAreaAtuacaoProponente href="http://api.convenios.gov.br/sicon
v/id/subarea_atuacao_proponente/81"/>
</subarea>
<proponente>
<Proponente href="http://api.convenios.gov.br/siconv/id/proponent
e/3637022000155">APB ASSOCIACAO POSITIVA DE BRASILIA</Proponente>
</proponente>
<orgao>
<Orgao href="http://api.convenios.gov.br/siconv/id/orgao/20408">F
UNDACAO CULTURAL PALMARES</Orgao>
</orgao>
<pessoa_responsavel>
<PessoaResponsavel href="http://api.convenios.gov.br/siconv/id/pe
ssoa_responsavel/69AA21548AFF23B69988D7004C962AE191BAB94D">LEILA
CALACA DA SILVA</PessoaResponsavel>
</pessoa_responsavel>
<cpf_responsavel>***521371**</cpf_responsavel>
<situacao>APROVADA</situacao>
```

```
<data_inicio>2015-01-20</data_inicio>
<data_vencimento>2015-12-31</data_vencimento>
</habilitacao_area_atuacao>
…
```

Disponível em http://api.convenios.gov.br/siconv/v1/consulta/habilitacoes_area_atuacao.xml

No caso em que o documento não contempla elementos de uma mesma categoria com as mesmas tags, a função não consegue estabelecer o nível de comparação entre esses elementos. Por exemplo, levando em consideração o trecho acima se o usuário expressou que prefere informações sobre as habilitações com data de aprovação após 2015-01-20, o sistema de consulta não consegue estabelecer ordem de preferência entre as habilitações que não possuem a tag <data_inicio>.

Em resumo, caso sistema o precise analisar a ordem de preferência entre dois objetos e um deles não possuir o mesmo conjunto de tags, o sistema não consegue identificar qual o resultado esperado pelo usuário porque não consegue estabelecer qual objeto é o preferido.

Diante destes aspectos, foi constatado que o ponto falho do estudo realizado com os dados abertos reais está na falta de padronização das tags utilizadas nos dados disponibilizados. Em alguns dos documentos onde ocorreram os erros, havia informações duplicadas ou até mesmo a falta da informação desejada. Tal fato explica o motivo da taxa de acerto das consultas nos documentos fictícios ser expressivamente maior do que quando realizada em dados reais.

5.3 Considerações Finais

O estudo de caso é fundamental para verificar a validade de uma proposta e nesta pesquisa se optou por dividir esta avaliação para facilitar a execução e análise dos testes. Diante dos resultados apresentados, tal escolha se mostrou ter sido a mais adequada tendo em vista que tal divisão facilitou a percepção de uma falha que se mostrou fator limitante da pesquisa.

Destaca-se que o sistema XQPref ainda necessita de maior avaliação antes de ser disponibilizada, tendo em vista que o seu foco são para consultas em grandes quantidades de informações. Tal avaliação deveria ser conduzida através da realização de testes automatizados, ficando resguardada como trabalho futuro.

No entanto, diante da avaliação da XQuery-Pref realizada neste Capítulo, é importante destacar a relevância deste trabalho na área de pesquisa de personalização de

consultas, especialmente em documentos XML. Com isso, o quadro abaixo ressalta o comparativo entre as contribuições desta dissertação e os trabalhos relacionados apresentados no Capítulo 3.

Quadro 5.1 Comparativo entre as contribuições desta pesquisa e os trabalhos relacionados.

Trabalhos	**Suporte a Personalização**	**Pref. Condicionais**	**XML**	**Transparência**
Koutrika[2010]	√			
Yan et al.[2001]	√		√	
Gomes[2002]	√		√	
Ribeiro[2008] e Pereira [2011]	√	√		
Medeiros e Soares [2015]	√	√	√	√

Com o estudo de caso foi possível verificar que apesar de ainda existir alguns pontos para serem esclarecidos, os resultados retornados estão respeitando as preferências expressas pelo usuário, possibilitando, que mesmo usuários que não conheçam os comandos da linguagem XQuery, realizem as consultas personalizadas de modo otimizado e transparente.

Uma limitação desta pesquisa, detectada através da avaliação experimental, foi a falta de padronização nos dados disponibilizados pelo governo federal, onde impossibilitou em alguns momentos que a XQuery-Pref identificasse a tag que estava sendo procurada ou que estabelecesse a ordem de preferência expressa pelo usuário. Sendo assim, mais uma abordagem que se propõe para o sequenciamento deste trabalho poderia ser uma proposta de padronização dos dados abertos governamentais ou uma forma da XQuery-Pref tratar esta falta de padronização.

Capítulo 6

Conclusão

Este trabalho apresentou a XQuery-Pref, uma extensão da linguagem de consulta XQuery, desenvolvida para viabilizar a personalização de consultas, com suportes a preferências condicionais em documentos XML, necessidade identificada a partir da sobrecarga de informações em domínios que utilizam este formato. Para isto, a XQuery-Pref usa um modelo de preferência baseado no conjunto de regras condicionais proposto por Wilson [2004] e do formalismo CP-net [Boutilier et al. 2004], para induzir uma ordem de preferência sobre os elementos de um documento XML.

Para tanto, foi realizado um estudo em torno da personalização de consultas, bem como das técnicas para representação e modelagem das preferências. Através da literatura estudada percebe-se que existem poucas ferramentas e técnicas que suprem essa necessidade de realizar consulta em dados abertos governamentais, e as que existem são para consultas específicas a determinado conjunto de dados. Através da literatura também foi possível concluir que a personalização de consultas através das preferências condicionais é uma solução eficiente e que pode ser usada para suprir tal necessidade.

Outra contribuição apresentada como resultado da pesquisa é o sistema XQPref, que é responsável pela elicitação de preferências dinâmicas e pelo processamento destas consultas personalizadas. Tal proposta tem como objetivo suprir a necessidade de um mecanismo de consulta personalizada com suporte a preferências condicionais em base de dados XML, tornando transparente para o usuário a execução de consultas escritas na linguagem XQuery-Pref.

A importância do XQPref para o presente trabalho destaca-se também pela necessidade de se realizar consultas testes utilizando a extensão proposta através da

linguagem XQuery-Pref, para verificar a consistência e acurácia da pesquisa através da avaliação experimental.

Através da avaliação realizada com o estudo de caso, também foi possível registrar que os resultados retornados estão respeitando as preferências expressas pelo usuário e, de fato, possibilita que mesmo usuários que não conheçam os comandos da linguagem XQuery, realizem as consultas personalizadas de modo otimizado e transparente.

Para melhor compreensão, o Capítulo foi dividido em três seções: Contribuições, Limitações da Pesquisa e Trabalhos Futuros.

6.1 Contribuições

Em resumo, o trabalho possui como contribuição:

- Estado da arte em duas linhas de pesquisas: Personalização de consultas em documentos XML através das funções extensíveis e personalização de consultas a dados abertos governamentais;
- XQuery-Pref, uma extensão a linguagem de consulta XQuery desenvolvida para viabilizar a personalização de consultas, com suportes a preferências condicionais em documentos XML;
- O sistema XQPref, capaz de elicitar preferências dinâmicas e processar consultas personalizadas;
- Publicações nas linhas de pesquisa envolvidas no trabalho:
    - MEDEIROS, A. F. ; SOARES, V. G. . XQPref: Um Sistema para Personalização de Consultas a Dados XML. In: 4ª Semana de Informática da Universidade Federal de Sergipe, 2014, Itabaiana - SE. Semana Global do Empreendedorismo, 2014. p. 5-8.
    - A submeter: A Proposal for Customizing Queries on XML documents based on Conditional Preferences.
    - A submeter: XQuery-Pref: Extensão de uma Linguagem de Consulta para Documentos XML para personalização de Consultas em Documentos XML.

Em resumo, a principal contribuição deste trabalho é, portanto, reduzir o esforço do usuário na busca por conteúdo relevante em bases de dados XML, para que qualquer interessado possa, ao processar livremente os dados governamentais, criar conteúdo a partir da reutilização dos dados. É interessante ressaltar também que, caso o usuário queira realizar

consultas em algum dado aberto governamental e o documento não esteja disponível em formato XML, o usuário pode solicitar a liberação do documento neste formato através do site http://dados.gov.br/.

6.2 Limitações da Pesquisa Contribuições

Um fator limitante nesta pesquisa foi o tempo, que impossibilitou o desenvolvimento mais aprimorado do XQPref e a realização de testes automatizados desta ferramenta, que pudessem avaliar melhor a sua utilização levando em consideração questões como confiabilidade, usabilidade e segurança.

Neste trabalho a execução dos testes foi realizada a partir da divisão dos testes em dois grupos: dados governamentais reais e dados sintéticos ou fictícios. Tal divisão se mostrou fundamental diante dos resultados apresentados, onde foi possível observar uma falha que se mostrou fator limitante da pesquisa: A heterogeneidade inerente aos dados abertos.

A heterogeneidade embora não gere impacto negativo no retorno das consultas, pode atrapalhar a consulta do usuário ao deixar algum conteúdo sem ser retornado pelo fato da tag não existir, ou estar com nomes distintos no decorrer do documento.

6.3 Trabalhos Futuros

A falta de padronização nos dados disponibilizados pelo governo federal, a qual impossibilitou em alguns momentos que a XQuery-Pref identificasse a tag que estava sendo procurada ou que fosse estabelecida uma ordem de preferência, além de ter sido uma limitação do trabalho surge também como ideia propulsora para sugestão de trabalhos futuros.

Sendo assim, uma das sugestões para trabalhos futuros seria proposta de padronização dos dados abertos governamentais ou uma forma da XQuery-Pref tratar esta falta de padronização.

Destaca-se por fim, que observando os objetivos propostos no início do trabalho e os confrontando com os resultados obtidos percebe-se que apesar das limitações encontradas no decorrer da pesquisa e já listadas acima, a presente pesquisa cumpriu o proposto e ainda abre espaço para novos estudos em duas linhas de pesquisas fundamentais: Personalização de consultas em documentos XML através das funções extensíveis e personalização de consultas a dados abertos governamentais.

Referências

[Aeroportos Brasil, 2012] **Aeroportos Brasil.** 2012. Disponível em: http://ison.stratebi.es/aerobrasil/. Acesso em Jun. 2014.

[Agrawal e Wimmers, 2000] Agrawal, R., Wimmers, E. A.; ***Framework for Expressing and Combining Preferences***. Proc. Of the ACM Int'l Conf. on Management of Data. 2000.

[Amo et al. 2012] Amo, S. A. ; Ferneda, T. M. ; Cattelan, R. G. ; Dias, V. V. S. ; Ferreira, H. M. N.; ***Contextual Preference Repositories for Personalized Query Answering.*** In: XXVII Simpósio Brasileiro de Banco de Dados, 2012, São Paulo. Anais do XXVII Simpósio Brasileiro de Banco de Dados. São Paulo, 2012. p. 17-24.

[Boutilier et al. 2004] Boutilier , C., Brafman, R. I., Domshlak, C., Hoos, H. H., Poole, D., ***CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements.*** Journal of Articial Intelligence Research 21, 2004.

[Brafman et al. 2006] Brafman, R. I., Domshlak, C., Shimony, S. E., e Silver, Y. (2006). ***Preferences over Sets.*** In Proceedings of the 21st National Conference on Arti cial Intelligence (AAAI), pp. 71-80, Menlo Park, California. AAAI Press.

[Brasil, 2011] Brasil. **Lei n. 12.527**. Brasília: Câmara dos Deputados, 2011.

[Bray et al. 2004] Bray, T; Paoli, J; Sperberg-MCqueen, C. M; Yergeau, F. ***Extensible Markup Language (XML) 1.0.*** 2004. Disponível em: http://www.w3.org/TR/2004/REC-xml-20040204/. Acesso em Jan.2014.

[Cartilha, 2011] **Cartilha técnica para publicação de dados abertos no Brasil v1.0.** [2011]. Disponível em: <http://wiki.gtinda.ibge.gov.br/GetFile.aspx?Page=Tecnologia&File=Cartilha%20T%C3%A9cnica%20para%20Publica%C3%A7%C3%A3o%20de%20Dados%20Abertos%20no%20Brasil%20v1.pdf>. Acesso em Mar.2014.

[Chamberlin et al. 2001] Chamberlin, D.; Clark, J.; Florescu, D.; Robie, J.; Siméon, J.; Stefanescu, M. ***XQuery 1.0: An Query Language for XML: W3C Working Draft***. 2001. Disponível em: <http://www.w3.org/ TR/xquery>. Acesso em Jan.2014.

[Chomicki, 2003] Chomicki, J; ***Preference Formulas in Relational Queries***. ACM TODS, 28(4), 427–466. 2003.

[Comai et al. 2001] Comai S., Damiani E., e Fraternali P. ***Computing Graphical Queries over XML Data***. ACM Transactions on Information Systems, Vol. 19, nro. 4, Outubro 2001, páginas 371-430.

[Deitel et al. 2003] Deitel, H. M. et al. **XML: como programar**. Tradução Luiz Augusto Salgado e Edson Furmankiewicz. Porto Alegre: Bookman, 2003.

[Deutsch et al. 1998] Deutsch, A.; Fernandez, M.; Florescu, D.; Levy, A.; Suciu, D. ***XML-QL: A Query Language for XML.*** Submission to the World Wide Web Consortium. 1998. Disponível em: <http://w3.org/TR/1998/NOTE-xml-ql-19980819>. Acesso em Fev.2014.

[Diniz, 2010] Diniz, V. **Como conseguir dados governamentais abertos.** In: CONGRESSO CONSAD DE GESTÃO PÚBLICA, III, Brasília, 2010.

[Eaves, 2009] Eaves, D. ***The three laws of open government***. Disponível em: http://eaves.ca/. Acesso em Mar. 2014.

[Elmasri e Navathe, 2005] Elmasri, Ramez; Navathe, Shamkant B. **Sistema de Banco de Dados.** Revisor técnico Luiz Ricardo de Figueiredo, 4ª ed. São Paulo: Pearson Addison Wesley, 2005.

[Gomes, 2002] Gomes, C. H. P. **Extensão de uma Linguagem de Consulta para Documentos XML com Características de Tempo e de Versão.** 2002. Dissertação - Universidade Federal do Rio Grande do Sul, UFRGS, Brasil. 2002.

[Graves, 2003] Graves, Mark. **Projeto de Banco de Dados com XML**; tradução Aldair José Coelho Corrêa da Silva; revisão técnica Marcos Jorge. ___1. Ed. São Paulo: Pearson Education do Brasil, 2003.

[Harold, 1999] Harold, E. R. ***The XML Bible***. IDG Books, 2 edition, 1999.

[Jung et al. 2005] Jung, S. Y., Hong, J., Kim, T., ***A Statistical Model for User Preference.*** IEEE Transactions on Knowledge and Data Engineering, VOL. 17, NO. 6, 2005.

[Kießling , 2002] Kießling, W. (2002). ***Foundations of preferences in database systems.*** Proc. Of the 28th Intl. VLDB Conf.

[Koutrika e Ioannidis, 2005] Koutrika, G., Ioannidis, Y. (2005). ***Personalized Queries under a Generalized Preference Model***. Proc. of 21st ICDE, 841-852, 5-8 April 2005, Tokyo, Japan.

[Koutrika, 2010] Koutrika, G. (2010) "***Query Personalization based on User Preferences***". V. 35, Abril, New York, USA.

[OpenGovData, 2007] Open Gov Data. (2007) ***Eight principles of open government data***. Dezembro de 2007. Disponível em:www.opengovdata.org/home/8principles. Acesso em Mar.2014.

[Pereira, 2011] Pereira, F. S. F. **CPrefSQL: Uma linguagem de consultas com suporte a preferências condicionais - Teoria e Implementação.** 2011. Dissertação - Universidade Federal de Uberlândia. Uberlândia - MG, 2011.

[Rambhia, 2002] Rambhia, A. M. ***XML Distributed Systems Design***. SMS, USA, 2002.

[Reputação S.A, 2014] **Reputação S.A.** 2014. Disponível em: http://reputacao-sa.org/. Acesso em Jun. 2014.

[Ribeiro, 2008] Ribeiro, M. R. **Linguagens de consulta para banco de dados com suporte a preferências condicionais.** 2008. Dissertação - Universidade Federal de Uberlândia. Uberlândia - MG, 2008.

[Robie, 1999] Robie, J. ***XQL (XML Query Language): W3C Recommendation.*** 1999. Disponível em: <http://www.ibiblio.org/xql/xql-proposal.html>. Acesso em Fev.2014.

[Santos et al. 2012] Santos, F. G. ; Pinheiro, R. ; Braganholo, V.; **Processamento de Consultas XML usando Máquinas de Inferência.** In: Simpósio Brasileiro de Banco de Dados (SBBD), 2012, São Paulo, SP. Simpósio Brasileiro de Banco de Dados (SBBD). Porto Alegre, RS: SBC, 2012. p. 129-136.

[Santos, 2007] Santos, Miriam Oliveira; **Armazenamento e Recuperação de documentos XML Heterogêneos: Aplicando técnicas de KDD para apoiar o projeto físico em SGBDs XML Nativos.** Rio de Janeiro, 2007. Disponível em: http://recreio.de9.ime.eb.br/dissertacoes/2007-Miriam_Santos.pdf. Acesso em Jan. 2014.

[Silva et al. 2008] Silva, Maria Estela Vieira da ; Borges, Eduardo Nunes ; Galante, Renata de Matos; **XSimilarity : Uma Ferramenta para Consultas por Similaridade embutidas na Linguagem XQuery**. In: IV Escola Regional de Bancos de Dados - ERBD, 2008, Florianópolis. Anais da IV Escola Regional de Bancos de Dados, 2008. p. 148-157.

[Silva, 2010] Silva, D. B. **Transparência na esfera pública interconectada.** 2010. (Dissertação de Mestrado), Faculdade Cásper Líbero, São Paulo.

[Thompson et al. 2004] Thompson, Henry S.; Beech, David; Maloney, Murray; Mendelsohn, Noah; ***XML Schema Part 1: Structures Second Edition***, World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, October 2004.

[Vaz et al. 2010] Vaz, J. C., Ribeiro, M. M., Matheus, R.; (2011) **Dados governamentais abertos e seus impactos sobre os conceitos e práticas de transparência no Brasil.** In: Cadernos PPG-AU/UFBA, v. 9, p. 45-62 (2010). Disponível em: http://goo.gl/Nf0t7N. Acesso em Mar.2014.

[W3C, 2010] World Wide Web Consortium. ***XML Query Requirements: W3C Working Draft.*** 14 December 2010. Disponível em: <http://www.w3.org/TR/xquery/>. Acesso em Fev.2014.

[Walmsley, 2007] Walmsley, P. (2007). ***XQuery. O'Reilly Media***, Inc.

[Wilson, 2004] Wilson, N. (2004). ***Extending CP-Nets with Stronger Conditional Preference Statements.*** In 19th National Conference on Arti cial Intelligence (AAAI), pp.735-741, San Jose, California, USA.

[Yan et al, 2011] Yan, Wei, Li Yan, and Z. M. Ma***. "Automated Ranking of Relaxing Query Results Based on XML Structure and Content Preferences."*** International Journal of Systems and Service-Oriented Engineering (IJSSOE) 2.1 (2011): 21-39.

Apêndice A

Trecho de Código XML utilizado para Demonstração de Exemplos na Formalização do Problema

Código 1: Código XML

```
...
<modalidade_educacao>
<educacao_especial>
<regiao>
<nordeste>
<dados_estatisticos>
<evasao>5.3 </evasao>
<aprovacao>9.1</aprovacao>
<reprovacao>0.9</reprovacao>
</dados_estatisticos>
<dados_orcamentais>
<empenho_anual>R$ 1.500.000,00 </empenho_anual>
<alimentacao>R$ 900.000,00 </alimentacao>
<material>R$ 7.000.000,00 </material>
</dados_orcamentais>
</nordeste>
<sudeste>
<dados_estatisticos>
<evasao>6.1 </evasao>
<aprovacao>8.7</aprovacao>
<reprovacao>1.3</reprovacao>
</dados_estatisticos>
<dados_orcamentais>
<empenho_anual>R$ 1.300.000,00 </empenho_anual>
<alimentacao>R$ 850.000,00 </alimentacao>
<material>R$ 7.200.000,00 </material>
</dados_orcamentais>
</sudeste>
<sul>
<dados_estatisticos>
```

```
31    <evasao>4.5 </evasao>
32    <aprovacao>8.5</aprovacao>
33    <reprovacao>1.5</reprovacao>
34    </dados_estatisticos>
35    <dados_orcamentais>
36    <empenho_anual>R$ 1.350.000,00 </empenho_anual>
37    <alimentacao>R$ 890.000,00 </alimentacao>
38    <material>R$ 7.500.000,00 </material>
39    </dados_orcamentais>
40    </sul>
41    <norte>
42    <dados_estatisticos>
43    <evasao>3.2 </evasao>
44    <aprovacao>8.7</aprovacao>
45    <reprovacao>1.3</reprovacao>
46    </dados_estatisticos>
47    <dados_orcamentais>
48    <empenho_anual>R$ 1.400.000,00 </empenho_anual>
49    <alimentacao>R$ 800.000,00 </alimentacao>
50    <material>R$ 7.600.000,00 </material>
51    </dados_orcamentais>
52    </norte>
53    </regiao>
54    </educacao_especial>
55    <educacao_a_distancia>
56    <regiao>
57    <nordeste>
58    <dados_estatisticos>
59    <evasao>4.2 </evasao>
60    <aprovacao>8.5</aprovacao>
61    <reprovacao>1.5</reprovacao>
62    </dados_estatisticos>
63    <dados_orcamentais>
64    <empenho_anual>R$ 2.500.000,00 </empenho_anual>
65    <material>R$ 7.000.000,00 </material>
66    </dados_orcamentais>
67    </nordeste>
68    <sudeste>
69    <dados_estatisticos>
70    <evasao>3.3 </evasao>
71    <aprovacao>9.0</aprovacao>
72    <reprovacao>1.0</reprovacao>
73    </dados_estatisticos>
74    <dados_orcamentais>
75    <empenho_anual>R$ 2.300.000,00 </empenho_anual>
76    <material>R$ 8.200.000,00 </material>
77    </dados_orcamentais>
78    </sudeste>
```

```
79   <sul>
80   <dados_estatisticos>
81   <evasao>2.5 </evasao>
82   <aprovacao>9.5</aprovacao>
83   <reprovacao>0.5</reprovacao>
84   </dados_estatisticos>
85   <dados_orcamentais>
86   <empenho_anual>R$ 2.150.000,00 </empenho_anual>
87   <material>R$ 9.500.000,00 </material>
88   </dados_orcamentais>
89   </sul>
90   <norte>
91   <dados_estatisticos>
92   <evasao>2.2 </evasao>
93   <aprovacao>8.2</aprovacao>
94   <reprovacao>1.8</reprovacao>
95   </dados_estatisticos>
96   <dados_orcamentais>
97   <empenho_anual>R$ 2.500.000,00 </empenho_anual>
98   <material>R$ 9.600.000,00 </material>
99   </dados_orcamentais>
100  </norte>
101  </regiao>
102  </educacao_a_distancia>
103  <educacao_infantil>
104  <regiao>
105  <nordeste>
106  <dados_estatisticos>
107  <evasao>3.7 </evasao>
108  <aprovacao>8.7</aprovacao>
109  <reprovacao>1.3</reprovacao>
110  </dados_estatisticos>
111  <dados_orcamentais>
112  <empenho_anual>R$ 7.500.000,00 </empenho_anual>
113  <alimentacao>R$ 900.000,00 </alimentacao>
114  <material>R$ 8.000.000,00 </material>
115  </dados_orcamentais>
116  </nordeste>
117  <sudeste>
118  <dados_estatisticos>
119  <evasao>4.4 </evasao>
120  <aprovacao>8.7</aprovacao>
121  <reprovacao>1.3</reprovacao>
122  </dados_estatisticos>
123  <dados_orcamentais>
124  <empenho_anual>R$ 7.300.000,00 </empenho_anual>
125  <alimentacao>R$ 850.000,00 </alimentacao>
126  <material>R$ 7.200.000,00 </material>
```

```
</dados_orcamentais>
</sudeste>
<sul>
<dados_estatisticos>
<evasao>2.5 </evasao>
<aprovacao>8.7</aprovacao>
<reprovacao>1.3</reprovacao>
</dados_estatisticos>
<dados_orcamentais>
<empenho_anual>R$ 7.350.000,00 </empenho_anual>
<alimentacao>R$ 890.000,00 </alimentacao>
<material>R$ 8.500.000,00 </material>
</dados_orcamentais>
</sul>
<norte>
<dados_estatisticos>
<evasao>2.6 </evasao>
<aprovacao>8.9</aprovacao>
<reprovacao>0.9</reprovacao>
</dados_estatisticos>
<dados_orcamentais>
<empenho_anual>R$ 7.400.000,00 </empenho_anual>
<alimentacao>R$ 800.000,00 </alimentacao>
<material>R$ 6.600.000,00 </material>
</dados_orcamentais>
</norte>
</regiao>
</educacao_infantil>
</modalidade_educacao>
...
```