UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDIMAR MANICA

Suporte a consultas temporais por palavras-chave em documentos XML

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof[a]. Dr. Renata Galante
Orientadora

Porto Alegre, dezembro de 2010

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Manica, Edimar

Suporte a consultas temporais por palavras-chave em documentos XML / Edimar Manica. – Porto Alegre: PPGC da UFRGS, 2010.

90 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2010. Orientadora: Renata Galante.

1. Consulta temporal. 2. Consulta por palavras-chave. 3. XML. I. Galante, Renata. II. Título.

“Descobri como é bom chegar quando se tem paciência.
E para se chegar, onde quer que seja, aprendi que
não é preciso dominar a força, mas a razão.
É preciso, antes de mais nada, querer.”
— AMYR KLINK

AGRADECIMENTOS

Meus sinceros agradecimentos a todos que contribuíram para o desenvolvimento desse trabalho. Em especial, gostaria de agradecer:

À minha orientadora, Profª. Dr. Renata Galante, por todos os valorosos conhecimentos e experiências transmitidos. Agradeço muito pela orientação, pelas oportunidades e pela super atenção que me foram dispensadas ao longo de todo o curso de mestrado, as quais foram de fundamental importância para a realização desse trabalho. Também, agradeço pela sua forma de orientar, que não tem como objetivo gerar um bom *paper*, mas formar um bom aluno, capaz de publicar bons *papers*, realizar boas apresentações, incentivado a participar de conferências e evoluir sempre. Agradeço ainda a confiança, a amizade, os conselhos, as viagens realizadas e pela oportunidade de cursar o mestrado.

À Profª. Dr. Carina F. Dorneles, pela colaboração nos artigos, especialmente pelos fins de semana que passou revisando os artigos. Agradeço, também, a confiança e os conselhos que me permitiram chegar ao mestrado e as direções, junto com a professora Renata, que me guiaram neste trabalho. Finalmente, agradeço por ser a pessoa que eu sei que posso contar quando surge qualquer dúvida.

Ao Prof. Msc. Cristiano Roberto Cervi pelo incentivo e motivação para ingressar no mestrado, bem como por todo auxílio fornecido no período de iniciação científica.

À UFRGS e ao Instituto de Informática pela infra-estrutura disponibilizada. Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao Instituto Nacional de Ciência e Tecnologia para a Web (InWeb) pelo apoio financeiro.

Aos professores do PPGC pelos ensinamentos transmitidos nas disciplinas por eles ministradas e aos funcionários sempre solícitos e prestativos. Em especial, aos professores Carlos Alberto Heuser, José Palazzo e Viviane Moreira pelas idéias e discussões durante a apresentação do PEP e semana acadêmica. Reforço o agradecimento ao professor Palazzo pelo auxílio financeiro dos projetos dos quais é coordenador e ao professor Heuser por ter me dado a oportunidade de prestar consultoria junto com ele e o colega Maurício, na qual consegui aprimorar meus conhecimentos.

Aos professores e alunos de diversas universidades que contribuíram com o trabalho durante as apresentações realizadas em conferências.

Aos membros da banca de defesa dessa dissertação: Profª. Dr. Valéria Cesário Times (UFPE), Prof. Dr. Carlos Alberto Heuser (UFRGS) e Prof. Dr. Leandro Krug Wives (UFRGS); por terem aceitado o convite, pelas importantes sugestões e correções que contribuíram no aprimoramento do texto final deste trabalho e pelo incentivo para o seguimento da pesquisa realizada.

Aos meus colegas de laboratório, que não citarei nomes para não correr o risco de esquecer alguém. Eles foram ótimos companheiros de pesquisa, parceiros para o vôlei, participantes nas minhas prévias e solícitos na realização de experimentos. Em especial,

ao Eduardo Borges, Guilherme Dal Bianco, Isabel Volpe e Giseli Lopes a quem recorri em vários momentos para me auxiliar em experimentos, escritas, apresentações, conflitos pessoais, etc.

Aos meus pais que muitas vezes abriram mão de seus sonhos para que eu realizasse o meu, me dando apoio incondicional nas minhas decisões. A eles, que tantas vezes de longe estiveram tão perto e me ajudaram muito mais do que imaginam. A distância que nos separou só me fez ver ainda mais que posso sempre contar com eles. Muito obrigado.

Por fim, agradeço imensamente a Deus por ter me dado forças em mais uma etapa de minha vida e por ter me propiciado mais esta oportunidade de crescimento pessoal e profissional. Em especial, agradeço a Deus por sempre ter colocado as pessoas certas ao meu lado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ACL	Meeting of the Association for Computational Linguistics
ACM	Association for Computing Machinery
CIKM	Conference on Information and Knowledge Management
CVLCA	Compact Valuable Lowest Commom Ancestor
DBLP	Digital Bibliography & Library Project
DTD	Document Type Descriptor
ECIR	European Conference on Information Retrieval Research
EDBT	International Conference on Extending Database Technology
ERBD	Escola Regional de Banco de Dados
ICWE	International Conference on Web Engineering
IEEE	Institute of Electrical and Electronics Engineers
IITA	International Symposium on Intelligent Information Technology Application
JIDM	Journal of Information and Data Management
SAC	Symposium on Applied Computing
SBBD	Simpósio Brasileiro de Banco de Dados
SIGIR	Special Interest Group on Information Retrieval
SQL	Structured Query Language
LCA	Lowest Commom Ancestor
MLCA	Meaningfully Lowest Commom Ancestor
POS	Part-of-Speech
PTR	Palavras Temporais de Referência
SIGMOD	Special Interest Group on Management Of Data
SLCA	Smallest Lowest Commom Ancestor
TERN	Temporal Expressions Recognition and Normalization
TISE	Time-Inspired Search Engine
TKC	Temporal Keyword Classification

TPI	Two Phase Interpection
T-Yago	Timely Yago
UFRGS	Universidade Federal do Rio Grande do Sul
VLDB	Very Large Data Base
WTDBD	Workshop de Teses e Dissertações de Banco de Dados
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Consultas por palavras-chave permitem o acesso fácil a dados XML, uma vez que não exigem que o usuário aprenda uma linguagem de consulta estruturada nem estude possíveis esquemas de dados complexos. Com isso, vários motores de busca XML foram propostos para permitir a extração de fragmentos XML relevantes para consultas por palavras-chave. No entanto, esses motores de busca tratam as expressões temporais da mesma forma que qualquer outra palavra-chave. Essa abordagem ocasiona inúmeros problemas, como por exemplo, considerar como casamentos para uma expressão temporal nodos do domínio preço ou código.

Este trabalho descreve TPI (*Two Phase Interception*), uma abordagem que permite o suporte a consultas temporais por palavras-chave em documentos XML orientados a dados. O suporte a consultas temporais é realizado através de uma camada adicional de software que executa duas interceptações no processamento de consultas, realizado por um motor de busca XML. Esta camada adicional de software é responsável pelo tratamento adequado das informações temporais presentes na consulta e no conteúdo dos documentos XML. O trabalho ainda especifica TKC (*Temporal Keyword Classification*), uma classificação de consultas temporais que serve de guia para qualquer mecanismo de consulta por palavras-chave, inclusive TPI. São apresentados os algoritmos de mapeamento das diferentes formas de predicados temporais por palavras-chave, especificadas em TKC, para expressões relacionais a fim de orientar a implementação do processamento das consultas temporais. É proposto um índice temporal e definidas estratégias para identificação de caminhos temporais, desambiguação de formatos de valores temporais, identificação de datas representadas por vários elementos e identificação de intervalos temporais. São demonstrados experimentos que comparam a qualidade, o tempo de processamento e a escalabilidade de um motor de busca XML com e sem a utilização de TPI. A principal contribuição desse trabalho é melhorar significativamente a qualidade dos resultados de consultas temporais por palavras-chave em documentos XML.

Palavras-chave: Consulta temporal, consulta por palavras-chave, XML.

Supporting Temporal Keyword Queries on XML Documents

ABSTRACT

Keyword queries enable users to easily access XML data, since the user does not need to learn a structured query language or study possibly complex data schemas. Therewith, several XML search engines have been proposed to extract relevant XML fragments in response to keyword queries. However, these search engines treat the temporal expressions as any other keyword. This approach may lead to several problems. It could, for example, consider prices and codes as matches to a temporal expression.

This work describes TPI (*Two Phase Interception*), an approach that supports temporal keyword queries on data-centric XML documents. The temporal query support is performed by adding an additional software layer that executes two interceptions in the query processing performed by a XML search engine. This additional software layer is responsible for the adequate treatment of the temporal expressions contained in the query and in the contents of the XML documents. This work also specifies TKC (*Temporal Keyword Classification*), a temporal query classification to be used as guidance for any keyword query mechanism, including TPI. We present the algorithms for mapping different temporal predicates expressed by keywords to relational expressions in order to guide the implementation of the temporal query processing. We propose a temporal index together with strategies to perform temporal path identification, format disambiguation, identification of dates represented by many elements and detection of temporal intervals. This work also reports on experiments which evaluate quality, processing time and scalability of an XML search engine with TPI and without TPI. The main contribution of this work is the significant improvement in the quality of the results of temporal keyword queries on XML documents.

Keywords: temporal query, keyword search, XML.

1 INTRODUÇÃO

O XML tornou-se o padrão para representação de dados Web (LIU; CHEN, 2007). Grande parte dos documentos XML possuem centenas de *megabytes* de dados. O tamanho do documento XML da DBLP[1], por exemplo, supera os 700MB. Com isso, consultas em tais documentos devem retornar apenas fragmentos XML relevantes para a consulta, ou seja, os nodos do documento XML que o usuário deseja como resposta.

Uma das alternativas para realizar consultas em documentos XML são as linguagens estruturadas, como XPath (CLARK; DEROSE, 2010) e XQuery (W3C, 2010). Essas linguagens são poderosas, uma vez que podem carregar significado semântico complexo e, portanto, recuperar precisamente os fragmentos XML desejados (LIU; CHEN, 2010). Porém, essas linguagens são de difícil compreensão por usuários que não são da área de banco de dados e requerem conhecimento do esquema, que pode ser aninhado e às vezes complexo (FENG et al., 2010).

Outra alternativa para consultar documentos XML é a utilização de consultas por palavras-chave. Tradicionalmente, cada documento (página Web, por exemplo) constitui uma *unidade básica de informação*, que é considerada um resultado se ela contém um subconjunto das palavras-chave da consulta (MARKOWETZ; YANG; PAPADIAS, 2009). Aplicando esse tipo de consulta para documentos XML, a *unidade básica de informação* passa a ser o fragmento XML. Consultas por palavras-chave representam uma abordagem útil e intuitiva para usuários leigos. Isso pode ser concluído a partir do sucesso dos motores de busca Web, que utilizam essa abordagem (FARFÁN et al., 2009). Dentro desse contexto, diversas propostas de motores de busca XML, como, por exemplo, XSeek (LIU; CHEN, 2007), XKSearch (XU; PAPAKONSTANTINOU, 2005), XSEarch (COHEN et al., 2003) e XRank (GUO et al., 2003), foram propostas para processamento de consulta por palavras-chave em documentos XML. A principal característica dessas propostas é a extração de fragmentos XML relevantes para a consulta.

A maioria das aplicações possuem informações temporais. As informações temporais são representadas em documentos XML de várias formas, tais como datas, períodos, entre outros. A DBLP, por exemplo, possui o ano de publicação dos artigos. O currículo Lattes[2] possui inúmeras informações temporais dos pesquisadores, tais como o período de vínculo com determinada instituição e o ano de publicação de seus artigos. O Health Level 7's [3] apresenta um formato para troca de registros médicos eletrônicos baseado em XML. Documentos desse domínio necessitam armazenar várias informações temporais de pacientes, tais como datas de consultas e períodos de internações.

Em documentos XML com informações temporais, os usuários podem estar interes-

[1] http://dblp.uni-trier.de/xml/
[2] http://lattes.cnpq.br/
[3] http://www.hl7.org/

sados apenas em informações referentes a um período de tempo específico e usam expressões temporais na consulta a fim de filtrar os resultados. No entanto, os motores de busca XML tratam essas expressões temporais da mesma forma que as demais palavras-chave da consulta, ou seja, procuram elementos onde seu nome ou valor contenha uma das palavras-chave da consulta. Isso acarreta inúmeros problemas, tais como:

- as expressões temporais são casadas com nodos de qualquer domínio. Por exemplo, um nodo preço ou código pode ser considerado como um casamento para uma expressão temporal;
- não há suporte para operadores temporais. Por exemplo, na consulta "`artigos` **`após`** `2008`", o motor de busca XML procura ocorrências do operador temporal "`após`" no documento XML;
- não é realizada a identificação de períodos. Por exemplo, as linhas 06-07 do `Doc C` (Figura 1.1) contém informação temporal referente a internação de um paciente. Essa informação temporal é representada por um período temporal que inicia em 23/11/2007 e termina em 25/11/2007. No entanto, um motor de busca XML não tem o conhecimento que o paciente estava internado em 24/11/2007, uma vez que não identifica a existência do intervalo temporal;
- não há identificação de datas representadas através de vários elementos. Por exemplo, as linhas 09-11 do `Doc B` (Figura 1.1) constituem uma data que é representada por três elementos (`dia`, `mes` e `ano`). No entanto, um motor de busca XML trata os três elementos como elementos com valores numéricos quaisquer;
- não há desambiguação de formato. Por exemplo, a linha 07 do `Doc A` (Figura 1.1) apresenta uma ambiguidade de formato. Não é possível saber se a data representa o valor 12 de março de 1997 ou 03 de dezembro de 1997. Um motor de busca XML não realiza a desambiguação desse formato.

```
01. <?xml version="1.0"?>
02. <posto>
03.   <cidade>Soledade</cidade>
04.   <vacina>
05.     <nome>Febre Amarela</nome>
06.     <pessoa>Edimar Manica</pessoa>
07.     <data>12/03/1997</data>
08.   </vacina>
09.   <vacina>
10.     <nome>Febre Amarela</nome>
11.     <pessoa>Edimar Manica</pessoa>
12.     <data>28/01/2009</data>
13.   </vacina>
14. </posto>
```

Doc A

```
01. <?xml version="1.0"?>
02. <posto cidade="Marau">
03.   <vacina>
04.     <nome>Febre Amarela</nome>
05.     <pessoa>Edimar Manica
06.     </pessoa>
07.     <idade>25</idade>
08.     <aplicacao>
09.       <dia>12</dia>
10.       <mes>01</mes>
11.       <ano>1998</ano>
12.     </aplicacao>
13. </vacina>
14. </posto>
```

Doc B

```
01. <?xml version="1.0"?>
02. <posto cidade="Passo Fundo">
03.   <internacao>
04.     <paciente>Edimar Manica
05.     </paciente>
06.     <baixa>23/11/2007</baixa>
07.     <alta>25/11/2007</alta>
08.   </internacao>
09. </posto>
```

Doc C

Figura 1.1: Exemplo de documentos XML com dados temporais

O objetivo desse trabalho é especificar uma abordagem para suportar consultas temporais por palavras-chave em documentos XML orientados a dados. Documentos XML orientados a dados (*data-centric*) são documentos XML cujos nomes dos nodos tipicamente representam anotação semântica, por exemplo, documentos XML exportados de banco de dados relacionais. Nos documentos XML orientados a documento (*document-centric*), os nomes dos nodos tipicamente descrevem formatações (LIU; CHEN, 2010).

A abordagem proposta, denominada **TPI** (***Two Phase Interception***), realiza duas interceptações no processamento de consultas, realizado por um motor de busca XML, para adicionar o tratamento adequado da informação temporal presente nas consultas e nos documentos XML. Foram realizados experimentos de qualidade, tempo de processamento e escalabilidade em três bases reais que comparam o motor de busca XML XSeek com e sem a utilização de TPI. Os resultados mostram que TPI aumenta significativamente a qualidade dos resultados e mantém o desempenho geral do sistema, comparando com um motor de busca XML convencional. A principal contribuição de TPI é a definição de uma camada adicional de software que inclui a exploração da informação temporal em motores de busca XML existentes.

O trabalho também especifica **TKC** (***Temporal Keyword Classification***), uma classificação genérica proposta para servir de guia para qualquer mecanismo de consulta temporal por palavras-chave. Essa classificação foi definida a partir de uma análise de consultas Web realizada com o objetivo de verificar os tipos primitivos de representação temporal, as granularidades, os operadores temporais, entre outros aspectos presentes em consultas temporais por palavras-chave. A principal contribuição de TKC é definir os requisitos que um mecanismo de consulta por palavras-chave deve considerar com relação às consultas temporais.

Detalhes do trabalho incluem a definição de algoritmos que orientam a implementação do processamento das consultas temporais definidas em TKC. Um índice temporal é proposto para permitir o acesso rápido e consistente dos dados temporais durante o processamento das consultas. Estratégias para identificação de caminhos temporais, datas formadas por vários elementos e intervalos temporais são definidas.

O restante do texto está organizado da seguinte forma. No Capítulo 2 são apresentados alguns conceitos necessários para a compreensão da classificação TKC e da abordagem TPI. O Capítulo 3 apresenta o contexto no qual a dissertação está inserida, a saber, banco de dados temporais e recuperação de informação em XML. Cada tópico é finalizado com uma análise comparativa entre as propostas. Além disso, são identificadas as principais necessidades e desafios para a realização de consultas temporais por palavras-chave em documentos XML. A análise de consultas temporais realizada nessa dissertação é apresentada no Capítulo 4. O Capítulo 5 propõe TKC, uma classificação de consultas temporais que serve de guia para qualquer mecanismo de consulta por palavras-chave. O Capítulo 6 especifica TPI, uma abordagem para permitir consultas temporais por palavras-chave em documentos XML. A avaliação experimental é apresentada no Capítulo 7, no qual TPI é comparado com um motor de busca convencional em termos de qualidade, tempo de processamento e escalabilidade. O Capítulo 8 revisa as contribuições da dissertação e apresenta possibilidades de trabalhos futuros. O Apêndice A apresenta a especificação completa de TKC. Finalmente, o Apêndice B apresenta o diagrama de classes completo de TKC.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos necessários para a compreensão dessa dissertação. A Seção 2.1 apresenta os conceitos gerais de tempo. A Seção 2.2 aborda os conceitos de tempo específicos da área de recuperação de informação. Finalmente, a Seção 3.4 apresenta conceitos de consulta por palavras-chave em documentos XML. Esses conceitos se relacionam, uma vez que o presente trabalho adiciona o tratamento apropriado da informação temporal à consultas por palavras-chave em documentos XML, o que constitui uma recuperação de informação temporal.

2.1 Conceitos Gerais de Tempo

Os conceitos gerais de tempo, definidos a seguir, foram extraídos do glossário de conceitos de banco de dados temporais (JENSEN et al., 1997), de (EDELWEISS, 1998), de (EDELWEISS; OLIVEIRA, 1994) e de (SNODGRASS; AHN, 1985).

Existem duas formas de representação da `variação temporal`: tempo contínuo e tempo discreto. O `tempo contínuo` é a forma natural de ver o tempo, na qual um dado pode ter um valor diferente em qualquer instante de tempo. O `tempo discreto` baseia-se em uma linha de tempo composta de uma sequência de intervalos temporais consecutivos, que não podem ser decompostos, de idêntica duração. Esses intervalos são denominados `chronons`.

A `granularidade temporal` consiste na duração de um *chronon*. Entretanto, dependendo da aplicação considerada, às vezes é necessário considerar simultaneamente diferentes granularidades. Embora o *chronon* seja único, é possível manipular diferentes granularidades através de funções e operações sobre o *chronon*. No entanto, granularidades menores que a duração do *chronon* definido para a aplicação não podem ser manipuladas. Com isso, a principal consequência de se escolher uma duração de *chronon* para uma aplicação - por exemplo, segundo - é que os eventos que ocorrem dentro do mesmo segundo são considerados eventos simultâneos, embora, na realidade, possam não sê-lo, uma vez que podem ocorrer em milissegundos diferentes.

Tempo `absoluto` consiste de uma informação temporal que define um tempo específico, definido com uma granularidade determinada, associada a um fato. Por exemplo, Fulano nasceu no dia 11/04/2010. Um tempo é `relativo` quando sua validade é relacionada ao momento atual (o salário aumentou **há duas semanas**, por exemplo), ou à validade de outro fato (a loja abriu **dois meses após a abertura do shopping**, por exemplo).

Há quatro `tipos primitivos de representação temporal` principais: instante no tempo, intervalo temporal, elemento temporal e duração temporal.

Um `instante no tempo`, ou simplesmente `instante`, consiste de um ponto

na linha do tempo. Quando utiliza-se tempo contínuo, um instante é o ponto no tempo de duração infinitesimal. Nesse caso, os instantes são isomórficos com números reais, o que significa que entre dois pontos do tempo sempre existe um outro ponto no tempo. No entanto, quando utiliza-se uma representação discreta do tempo, os instantes são isomórficos aos números inteiros ou a um subconjunto desses. Dessa forma, entre dois pontos do tempo consecutivos não existe outro ponto no tempo.

Um `intervalo temporal` é o tempo compreendido entre dois instantes. Um intervalo temporal é caracterizado por dois valores, identificando um intervalo, representando o tempo inicial e final da validade da informação. Elmasri e Navathe (2005) referem-se a um intervalo de tempo como `período de tempo` ou `período temporal`. Dependendo da pertinência ou não dos instantes limites ao intervalo, esse pode ser `aberto` (os dois limites não pertencem ao intervalo), `semiaberto` (apenas um dos limites pertence ao intervalo) ou `fechado` (os dois limites pertencem ao intervalo).

Um `elemento temporal` é uma união finita de intervalos de tempo. Elmasri e Navathe (2005) definem elemento temporal como uma coleção de um ou mais períodos de tempo disjuntos, de tal modo que nenhum dos dois períodos de tempo de um elemento temporal seja diretamente adjacente. Nesse caso, para quaisquer dois períodos de tempo [T1,T2] e [T3, T4], em um elemento temporal, deve-se garantir que:

- [T1, T2] interseção [T3, T4] seja vazio;
- T3 não é o ponto de tempo seguinte a T2, em determinada granularidade;
- T1 não é um ponto de tempo seguinte a T4, em determinada granularidade. Se dois períodos de tempo [T1,T2] e [T3,T4] são adjacentes, eles serão combinados em um único período de tempo [T1, T4]. Essa combinação é chamada de fusão de períodos de tempo.

Uma `duração temporal` também pode ser considerada como tipo primitivo. Durações temporais podem ser basicamente de dois tipos, dependendo do contexto em que são definidas: fixas e variáveis. Uma `duração fixa` independe do contexto de sua definição. Por exemplo, uma hora possui uma duração fixa, pois sempre tem duração de 60 minutos. Já a `duração variável` depende do contexto. Por exemplo, um mês possui duração variável que pode ser de 28 a 31 dias.

A operação de união de intervalos temporais sobrepostos com o objetivo de eliminar duplicatas é denominada *`coalesce`*. A operação *coalesce* é uma operação unária que tem como entrada uma relação temporal e retorna uma relação temporal com o mesmo esquema. Sua finalidade é efetuar um tipo de normalização com relação a uma ou várias dimensões temporais. Isso é alcançado através da compactação de todas as tuplas de valor equivalente em uma única tupla, o que, possivelmente, ampliará o intervalo temporal das respectivas dimensões temporais.

Três diferentes tipos de tempo podem ser identificados em aplicações de banco de dados: (i) `tempo de transação`, tempo no qual o fato é registrado no banco de dados; (ii) `tempo de validade`, tempo durante o qual um fato do banco de dados é verdadeiro na realidade modelada; e (iii) `tempo definido pelo usuário`, tempo que consiste de propriedades temporais definidas explicitamente pelos usuários em um domínio temporal (data, por exemplo) e manipuladas pelos programas de aplicação.

Uma consulta temporal é uma consulta que permite a recuperação de todas as informações armazenadas em um banco de dados (temporal ou não), de modo que seja tirado real proveito do acréscimo da dimensão temporal. Consultas temporais permitem:

- fornecer valores de propriedades cujo domínio é temporal. Por exemplo, fornecer o valor da propriedade que armazena a data de nascimento de uma pessoa;
- se referir a um determinado instante ou intervalo temporal. Por exemplo, qual o valor do salário no dia 01/01/2004;
- recuperar valores com base em condições temporais. Por exemplo, recuperar todos os valores do salário antes do dia 01/01/2004;
- fornecer informações temporais (datas, intervalos). Por exemplo, qual a data em que foi alterado o salário de um funcionário.

Uma consulta temporal apresenta dois componentes ortogonais: um `componente de seleção`, responsável por filtrar os dados; e um `componente de saída`, responsável pela projeção dos resultados.

O componente de seleção geralmente é representado através de uma condição lógica. Quando a condição envolve valores temporais ela é denominada `condição temporal`, `predicado temporal` ou `restrição temporal`. No entanto, nesta dissertação, o termo `restrição temporal` não é utilizado, pois alguns trabalhos empregam esse termo para se referir a restrições de integridade temporal. As restrições de integridade temporal são encontradas no esquema e especificam um predicado que todas as instâncias de dados devem satisfazer. Por exemplo, uma instância cujo domínio é data e o valor do mês é abril não pode ter 31 como valor de dia.

Além dos operadores condicionais das linguagens de consulta convencionais, o contexto temporal exige operadores específicos para tratar valores temporais (antes, depois e durante, por exemplo). Esses operadores definem relações temporais entre dois instantes, dois intervalos ou um intervalo e um instante e são denominados `operadores temporais`. Allen (1983) especificou 13 relações entre intervalos temporais: `after`, `before`, `contains`, `during`, `equal`, `finished by`, `finishes`, `meets`, `met by`, `overlapped by`, `overlaps`, `started by` e `starts`. A Figura 2.1 ilustra essas relações. Nesse trabalho essas relações são referenciadas como `relações de Allen`.

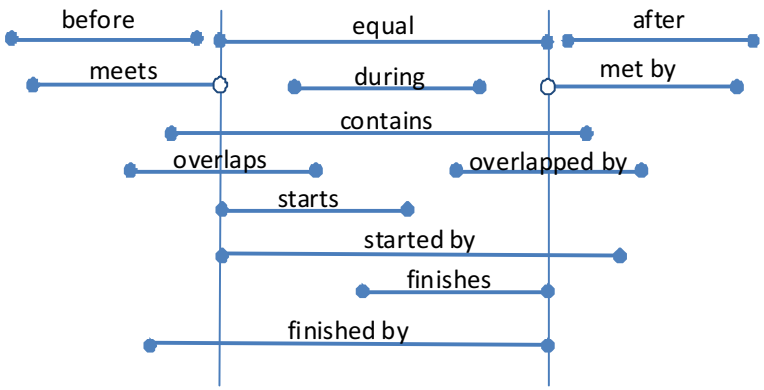


Figura 2.1: Relações de Allen

2.2 Recuperação de Informação Temporal

Recuperação de informação temporal é a recuperação de informação que explora as informações temporais presentes na consulta e nos documentos para melhorar a qualidade dos resultados. Os conceitos de recuperação de informação temporal, definidos a seguir, foram extraídos de (ALONSO; GERTZ; BAEZA-YATES, 2007a).

As informações temporais presentes em documentos texto, tais como: um ponto no tempo, um evento ou um período de tempo, são descritas em um nível conceitual por uma `entidade temporal`. Uma sequência de *tokens* que representa uma instância de uma entidade temporal é denominada `expressão temporal`. Existem três categorias de expressões temporais:

- `Explícitas` - expressões temporais que descrevem diretamente uma entrada em uma linha de tempo, tal como uma data exata ou ano específico. Por exemplo, a expressão "`dezembro de 2004`" ou "`12 de janeiro de 2006`" em um fragmento de texto são expressões temporais explícitas e podem ser mapeadas diretamente para *chronons* em uma linha de tempo. Essas expressões representam tempo absoluto.

- `Implícitas` - são expressões temporais que precisam de conhecimento predefinido (ontologias de tempo, por exemplo) para serem mapeadas para uma entrada em uma linha de tempo. Nome de feriados e eventos específicos são típicos exemplos de expressões temporais implícitas. Por exemplo, a expressão "`Natal de 2005`" precisa ser mapeada para "`25 de dezembro de 2005`". Essas expressões também representam tempo absoluto.

- `Relativas` - são expressões temporais que representam entidades temporais que apenas podem ser mapeadas para uma entrada em uma linha de tempo em referência à uma expressão temporal explícita, implícita ou ainda ao momento em que o texto foi escrito. Por exemplo, a expressão "`ontem`" só pode ser mapeada com base no momento em que o texto foi escrito. Essas expressões representam tempo relativo.

O processo de mapear as expressões temporais para um formato único, representando de forma padronizada um ponto na linha de tempo, é denominado `normalização de expressões temporais`.

Neste trabalho, as consultas temporais foram classificadas em: `consulta de seleção temporal`, nas quais o usuário utiliza expressões temporais a fim de filtrar os resultados da consulta; e `consulta de saída temporal`, nas quais o usuário está interessado em descobrir quando um dado evento ocorreu. Por exemplo, "`Quando o Brasil venceu a Copa do Mundo?`" e "`Quanto tempo durou a Revolução Farroupilha?`".

2.3 Consultas por Palavras-chave em Documentos XML

Uma `consulta por palavras-chave` ou `busca` é uma consulta expressa pelo usuário através de um conjunto de palavras-chave. Tradicionalmente, cada documento (página Web, por exemplo) constitui uma *unidade básica de informação*, que é considerada um resultado se ela contém um subconjunto das palavras-chave da consulta (MARKOWETZ; YANG; PAPADIAS, 2009). Porém, aplicando esse tipo de consulta para XML a *unidade básica de informação* passa a ser o fragmento XML.

Liu e Chen (2007) classificam as palavras-chave de uma consulta em:

- `predicado` - quando a palavra-chave visa filtrar a consulta, correspondendo à cláusula WHERE em XQuery ou SQL;
- `nodos de retorno explícito` - quando a palavra-chave visa especificar um tipo de saída desejável, correspondendo à cláusula RETURN em XQuery ou SELECT em SQL.

As consultas por palavras-chave nas quais todas as palavras-chave são predicados possuem `nodos de retorno implícito`, ou seja, o tipo de saída desejável precisa ser inferido a partir dos predicados. Nesta dissertação, uma palavra-chave que é uma expressão temporal é referenciada como um `predicado temporal`.

Um `motor de busca XML` é responsável por realizar o processamento de consultas por palavras-chave em XML. A abordagem mais utilizada para indexação dos documentos XML pelos motores de busca XML é através da utilização de *`Dewey Labels`* como identificadores de nodos XML. Tatarinov et al. (2002) demonstra que a utilização de *Dewey Labels* como identificadores é uma boa escolha.

O *Dewey Label* de um nodo é formado pelo *Dewey Label* de seu nodo pai, concatenado com um ponto final (“.”), concatenado com a posição relativa do nodo entre seus irmãos. A primeira posição relativa é 0. Uma vez que a raiz não possui nodos pais, nem nodos irmãos, seu *Dewey Label* é 0. Na Figura 2.2, por exemplo, o nodo com *Dewey Label* `0.1.2` é o terceiro filho do nodo `0.1`.

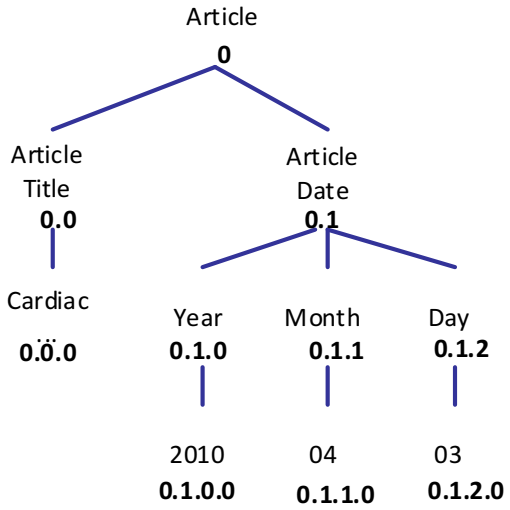


Figura 2.2: Exemplo da utilização de *Dewey Labels* como identificadores de nodos XML

O *Dewey Label* pode ser utilizado para detectar a ordem dos nodos, nodos irmãos e nodos antecessores da seguinte forma:

- O rótulo de início de um nodo *u* aparece antes de um nodo *v* em um documento XML se e somente se *Dewey(u)* for menor que *Dewey(v)* comparando cada componente em ordem.
- Um nodo *u* é um irmão de um nodo *v* se e somente se *Dewey(u)* for diferente de *Dewey(v)* apenas no último componente.

- Um nodo *u* é um antecessor de um nodo *v* se e somente se *Dewey(u)* for um prefixo de *Dewey(v)*.

Duas importantes etapas da execução de um motor de busca XML são: (i) `encontrar os casamentos` para cada palavra-chave da consulta, onde um nodo XML (elemento, atributo ou texto) é considerado um `casamento (match)` se ele contém uma palavra-chave da consulta; e (ii) `encontrar nodos de retorno` (explícitos ou implícitos) através do processamento dos casamentos encontrados.

Um importante conceito utilizado pelos motores de busca XML é a noção de LCA (*lowest common ancestor* - menor antecessor comum), que foi introduzida por (GUO et al., 2003). O LCA entre dois nodos v e w é o nodo com o maior nível que tem v e w como seus descendentes (onde considera-se que um nodo pode ser descendente dele mesmo). Para encontrar o LCA entre dois nodos *u* e *v*, usando a noção de *Dewey Label*, basta encontrar o mais longo prefixo comum entre *Dewey(u)* e *Dewey(v)*. O termo LCA é utilizado ao longo desta dissertação para referenciar qualquer variante da noção de LCA. Na Figura 2.2, por exemplo, o nodo `0.1` é LCA dos nodos `0.1.0.0` e `0.1.2`.

Nesta dissertação, um nodo XML (elemento ou atributo) que tem como filho um nodo texto cujo valor contém uma ou mais expressões temporais é referenciado como `nodo temporal`. Um caminho XML que vai da raiz até um nodo temporal é referenciado como `caminho temporal`. Por exemplo, considere o documento XML apresentado na Figura 2.3. A expressão `12/11/2000` é uma expressão temporal, logo, o elemento `nascimento` é um nodo temporal e o caminho `/pessoas/pessoa/nascimento` é um caminho temporal.

```
<pessoas>
  <pessoa>
   <sexo>F</sexo>
   <nascimento>12/11/2000</nascimento
  </pessoa>
</pessoas>
```

Figura 2.3: Exemplo de documento XML com um nodo temporal e um caminho temporal

A literatura pesquisada não apresenta uma definição para consultas temporais por palavras-chave. Com isso, utilizou-se a seguinte definição: uma `consulta temporal por palavras-chave` é uma consulta por palavras-chave onde pelo menos uma das palavras-chave é uma expressão temporal.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta o contexto no qual essa dissertação está inserida. As áreas de pesquisa são apresentadas, identificando as principais necessidades e desafios quanto a processamento de consultas temporais por palavras-chave em documentos XML, visando delimitar o escopo do problema. Dentro desse contexto, são descritos os principais trabalhos relacionados, que são comparados entre si e, posteriormente, com a solução proposta pela dissertação.

Este capítulo está organizado da seguinte forma. Na Seção 3.1 são discutidas ferramentas de anotação de expressões temporais, enquanto a Seção 3.2 apresenta uma pesquisa sobre a utilização de expressões temporais em consultas Web. Alguns motores de busca temporais para páginas Web são apresentados na Seção 3.3. Na Seção 3.4 são apresentados trabalhos que objetivam permitir o suporte a consultas temporais sobre dados semi-estruturados. Finalmente, na Seção 3.5, são apresentados motores de busca por palavras-chave em documentos XML.

3.1 Ferramentas de Extração de Expressões Temporais

As ferramentas atuais de extração de expressões temporais em documentos Web usam técnicas de extração de entidades nomeadas (*named-entities*) para identificar as expressões temporais. Uma vez identificadas as expressões temporais, gera-se um documento no formato XML com essas expressões anotadas (ALONSO; GERTZ; BAEZA-YATES, 2007a). TimeML (TIMEML, 2010) é um padrão emergente para anotação de expressões temporais e eventos. No entanto, existem várias ferramentas de anotação de expressões temporais que definem sua própria forma de anotação. A seguir são apresentadas as principais ferramentas de anotação temporal encontradas na literatura pesquisada, bem como um breve comparativo entre elas.

ANNIE (ANNIE, 2010) é uma ferramenta de código aberto para extração de informação que faz parte do *framework* GATE (CUNNINGHAM et al., 2002). Além de informação temporal, ANNIE extrai informações de localização, pessoas, organizações, esportes, etc. A extração é feita a partir de entidades nomeadas. Algumas entidades pré-definidas estão disponíveis e, se necessário, é possível definir novas entidades através da linguagem de regras que vem embutida na ferramenta. Como resultado, tem-se um arquivo XML com as anotações das entidades identificadas em uma linguagem específica da ferramenta. A manipulação desse XML pode ser realizada através de uma API própria, que é oferecida junto com o *framework*. ANNIE anota expressões temporais explícitas, implícitas e relativas. Porém, não realiza a normalização dessas expressões temporais.

GUTime (GUTIME, 2010) é uma ferramenta de código aberto especificada para anotação temporal, disponível junto com o pacote TARSQI (TARSQI, 2010). Junto com essa

ferramenta é necessário instalar o TreeTagger (TREETAGGER, 2010). O TreeTagger é um POS (*Part-of-Speech*) *tagger* que rotula as palavras de um texto com suas funções morfossintáticas, como verbo e substantivo. GUTime utiliza esses rótulos nas suas regras de inferência de expressões temporais. Como resultado, tem-se um documento XML com as expressões temporais anotadas de acordo com a linguagem TimeML. Essas anotações podem ser manipuladas por uma linguagem de processamento de consulta XML como XQuery. GUTime anota e normaliza expressões temporais explícitas, implícitas e relativas. Para normalizar expressões relativas, tais como: hoje, amanhã, ontem, próximo mês e ano passado, GUTime verifica o contexto local a fim de identificar o ponto de referência temporal no qual esses tempos são relativos. Normalmente, o ponto de referência temporal é a data de publicação do documento.

PorTexto (CRAVEIRO; MACEDO; MADEIRA, 2009) é uma ferramenta de reconhecimento de entidades temporais em textos de língua portuguesa. O processamento dos documentos é efetuado frase a frase. A identificação das expressões temporais é feita usando padrões de expressões, criados a partir de co-ocorrências existentes em referências temporais. A determinação das co-ocorrências é realizada utilizando um conjunto de palavras temporais de referência (PTR). A lista das PTR é criada manualmente e deve ser constituída por todas as palavras temporais que aparecem em expressões com, no mínimo, duas palavras, para que realmente haja a necessidade de determinar as palavras que ocorrem conjuntamente. Os meses do ano e dias da semana são exemplos de PTR, pois ocorrem em expressões como "`em janeiro`" e "`na próxima segunda-feira`". Os padrões extraídos são definidos por expressões regulares e armazenados em um arquivo. É possível alterar os padrões existentes e até mesmo incluir novos padrões. Como resultado, tem-se um documento XML com as anotações temporais que seguem as diretivas gerais e de tempo do HAREM (SANTOS et al., 2008). O HAREM é uma avaliação conjunta na área do reconhecimento de entidades nomeadas em português. PorTexto anota expressões temporais explícitas, implícitas e relativas. Porém, não as normaliza. Além disso, não está disponível para utilização.

Chronos (NEGRI; MARSEGLIA, 2004) é uma ferramenta projetada para realizar o reconhecimento e normalização de expressões temporais. O processamento do texto em Chronos envolve a extração de *tokens*, um processamento linguístico e o reconhecimento de multi-palavras baseado em uma lista de 5.000 entradas recuperadas da WordNet[1]. Em seguida, o texto é processado por um conjunto de aproximadamente 1.000 regras básicas que reconhecem construções temporais e extrai informações sobre elas que são úteis para o processo de normalização. Em seguida, são executadas regras de composição que resolvem ambiguidades quando múltiplos rótulos são possíveis. Como resultado, tem-se um documento XML com as anotações temporais na linguagem TIMEX2[2] (FERRO et al., 2001). Chronos anota expressões temporais explícitas e relativas. Além disso, realiza a normalização das expressões temporais.

A Tabela 3.1 apresenta um resumo comparativo das ferramentas de anotação temporal discutidas nessa seção. Nessa tabela são considerados os seguintes itens de comparação:

- `POS` - indica se as ferramentas utilizam processamento linguístico;
- `Linguagem` - indica qual linguagem é utilizada para anotação das expressões temporais;

[1] http://wordnet.princeton.edu/

[2] TimeML é uma extensão de TIMEX2.

- `Disponível` - indica se as ferramentas estão disponíveis para utilização;
- `Tipos` - indica quais os tipos de expressões temporais (explícitas (`E`), implícitas (`I`) e relativas (`R`)) são tratados pelas ferramentas;
- `Normalização` - indica se as ferramentas normalizam as expressões temporais;
- `Isolamento` - indica se cada expressão é avaliada isoladamente.

Tabela 3.1: Comparativo entre as ferramentas de anotação de expressões temporais

	ANNIE	**GUTime**	**PorTexto**	**Chronos**
POS	Sim	Sim	Não	Sim
Linguagem	Própria	TimeML	Diretivas HAREM	TIMEX2
Disponível	Sim	Sim	Não	Não
Tipos	E, I, R	E, I, R	E, I, R	E, R
Normalização	Não	Sim	Não	Sim
Isolamento	Sim	Sim	Sim	Sim

Legenda: E: explícita I: implícita R: relativa

Constata-se que a maioria das ferramentas utilizam processamento linguístico. Essa característica não é desejada quando o objetivo é realizar a identificação de expressões temporais em documentos XML orientados a dados, pois esses documentos não contêm frases compostas por vários elementos morfossintáticos. Geralmente, tem-se nodos apenas com substantivos. Cada ferramenta utiliza uma linguagem de anotação das expressões temporais diferente, embora todas as linguagens sejam baseadas em XML. Apenas GUTime e ANNIE estão disponíveis para utilização. A maioria das ferramentas tratam expressões temporais explícitas, implícitas e relativas. Apenas GUTime e Chronos realizam a normalização das expressões temporais. Todas as ferramentas avaliam as expressões isoladamente para identificar e normalizar as expressões temporais. No entanto, em documentos XML, agrupar as expressões de acordo com o caminho que as contém e analisar em conjunto todas as expressões de um mesmo caminho fornece maiores informações para o processo de normalização, pois permite verificar se todos os valores do caminho satisfazem as restrições temporais e realizar a desambiguação de formatos.

3.2 Uso de Expressões Temporais em Consultas por Palavras-chave

Nunes, Ribeiro e David (2008) realizaram alguns experimentos para verificar como as expressões temporais são utilizadas em consultas Web. Foram utilizados dois conjuntos de dados contendo *logs* de consultas realizadas em um motor de busca Web:

- uma coleção contendo 23.781 consultas submetidas ao motor de busca AOL[3], manualmente classificadas em: automóveis (2,9%), negócios (5,1%), computação (1,4%), entretenimento (10,6%), jogos (2%), saúde (5%), feriados (1,4%), casa e jardim (3,2%), notícias e sociedade (4,9%), organizações (3,7%), finanças pessoais (1,4%), esportes (2,8%), viagens (2,6%), URLs (5,7%), erros ortográficos (5,5%) e outros (13,2%);

[3]http://www.aol.com/

- uma coleção contendo 30 milhões de consultas coletadas de mais de 650.000 usuários durante três meses, possuindo o identificador do usuário que realizou a consulta e o *timestamp* em que cada consulta foi realizada, permitindo normalizar as expressões temporais relativas ao momento atual da consulta. Ao contrário da primeira coleção, as consultas não estavam classificadas, logo, não foi possível identificar os tópicos relacionados às consultas.

Para extrair as expressões temporais das consultas, o texto das consultas foi rotulado usando Aaron Coburn's Lingua::EN::Tagger[4], um *POS tagger* para inglês. Em seguida, a saída foi redirecionada para o TempEx (MANI; WILSON, 2000), um rotulador de texto que anota expressões temporais. TempEx cobre a maioria dos tipos de expressões temporais contidas no padrão TIMEX2 (FERRO et al., 2001)[5]. TempEx anota e normaliza expressões temporais explícitas, implícitas e relativas.

Para avaliar a qualidade da anotação do TempEx tagger quando aplicado em consultas de busca na Web, um subconjunto da coleção contendo 1.000 consultas foi manualmente classificado. A classificação automática realizada pelo TempEx obteve os seguintes resultados: 92% de precisão[6] e 63% de revocação[7]. Em seguida, foi avaliada a homogeneidade da classificação automática com a classificação manual usando um teste estatístico, o qual confirmou que a classificação realizada pelo TempEx é equivalente à classificação humana ($p > 0,05$).

Para investigar como as expressões temporais são distribuídas nos diversos tópicos de consultas, foi utilizada a primeira coleção por estar manualmente anotada com as classes de tópicos. Os tópicos que continham o maior percentual de consultas com expressões temporais foram: `automóveis` (7,8%), `esportes` (5,2%), `notícias e sociedade` (3,9%) e `feriados` (2,5%). Exemplos de consultas contendo expressões temporais são: "`1985 ford ranger engine head`" (automóveis), "`chicago national slam 2003`" (esportes) e "`los angeles times newspaper april 1946`" (notícias e sociedade).

Nunes, Ribeiro e Davi (2008) constataram que a utilização das expressões temporais nas consultas em geral é pequena, comprendendo 1,5% das consultas na primeira coleção e 1,5% na segunda. Removendo consultas duplicadas obteve-se 1,6% na primeira coleção e 1,9% na segunda.

Uma vez que a segunda coleção possuía o *timestamp* em que a consulta foi realizada, foi possível identificar que 42,5% das expressões temporais indicavam o ano da consulta, 49,6% indicavam anos anteriores à consulta e 4,2% indicavam anos sucessores à consulta. As 10 expressões mais utilizadas foram: `algum ano` (45,7%), `Páscoa` (5,6%), `diariamente` (5,4%), `algum mês` (4,6%), `agora` (2,3%), `hoje` (2,1%), `dia das mães` (1,8%), `atual` (1,2%), `Natal` (1,1%) e `semanalmente` (0,8%). Nunes, Ribeiro e Davi (2008) também tinham a hipótese de que as pessoas utilizavam expressões temporais para expandir as consultas, porém os resultados foram baixos (apenas 1,4% das reformulações continham expressões temporais). A identificação da reformulação foi realizada através da comparação de cada consulta com sua consulta anterior, identificada pelo *timestamp* da consulta e identificador de usuário presentes na segunda coleção, verificando se havia uma expansão de termos usados, como por exemplo: "`Páscoa`

[4]http://search.cpan.org/ acoburn/Lingua-EN-Tagger/

[5]A ferramenta GUTime extende TempEx para manipular expressões temporais baseado-se no padrão TimeML TIMEX3.

[6]precisão mede o percentual de expressões anotadas que são temporais.

[7]revocação mede o percentual de expressões temporais que foram anotadas.

Feriado" é considerado uma reformulação de "Páscoa".

Nunes, Ribeiro e Davi (2008) concluíram que expressões temporais são pouco utilizadas tanto em consultas de modo geral (aproximadamente 1,5% das consultas possuem expressões temporais) quanto em reformulações de consultas (aproximadamente 1,4% das reformulações possuem expressões temporais). Nunes, Ribeiro e Davi (2008) atribuem três possíveis razões a esse fato: (**i**) usuários Web normalmente pesquisam sobre eventos atuais; (**ii**) usuários geralmente ficam satisfeitos com os resultados obtidos usando consultas curtas; e (**iii**) usuários procuram interfaces mais avançadas quando precisam de informação temporal. Outras considerações encontradas foram: (**i**) as expressões temporais referenciam normalmente o ano corrente e anos passados, sendo raramente referenciados anos futuros; e (**ii**) as expressões temporais são geralmente usadas nos seguintes tópicos: automóveis, esportes, notícias e feriados.

Nunes, Ribeiro e Davi (2008) destacam que embora as expressões temporais apareçam em apenas uma pequena fração das consultas, a escalabilidade da Web traduz esse percentual a um grande número de usuários. Além disso, afirmam que as expressões temporais podem melhorar o ranqueamento das consultas e permitir um melhor agrupamento dos resultados. A conclusão, portanto, é que com o passar do tempo, mais e mais páginas serão acumuladas e a necessidade por informação temporal aumentará.

A Tabela 3.2 apresenta um resumo dos itens avaliados em (NUNES; RIBEIRO; DAVID, 2008). Nessa tabela são considerados os seguintes itens:

- Percentual - indica se a análise verifica o percentual de consultas que são temporais;
- Tópicos - indica se a análise verifica quais os tópicos contém o maior percentual de consultas temporais;
- Referência - indica se a análise verifica o percentual de consultas referentes ao passado, presente e futuro;
- Tipo - indica se a análise verifica quais os tipos primitivos de tempo são utilizados em consultas temporais;
- Granularidades - indica se a análise verifica quais as granularidades são utilizadas em consultas temporais;
- Operadores - indica se a análise verifica quais os operadores temporais são utilizados em consultas temporais.

Tabela 3.2: Resumo dos itens avaliados

	Percentual	**Tópicos**	**Referência**	**Tipos**	**Granularidades**	**Operadores**
(NUNES; RIBEIRO; DAVID, 2008)	Sim	Sim	Sim	Não	Não	Não

Constata-se que a análise realizada em (NUNES; RIBEIRO; DAVID, 2008) avaliou o percentual de utilização de consultas temporais, os tópicos com maior número de consultas temporais e a que ponto no tempo as expressões temporais se referem. No entanto, não é avaliada a estrutura da consulta, por exemplo, se possuem tipos primitivos de tempo, granularidades, operadores temporais, entre outros aspectos.

3.3 Motores de Busca Temporais sobre Páginas Web

Recentemente, os motores de busca tradicionais, tais como Google[8] e MSN Search[9], começaram a notar o valor da informação temporal na busca Web. Esses motores de busca fornecem algumas formas para os usuários realizarem buscas baseadas no tempo. O Google, por exemplo, disponibiliza uma opção de intervalo de data para expressar um predicado temporal e também permite a visualização dos resultados em uma *timeline*[10]. Esses motores de busca tradicionais suportam a data de coleta das páginas Web (JIN et al., 2008). Há, também, trabalhos que consideram as expressões temporais presentes no conteúdo das páginas Web quando avaliam o predicado temporal da consulta. Os motores de busca que realizam algum tratamento especial para a informação temporal são chamados `motores de busca temporais`.

TISE (JIN et al., 2008) é um motor de busca temporal que considera as expressões temporais presentes nas páginas Web quando avalia o predicado temporal da consulta. O predicado temporal é especificado na forma de um intervalo, que deve ser definido em um campo diferente do campo utilizado para as demais palavras-chave da consulta. TISE indexa uma expressão temporal por página, selecionando a expressão temporal que descreve mais apropriadamente os eventos da página Web. O predicado temporal da consulta é aplicado na expressão temporal indexada para a página. A informação temporal é representada como um intervalo.

TERN (VICENTE-DÍEZ; MARTÍNEZ, 2009) é um motor de busca temporal que considera as expressões temporais presentes nas páginas Web quando avalia o predicado temporal da consulta. O predicado temporal é expresso no mesmo campo onde são expressas as demais palavras-chave da consulta. TERN indexa todas as expressões temporais de cada página. O predicado temporal da consulta é aplicado em qualquer expressão temporal indexada para a página. A informação temporal é representada como um instante.

Chronica (EFENDIOGLU; FASCHETTI; PARR, 2006) é um motor de busca temporal que considera a data de coleta da página. O predicado temporal é especificado na forma de um intervalo, que deve ser definido em um campo diferente do campo utilizado para as demais palavras-chave da consulta. O predicado temporal da consulta é aplicado na data indexada para a página. A informação temporal é representada como um instante.

Pasca (2008) propõem um motor de busca temporal utilizado quando o usuário deseja descobrir quando determinado evento ocorreu. Com isso, o usuário não informa um predicado temporal e sim recebe um valor temporal como resultado. Esse motor de busca indexa uma expressão temporal para cada *nugget* temporal, formando um pseudo-documento. Um *nugget* temporal é um fragmento de uma sentença que informa fatos de domínio aberto associados com alguma entidade. A informação temporal é representada como um instante.

A Tabela 3.3 apresenta um resumo comparativo entre os motores de busca temporais discutidos nessa seção. Nessa tabela são considerados os seguintes itens de comparação:

- `Predicado` - verifica se o predicado temporal é embutido na consulta ou se é informado em um campo separado;

[8] http://www.google.com/

[9] http://www.msn.com/

[10] Uma *timeline* (linha de tempo), também conhecida como cronologia, é uma representação linear de eventos na ordem em que eles ocorreram. Esta seqüência de eventos é normalmente disposta em ordem cronológica e apresentada em uma linha desenhada da esquerda para a direita ou de cima para baixo (ALONSO; GERTZ; BAEZA-YATES, 2007b).

- `Rótulo` - indica se a informação temporal indexada é representada como um instante ou como um intervalo;

- `Tipo de Tempo` - indica qual informação temporal é considerada na consulta: (i) `Atualização` (*update time*), definido como a data de coleta da página Web e (ii) `Conteúdo` (*content time*), definido como a informação temporal embutida no conteúdo textual da página Web (JIN et al., 2008);

- `Índice Temporal` - indica qual informação temporal é indexada;

- `Tipo de Consulta` - indica qual o tipo de consulta pode ser especificada: (i) `Seleção Temporal`, consultas nas quais utiliza-se um predicado temporal para filtrar a consulta e (ii) `Saída Temporal`, consultas nas quais o usuário está interessado em saber qual o tempo em que um determinado evento ocorreu;

- `Docs` - indica o tipo de documento consultado (HTML ou XML).

Tabela 3.3: Comparativo entre motores de busca temporais

	Predicado	**Rótulo**	**Tipo de Tempo**	**Índice Temporal**	**Tipo de Consulta**	**Docs**
TISE	Separado	Intervalo	Conteúdo	1 timex por página	Seleção Temporal	HTML
TERN	Embutido	Instante	Conteúdo	Todas as timex para página	Seleção Temporal	HTML
Chronica	Separado	Instante	Atualização	Data de coleta	Seleção Temporal	HTML
(PASCA, 2008)	NA	Instante	Conteúdo	1 timex por *nugget* temporal	Saída Temporal	HTML

Legenda: NA: Não se aplica Timex: Expressão temporal

Constata-se que TERN é o único motor de busca que permite que o predicado temporal seja inserido no mesmo campo que as demais palavras-chave da consulta. Essa característica exige uma etapa de identificação e normalização das expressões temporais presentes na consulta. No entanto, simplifica e agiliza a criação da consulta, uma vez que não é necessário preencher vários campos. TISE é o único motor de busca que representa a informação temporal indexada como um intervalo. A representação como intervalo é mais abrangente uma vez que permite, por exemplo, para a informação "`presidiu o senado entre 2001 e 2003`", expressar que em 2002 a informação também era válida. A maioria dos motores de busca utilizam a informação temporal presente no conteúdo das páginas. Porém, Chronica, assim como os motores de busca tradicionais, utiliza apenas a data de coleta da página. Cada motor de busca indexa uma informação temporal diferente. Indexar apenas uma expressão temporal para a página (TISE), ocasiona o descarte de várias informações temporais relevantes. Indexar apenas a data de coleta (Chronica) traz problemas quando a página refere-se a informações do passado ou futuro. Indexar cada *nugget* temporal associado a uma expressão temporal (PASCA, 2008), ao invés de indexar todas as expressões temporais para a página (TERN), tem a vantagem de associar a uma expressão temporal apenas os termos que estão próximos no conteúdo da página. A maioria dos motores de busca permitem consultas de seleção temporal. Finalmente, todos os motores de busca têm como foco consultas temporais sobre páginas Web, ou seja, retornam documentos inteiros. No entanto, consultas por palavras-chave em um documento XML devem retornar fragmentos XML relevantes para a consulta e não o documento inteiro.

3.4 Consulta Temporal em Dados Semi-estruturados

Basicamente, toda aplicação manipula algum tipo de informação temporal (EDELWEISS; OLIVEIRA, 1994). Há décadas já existem pesquisas para modelar, manipular e consultar a informação temporal em banco de dados relacionais e banco de dados orientados a objetos. Com a utilização de dados semi-estruturados para representação de informação, surgem trabalhos que aplicam o conceito de tempo nessa estrutura de dados.

Rizzolo e Vaisman (2008) propõem um modelo de dados de rastreamento de informações históricas em um documento XML e uma forma de recuperar o estado dos documentos em qualquer momento. O modelo de dados temporal utilizado trata o tempo como discreto. A informação temporal é adicionada aos documentos XML através dos atributos `From` e `To`, que representam, respectivamente, o instante inicial e final de um intervalo fechado. As consultas são realizadas em uma extensão de XPath, denominada TXPath. Dentre as características suportadas por TXPath, incluem-se: operação *coalesce* e operadores de agregação. Os operadores temporais suportados são `MEETS`, `CONTAINS` e `IN`. Eles também propõem uma abordagem para sumarizar e indexar documentos XML temporais.

Gao e Snodgrass (2003) apresentam uma linguagem de consulta temporal denominada TXQuery, que adiciona o suporte temporal para XQuery, estendendo sua sintaxe e semântica. O objetivo é mover a manipulação do tempo do código da aplicação para o processador TXQuery. O modelo de dados adota o tempo de validade para expressar o tempo nos documentos XML. No entanto, os autores afirmam que o tempo de transação pode ser utilizado de forma análoga. A informação temporal é adicionada nos documentos XML através de um elemento `TIMESTAMP` com os atributos `vtBegin` e `vtEnd` que representam, respectivamente, o instante inicial e final de validade. TXQuery suporta três tipos de consultas: (i) `representacionais`, consultas com a mesma semântica de XQuery; (ii) `atuais`, consultas sobre os dados válidos atualmente; e (iii) `sequênciais`, consultas sobre o histórico de validade dos dados. Dentre as características suportadas por TXQuery, incluem-se: operação *coalesce* e operadores de agregação. Nenhum operador temporal é suportado por TXQuery, logo eles devem ser construídos através de operadores relacionais.

Timely YAGO (T-YAGO) (WANG et al., 2010) é um protótipo que extrai fatos temporais da Wikipédia e os integra em uma base de conhecimento. A extração dos fatos temporais é focada nos elementos semi-estruturados da Wikipédia (categorias, *infoboxes* e listas) e usa regras compostas por expressões regulares para a sua realização. Os fatos temporais são armazenados em RDF. O protótipo suporta consultas sobre os fatos temporais armazenados, utilizando predicados temporais em uma extensão da linguagem SPARQL[11]. T-YAGO disponibiliza os operadores temporais `before`, `after`, `equal`, `during`, `overlaps` e `sameYear`. T-YAGO permite que os dois operandos do predicado temporal sejam eventos. Por exemplo, jogadores de futebol que jogaram no mesmo time que Beckham durante períodos sobrepostos. A visualização dos resultados da consulta em T-Yago pode ser realizada em uma *timeline*.

A Tabela 3.4 apresenta um resumo comparativo entre os trabalhos de consulta temporal. Nessa tabela, são considerados os seguintes itens de comparação:

- `Documentos` - indica o tipo de documento consultado (HTML ou XML);

[11] http://www.w3.org/TR/rdf-sparql-query/

- `Modelo` - indica se para a realização da consulta é necessário um modelo temporal específico, ou se o projetista tem uma maior liberdade para definir a forma de representação temporal (*`ad-hoc`*);
- `Tipo de Consulta` - indica se as consultas temporais são realizadas através de uma linguagem de consulta estruturada ou através de palavras-chave;
- `Operadores Temporais` - indica o número de operadores temporais suportados;
- `Agregação` - indica se o trabalho permite operadores de agregação;
- *`Coalesce`* - indica se o trabalho suporta a operação *coalesce*.

Tabela 3.4: Comparativo entre as abordagens de consulta temporal em dados semi-estruturados

	Documentos	**Modelo**	**Tipo de Consulta**	**Operadores Temporais**	**Agregação**	***Coalesce***
TXPath	XML	Específico	Estruturada	3	Sim	Sim
TXQuery	XML	Específico	Estruturada	0	Sim	Sim
T-Yago	HTML	*Ad-hoc*	Estruturada	6	Não	Não

Constata-se que existem abordagens para a realização de consultas temporais tanto em documentos XML quanto em páginas Web. Além disso, as abordagens para XML exigem que o documento XML siga um modelo temporal específico para que as consultas sejam realizadas, limitando imensamente os documentos XML que podem ser consultados. Todos os trabalhos suportam as consultas temporais em uma linguagem estruturada. No entanto, consultas por palavras-chave representam uma abordagem amigável para descoberta de informação que tem sido amplamente estudada para documentos texto (BHALOTIA et al., 2002; HRISTIDIS et al., 2006; LIU; CHEN, 2007). T-Yago é o trabalho que suporta o maior número de operadores temporais. No entanto, é o único que não suporta operadores de agregação e a operação *coalesce*.

3.5 Consultas por Palavras-Chave sobre Documentos XML

Devido à falta de expressividade e inerente ambiguidade das consultas por palavras-chave, há dois desafios principais na interpretação da semântica quando realizadas em documentos XML (LIU; CHEN, 2010):

- identificar os predicados da consulta, equivalente à cláusula `WHERE` em linguagens estruturadas como XQuery e SQL.
- identificar nodos de retorno, ou seja, identificar eficientemente a informação desejável de retorno, equivalente a cláusula `RETURN` em XQuery e à cláusula `SELECT` em SQL.

Vários trabalhos têm sido propostos para atacar o primeiro desafio, tais como XRank (GUO et al., 2003), XSEarch (COHEN et al., 2003), XKSearch (XU; PAPAKONSTANTINOU, 2005) e (LI; YU; JAGADISH, 2008). Por exemplo, considere o documento XML (`Doc Livros`) apresentado na Figura 3.1 sobre livros, onde cada nodo possui um identificador único (`ID`) associado, de acordo com o conceito de *Dewey Label*.

Considere a consulta Q1: "TSQL2 ISBN" com o objetivo de retornar o ISBN do livro cujo título é TSQL2. Há dois nodos (0.0.1 e 0.1.1) que casam com a palavra-chave ISBN. Apenas o nodo 0.0.1 deve ser considerado como intimamente relacionado com o nodo 0.0.0.0, que é o casamento para a palavra-chave TSQL2 (os nodos 0.0.1 e 0.0.0.0 se referem ao mesmo livro). Isso é alcançado usando variantes do conceito de LCA, tais como SLCA (XU; PAPAKONSTANTINOU, 2005), MLCA (LI; YU; JAGADISH, 2004), *interconnection* (COHEN et al., 2003), CVLCA (LI et al., 2007), MaxMatch (LIU; CHEN, 2008), etc. O LCA no exemplo acima é o nodo 0.0.

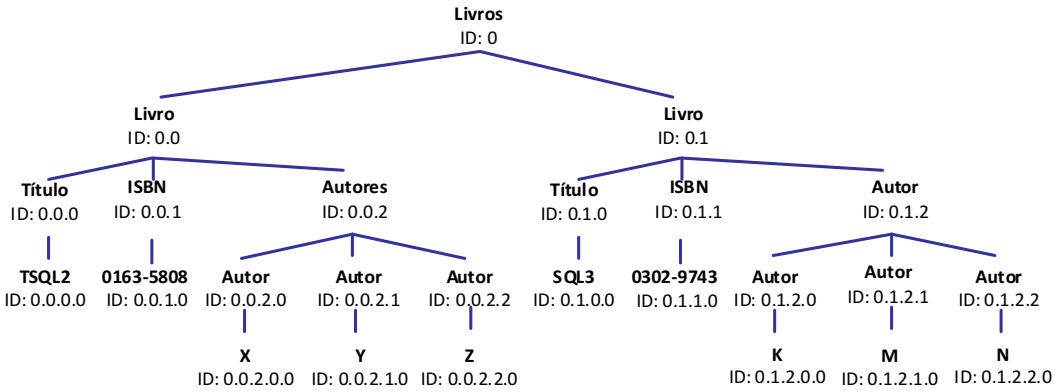


Figura 3.1: Documento XML Doc Livros

Com relação ao segundo desafio, os motores de busca XML são classificados em: *Path Return*, *Subtree Return* e *Inferred Nodes Return*. Os motores de busca XML *Path Return* retornam os caminhos (*paths*) na árvore XML a partir de cada nodo LCA até seus descendentes que casam com uma palavra-chave da consulta. A Figura 3.2(a) apresenta o resultado da consulta Q1 sobre o documento Doc Livros de acordo com essa categoria. Observa-se que o nodo com o valor do ISBN não foi retornado, embora fosse a informação que o usuário desejava. Os trabalhos propostos por Bhalotia et al. (2002) e Hristidis et al. (2006) são exemplos desse tipo de motor de busca.

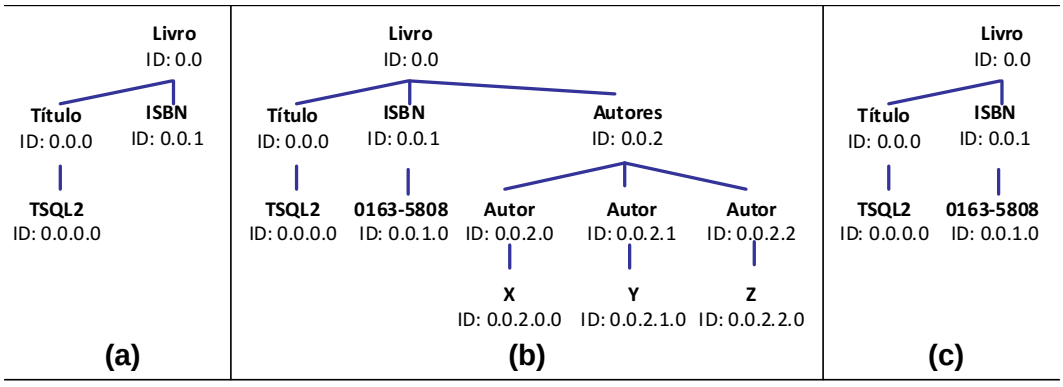


Figura 3.2: Resultado de acordo com *Path Return* (a), *Subtree Return* (b) e *Inferred Nodes Return* (c) para a consulta Q1 sobre o documento Doc Livros

Os motores de busca XML *Subtree Return* retornam as subárvores enraizadas nos nodos que são LCAs. A Figura 3.2(b) apresenta o resultado da consulta Q1 sobre o documento Doc Livros de acordo com essa categoria. Observa-se que foram retornadas informações sobre os autores dos livros, poluindo o resultado, pois não representa a

informação que o usuário buscava. XRank (GUO et al., 2003) e XKSearch (XU; PAPAKONSTANTINOU, 2005) são exemplos desse tipo de motor de busca.

Os motores de busca XML *Inferred Nodes Return* inferem a semântica da consulta analisando a estrutura da consulta e do documento XML a fim de identificar os nodos pertencentes às subárvores enraizadas pelo nodos LCAs, que são relevantes para a consulta. A Figura 3.2(c) apresenta o resultado da consulta `Q1` sobre o documento `Doc Livros` de acordo com essa categoria. Observa-se que apenas foi retornado o nodo `Título` e seu valor por representar o predicado da consulta e o nodo `ISBN` e seu valor por representar o nodo de retorno explícito da consulta. XSeek (LIU; CHEN, 2007) é um exemplo desse tipo de motor de busca XML.

A Tabela 3.5 apresenta um resumo comparativo entre os motores de busca XML. Nessa tabela são considerados os seguintes itens de comparação:

- `Predicado` - indica a forma como é definido o predicado da consulta;
- `Retorno` - indica a forma como são definidos os nodos de retorno;
- `Tratamento Temporal` - indica se algum tratamento específico para a informação temporal é realizado.

Tabela 3.5: Comparativo entre os motores de busca XML

	Predicado	**Retorno**	**Tratamento Temporal**
(BHALOTIA et al., 2002)	LCA	*Path Return*	Não
(HRISTIDIS et al., 2006)	LCA	*Path Return*	Não
XRank (GUO et al., 2003)	LCA	*Subtree Return*	Não
XKSearch (XU; PAPAKONSTANTINOU, 2005)	LCA	*Subtree Return*	Não
XSeek (LIU; CHEN, 2007)	LCA	*Inferred Nodes Return*	Não

Constata-se que todos os motores de busca XML utilizam alguma variação da noção de LCA para encontrar os predicados da consulta. Os motores de busca XML propostos por Bhalotia et al. (2002) e Hristidis et al. (2006) definem os nodos a serem retornados de acordo com a categoria *Path Return*. XRank e XKSearch definem os nodos a serem retornados de acordo com a categoria *Subtree Return*. XSeek define os nodos a serem retornados de acordo com a categoria *Inferred Nodes Return*. Todos os motores de busca XML apresentados não possuem um tratamento específico para a informação temporal presente na consulta e no conteúdo dos documentos XML.

XSeek é um motor de busca XML que permite aos usuários executarem consultas por palavras-chave em documentos XML, e que identifica nodos de retorno significantes sem a identificação do perfil do usuário. Nodos de retorno referem-se ao nome e ao conteúdo dos nodos que são o objetivo da consulta do usuário. XSeek analisa tanto a estrutura do XML como os casamentos para as palavras-chave da consulta. Em XSeek, os nodos XML são classificados em: (i) entidades do mundo real; (ii) atributos de entidades; e (iii) nodos de conexão. As palavras-chave da consulta são categorizadas em: (i) predicados da consulta; e (ii) informação de retorno que o usuário está procurando. XSeek gera nodos de retorno que podem ser explicitamente inferidos das palavras-chave da consulta ou dinamicamente construídos de acordo com as entidades presentes nas subárvores dos LCAs da consulta. XSeek utiliza o conceito de *Dewey Label* para atribuir identificadores para os nodos do documento XML. XSeek é o motor de busca XML utilizado como *baseline* nessa dissertação, pois é o único motor de busca XML que apresenta tanto regras para identificação de predicados quanto regras para identificação de nodos de retorno.

4 ANÁLISE DE CONSULTAS

Com o intuito de identificar requisitos e funcionalidades necessários a um mecanismo de consulta temporal por palavras-chave, foi realizado um experimento que analisou consultas Web em um motor de busca. O objetivo dessa análise foi identificar a estrutura das consultas temporais por palavras-chave através da identificação de tipos primitivos de representação temporal, granularidades, operadores temporais, entre outras características.

Esse capítulo está organizado da seguinte forma. A Seção 4.1 descreve a metodologia utilizada no experimento. A análise dos resultados é apresentada na Seção 4.2. O capítulo é finalizado na Seção 4.3 com uma análise geral dos resultados.

4.1 Configuração do Experimento

A análise das consultas foi realizada sobre a coleção de consultas WBR99. Essa coleção contém consultas Web submetidas ao extinto motor de busca chamado TodoBR[1]. WBR99 contém consultas realizadas na Web brasileira em novembro de 1999. São disponibilizadas 65.737 consultas, sendo 33.154 consultas distintas. Para clareza, no decorrer desse capítulo, as consultas do conjunto que considera duplicatas são referenciadas como `duplicatas` e as consultas do conjunto que elimina as duplicatas são referenciadas como `consultas distintas`.

Cada consulta da coleção WBR99 foi submetida às ferramentas GUTime (GUTIME, 2010) e ANNIE (ANNIE, 2010), que realizam a anotação de expressões temporais. As consultas que tiveram pelo menos uma palavra-chave anotada como expressão temporal, por ANNIE e/ou por GUTime, foram analisadas manualmente. Essa análise manual tinha como primeiro objetivo verificar se as palavras-chave anotadas realmente representavam expressões temporais. O segundo objetivo foi verificar as características das consultas com expressões temporais, tais como tipos primitivos de tempo, granularidades, operadores temporais, etc.

Nesse capítulo, as consultas com expressões temporais são referenciadas como `consultas temporais`. As consultas com versões de softwares ("`Windows 2000`", por exemplo) não foram consideradas como consultas temporais, uma vez que a versão de um software pode não representar realmente seu lançamento. Por exemplo, "`Windows 2000 Advanced Server - Limited Edition`" foi lançado em 2001.

[1]TodoBR é uma marca registrada de *Akwan Information Technologies*, que foi adquirida pelo Google em julho de 2005.

4.2 Análise dos Resultados

A Tabela 4.1 mostra que, das 65.737 duplicatas, 318 (0,48%) são temporais e 223 (0,34%) possuem informação sobre versões de softwares. Considerando apenas as consultas distintas, 202 (0,61%) são temporais e 96 (0,29%) possuem informação sobre versões.

Tabela 4.1: Uso de expressões temporais em consultas na Web brasileira

	Distintas	Duplicatas	Distintas (%)	Duplicatas (%)
Total de consultas	33.154	65.737	-	-
Expressões temporais	202	318	0,61	0,48
Versões	96	223	0,29	0,34

Verificou-se o uso de operadores temporais explícitos[2] a fim de especificar relações temporais entre as expressões temporais e as demais palavras-chave da consulta. No conjunto das consultas temporais distintas foram encontradas 61 (30,20%) consultas com operadores temporais explícitos. No conjunto das consultas temporais considerando as duplicatas foram encontradas 108 (33,96%) consultas com operadores temporais explícitos. Os operadores temporais explícitos encontrados são apresentados na Figura 4.1. É importante notar que alguns operadores foram agrupados:

- **`em X`** - inclui `no ano X`, `nos anos X` e `na década X`;
- **`de X a Y`** - inclui `de X a Y`, `de X e Y`, `de X Y`, `no período de X a Y` e `dos anos X e Y`;
- **`de X`** - inclui `de X`, `do ano X` e `da década X`;
- **`após X`** - inclui `após X` e `depois de X`.

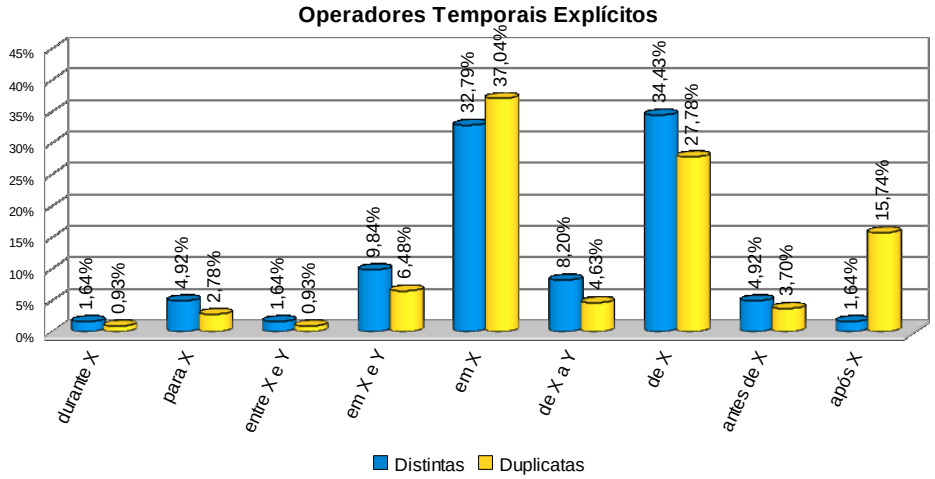


Figura 4.1: Uso de operadores temporais explícitos em consultas temporais

[2]Nesta dissertação, considera-se como operador temporal explícito uma palavra-chave da consulta que expressa uma relação temporal. Por exemplo, `antes` e `após`.

Constatou-se que alguns operadores temporais utilizados nas consultas representam a mesma relação temporal. Por exemplo, `após X` e `depois de X` representam a relação temporal `after`. Além disso, consultas temporais sem um operador temporal explícito, também possuem uma relação temporal entre a expressão temporal e as demais palavras-chave da consulta. Essa relação temporal pode ser inferida pela semântica e contexto da consulta. A Figura 4.2 apresenta um agrupamento das consultas de acordo com as relações entre intervalos definidas em (ALLEN, 1983). Esse foi um agrupamento realizado de forma subjetiva que considerou a semântica e contexto da consulta, bem como os operadores temporais (quando eles estavam explícitos). Por exemplo, a consulta "`temperatura em 14 8 99`" contém o operador temporal explícito `em X` representando a relação temporal `equal`[3]. Isso significa que o usuário deseja a temperatura válida em 14/08/1999. Outro exemplo, a consulta "`regime militar 64 84`" não possui um operador temporal explícito, mas sabe-se que o regime militar no Brasil ocorreu de 1964 a 1984. Então, inferiu-se que a relação temporal é `equal`. Isso significa que o usuário deseja informações sobre o regime militar, que iniciou em 1964 e terminou em 1984.

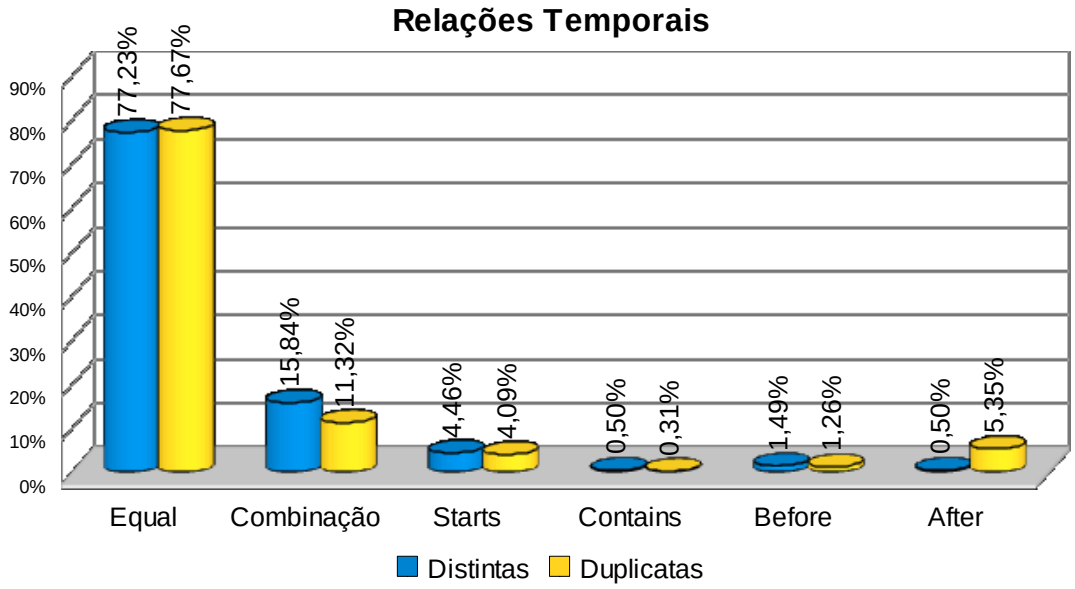


Figura 4.2: Relações temporais em consultas temporais

Constatou-se em 15,84% das consultas distintas (11,32% das duplicatas) a necessidade de restringir a consulta apenas a um intervalo que intersecione outro intervalo (Figura 4.2 - `Combinação`). Para isso é necessária uma combinação das relações entre intervalos definidas em (ALLEN, 1983). Por exemplo, a consulta "`salario em 2005`" deve retornar tanto os salários cuja validade iniciou em 01/01/2005 e terminou em 31/12/2005 (relação `equal`); os salários cuja validade iniciou antes de 2005 e acabou após 01/01/2005 e antes de 31/12/2005 (relação `overlaps`), entre outras relações.

Também, foram verificados os tipos primitivos de representação temporal das expressões temporais utilizadas nas consultas. A Tabela 4.2 ilustra que em 184 (91,09%) consultas temporais distintas o tipo primitivo era instante, enquanto que 18 (8,91%) consultas

[3]Ressalta-se que Allen (1983) definiu relações entre intervalos e neste trabalho um instante é tratado como um intervalo que inicia e termina no mesmo instante.

temporais distintas tinham intervalo temporal como tipo primitivo e nenhuma consulta possuía elemento temporal nem duração temporal como tipo primitivo. Considerando consultas com duplicatas, 297 (93,40%) duplicatas tinham instante como tipo primitivo, enquanto que 21 (6,60%) tinham intervalo temporal e nenhuma tinha elemento temporal nem duração temporal. Foram encontradas duas representações para o tipo primitivo intervalo temporal: (i) intervalos de tempo absoluto, definidos por dois instantes, representando o instante inicial e o instante final do intervalo; e (ii) intervalos de tempo relativo à data atual, definidos por expressões temporais representando um deslocamento temporal a partir da data atual, como, por exemplo, `últimos 10 anos`.

Tabela 4.2: Tipos primitivos de representação temporal nas consultas temporais

	Distintas	Duplicatas	Distintas (%)	Duplicatas (%)
Instante	184	297	91,09	93,40
Intervalo Temporal	18	21	8,91	6,60

A Figura 4.3 mostra que a maioria das consultas temporais (72,77% das consultas distintas e 65,41% das duplicatas) possuem granularidade `ano`, seguido por `década` (21,78% nas consultas distintas e 28,62% nas duplicatas). É importante notar que a granularidade da consulta pode ser diferente da granularidade dos dados. Por exemplo, a consulta "`músicas nos anos 60`" deve retornar as músicas dos anos 1960, 1961, até 1969.

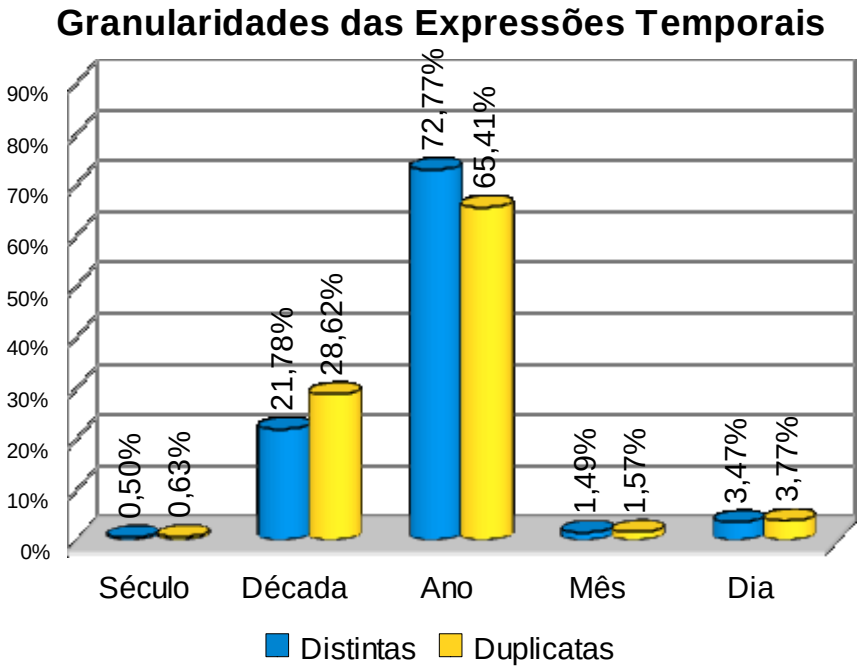


Figura 4.3: Granularidades utilizadas nas consultas temporais

A Tabela 4.3 apresenta a localização das expressões temporais na consulta. Na análise da localização, foi verificado se a expressão temporal presente na consulta estava antes das palavras-chave não temporais (`Início`), entre as palavras-chave não temporais (`Meio`), ou após as palavras-chave não temporais (`Fim`). É importante notar que em 25 consultas distintas (58 duplicatas) todas as palavras-chave da consulta representavam expressões

temporais. Essas consultas não foram consideradas na análise da localização, uma vez que não possuíam palavras-chave não temporais. A análise mostra que na grande maioria das consultas (93,22% das consultas distintas e 94,62% das duplicatas), a expressão temporal está no fim da consulta.

Tabela 4.3: Localização das expressões temporais na consulta

	Distintas	Duplicatas	Distintas (%)	Duplicatas (%)
Início	2	2	1,13	0,77
Meio	10	12	5,65	4,62
Fim	165	246	93,22	94,62

A Figura 4.4 mostra o número de palavras-chave não temporais nas consultas temporais (removidos os artigos, os pronomes, as preposições e as conjunções). A maioria das consultas possui entre zero e três palavras-chave não temporais.

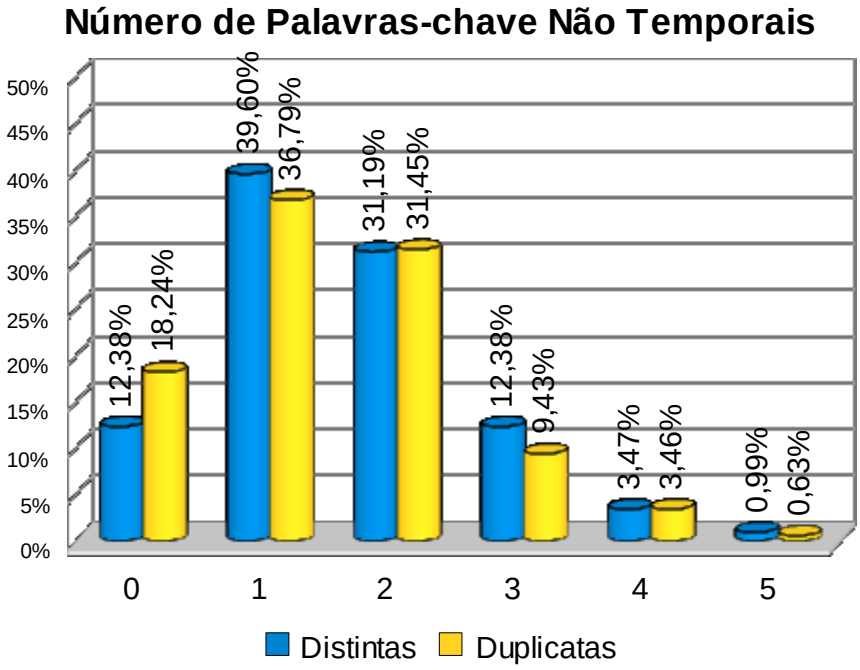


Figura 4.4: Número de palavras-chave não temporais presentes nas consultas temporais

4.3 Considerações Finais

Este capítulo apresentou a análise de consultas Web realizada. O objetivo dessa análise foi identificar a estrutura das consultas temporais por palavras-chave.

Nessa análise foram encontrados os tipos primitivos de representação temporal instante e intervalo temporal. O intervalo temporal é caracterizado de duas formas: (i) delimitado por um instante inicial e um instante final; e (ii) representado por um deslocamento temporal a partir da data atual.

Algumas consultas possuem operadores temporais explícitos para definir a relação temporal entre as palavras-chave não temporais e as expressões temporais. Nas consultas que não possuem essa característica, a relação temporal precisa ser inferida através da

semântica e do contexto da consulta. Várias palavras-chave da consulta representam uma mesma relação temporal.

Algumas consultas expressam a necessidade de restringir a consulta apenas a um intervalo que interseccione outro intervalo. Para isso é necessário uma combinação das relações entre intervalos definidas em (ALLEN, 1983). Acredita-se que isso ocorra, uma vez que as relações entre intervalos definidas em (ALLEN, 1983) são mutuamente excludentes, e existem certas ocasiões nas quais o usuário não deseja expressar tanta especificidade.

As granularidades encontradas foram: ano, década, dia, mês e século. A maioria das consultas temporais possui a parte temporal após as demais palavras-chave. Removidas as *stop words*, a maioria das consultas possui entre zero e três palavras-chave não temporais.

A Tabela 4.4 apresenta um resumo comparativo entre a análise de consultas temporais realizada nessa dissertação (`Dissertação`) e a análise de consultas temporais realizada por (NUNES; RIBEIRO; DAVID, 2008). O principal diferencial da análise de consultas realizada nesta dissertação é a verificação da estrutura das consultas temporais (tipos primitivos de tempo, granularidades, operadores temporais, etc.).

Tabela 4.4: Comparativo entre a análise de consultas realizada nessa dissertação e a análise de consultas realizada no trabalho relacionado

	Percentual	**Tópicos**	**Referência**	**Tipos**	**Granularidades**	**Operadores**
(NUNES; RIBEIRO; DAVID, 2008)	Sim	Sim	Sim	Não	Não	Não
Dissertação	Sim	Não	Não	Sim	Sim	Sim

5 TEMPORAL KEYWORD CLASSIFICATION (TKC)

Este capítulo apresenta **TKC** - ***Temporal Keyword Classification*** - que é uma classificação genérica proposta para servir de guia para qualquer mecanismo de consulta temporal por palavras-chave. Primeiramente, é apresentada uma visão geral para o formato de expressar consultas temporais por palavras-chave. Em seguida, a sintaxe da classificação é brevemente especificada. São apresentados os algoritmos de mapeamento dos predicados temporais, definidos por palavras-chave, para expressões relacionais. Por fim, são expostas as considerações finais e um comparativo com os trabalhos relacionados.

5.1 TKC: Uma Visão Geral

TKC especifica que uma consulta temporal é constituída por um conjunto de palavras-chave. Cada palavra-chave pode especificar um predicado da consulta ou uma informação que o usuário deseja como retorno. Uma palavra-chave deve especificar uma relação temporal e uma ou mais palavras-chave devem compor um instante ou um intervalo de tempo a ser utilizado para filtrar a consulta.

Considere as consultas apresentadas na Figura 5.1. A consulta Q4.1 tem como objetivo retornar os artigos publicados no VLDB após 2005. A consulta Q4.2 tem como objetivo retornar os artigos publicados no VLDB entre 2005 e 2008. A Q4.3 tem como objetivo retornar os artigos publicados no VLDB nos últimos 10 anos.

```
Q4.1 - articles VLDB after 2005
Q4.2 - articles VLDB intersect [2005, 2008]
Q4.3 - articles VLDB intersect last 10 years
```

Figura 5.1: Consultas utilizadas nos exemplos

A Figura 5.2 apresenta a visão geral de TKC. TKC propõe que uma consulta temporal forma um `predicado temporal`. Um predicado temporal é formado por um primeiro operando, um operador temporal e um segundo operando. O `primeiro operando` é caracterizado pelas palavras-chave não temporais da consulta, sobre as quais infere-se em quais intervalos deve-se aplicar a condição temporal. Nas consultas Q4.1, Q4.2 e Q4.3, o primeiro operando é caracterizado por "`articles VLDB`". O `operador temporal` representa a relação temporal entre o primeiro e o segundo operando. Na consulta Q4.1, o operador temporal é "`after`" e nas consultas Q4.2 e Q4.3, o operador temporal é "`intersect`". O `segundo operando` expressa o tempo desejado como filtro para a consulta. Na consulta Q4.1, o segundo operando é "`2005`". Na consulta Q4.2, o segundo operando é "`[2005, 2008]`". Na consulta Q4.3, o segundo operando é "`last 10 years`".

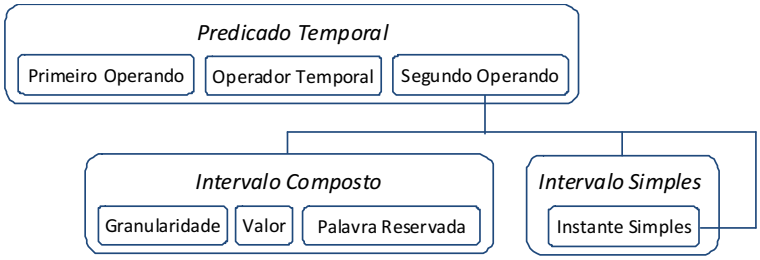


Figura 5.2: Visão geral de TKC

O segundo operando pode ser um instante simples, um intervalo simples ou um intervalo composto. Um `instante simples` é caracterizado por um ponto no tempo (segundo operando de Q4.1, por exemplo). Um `intervalo simples` é caracterizado por dois instantes simples que representam, respectivamente, o instante inicial e final de um intervalo (segundo operando de Q4.2, por exemplo). Um `intervalo composto` é caracterizado por um deslocamento temporal a partir da data atual, ou seja, é um tempo relativo ao momento atual (segundo operando de Q4.3, por exemplo). Um intervalo composto é formado por uma palavra reservada, uma granularidade e um valor. A `palavra reservada` indica a direção do deslocamento, a `granularidade` define a unidade de tempo utilizada, e o `valor` especifica o tamanho do deslocamento. Na consulta Q4.3, "`last`" é a palavra reservada, "`10`" é o valor e "`years`" é a granularidade.

5.2 Especificação de TKC

Esta seção apresenta brevemente a especificação da classificação TKC, sendo sua sintaxe geral e simplificada ilustrada na Figura 5.3. A BNF completa é apresentada no Apêndice A. Cabe ressaltar que TKC especifica um formato para expressar consultas temporais por palavras-chave, mantendo inalterado o formato das consultas básicas por palavras-chave.

```
tempQuery ::= {kws} tempOp op
kws ::= string
tempOp ::= ( After | Before | Contains | During | Equals | FinishedBy |
             Finishes | Intersect | Meets | MetBy | OverlappedBy |
             Overlaps | StartedBy | Starts )
op ::=  ( instant | interval | compositeInterval )
instant ::= [ number / ] [ number / ] number
interval ::= [ instant , instant ]
compositeInterval ::= word  [value] granularity
word ::= ( Current | Last | Next )
granularity ::= ( Day(s) | Week(s) | Month(s) | Year(s) |
                  Decade(s) | Century(ies) )
value ::= integer
```

Legenda: colchetes [] indicam segmentos opcionais, chaves {} indicam segmento repetitivo opcional (zero ou mais), parênteses () indicam um conjunto de opções, | indica "ou", e palavras em negrito são usadas como termos pré-definidos do formato proposto.

Figura 5.3: Sintaxe geral de TKC

Uma consulta temporal (`tempQuery`) é composta, em ordem, por: (i) `kws`, que representa um conjunto de palavras-chave convencional sem informação temporal, ca-

racterizando o primeiro operando; (ii) `tempOp`, que representa o operador temporal; e (iii) `op`, que representa o segundo operando.

Observa-se que a parte temporal é adicionada no final da consulta. Ressalta-se que isso não gera grandes perdas ao usuário, uma vez que, na análise de consultas Web realizada, as expressões temporais estavam no final da consulta na grande maioria dos casos (aproximadamente 94% das consultas temporais).

O formato inclui como operadores temporais todas as relações de Allen (1983). Além disso, inclui a relação `intersect` que restringe a consulta a um intervalo que intersecciona outro intervalo. Isso significa que a relação `intersect` combina as relações de Allen `equal`, `during`, `contains`, `overlaps`, `overlapped by`, `starts`, `started by`, `finishes` e `finished by`, que são mutuamente excludentes, fazendo uma generalização.

O segundo operando pode ser: (i) `instant`, que representa um instante simples; (ii) `interval`, que representa um intervalo simples; e (iii) `compositeInterval`, que representa um intervalo composto. O instante simples deve seguir o formato `dd/mm/aaaa`, sendo o dia e o mês elementos opcionais. Por exemplo, `2005`, `11/2008` e `14/05/2010`. O intervalo simples deve conter dois instantes simples que representam o instante inicial e final do intervalo. Esses dois instantes devem estar entre colchetes e separados por vírgula. Por exemplo, `[01/01/2008, 30/06/2008]`.

Um intervalo composto é formado por uma palavra reservada (`word`), uma granularidade (`granularity`) e opcionalmente por um valor (`value`). O valor deve ser um número inteiro. As granularidades especificadas são: `dia`, `semana`, `mês`, `semestre`, `ano`, `década` e `século`. Essa especificação cobre todas as granularidades encontradas na análise das consultas.

Três palavras reservadas foram especificadas: Next, Last e Current. `Next` inicia um intervalo temporal na data corrente e avança *n* unidades de tempo. `Last` termina um intervalo temporal na data corrente e retorna *n-1* unidades de tempo. Para ambos os casos, `Next` e `Last`, *n* indica um valor especificado, ou 1 se o valor não for informado. É necessário subtrair 1 do valor para a palavra reservada `Last` para descontar o valor corrente. Por exemplo, a expressão "`last 2 months`" cobre o mês corrente e o mês passado, então o deslocamento deve retornar um mês. `Current` cobre o período atual de acordo com a granularidade informada. Por exemplo, a expressão "`current month`" representa um intervalo que inicia no primeiro dia do mês corrente e termina no último dia do mês corrente.

Destaca-se que foram definidos sinônimos, incluindo termos em inglês e português, para cada relação temporal, granularidade e palavra-reservada. Isso se faz necessário devido a riqueza dos idiomas, nos quais uma mesma relação temporal, granularidade ou palavra-reservada pode ser expressa por diferentes termos. Para clareza do texto, os sinônimos apenas são apresentados no Apêndice A, no qual observa-se para a relação temporal `after`, os termos `Following`, `Após` e `Depois de`; para a palavra reservada `Last`, o termo `Último(s)`; e para a granularidade `Day`, o termo `Dia(s)`.

A Figura 5.4 apresenta alguns exemplos de utilização do segundo operando. Em (a) é esperado como retorno o mês corrente (`current month`), então o intervalo retornado inicia no primeiro dia e termina no último dia do mês atual. Em (b) é esperado como retorno o ano corrente (`current year`), então é retornado um intervalo que inicia no primeiro dia e termina no último dia do ano atual. Em (c) é esperado o retorno dos últimos dois meses (`last 2 months`), então o intervalo retornado inicia no primeiro dia do mês passado e termina na data atual. Em (d) é esperado os próximos dois meses

(`next 2 months`), então o intervalo retornado inicia na data atual e termina no último dia do mês seguinte ao próximo mês. É importante notar que em (c) ocorre um retorno até o instante 01/03/2010 e em (d) ocorre um avanço até o instante 30/06/2010, mesmo o instante inicial sendo 05/04/2010. Isso se deve ao fato de que as duas consultas possuem granularidade `mês` (`months`), logo o deslocamento ocorre considerando mês como uma unidade de tempo atômica. Para realizar um deslocamento em dias é necessário utilizar a granularidade `dia`.

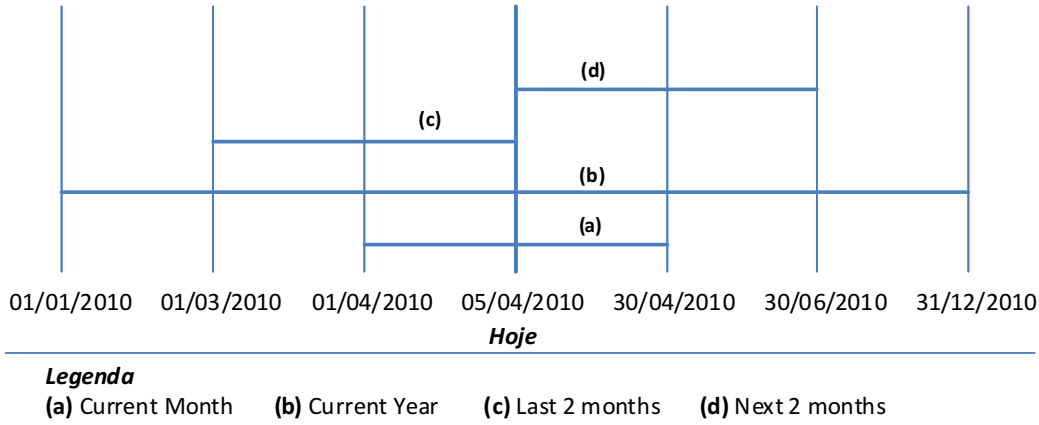


Figura 5.4: Exemplos de segundos operandos

Ressalta-se que a ordem dos elementos que compõem o segundo operando não é fixa. Por exemplo, em inglês tem-se o segundo operando "`current year`", onde primeiro tem-se a palavra reservada (`current`) seguida pela granularidade (`year`). No entanto, em português, tem-se o segundo operando "`ano atual`" com o mesmo significado, onde primeiro tem-se a granularidade (`ano`) seguida pela palavra reservada (`atual`).

5.3 Mapeamento

Esta seção apresenta o mapeamento dos predicados temporais formados por palavras-chave para expressões relacionais[1]. Os mapeamentos são especificados através de algoritmos que servem de guia para a implementação do processamento das consultas temporais. O mapeamento é necessário porque as linguagens de programação não suportam predicados temporais por palavras-chave, porém permitem a utilização de expressões relacionais para expressar condições de comparação.

Mapear um predicado temporal para uma expressão relacional é uma tarefa realizada através de uma série de algoritmos criados. Cada algoritmo realiza um passo específico do mapeamento. A seguir, cada algoritmo é apresentado e explicado.

O Algoritmo 1 contém a ordem em que os mapeamentos devem ser executados. Esse algoritmo tem como entrada um predicado temporal (PT) e retorna como saída uma condição temporal formada por expressões relacionais (TC). Na linha 4, a função $get\_temporal\_operator$ extrai o operador temporal do predicado temporal através da identificação da palavra-chave pertencente ao conjunto pré-definido de termos que correspondem a um operador temporal. Na linha 5, a função $get\_first\_operand$ extrai as palavras-chave convencionais, que correspondem ao primeiro operando, através da

[1]Expressões relacionais são expressões que realizam uma comparação através de operadores relacionais. Os operadores relacionais são: igual (=), maior (>), maior ou igual (>=), menor (<), menor ou igual (<=) e diferente (!=).

identificação das palavras-chave à esquerda do operador temporal. Na linha 6, a função $get\_second\_operand$ extrai o segundo operando, através da identificação das palavras-chave à direita do operador temporal. Na linha 7 é realizado o mapeamento do primeiro operando. Na linha 8 é realizado o mapeamento do segundo operando. Finalmente, na linha 9 é realizado o mapeamento do operador temporal.

Algorithm 1 Mapping Algorithm

```
1: INPUT: PT;
2: OUTPUT: TC;
3: begin
4: temporal_operator ← get_temporal_operator(PT);
5: first_operand ← get_first_operand(PT, temporal_operator);
6: second_operand ← get_second_operand(PT, temporal_operator);
7: op1 ← map_first_operand(first_operand);
8: op2 ← map_second_operand(second_operand);
9: TC ← map_temporal_operator(temporal_operator, op1, op2);
10: return TC;
11: end
```

O mapeamento do primeiro operando, realizado pela função $map\_first\_operand$, deve identificar onde a condição temporal deve ser aplicada. Esse algoritmo não é descrito, pois ele é dependente da aplicação. Por exemplo, em um motor de busca XML, a condição temporal deve ser aplicada em nodos temporais. No entanto, em um motor de busca de páginas Web, a condição temporal é aplicada nas expressões temporais do conteúdo da página, na data de coleta ou na data de criação/última alteração da página.

O Algoritmo 2 recebe como entrada um instante extraído da consulta (Ins_{efc}) e retorna como saída esse instante formatado (Ins_f). Na linha 4, o instante recebido como entrada é dividido em n partes, de acordo com a ocorrência do caractere "/". A última parte corresponde ao valor do ano (linha 5). Caso a divisão tenha resultado em mais de uma parte, a penúltima parte corresponde ao valor do mês (linhas 6-8). Caso a divisão tenha resultado em três partes, a primeira parte corresponde ao valor do dia (linhas 9-11).

Algorithm 2 Format Instant

```
1: INPUT: Ins_efc;
2: OUTPUT: Ins_f;
3: begin format_instant(Ins_efc)
4: parts[] ←split(Ins_efc, "/");
5: Ins_f.year ←parts[count(parts)];
6: if count(parts) > 1 then
7:       Ins_f.month ←parts[count(parts) − 1];
8: end if
9: if count(parts) == 3 then
10:      Ins_f.day ←parts[1];
11: end if
12: return Ins_f
13: end
```

O Algoritmo 3 recebe como entrada dois instantes ($Ins_{initial}, Ins_{final}$), respectivamente, o instante inicial e final do intervalo, e retorna como saída um intervalo bem for-

mado (Int), ou seja, um intervalo onde os valores de dia, mês e ano para os instantes inicial e final não são nulos e não violam as regras de integridade temporal. Se o mês do primeiro instante é nulo, então atribui-se a ele o primeiro mês do ano (linhas 4-6). Se o dia do primeiro instante é nulo, então atribui-se a ele o primeiro dia do mês (linhas 7-9). Se o mês do segundo instante é nulo, então atribui-se a ele o último mês do ano (linhas 10-12). Se o dia do segundo instante é nulo, então atribui-se a ele o último dia do mês naquele ano, obtido através da função get_last_day (linhas 13-15). Finalmente, o primeiro instante é atribuído como instante inicial do intervalo a ser formado (linha 16) e o segundo instante é atribuído como instante final (linha 17).

Algorithm 3 Build Interval

```
1: INPUT: Ins_initial, Ins_final;
2: OUTPUT: Int;
3: begin build_interval(Ins_initial, Ins_final)
4: if Ins_initial.month is null then
5:       Ins_initial.month ←1;
6: end if
7: if Ins_initial.day is null then
8:       Ins_initial.day ←1;
9: end if
10: if Ins_final.month is null then
11:       Ins_final.month ←12;
12: end if
13: if Ins_final.day is null then
14:       Ins_final.day ←get_last_day(Ins_final.month, Ins_final.year);
15: end if
16: Int.initial ←Ins_initial;
17: Int.final ←Ins_final;
18: return Int
19: end
```

Algorithm 4 Second Operand Mapping Algorithm

```
1: INPUT: SO_efc;
2: OUTPUT: SO_m;
3: begin map_second_operand(SO_efc)
4: if SO_efc is Simple Instant then
5:       SO_m ←map_simple_instant(SO_efc);
6: end if
7: if SO_efc is Simple Interval then
8:       SO_m ←map_simple_interval(SO_efc);
9: end if
10: if SO_efc is Composite Interval then
11:       SO_m ←map_composite_interval(SO_efc);
12: end if
13: return SO_m
14: end
```

O Algoritmo 4 invoca o algoritmo de mapeamento adequado para cada tipo de se-

gundo operando. A identificação do tipo de segundo operando é realizado por expressões regulares. Esse algoritmo tem como entrada o segundo operando extraído do predicado temporal (SO_{efc}) e retorna como saída o segundo operando mapeado para um formato de intervalo padronizado (SO_m).

O Algoritmo 5 executa o mapeamento de um instante simples. Esse algoritmo tem como entrada um segundo operando do tipo instante simples (Ins_{efc}) e retorna como saída o segundo operando mapeado para um formato de intervalo padronizado (Int_m). O instante é formatado pela função $format\_instant$ (linha 4). A função $build\_interval$ cria um intervalo que tem o instante formatado como instante inicial e final (linha 5).

Algorithm 5 Simple Instant Mapping Algorithm

```
1: INPUT: Ins_efc;
2: OUTPUT: Int_m;
3: begin map_simple_instant(Ins_efc)
4: Ins_f ←format_instant(Ins_efc);
5: Int_m ←build_interval(Ins_f, Ins_f);
6: return Int_m
7: end
```

O Algoritmo 6 executa o mapeamento de um intervalo simples. Esse algoritmo tem como entrada um segundo operando do tipo intervalo simples (Int_{efc}) e retorna como saída o segundo operando mapeado para um formato de intervalo padronizado (Int_m). O instante inicial é obtido através da função $get\_initial\_instant$ (linha 4), que extrai o segmento entre o abre colchetes (“[“) e a vírgula (“,”). O instante final é obtido através da função $get\_final\_instant$ (linha 5), que extrai o segmento entre a vírgula (“,”) e o fecha colchetes “]”. O intervalo é construído através da função $build\_interval$ (linha 6), tendo como parâmetros os instantes obtidos.

Algorithm 6 Simple Interval Mapping Algorithm

```
1: INPUT: Int_efc;
2: OUTPUT: Int_m;
3: begin map_simple_interval(Int_efc)
4: Ins_initial ←get_initial_instant(Int_efc);
5: Ins_final ←get_final_instant(Int_efc);
6: Int_m ←build_interval(Ins_initial, Ins_final);
7: return Int_m
8: end
```

O Algoritmo 7 executa o mapeamento de um intervalo composto. Esse algoritmo tem como entrada um segundo operando do tipo intervalo composto (CI_{efc}) e retorna como saída o segundo operando mapeado para um formato de intervalo padronizado (Int_m). A granularidade é extraída pela função $get\_granularity$ (linha 4), através da identificação da palavra-chave pertencente ao conjunto pré-definido de termos que correspondem a uma granularidade. Na linha 5, a palavra reservada é extraída pela função $get\_reserved\_word$, através da identificação da palavra-chave pertencente ao conjunto pré-definido de termos que correspondem a uma palavra reservada. Na linha 6, o valor é extraído pela função get_value, extraindo o valor inteiro presente no intervalo composto. Caso esse valor não esteja disponível, atribui-se o valor 1. Em seguida, é chamada a função responsável pelo mapeamento de acordo com a palavra reservada (linhas 7-15).

Algorithm 7 Composite Interval Mapping Algorithm

1: INPUT: CI_{efc};
2: OUTPUT: Int_m;
3: **begin** *map_composite_interval(CI_{efc})*
4: $gran \leftarrow$*get_granularity(CI_{efc})*;
5: $word \leftarrow$*get_reserved_word(CI_{efc})*;
6: $value \leftarrow$*get_value(CI_{efc})*;
7: **if** $word$ is current **then**
8: $Int_m \leftarrow$*map_current($gran$)*;
9: **end if**
10: **if** $word$ is last **then**
11: $Int_m \leftarrow$*map_last($gran, value$)*;
12: **end if**
13: **if** $word$ is next **then**
14: $Int_m \leftarrow$*map_next($gran, value$)*;
15: **end if**
16: **return** Int_m
17: **end**

Os algoritmos 8, 9 e 10 executam o mapeamento de um intervalo composto de acordo com cada palavra reservada, respectivamente, `current`, `last` e `next`. Esses algoritmos tratam as granularidades da seguinte forma. São definidas três granularidades básicas: dia, mês e ano. Qualquer outra granularidade maior deve ser decomposta para uma das granularidades básicas. Nenhuma granularidade menor que as granularidades básicas é permitida. Para a realização da decomposição, dois conceitos são utilizados: `calendar type` e `base`. O `calendar type` define para qual granularidade básica a granularidade em questão deve ser decomposta. A `base` define o número de unidades da granularidade básica que compõem a granularidade em questão. Por exemplo, a granularidade semestre deve ser decomposta para a granularidade mês (`calendar type`) e é composta por 6 (`base`) meses. A Tabela 5.1 apresenta a lista completa das granularidades especificadas juntamente com o `calendar type` e `base` de cada uma.

Tabela 5.1: Granularidades suportadas e informações para sua decomposição

Granularidade	Calendar Type	Base
Dia	Dia	1
Semana	Dia	7
Mês	Mês	1
Semestre	Mês	6
Ano	Ano	1
Década	Ano	10
Século	Ano	100

Foram especificadas três granularidades básicas, em vez de decompor todas as granularidades para a menor granularidade (dia), pois algumas granularidades possuem duração de dias variável. Por exemplo, últimos 2 meses não são necessariamente os últimos 60 dias, uma vez que alguns meses possuem 31 dias, outros 30 e fevereiro pode possuir 28 ou 29 dias, dependendo do ano.

Algorithm 8 Current Mapping Algorithm

```
1: INPUT: gran;
2: OUTPUT: Int_m;
3: begin map_current(gran)
4: if gran.calendarType is year then
5:       Int_m.initial.year ←floor(current_year/gran.base)*gran.base;
6:       if gran.name is century then
7:             Int_m.initial.year ←floor((current_year − 1)/gran.base)*gran.base;
8:             Int_m.initial.year ←Int_m.initial.year+1;
9:       end if
10:      Int_m.initial.month ← Int_m.initial.day ←1;
11:      Int_m.final.year ←Int_m.initial.year+gran.base-1;
12:      Int_m.final.month ←12;
13:      Int_m.final.day ←get_last_day(Ins_final.month, Ins_final.year);
14: end if
15: if gran.calendarType is month then
16:      Int_m.initial.year ← current_year;
17:      Int_m.initial.month ←floor((current_month − 1)/gran.base)*gran.base;
18:      Int_m.initial.month ←Int_m.initial.month+1;
19:      Int_m.initial.day ←1;
20:      Int_m.final.year ← Int_m.initial.year;
21:      Int_m.final.month ←Int_m.initial.month+gran.base-1;
22:      Int_m.final.day ←get_last_day(Ins_final.month, Ins_final.year);
23: end if
24: if gran.calendarType is day then
25:      Int_m.initial.year ← current_year;
26:      Int_m.initial.month ← current_month;
27:      Int_m.initial.day ← current_day;
28:      if gran.name is week then
29:            Int_m.initial.day ←current_day − get_day_of_week(current_date);
30:      end if
31:      Int_m.final.year ← Int_m.initial.year;
32:      Int_m.final.month ← Int_m.initial.month;
33:      Int_m.final.day ←Int_m.initial.day+gran.base-1;
34: end if
35: return Int_m
36: end
```

O Algoritmo 8 executa o mapeamento de intervalos compostos cuja palavra reservada é `current`. Esse algoritmo tem como entrada uma granularidade ($gran$) e retorna o intervalo atual de acordo com a granularidade informada (Int_m). Nas linhas 4-14 são tratadas as granularidades cujo `calendar type` é ano. Na linha 5 obtém-se o quociente do ano corrente pela `base`. Esse quociente é arredondado para baixo e, em seguida, multiplicado pela `base`. O resultado obtido é atribuído ao ano do instante inicial. Nas linhas 6-9 é realizado um tratamento especial para a granularidade século, devido ao fato do século iniciar no ano 1 e não no ano 0. Por exemplo, o século XX inicia no ano 1901 e termina no ano 2000. Na linha 10 atribui-se o primeiro mês do ano para o mês do instante inicial e o primeiro dia do primeiro mês do ano para o dia do instante inicial. Na linha

11 calcula-se a soma entre o ano do instante inicial e a `base`. Em seguida, obtém-se a diferença entre o valor da soma e 1. O resultado obtido é atribuído ao ano do instante final. Nas linhas 12 e 13 atribui-se o último mês do ano para o mês do instante final e o último dia do último mês do ano para o dia do instante final.

Nas linhas 15-23 são tratadas as granularidades cujo `calendar type` é mês. O ano do instante inicial recebe o valor do ano corrente (linha 16). Na linha 17 obtém-se o quociente da diferença entre o mês atual e 1 pela `base`. Esse quociente é arredondado para baixo e, em seguida, multiplicado pela `base`. O resultado obtido é atribuído ao mês do instante inicial. A subtração por 1 realizada na linha 17 e a adição de 1 na linha 18 devem-se ao fato de que os meses iniciam em 1. O valor do dia do instante inicial é definido como o primeiro dia do mês (linha 19). Na linha 20 atribui-se ao ano do instante final o ano do instante inicial. Na linha 21 calcula-se a soma entre o mês do instante inicial e a `base`. Em seguida, obtém-se a diferença entre o valor da soma e 1. O resultado obtido é atribuído ao mês do instante final. Na linha 22 atribui-se ao dia do instante final o último dia do mês do instante final no ano do instante final.

Nas linhas 24-34 são tratadas as granularidades cujo `calendar type` é dia. O ano, mês e dia do instante inicial recebem, respectivamente, os valores de ano, mês e dia correntes (linha 25-27). As linhas 28-30 apresentam um tratamento especial para a granularidade semana, uma vez que essa granularidade não tem um dia fixo de início. Esse tratamento constitui-se em atribuir ao dia do instante inicial a diferença entre o dia corrente e o valor retornado pela função $get\_day\_of\_week$. Essa função retorna 0 se a data passada por parâmetro corresponde ao primeiro dia da semana, 1 se a data passada por parâmetro corresponde ao segundo dia da semana, e assim sucessivamente até 6 para o último dia da semana. Nas linhas 31 e 32 atribui-se, respectivamente, ao ano e mês do instante final os valores de ano e mês do instante inicial. Na linha 33 calcula-se a soma entre o dia do instante inicial e a `base`. Em seguida, obtém-se a diferença entre o valor da soma e 1. O resultado obtido é atribuído ao dia do instante final.

Algorithm 9 Last Mapping Algorithm

```
1: INPUT: gran, value;
2: OUTPUT: Int_m;
3: begin map_last(gran, value)
4: Ins_final ←current_date;
5: Ins_initial ←map_current(gran).initial;
6: if gran.calendarType is year then
7:       Ins_initial.year ← Ins_initial.year − ((value − 1) ∗ gran.base);
8: end if
9: if gran.calendarType is month then
10:      Ins_initial.month ← Ins_initial.month − ((value − 1) ∗ gran.base);
11: end if
12: if gran.calendarType is day then
13:      Ins_initial.day ← Ins_initial.day − ((value − 1) ∗ gran.base);
14: end if
15: Int_m ←build_interval(Ins_initial, Ins_final);
16: return Int_m
17: end
```

O Algoritmo 9 executa o mapeamento de intervalos compostos cuja palavra reservada é `last`. Esse algoritmo tem como entrada uma granularidade ($gran$) e um valor ($value$)

e retorna como saída o intervalo (Int_m) de acordo com um deslocamento para o passado, no tamanho de $value$, de acordo com a granularidade $gran$. O instante final recebe a data atual (linha 4). O instante inicial recebe o instante inicial do intervalo atual, obtido pela função $map\_current$ (linha 5). As linhas 6-14 tratam cada um dos `calendar type`. Nesse tratamento, multiplica-se a diferença entre o valor recebido por parâmetro e 1, pela `base`. Em seguida, obtém-se a diferença entre o valor da granularidade X do instante inicial e o resultado da multiplicação. O valor obtido é atribuído à granularidade X do instante inicial, onde X é o `calendar type`. É necessário subtrair 1 do valor especificado para descontar o valor corrente. Por exemplo, a expressão "`últimos 2 meses`" cobre o mês corrente e o mês passado. Logo, o deslocamento deve ser de um mês. Finalmente, na linha 15 é construído o intervalo, através da função $build\_interval$ com o instante inicial e final como parâmetros.

O Algoritmo 10 executa o mapeamento de intervalos compostos cuja palavra reservada é `next`. Esse algoritmo tem como entrada uma granularidade ($gran$) e um valor ($value$) e retorna como saída o intervalo (Int_m) de acordo com um deslocamento para o futuro, no tamanho de $value$, de acordo com a granularidade $gran$. O instante inicial recebe a data atual (linha 4). O instante final recebe o instante final do intervalo atual, obtido pela função $map\_current$ (linha 5). As linhas 6-14 tratam cada um dos `calendar type`. Nesse tratamento, multiplica-se o valor recebido por parâmetro pela `base`. Em seguida, obtém-se a soma entre o valor da granularidade X do instante final e o resultado da multiplicação. O valor obtido é atribuído à granularidade X do instante final, onde X é o `calendar type`. Finalmente, na linha 15 é construído o intervalo, através da função $build\_interval$, com o instante inicial e final como parâmetros.

Algorithm 10 Next Mapping Algorithm

1: INPUT: $gran, value$;
2: OUTPUT: Int_m;
3: **begin** *map_next(*$gran, value$*)*
4: $Ins_{initial}$ ←*current_date*;
5: Ins_{final} ←*map_current(*$gran$*).final*;
6: **if** $gran.calendarType$ is year **then**
7: $Ins_{final}.year \leftarrow Ins_{final}.year + (value * gran.base)$;
8: **end if**
9: **if** $gran.calendarType$ is month **then**
10: $Ins_{final}.month \leftarrow Ins_{final}.month + (value * gran.base)$;
11: **end if**
12: **if** $gran.calendarType$ is day **then**
13: $Ins_{final}.day \leftarrow Ins_{final}.day + (value * gran.base)$;
14: **end if**
15: Int_m ←*build_interval(*$Ins_{initial}, Ins_{final}$*)*;
16: **return** Int_m
17: **end**

O algoritmo de mapeamento do operador temporal é realizado de acordo com a Tabela 5.2, considerando que operadores relacionais possuem precedência maior que os operadores lógicos. Esse algoritmo tem como entrada o operador temporal, o primeiro operando ($op1$) e o segundo operando ($op2$) e retorna como saída uma condição temporal formada por expressões relacionais. Ressalta-se que $op1$ e $op2$ são intervalos e possuem um instante inicial (`InitialInstant`) e um instante final (`FinalInstant`). Além

disso, há a função `getChronon` que retorna a duração da menor unidade suportada, ou seja, um dia.

Tabela 5.2: Mapeamento dos operadores temporais

Operador	Mapeamento
After	`op1.InitialInstant > (op2.FinalInstant + getChronon())`
Before	`op1.FinalInstant < (op2.InitialInstant - getChronon())`
Contains	`(op2.InitialInstant > op1.InitialInstant AND op2.FinalInstant < op1.FinalInstant)`
During	`(op1.InitialInstant > op2.InitialInstant AND op1.FinalInstant < op2.FinalInstant)`
Equals	`op1.InitialInstant = op2.InitialInstant AND op1.FinalInstant = op2.FinalInstant`
FinishedBy	`op2.InitialInstant > op1.InitialInstant AND op2.FinalInstant = op1.FinalInstant`
Finish	`op1.InitialInstant > op2.InitialInstant AND op1.FinalInstant = op2.FinalInstant`
Intersect	`op1.InitialInstant <= op2.FinalInstant AND op1.FinalInstant >= op2.InitialInstant`
Meets	`op1.FinalInstant + getChronon() = op2.InitialInstant`
MetBy	`op2.FinalInstant + getChronon() = op1.InitialInstant`
OverlappedBy	`op2.InitialInstant < op1.InitialInstant AND op2.FinalInstant < op1.FinalInstant AND op2.FinalInstant > op1.InitialInstant`
Overlaps	`op1.InitialInstant < op2.InitialInstant AND op1.FinalInstant < op2.FinalInstant AND op1.FinalInstant > op2.InitialInstant`
StartedBy	`op2.InitialInstant = op1.InitialInstant AND op2.FinalInstant < op1.FinalInstant`
Starts	`op1.InitialInstant = op2.InitialInstant AND op1.FinalInstant < op2.FinalInstant`

A Figura 5.5 apresenta o diagrama de classes de TKC simplificado. O diagrama completo é apresentado no Apêndice B. A classe `TemporalPredicate` modela o predicado temporal. O mapeamento do predicado temporal é executado pelo método `map`. A entrada desse método é um predicado temporal que foi previamente extraído da consulta temporal. O predicado temporal é dividido em três fragmentos (operador, primeiro operando e segundo operando) e, então, cada fragmento é separadamente tratado em suas próprias classes. A classe `FirstOperand` modela o primeiro operando. Foram definidos dois métodos (`getInitialInstant()` e `getFinalInstant()`) para obter, respectivamente, o instante inicial e final dos intervalos sobre os quais deve-se aplicar a condição temporal. A classe `TemporalOperator` modela o operador temporal. A relação temporal entre o primeiro e o segundo operando é mapeada para expressões relacionais pelo método `getMapping()`. Também, foi especificado o método *getChronon* que retorna a duração da menor unidade de tempo suportada. A classe `SecondOperand` modela o segundo operando. Foi definido o método `getInterval()` para obter o intervalo que representa o tempo desejado como filtro da consulta. Os instantes inicial e final desse intervalo podem ser obtidos, respectivamente, pelos métodos `getInitialInstant()` e `getFinalInstant()`.

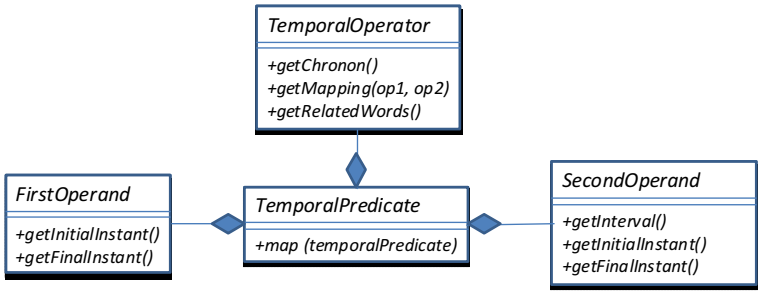


Figura 5.5: Diagrama de classes de TKC simplificado

5.4 Considerações Finais

Este capítulo apresentou TKC, que é uma classificação genérica que serve de guia para qualquer mecanismo de consulta temporal por palavras-chave. Foram apresentados a visão geral para o formato de expressar consultas temporais por palavras-chave, a especificação da sintaxe da classificação e os algoritmos de mapeamento dos predicados temporais, definidos por palavras-chave, para expressões relacionais.

A principal contribuição desse capítulo é especificar a classificação TKC, que define os requisitos que um mecanismo de consulta por palavras-chave deve considerar com relação às consultas temporais. O diferencial de TKC é o fato de ser voltado para consultas temporais por palavras-chave, enquanto os trabalhos relacionados destinam-se a consultas temporais em linguagens estruturadas. Além disso, nossa proposta não altera o formato das consultas por palavras-chave padrão.

TKC traz contribuições tanto para a área de gerenciamento de dados temporais quanto para a área de recuperação de informação. Enquanto que para a área de gerenciamento de dados temporais a contribuição é a especificação dos requisitos temporais em consultas por palavras-chave, para recuperação de informação a contribuição são os algoritmos de mapeamento que servem de guia para a implementação dos requisitos temporais em mecanismos de consulta por palavras-chave, como, por exemplo, motores de busca Web.

A Tabela 5.3 apresenta um comparativo entre TKC e os trabalhos de consulta temporal apresentados nos trabalhos relacionados. O principal diferencial de TKC é o fato de ser aplicado em consultas por palavras-chave, enquanto os trabalhos relacionados suportam consultas temporais em linguagens estruturadas. Além disso, TKC tem uma abrangência maior em relação aos operadores temporais. TKC cobre todas as 13 relações de Allen e ainda adiciona o operador `intersect`, que combina as relações de Allen, fazendo uma generalização. Suportar operadores de agregação e a operação *coalesce* são trabalhos futuros.

Tabela 5.3: Comparativo entre TKC e trabalhos relacionados: consultas temporais

	Tipo de Consulta	Operadores Temporais	Agregação	*Coalesce*
TXPath	Estruturada	3	Sim	Sim
TXQuery	Estruturada	0	Sim	Sim
T-Yago	Estruturada	6	Não	Não
TKC	Palavras-chave	14	Não	Não

6 TWO PHASE INTERCEPTION (TPI)

Este capítulo apresenta **TPI** - ***Two Phase Interception*** - que é uma abordagem proposta para suportar consultas temporais por palavras-chave sobre documentos XML. São apresentadas a visão geral e a especificação de seus componentes, que incluem um motor de busca XML, um processador temporal, um índice convencional e um índice temporal. A principal contribuição de TPI é realizar o tratamento adequado das informações temporais presentes nas consultas e nos documentos XML, melhorando a qualidade dos resultados de consultas temporais por palavras-chave sobre documentos XML.

Esse capítulo está organizado da seguinte forma. Na Seção 6.1 é apresentada a visão geral de TPI, juntamente com um exemplo de execução de uma consulta temporal processada por um motor de busca XML convencional com a utilização de TPI e sem a utilização de TPI, a fim de evidenciar sua contribuição. A Seção 6.2 detalha a criação do índice temporal, utilizado pelo processador temporal. O processador temporal é detalhado na Seção 6.3. Por fim, são expostas as considerações finais e um comparativo de TPI com os trabalhos relacionados.

A proposta inicial do trabalho foi publicada em (MANICA; GALANTE, 2009). Os detalhes do processo de indexação temporal foram publicados em (MANICA; GALANTE; DORNELES, 2010a). Uma visão geral de TPI foi publicada em (MANICA; GALANTE; DORNELES, 2010b). Uma versão completa de TPI foi publicada em (MANICA; DORNELES; GALANTE, 2010).

6.1 Visão Geral

TPI suporta consultas temporais por palavras-chave, que seguem a classificação TKC, sobre documentos XML. Basicamente, esse suporte é realizado através de duas interceptações no processamento da consulta, realizado por um motor de busca XML convencional. A primeira interceptação (Figura 6.1 (1)) extrai da consulta a parte temporal. A segunda interceptação (Figura 6.1 (2)) devolve ao motor de busca XML convencional a parte temporal da consulta já manipulada por um processador temporal.

A Figura 6.1 apresenta a arquitetura de TPI. O documento XML representa uma base de dados XML orientada a dados. Os quatro componentes principais são:

- `Motor de Busca XML Convencional` - representa um motor de busca XML convencional, que não possui tratamento especial da informação temporal. Esse motor de busca é responsável por encontrar os casamentos para as palavras-chave não temporais e por definir os nodos XML que devem ser retornados. TPI é independente do motor de busca XML convencional utilizado. Para clareza do

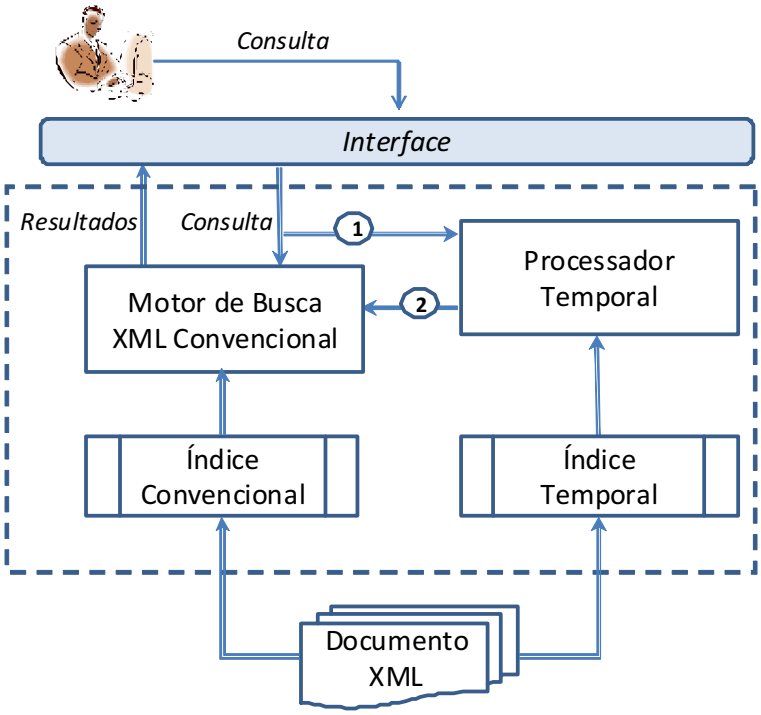


Figura 6.1: Visão geral de TPI

texto, no decorrer deste capítulo, um motor de busca XML convencional é referenciado simplesmente como motor de busca;

- `Processador Temporal` - representa uma camada adicional de software responsável pela manipulação da parte temporal da consulta. Essa manipulação inclui mapear o predicado temporal da consulta para uma condição temporal e identificar os nodos temporais que satisfazem essa condição temporal;

- `Índice Convencional` - representa o(s) índice(s) exigido(s) pelo motor de busca. Basicamente, cada nodo XML é indexado, recebendo um identificador de acordo com o conceito de *Dewey Label*. Esse identificador permite detectar a ordem dos nodos, nodos irmãos, nodos antecessores e nodos descendentes;

- `Índice Temporal` - representa o índice utilizado pelo processador temporal. Esse índice permite ao processador temporal ter um acesso rápido, consistente e padronizado às informações temporais presentes no documento XML. O índice temporal armazena os valores temporais presentes no documento XML no formato de um intervalo temporal. Cada intervalo temporal aponta para o nodo elemento ou nodo atributo que representa o LCA dos nodos texto que compõem o intervalo temporal.

6.1.1 Exemplificando o processamento de uma consulta temporal

Nesta subseção a execução de TPI é exemplificada passo a passo para deixar evidente a diferença entre o *baseline* e TPI. Inicialmente, é apresentada a execução de TPI para uma consulta temporal, bem como os resultados retornados. Em seguida, é apresentada a execução do *baseline* para a mesma consulta temporal, bem como os resultados retornados.

Considere a consulta Q6.1, como exemplo. O objetivo de Q6.1 é retornar os títulos dos livros publicados em 15 de março de 1998. Considere, também, o documento XML Books, apresentado na Figura 6.2, que armazena informações sobre livros.

```
Q6.1 - book title intersect 15/03/1998
```

O índice convencional é representado pelos identificadores (ID) apresentados nos nodos na Figura 6.2. O documento XML Books possui dois instantes temporais: 15/03/1998 e 03/06/1998. No índice temporal (Figura 6.3), esses instantes são representados como intervalos que iniciam e terminam no mesmo instante: [15/03/1998, 15/03/1998] e [03/06/1998, 03/06/1998]. O primeiro instante aponta para o nodo 0.0.2, que é o LCA dos nodos que formam o instante (0.0.2.0.0, 0.0.2.1.0 e 0.0.2.2.0). O segundo instante aponta para o nodo 0.1.2, que é o LCA dos nodos que formam o instante (0.1.2.0.0, 0.1.2.1.0 e 0.1.2.2.0).

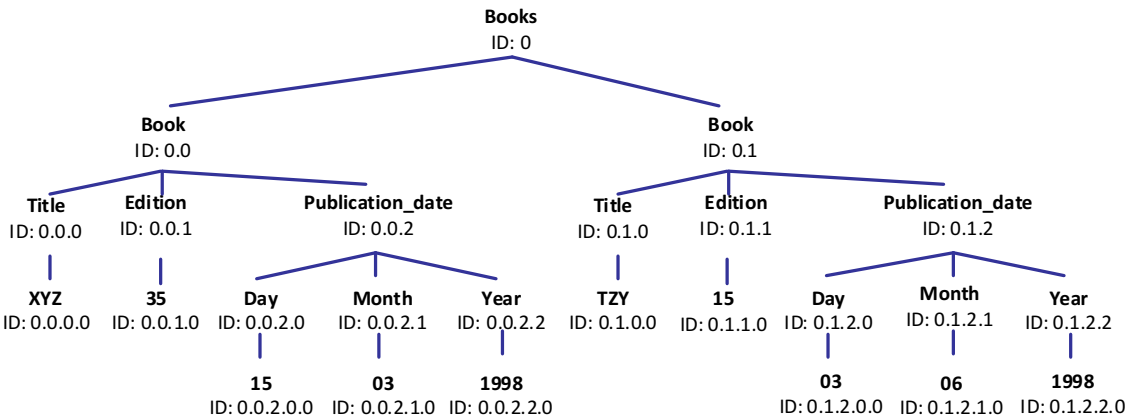


Figura 6.2: Documento XML Books

Instante Inicial			Instante Final			Dewey Label
Dia	Mês	Ano	Dia	Mês	Ano	
15	03	1998	15	03	1998	0.0.2
03	06	1998	03	06	1998	0.1.2

Figura 6.3: Índice temporal do documento XML Books

Inicialmente o usuário submete uma consulta temporal por palavras-chave para uma interface de consultas (por exemplo, a consulta Q6.1). A consulta é enviada para o motor de busca e, antes que ela seja processada, ocorre a **primeira interceptação (1)**. A primeira interceptação extrai da consulta a parte temporal ("intersect 15/03/1998") e a envia para o processador temporal. As palavras-chave não temporais ("book title") seguem para o motor de busca.

O motor de busca encontra os nodos XML que casam com as palavras-chave não temporais (nodos 0.0 e 0.1 para a palavra-chave book e nodos 0.0.0 e 0.1.0 para a palavra-chave title).

O processador temporal mapeia o predicado temporal ("intersect 15/03/1998") para uma condição temporal "instante inicial <=

15/03/1998 e instante final >= 15/03/1998". Em seguida, o processador temporal busca no índice temporal os nodos que satisfazem a condição temporal (nodo 0.0.2).

Quando o processador temporal conclui a busca pelos nodos que satisfazem a condição temporal ocorre a **segunda interceptação (2)**. A segunda interceptação adiciona na lista de identificadores do motor de busca, que contém os identificadores dos nodos que casam com as palavras-chave não temporais, os identificadores dos nodos que satisfazem a condição temporal. A partir desse ponto, o motor de busca define os nodos XML a serem retornados, através da lista de identificadores, sem nenhuma intervenção. Os identificadores dos nodos que satisfazem a condição temporal da consulta são tratados como se fossem casamentos para uma palavra-chave da consulta fictícia (tempcond). A Figura 6.4 apresenta os casamentos para cada palavra-chave após a inclusão dos identificadores dos nodos que satisfazem a condição temporal.

```
book: 0.0, 0.1
title: 0.0.0, 0.1.0
tempcond: 0.0.2
```

Figura 6.4: Casamentos para a consulta Q6.1 após as interceptações de TPI

A Figura 6.5 apresenta os resultados para a consulta Q6.1 de acordo com XSeek utilizando as interceptações de TPI. XSeek é o motor de busca utilizado nos experimentos. Apenas são retornados nodos referentes ao livro XYZ, pois esse livro foi publicado em 15/03/1998, que corresponde a data desejada. Por outro lado, nenhum nodo do livro TZY foi retornado, pois sua publicação é posterior à data desejada, logo, esse livro não possui nenhum nodo que satisfaça o predicado temporal. É importante notar que haviam casamentos para as palavras-chave book e title na subárvore referente ao livro TZY (nodos 0.1 e 0.1.0, respectivamente), porém não havia nenhum casamento nessa subárvore para a palavra-chave tempcond (palavra-chave fictícia que representa o predicado temporal), logo, os casamentos 0.1 e 0.1.0 são descartados.

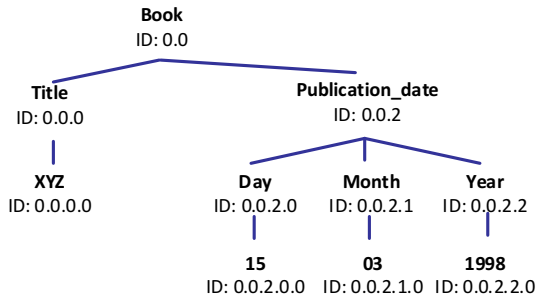


Figura 6.5: Resultado da consulta Q6.1 de acordo com XSeek utilizando as interceptações de TPI

A seguir é apresentada a execução da consulta Q6.1 diretamente no motor de busca XSeek, sem as interceptações realizadas por TPI. Isso é realizado para justificar a contribuição de TPI.

Uma vez que XSeek não possui tratamento especial para a informação temporal da consulta, a estrutura da consulta deve ser alterada para "book title 15 03 1998". Primeiramente, são encontrados os casamentos para cada palavra-chave. A Figura 6.6

apresenta os casamentos para a consulta `Q6.1` encontrados por XSeek sem as interceptações de TPI.

```
book: 0.0, 0.1
title: 0.0.0, 0.1.0
15: 0.0.2.0.0, 0.1.1.0
03: 0.0.2.1.0, 0.1.2.0.0
1998: 0.0.2.2.0, 0.1.2.2.0
```

Figura 6.6: Casamentos para a consulta `Q6.1` encontrados de acordo com XSeek sem as interceptações de TPI

A Figura 6.7 apresenta os resultados para a consulta `Q6.1` de acordo com XSeek sem as interceptações de TPI. São retornados nodos do livro `XYZ` e nodos do livro `TZY`. No entanto, o livro `TZY` foi publicado em 3 de julho de 1998. Portanto, não deve ser retornado, uma vez que foi publicado após a data desejada. Esse erro ocorre porque XSeek encontra um casamento para a palavra-chave `15`, que refere-se ao valor do dia, no valor da edição do livro `TZY` (nodo `0.1.1.0`). Além disso, encontra um casamento para a palavra-chave `03`, que refere-se ao valor do mês, no valor do dia da publicação do livro `TZY` (nodo `0.1.2.0.0`). Isso se deve ao fato dos motores de busca XML presentes na literatura tratarem a informação temporal presente na consulta como uma palavra-chave qualquer. Com isso, comprova-se a importância da aplicação de TPI para consultas temporais por palavras-chave sobre documentos XML.

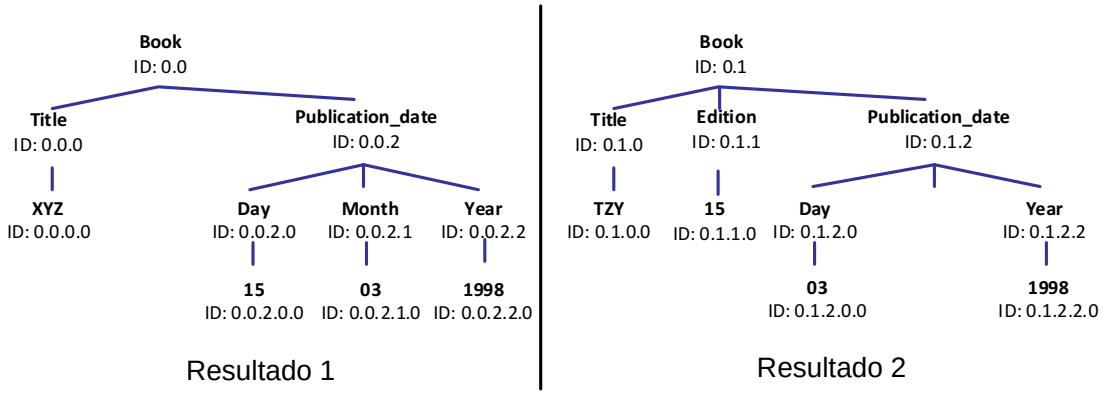


Figura 6.7: Resultado da consulta `Q6.1` de acordo com XSeek sem as interceptações de TPI

6.2 Indexação Temporal

O principal objetivo da indexação temporal é armazenar os valores temporais, estruturados das mais diversas formas no documento XML, em um índice de uma forma padronizada. Esse índice permite um acesso rápido e consistente aos dados temporais durante a execução de uma consulta temporal. Cada valor temporal é representado como um intervalo temporal. Cada intervalo temporal aponta para o nodo elemento ou nodo atributo que representa o LCA dos nodos texto que compõem o intervalo temporal.

Considere o documento XML `Pessoa` apresentado na Figura 6.8. Esse documento possui três valores temporais: `Nascimento`, `Casamento` e `Graduação`. Cada valor temporal está estruturado de uma forma diferente. O tempo referente ao nascimento é

caracterizado por um instante representado através de três nodos (`0.1.0.0`, `0.1.1.0` e `0.1.2.0`). Esse tipo de instante é referenciado ao longo desta seção como instante fragmentado. O tempo referente ao casamento é caracterizado por um instante representado através de um único nodo (`0.2.0`). O tempo referente à graduação é caracterizado por um intervalo representado através de dois nodos (`0.3.0.0` e `0.3.1.0`).

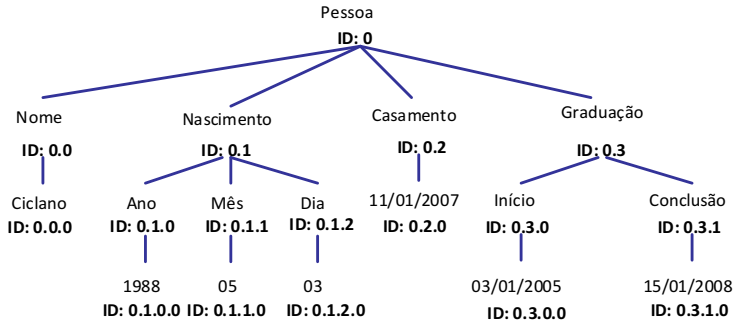


Figura 6.8: Documento XML `Pessoa`

A Figura 6.9 apresenta o índice temporal construído para o documento XML `Pessoa`. A Linha 01 refere-se ao valor temporal do nascimento. A Linha 02 refere-se ao valor temporal do casamento. A linha 03 refere-se ao valor temporal da graduação. Ressalta-se que instantes são representados como um intervalo que inicia e termina no mesmo instante. Cada intervalo temporal aponta para o nodo elemento ou nodo atributo que representa o LCA dos nodos que compõem o valor temporal.

	Instante Inicial			Instante Final			Dewey Label
	Dia	Mês	Ano	Dia	Mês	Ano	
Linha 01	03	05	1988	03	05	1988	0.1
Linha 02	11	01	2007	11	01	2007	0.2
Linha 03	03	01	2005	15	01	2008	0.3

Figura 6.9: Índice temporal do documento XML `Pessoa`

A indexação temporal possui cinco etapas distintas (Figura 6.10): `Identificação de Caminhos Temporais`, `Identificação de Instantes Fragmentados`, `Identificação de Intervalos`, `Normalização de Expressões Temporais` e `Indexação de Expressões Temporais`, conforme detalhado nas subseções seguintes. É importante notar que a indexação temporal é realizada uma única vez para cada documento XML e sua execução ocorre em modo *off-line*.

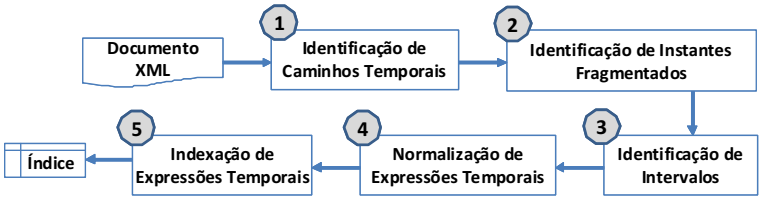


Figura 6.10: Indexação temporal

6.2.1 Identificação de Caminhos Temporais

A identificação de caminhos temporais tem como objetivo verificar quais caminhos XML possuem informações temporais. Além disso, é necessário identificar o formato dos valores temporais de cada caminho temporal através da análise do comportamento do caminho XML. Entende-se como análise do comportamento do caminho XML a análise de todos os valores do caminho a fim de identificar formatos, violações de regras de integridade, etc.

A Tabela 6.1 ilustra alguns exemplos de caminhos temporais. Para cada caminho temporal é definido um domínio e um formato. O documento XML `A` tem duas expressões temporais agrupadas em um mesmo caminho temporal. A vantagem de analisar caminhos temporais em vez de analisar ocorrências isoladas das expressões temporais é a possibilidade de verificação de um comportamento padrão no caminho temporal. Portanto, realiza-se a desambiguação de formato para muitas expressões temporais. Por exemplo, analisando isoladamente a primeira expressão temporal do documento XML `A` (`12/03/1997`) não é possível descobrir se o seu formato é "`DD/MM/YYYY`" ou "`MM/DD/YYYY`", ou seja, seu formato é ambíguo. Porém, analisando o comportamento do caminho temporal (`/clinic/vaccines/vaccine/date`) observa-se que há uma expressão temporal pertencente a ele (`28/01/2009`) que não possui ambiguidade de formato. Logo, infere-se que o formato de todas as expressões temporais desse caminho é "`DD/MM/YYYY`".

Tabela 6.1: Exemplos de caminhos temporais

Documento	Caminho Temporal	Expressões Temporais	Domínio	Formato
A	/clinic/vaccines/vaccine/date	12/03/1997, 28/01/2009	DATE	DMY4
B	/clinic/vaccine/when/day	12	DAY	D
B	/clinic/vaccine/when/month	01	MONTH	M
B	/clinic/vaccine/when/year	1998	YEAR	Y4
C	/clinic/hospitalization/patient/entrance	23/11/2007	DATE	DMY4
C	/clinic/hospitalization/patient/departure	24/11/2007	DATE	DMY4

Podem existir bases de dados que possuam caminhos temporais com todos os valores temporais ambíguos, impossibilitando a desambiguação de formato. Nesse caso, todos os formatos possíveis para o conjunto de valores são indexados. Por exemplo, se o caminho temporal que contém o valor `05/12/2010` não tiver seu formato desambiguado, esse valor será indexado duas vezes, uma como `05/dezembro/2010` e outra como `12/maio/2010`.

6.2.1.1 Arquivo de Configuração

O arquivo de configuração é um arquivo XML com regras genéricas para inferir caminhos temporais e seus formatos. A Figura 6.11 apresenta o DTD que especifica a estrutura do arquivo de configuração. Identificam-se três elementos principais (linha 01): (i) `candidates`; (ii) `eliminations`; e (iii) `decision_tree`.

O elemento `candidates` (linha 02) define expressões regulares, através do atributo `regex` de cada elemento `candidate` (linha 04), que indicam os caminhos temporais candidatos. O elemento `eliminations` (linha 05) define quais caminhos temporais candidatos devem ser eliminados, ou seja, exceções para as regras definidas no elemento `candidates`. Por exemplo, considere a regra "*caminhos cujos valores são numéricos*", que expressa candidatos a serem avaliados como caminho temporal. Nesse caso, uma exceção seria "*se o nome do nodo contém a substring preço*". A definição de `candidates`

```
01 <!ELEMENT cfg ((candidates, eliminations, decision_tree))>
02 <!ELEMENT candidates ((candidate+))>
03 <!ELEMENT candidate EMPTY>
04 <!ATTLIST candidate regex CDATA #IMPLIED>
05 <!ELEMENT eliminations ((condition+))>
06 <!ELEMENT decision_tree ((condition+))>
07 <!ELEMENT condition ((condition+ | annotation)?)>
08 <!ATTLIST condition test (values | tag ) #REQUIRED
09                     op1_f CDATA #REQUIRED
10                     op1_args CDATA #IMPLIED
11                     op2 CDATA #IMPLIED
12                     operator (equals | lessthan |
13                                   greaterthan) #IMPLIED
14                     abrag (some | none | every) #IMPLIED>
15 <!ELEMENT annotation EMPTY>
16 <!ATTLIST annotation format CDATA #REQUIRED
17                      type CDATA #REQUIRED>
```

Figura 6.11: DTD do arquivo de configuração

e `eliminations` funciona como um filtro rápido para reduzir o número de caminhos XML que devem ser analisados em detalhes a fim de diminuir o custo de processamento.

O elemento `decision_tree` (linha 06) é a raiz de uma árvore de decisão que define um conjunto de condições para que determinado caminho XML seja considerado temporal. As classes da árvore de decisão definem se o nodo é temporal ou não. Além disso, definem o domínio e formato dos valores do caminho, caso ele seja temporal.

O elemento `annotation` (linha 15) é utilizado para definir a classe da árvore de decisão. Seus atributos `type` e `format` representam o domínio e o formato dos valores do caminho temporal, respectivamente. Se esses atributos estiverem vazios, significa que o caminho não é temporal.

O elemento `condition` (linha 08) é utilizado para definir as condições da árvore de decisão, que são operações binárias. O elemento `condition` possui os seguintes atributos:

- `test` - define se a condição deve ser aplicada sobre o nome (`tag`) ou sobre os valores do caminho (`values`);
- `abrag` - define se a condição deve ser verdadeira para: (i) pelo menos um valor do caminho (`some`); (ii) todos os valores do caminho (`every`); ou (iii) nenhum valor do caminho (`none`) . Esse atributo só deve ser definido quando o valor do atributo `test` for `values`;
- `op1_f` - define uma função cujo valor de retorno representa o valor do primeiro operando da condição. As funções disponíveis incluem todas as funções de XQuery e a função `nesting`. A função `nesting` foi definida para permitir que o retorno de uma função seja direcionado como parâmetro de outra função;
- `op1_args` - define os parâmetros da função definida no atributo `op1_f`;
- `operator` - define o operador da condição;
- `op2` - define o valor do segundo operando da condição.

A Figura 6.12 ilustra um pequeno exemplo de arquivo de configuração[1]. A linha 03 define que são caminhos temporais candidatos os caminhos onde todos os seus valores contêm apenas dígitos e separadores (“-”, espaço, ou “/”) ou que contêm parte do nome ou sigla de meses do ano. Nas linhas 05-08 são definidas as eliminações de candidatos temporais, sendo eliminados os candidatos temporais cujo nome contém o termo `price` (linha 06) ou o termo `id` (linha 07). Nas linhas 09-20 é definida a árvore de decisão composta por:

```
01.<cfg>
02.  <candidates>
03.    <candidate regex="(^(\d|-|/|\s)*$|^.*(jan|...|dec).*$)"/>
04.  </candidates>
05.  <eliminations>
06.    <condition test="tag" op1_f="contains" op1_args="price"/>
07.    <condition test="tag" op1_f="contains" op1_args="id"/>
08.  </eliminations>
09.  <decision_tree>
10.    <condition test="values" abrag="every" op1_f="matches"
                op1_args=".*(-|/|\s).*(-|/|\s).*">
11.      <condition op1_f="em:nesting" op1_args="em:tokenize#3#(\-|/|\s)#1#string-length#1"
                  operator="equals" op2="4"  test="values" abrag="every">
12.        <condition test="values" abrag="every" op1_f="em:nesting"
                    op1_args="em:tokenize#3#(\-|/|\s)#2#matches#2#^(jan|...|dec).*$">
13.          <annotation format="Y4WD" type="DATE"/>
14.        </condition>
15.        <condition abrag="some" test="values" op1_f="em:tokenize" op1_args="-|/|\s#2"
                    operator="greaterthan" op2="12">
16.          <annotation type="DATE" format="Y4DM"/>
17.        </condition>
18.      </condition>
19.    </condition>
20.  </decision_tree>
21.</cfg>
```

Figura 6.12: Exemplo de arquivo de configuração

- linha 10 - especifica a condição na qual todos os valores do caminho devem ter dois separadores;
- linha 11 - especifica a condição na qual todos os valores devem ter quatro dígitos antes do primeiro separador;
- linha 12 - especifica a condição na qual todos os valores devem conter um nome ou abreviatura de mês entre o primeiro e segundo separador;
- linha 13 - especifica a classe na qual o caminho XML é temporal e seus valores possuem domínio `DATE` e formato `Y4WD`. Esse formato representa um valor temporal formado por um ano com quatro dígitos, seguido por um separador, seguido por um mês nominal, seguido por um separador, seguido por um dia numérico;
- linha 15 - especifica a condição na qual pelo menos um valor do caminho deve conter um número maior que 12 entre o primeiro e o segundo separador;
- linha 16 - especifica a classe na qual o caminho XML é temporal e seus valores possuem domínio `DATE` e formato `Y4DM`. Esse formato representa um valor temporal

[1]O arquivo de configuração utilizado nos experimentos está disponível em http://inf.ufrgs.br/~emanica/materiais/TPI/arquivoConfiguracao.xml.

formado por um ano com quatro dígitos, seguido por um separador, seguido por um dia numérico, seguido por um separador, seguido por um mês numérico.

6.2.1.2 Domínios e Formatos de Valores Temporais

Esta subseção especifica os principais domínios e formatos de valores temporais encontrados em documentos XML. Com isso, esses domínios e formatos devem ser contemplados no arquivo de configuração para a realização da identificação de caminhos temporais.

A Tabela 6.2 apresenta os domínios de valores temporais. Além disso, para cada domínio é apresentada uma descrição.

Tabela 6.2: Domínios de valores temporais

Tipo	**Descrição**
DAY	dia numérico
MONTH	mês numérico ou nominal
YEAR	ano com dois ou quatro dígitos
DATE	data composta por ano, mês e opcionalmente dia (tendo "/", "-", "." ou espaço em branco como separador)

A Tabela 6.3 exibe os formatos de valores temporais de acordo com cada domínio. Os separadores não são apresentados nos formatos. Cada valor temporal deve obrigatoriamente ter um separador. O separador pode ser: "/", "-", "." ou espaço em branco. Os símbolos utilizados nessa tabela são descritos na Tabela 6.4.

Tabela 6.3: Formatos de valores temporais

Domínio	**Formato**	**Domínio**	**Formato**	**Domínio**	**Formato**
DATE	Y4WD	DATE	Y2DM	DATE	WY4
	Y4DW		Y2MD		MY2
	Y4DM		Y2XX		WY2
	Y4MD		DWY2	DAY	D
	Y4XX		WDY2	MONTH	M
	DWY4		DMY2	MONTH	W
	WDY4		MDY2	YEAR	Y4
	DMY4		XXY2	YEAR	Y2
	MDY4		XXX	DAY|MONTH	D|M
	XXY4		Y4M	DAY|YEAR	D|Y2
	Y2WD		Y4W	DAY|MONTH|YEAR	X
	Y2DW		MY4		

Tabela 6.4: Descrição dos símbolos dos formatos

Símbolo	**Descrição**
Y4	ano com quatro dígitos
Y2	ano com dois dígitos
M	mês numérico
W	mês nominal
D	dia numérico
X	qualquer opção acima
|	disjunção

6.2.2 Identificação de Instantes Fragmentados

Um instante fragmentado é composto de nodos temporais que são filhos do mesmo nodo pai, de domínios diferentes, porém complementares. Esses nodos temporais devem ter valores dos seguintes domínios: `ANO`, `MÊS` e `DIA` (opcional). A Figura 6.13 (a)

```
<person>
    <born_year>1980</born_year>
    <born_month>05</born_month>
    <born_day>29</born_day>
</person>
```

(a)

```
<importantDates>
    <graduation_year>2007</graduation_year>
    <graduation_month>06</graduation_month>
    <graduation_day>29</graduation_day>
    <wedding_year>2008</wedding_year>
    <wedding_month>12</wedding_month>
    <wedding_day>01</wedding_day>
</importantDates>
```

(b)

Figura 6.13: Exemplos de instantes fragmentados

apresenta um exemplo que contém um instante fragmentado formado pelos elementos `born_year`, `born_month` e `born_day`.

Quando o documento XML possui dois instantes fragmentados irmãos, é necessário identificar os nodos que formam um instante e os nodos que formam o outro instante. Isso é resolvido agrupando o nodo que possui domínio `ANO` com os nodos que possuem domínio `MÊS` e `DIA` pela similaridade[2] de seus nomes. De acordo com essa regra, a identificação de instantes fragmentados sobre o exemplo apresentado na Figura 6.13 (b), deve agrupar `graduation_year` com `graduation_month` e `graduation_day`, formando um instante. Além disso, deve agrupar `wedding_year` com `wedding_month` e `wedding_day`, formando outro instante.

6.2.3 Identificação de Intervalos Temporais

Um intervalo temporal é identificado quando um fragmento XML possui dois nodos temporais (ou instantes fragmentados) que são filhos do mesmo nodo onde o comportamento de seus caminhos implica no valor de um deles ser sempre menor ou igual ao valor do outro. Um intervalo temporal pode ter instantes fragmentados que são netos de um mesmo nodo e filhos de nodos com nomes diferentes.

A Figura 6.14 (a) apresenta um exemplo de um intervalo temporal composto por dois nodos temporais filhos de um mesmo nodo. A Figura 6.14 (b) apresenta um exemplo de um intervalo temporal composto por dois instantes fragmentados netos de um mesmo nodo e filhos de nodos com nomes diferentes. A Figura 6.14 (c) apresenta um exemplo de um intervalo temporal composto por dois instantes fragmentados filhos de um mesmo nodo.

6.2.4 Normalização de Expressões Temporais

A normalização das expressões temporais consiste em transformar os valores temporais para um formato padronizado. Esse formato é constituído por dois instantes, que

[2]Nos experimentos, utilizou-se a função de similaridade *Levenshtein* (LEVENSHTEIN, 1966). Essa função realizou o agrupamento correto em todas as bases.

(a)
```
<event>
   <begin>2010-10-05</begin>
   <end>2010-10-08</end>
</event>
```

(b)
```
<event>
   <begin>
    <year>2010</year>
    <month>10</month>
    <day>05</day>
   </begin>
   <end>
    <year>2010</year>
    <month>10</month>
    <day>08</day>
   </end>
</event>
```

(c)
```
<event>
  <begin_year>2010</begin_year>
  <begin_month>10</begin_month>
  <begin_day>15</begin_day>
  <end_year>2010</end_year>
  <end_month>10</end_month>
  <end_day>18</end_day>
</event>
```

Figura 6.14: Exemplos de intervalos temporais

representam o instante inicial e final de um intervalo temporal. Cada instante deve estar no formato `dia/mês/ano`.

Valores temporais pertencentes a intervalos temporais têm mapeamento direto. Valores temporais pertencentes a instantes formam um intervalo que inicia e termina no mesmo instante. Valores temporais cujo formato não foi desambiguado são mapeados para vários intervalos temporais de acordo com todos os formatos possíveis.

A normalização do valor temporal contido no exemplo apresentado na Figura 6.14 (c) gera o intervalo temporal `[15/10/2010, 18/10/2010]`. A normalização do valor temporal contido no exemplo apresentado na Figura 6.13 (a) gera o intervalo temporal `[29/05/1980, 29/05/1980]`.

6.2.5 Indexação das Expressões Temporais

A indexação das expressões temporais consiste em criar o índice temporal. Esse índice é composto pelos intervalos temporais criados na normalização das expressões temporais. Cada intervalo temporal deve apontar para o nodo LCA dos nodos que compõem o intervalo temporal. Por exemplo, o intervalo temporal criado na normalização do valor temporal do exemplo apresentado na Figura 6.14 (c) deve apontar para o nodo cujo nome é `event`. O intervalo temporal criado na normalização do valor temporal do exemplo apresentado na Figura 6.13 (a) deve apontar para o nodo cujo nome é `person`.

A forma de identificar cada nodo deve ser a mesma do motor de busca utilizado. Esse trabalho utiliza o conceito de *Dewey Label*, uma vez que é o identificador mais utilizado pelos motores de busca. Dessa forma, para encontrar o nodo a ser referenciado, basta encontrar o mais longo prefixo comum entre os *Dewey Labels* de todos os nodos que compõem o intervalo temporal.

6.3 Processador Temporal

O objetivo do processador temporal é realizar o tratamento adequado das informações temporais presentes nas consultas por palavras-chave. Esse tratamento inclui mapear o predicado temporal da consulta para uma condição temporal padronizada e identificar os nodos XML que satisfazem essa condição temporal.

Considere a consulta `Q6.2`, cujo objetivo é retornar os títulos dos livros publicados após maio de 1998, e o índice temporal apresentado na Figura 6.15.

```
Q6.2 - book title after 05/1998
```

Instante Inicial			Instante Final			Dewey Label
Dia	Mês	Ano	Dia	Mês	Ano	
15	03	1998	15	03	1998	0.0.2
03	06	1998	03	06	1998	0.1.2

Figura 6.15: Exemplo de índice temporal

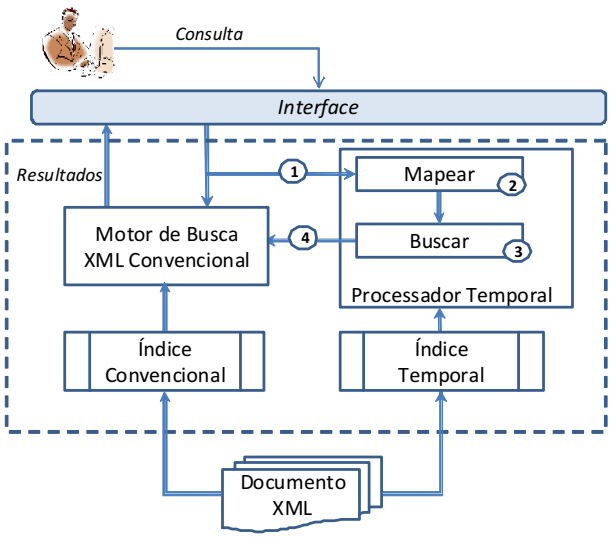


Figura 6.16: Visão Detalhada do Processador Temporal

A Figura 6.16 apresenta a visão detalhada do processador temporal, que executa as seguintes etapas:

1. o processador temporal recebe como entrada um predicado temporal (`after 05/1998`);

2. o predicado temporal é mapeado para uma condição temporal (`instante inicial > 31/05/1998`);

3. o processador temporal busca no índice temporal os nodos que satisfazem a condição temporal (`0.1.2`);

4. os nodos obtidos são enviados para o motor de busca, na segunda interceptação realizada por TPI, onde são tratados como casamentos para uma palavra-chave da consulta fictícia (`tempcond`).

Em experimentos preliminares, foi testada a busca de todos os nodos temporais que satisfazem a condição temporal presentes no índice temporal. Essa estratégia resulta em uma baixa precisão, pois aplica a condição temporal em nodos indesejados. Por exemplo, considere a consulta `Q6.3`, cujo objetivo é retornar o título dos artigos cujo periódico foi publicado em 2010. A condição temporal resultante do mapeamento é `instante inicial < 31/12/2010 e instante final > 01/01/2010`. Ao buscar os nodos que satisfazem essa condição no índice temporal (Figura 6.18) para o documento XML `Journals` (Figura 6.17), são retornados dois nodos: `0.1.3` e `0.1.0.1`. O nodo `0.1.3` refere-se a data de publicação do periódico, que realmente deve ser retornado. No entanto, o nodo `0.1.0.1` refere-se a data de submissão do artigo, logo não deve ser retornado.

`Q6.3 - articletitle pubdate intersect 2010`

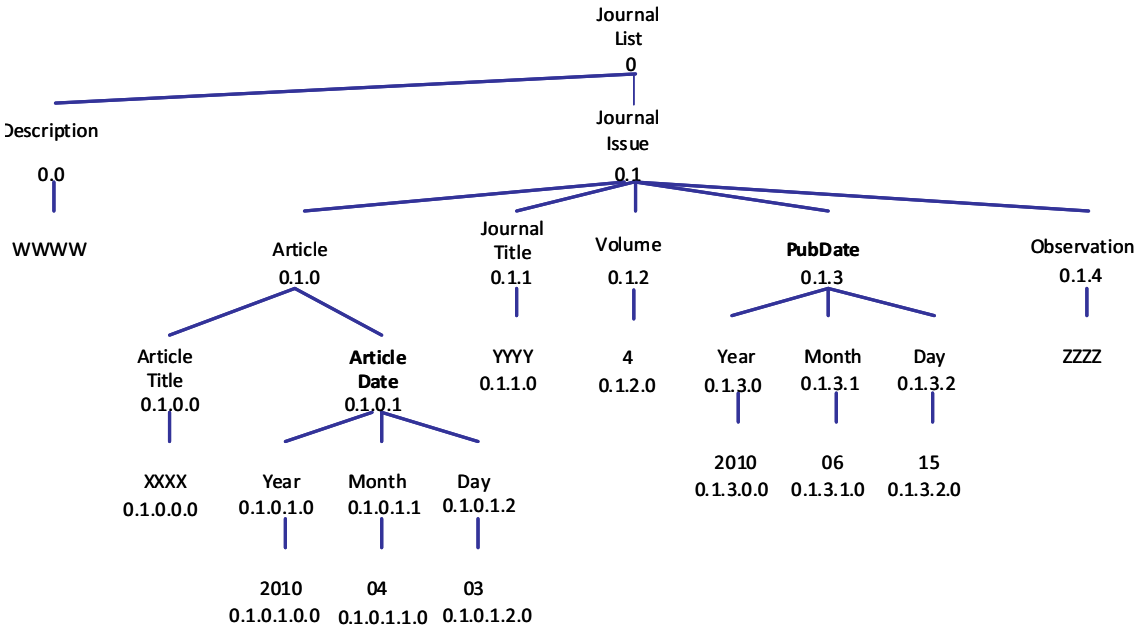


Figura 6.17: Documento XML `Journals`

Instante Inicial			Instante Final			Dewey Label
Dia	Mês	Ano	Dia	Mês	Ano	
15	06	2010	15	06	2010	0.1.3
03	04	2010	03	04	2010	0.1.0.1

Figura 6.18: Índice temporal do documento XML `Journals`

TPI soluciona o problema aplicando a condição temporal apenas nos nodos temporais mais próximos da consulta. Entende-se por nodos temporais mais próximos da consulta

os nodos temporais à menor distância de um casamento da consulta. A seguinte fórmula de distância é utilizada:

$$dist(m, nt) = (level_m - level_{lca}) + (level_{nt} - level_{lca})$$

onde $dist(m, nt)$ é a distância entre o casamento da consulta m e o nodo temporal candidado nt, $level_m$ é o nível de m, $level_{lca}$ é o nível do LCA entre m e nt e $level_{nt}$ é o nível de nt.

Por exemplo, a consulta `Q6.3`, possui dois casamentos: `0.1.0.0`, referente à palavra-chave `articletitle` e `0.1.3` referente à palavra-chave `pubdate`. Os nodos temporais indexados são: `0.1.3` e `0.1.0.1`. As distâncias entre os casamentos da consulta e os nodos temporais indexados são apresentadas na Tabela 6.5. O nodo temporal mais próximo da consulta é `0.1.3`, , pois está à menor distância de um casamento da consulta (`distância = 0`). Esse nodo condiz com o objetivo da consulta.

Tabela 6.5: Distâncias entre os casamentos da consulta `Q6.3` e os nodos temporais indexados

Casamento	Nodo Temporal	Distância
0.1.0.0	0.1.3	3
0.1.0.0	0.1.0.1	2
0.1.3	0.1.3	0
0.1.3	0.1.0.1	3

Para a consulta `Q6.4`, cujo objetivo é retornar o título dos periódicos publicados em 2010, o casamento da consulta é: `0.1.1`. As distâncias entre os casamentos da consulta e os nodos temporais indexados são apresentadas na Tabela 6.6. O nodo temporal mais próximo da consulta é `0.1.3`, , pois está à menor distância de um casamento da consulta (`distância = 2`). Esse nodo condiz com o objetivo da consulta.

```
Q6.4 - journaltitle intersect 2010
```

Tabela 6.6: Distâncias entre os casamentos da consulta `Q6.4` e os nodos temporais indexados

Casamento	Nodo Temporal	Distância
0.1.1	0.1.3	2
0.1.1	0.1.0.1	3

Para a consulta `Q6.5`, cujo objetivo é retornar o título dos artigos cujo periódico foi publicado em 2010, o casamento da consulta é: `0.1.0.0`. As distâncias entre os casamentos da consulta e os nodos temporais indexados são apresentadas na Tabela 6.7. O nodo temporal mais próximo da consulta é `0.1.0.1`, pois está à menor distância de um casamento da consulta (`distância = 2`). Esse nodo não condiz com o objetivo da consulta, pois refere-se à data de submissão do artigo, enquanto que o predicado temporal da consulta refere-se à data de publicação.

```
Q6.5 - articletitle intersect 2010
```

Tabela 6.7: Distâncias entre os casamentos da consulta `Q6.5` e os nodos temporais indexados

Casamento	Nodo Temporal	Distância
0.1.0.0	0.1.3	3
0.1.0.0	0.1.0.1	2

Para solucionar os casos onde os nodos temporais mais próximos da consulta não são os nodos onde o usuário deseja aplicar a condição temporal, é definida a seguinte estratégia. Reporta-se ao usuário, os caminhos temporais onde o predicado temporal foi aplicado. Se o usuário não estiver satisfeito com os resultados, então é possível visualizar todos os caminhos temporais ranqueados em ordem de proximidade com a consulta. Junto com cada caminho temporal, uma palavra-chave é apresentada. Essa palavra-chave deve ser utilizada na consulta para garantir que o predicado temporal seja aplicado naquele caminho temporal. Isso auxilia o usuário na reformulação de sua consulta. A Figura 6.19 apresenta um exemplo de mensagem fornecida ao usuário a fim de auxiliá-lo na reformulação da consulta, caso ele não esteja satisfeito com os resultados.

Condição temporal aplicada no caminho: `JournalList/JournalIssue/Article/ArticleDate`

A seguir é apresentada uma lista dos caminhos temporais ranqueados em ordem de proximidade com a consulta. Se a condição temporal foi aplicada no caminho errado, escolha o caminho correto, entre os caminhos a seguir, e reformule a consulta adicionando o termo à direita do caminho escolhido:

JournalList/JournalIssue/Article/ArticleDate: `ArticleDate`
JournalList/JournalIssue/PubDate: `PubDate`

Resultados da Consulta
...

Figura 6.19: Auxílio ao usuário para reformular consultas

O mapeamento do predicado temporal para uma condição temporal é realizado de acordo com os algoritmos apresentados no Capítulo 5, exceto o mapeamento do primeiro operando. O mapeamento do primeiro operando deve identificar onde a condição temporal deve ser aplicada. Esse algoritmo é dependente da aplicação. Um mecanismo de consultas em documentos XML aplica a condição temporal em nodos XML. Por outro lado, um mecanismo de consultas em páginas Web aplica a condição temporal em expressões temporais contidas no conteúdo da página, na data de coleta da página ou na data de criação/última alteração da página. Em TPI, a condição temporal deve ser aplicada nos nodos temporais mais próximos da consulta.

O Algoritmo 11 efetua o mapeamento do primeiro operando de acordo com TPI. Esse algoritmo tem como entrada um vetor com os casamentos da consulta ($matches[]$), ou seja, um vetor com todos os casamentos para cada palavra-chave não temporal da consulta. A saída é uma lista com os nodos temporais mais próximos da consulta (nts), sobre os quais deve-se aplicar a condição temporal. Inicialmente, a menor distância entre os casamentos da consulta para o(s) nodo(s) temporal(is) mais próximo(s) ($min\_distance$) é a distância entre o primeiro casamento da consulta e o seu nodo temporal mais próximo (linha 4). Em seguida, percorre-se o vetor de casamentos (linhas 5-10). Para cada casamento ($match$), obtém-se a distância entre o casamento e o seu nodo temporal mais

próximo (linha 6). Se essa distância for menor que a menor distância, então ela passa a ser a menor distância (linhas 7-9). Em seguida, percorre-se novamente o vetor de casamentos (linhas 11-13). Para cada casamento ($match$), obtém-se os nodos temporais que estão a uma distância igual a menor distância e adiciona-se esses nodos a lista de nodos temporais mais próximos da consulta (linha 12).

Algorithm 11 Map First Operand

```
 1: INPUT: matches[];
 2: OUTPUT: nts;
 3: begin map_first_operand(matches)
 4: min_distance ←get_distance_closest_temporal_node(matches[1]);
 5: for each match IN matches do
 6:       distance ←get_distance_closest_temporal_node(match);
 7:       if distance < min_distance then
 8:             min_distance = distance;
 9:       end if
10: end for
11: for each match IN matches do
12:       nts.add(get_temporal_node_by_distance(match, min_distance));
13: end for
14: return nts
15: end
```

O Algoritmo 11 utiliza duas funções: `get_distance_closest_temporal_node` e `get_temporal_node_by_distance`. A primeira retorna a distância entre um casamento informado e o seu nodo temporal mais próximo. A segunda retorna os nodos temporais a uma distância informada do casamento informado. Essas funções são apoiadas por um índice que armazena, para cada nodo XML, o(s) nodo(s) temporal(is) mais próximo(s), bem como a distância entre eles.

6.4 Considerações Finais

Este capítulo apresentou TPI, uma abordagem proposta para permitir o suporte a consultas temporais por palavras-chave em documentos XML. Foram apresentadas a visão geral de TPI e a especificação de seus componentes, que incluem um motor de busca XML, um processador temporal, um índice convencional e um índice temporal.

A Tabela 6.8 apresenta um comparativo entre TPI e as ferramentas de anotação de expressões temporais apresentadas nos trabalhos relacionados. O principal diferencial de TPI é não analisar cada expressão isoladamente na identificação e normalização de expressões temporais. TPI agrupa as expressões de acordo com os caminhos que as contém e analisa em conjunto todas as expressões de um mesmo caminho, de modo a ter maiores informações durante o processo de identificação e normalização das expressões temporais. Além disso, TPI não utiliza processamento linguístico, uma vez que os nodos dos documentos XML orientados a dados não possuem frases. Também, TPI trata apenas expressões temporais explícitas, uma vez que raramente os documentos XML orientados a dados possuem expressões temporais relativas ou implícitas, devido a sua característica estruturada. TPI não realiza sua anotação no padrão TimeML, pois anota caminhos temporais ao invés de expressões temporais.

Tabela 6.8: Comparativo entre TPI e trabalhos relacionados: anotação de expressões temporais

	ANNIE	**GUTime**	**PorTexto**	**Chronos**	**TPI**
POS	Sim	Sim	Não	Sim	Não
Linguagem	Própria	TimeML	Diretivas HAREM	TIMEX2	Própria
Disponível	Sim	Sim	Não	Não	Não
Tipos	E, I, R	E, I, R	E, I, R	E, R	E
Normalização	Não	Sim	Não	Sim	Sim
Isolamento	Sim	Sim	Sim	Sim	Não

Legenda: E: explícita I: implícita R: relativa

A Tabela 6.9 apresenta um comparativo entre TPI e os motores de busca temporais apresentados nos trabalhos relacionados. O principal diferencial de TPI é permitir consultas temporais em documentos XML, nas quais são retornados fragmentos XML relevantes para a consulta. Os trabalhos relacionados permitem consultas temporais sobre páginas Web, ou seja, retornam documentos inteiros. TPI suporta predicado temporal embutido na consulta e representa a informação temporal como um intervalo. TPI suporta consultas de seleção temporal, que é o foco do trabalho.

Tabela 6.9: Comparativo entre TPI e trabalhos relacionados: motores de busca temporais

	Predicado	**Rótulo**	**Tipo de Tempo**	**Índice Temporal**	**Tipo de Consulta**	**Docs**
TISE	Separado	Intervalo	Conteúdo	1 timex por página	Seleção Temporal	HTML
TERN	Embutido	Instante	Conteúdo	Todas as timex para página	Seleção Temporal	HTML
Chronica	Separado	Instante	Atualização	Data de coleta	Seleção Temporal	HTML
(PASCA, 2008)	NA	Instante	Conteúdo	1 timex por *nugget* temporal	Saída Temporal	HTML
TPI	Embutida	Intervalo	Conteúdo	1 timex por nodo temporal	Seleção Temporal	XML

Legenda: NA: Não se aplica Timex: Expressão temporal

A Tabela 6.10 apresenta um comparativo entre TPI e as abordagens de consulta em dados semi-estruturados apresentadas nos trabalhos relacionados. O principal diferencial de TPI é permitir consultas temporais em documentos XML, sem exigir um modelo de dados temporal específico. Essa característica aumenta o número de documentos XML que podem ser consultados.

Tabela 6.10: Comparativo entre TPI e trabalhos relacionados: consulta temporal em dados semi-estruturados

	Documentos	**Modelo**
TXPath	XML	Específico
TXQuery	XML	Específico
T-Yago	HTML	*Ad-hoc*
TPI	XML	*Ad-hoc*

7 VALIDAÇÃO EXPERIMENTAL

Este capítulo apresenta os experimentos realizados com o objetivo de determinar a qualidade dos resultados de TPI para o processamento de consultas temporais por palavras-chave em documentos XML. O impacto no desempenho do motor de busca XML causado pela utilização de TPI também foi avaliado. Além disso, a escalabilidade de TPI também foi verificada.

Este capítulo está organizado da seguinte forma. A Seção 7.1 descreve o projeto da validação experimental. Os experimentos realizados e os resultados obtidos são apresentados na Seção 7.2. O capítulo é finalizado na Seção 7.3, com uma análise geral dos resultados. Os experimentos estão publicados em (MANICA; DORNELES; GALANTE, 2010).

7.1 Projeto Experimental

Esta seção descreve o projeto da validação experimental. A Subseção 7.1.1 apresenta as métricas de avaliação dos experimentos. A plataforma de *hardware* e *software* utilizada é definida na Subseção 7.1.2. A Subseção 7.1.3 descreve as bases de dados utilizadas. Finalmente, a Subseção 7.1.4 descreve a metodologia utilizada.

7.1.1 Métricas de Avaliação

Os experimentos realizados utilizaram as seguintes métricas para avaliação dos resultados: revocação, precisão, F1-*measure*, tempo de processamento e Teste T.

A revocação mede o percentual de documentos relevantes recuperados em relação ao total de documentos relevantes presentes na coleção (BAEZA-YATES; RIBEIRO-NETO, 1999). Quando aplicada a consultas por palavras-chave em documentos XML, a revocação mede o percentual de nodos relevantes que são retornados. Nodos relevantes são os nodos XML que o usuário deseja que sejam retornados para a consulta.

A precisão mede o percentual de documentos corretamente recuperados (BAEZA-YATES; RIBEIRO-NETO, 1999). Quando aplicada a consultas por palavras-chave em documentos XML, a precisão mede o percentual de nodos retornados que são relevantes.

Revocação e precisão são definidas como segue:

$$\textit{revocação} = \frac{|Rel \cap Ret|}{|Rel|}, \textit{precisão} = \frac{|Rel \cap Ret|}{|Ret|}$$

onde Rel é o conjunto de nodos relevantes, Ret é o conjunto dos nodos retornados por um motor de busca XML e $|S|$ denota o número de elementos em um conjunto S.

F1-*measure* é uma média ponderada harmônica entre revocação e precisão, atribuindo o mesmo peso para as duas métricas. F1-*measure* foi calculada, nos experimentos, da seguinte forma:

$$F1 = \frac{(1+\alpha) \times P \times R}{\alpha \times P + R}$$

onde $\alpha = 1$, P é o valor da precisão e R é o valor da revocação.

O tempo de processamento indica o tempo gasto na execução de um determinado processamento.

O Teste T (WIKIPEDIA, 2010) avalia se as médias de duas distribuições normais de valores são estatisticamente diferentes. Segundo Hull (1993), o Teste T apresenta bons resultados mesmo quando as distribuições não são perfeitamente normais. O limiar de significância estatística utilizado nos experimentos foi $\alpha = 0,05$. Quando o valor de P bicaudal calculado é menor que α, existe uma diferença significativa entre os desempenhos dos motores de busca XML analisados. Com relação a qualidade, o melhor resultado é do motor de busca XML com a maior média de distribuição. Com relação ao tempo, o melhor resultado é o do motor de busca XML com a menor média de distribuição.

7.1.2 Plataforma de Trabalho

A implementação do protótipo, realizada para avaliar a abordagem, foi feita em Java 1.6. Os experimentos foram executados em uma máquina com processador Intel Core 2 Duo de 2GHZ rodando Ubuntu 9.10, com 3GB de memória principal e 120GB de espaço em disco (7200rmp).

7.1.3 Descrição das Coleções

Três bases de dados XML reais foram utilizadas:

- Mondial[1] é uma base de dados geográficos integrados do CIA *World Factbook*, *International Atlas*, TERRA, entre outras fontes. Para os experimentos, foi utilizado um documento XML com 1,8MB de dados, 4 caminhos temporais, 0 instantes fragmentados, 0 intervalos temporais, 3.247 nodos temporais indexados, e 105.178 nodos XML indexados.

- Pubmed[2] é uma biblioteca digital de periódicos de biomedicina e ciências da vida. A base de dados utilizada nos experimentos foi um documento XML obtido como resultado para a consulta *"alzheimer's disease pathology prevention"* submetida para a Pubmed no dia 26 de abril de 2010. Os experimentos foram executados sobre um documento de 2,6MB com 6 caminhos temporais, 2 instantes fragmentados, 0 intervalos temporais, 2.362 nodos temporais indexados e 72.590 nodos XML indexados.

- Lattes[3] contém todos os currículos dos pesquisadores brasileiros. Esses currículos são armazenados em uma base de dados centralizada e hospedados em uma agência de fomento do governo federal. Os experimentos foram executados sobre um subconjunto dessa base de dados com 6,9MB, 73 caminhos temporais, 22 instantes fragmentados, 22 intervalos temporais, 6.996 nodos temporais indexados e 214.546 nodos XML indexados.

[1] http://www.cs.washington.edu/research/xmldatasets/

[2] http://www.ncbi.nlm.nih.gov/sites/entrez/

[3] http://lattes.cnpq.br/

7.1.4 Metodologia

Os experimentos foram executados usando duas implementações: (i) XSeek como originalmente implementado (LIU; CHEN, 2007), explicado na Seção 3.5, que é utilizado como *baseline*; e (ii) uma extensão de XSeek, onde foi aplicada a abordagem Two Phase Interception (TPI). Para clareza do texto, a implementação original de XSeek é referenciada como baseline e a proposta de extensão é referenciada como TPI.

7.2 Experimentos

Esta seção descreve os experimentos realizados e os resultados obtidos. A Subseção 7.2.1 descreve o experimento realizado para avaliar o aspecto qualidade dos resultados. A Subseção 7.2.2 descreve o experimento realizado para avaliar o aspecto tempo de processamento. Finalmente, a Subseção 7.2.3 descreve os experimentos realizados para avaliar o aspecto escalabilidade.

7.2.1 Qualidade dos Resultados

Este experimento analisou a qualidade dos resultados de 32 consultas, criadas por quatro usuários, para cada base de dados. Para cada consulta o usuário preencheu o objetivo da consulta em linguagem natural, a consulta por palavras-chave para o TPI (TQuery) e a consulta por palavras-chave para um motor de busca XML convencional (CQuery). Todas as consultas estão disponíveis em http://inf.ufrgs.br/~emanica/materiais/TPI/consultas.txt. Os exemplos a seguir mostram uma consulta para cada base de dados, respectivamente, para as bases Mondial, Pubmed e Lattes:

```
Objetivo: Retornar o países cuja independência ocorreu entre 2000 e 2002
TQuery: country name indep_date intersect [2000, 2002]
CQuery: country name indep_date 2000 2001 2002

Objetivo: Retornar os artigos publicados em 2009
TQuery: articletitle pubdate equal 2009
CQuery: articletitle pubdate 2009

Objetivo: Retornar os projetos de pesquisa concluídos em 2000
TQuery: RESEARCH-PROJECT finishes 2000
CQuery: RESEARCH-PROJECT FINAL-YEAR 2000
```

TQuery foi submetida para TPI. CQuery foi submetida para o *baseline*. O objetivo da consulta foi utilizado para criar consultas manualmente na linguagem de consulta XML estruturada XQuery. Essas consultas em XQuery foram utilizadas para descobrir os nodos relevantes para cada consulta que existiam em cada uma das bases de dados, a fim de efetuar a avaliação da qualidade dos resultados.

Para a avaliação da qualidade dos resultados foram utilizadas as métricas revocação, precisão e F1-*measure*. Essas são métricas de qualidade bem conhecidas da comunidade de recuperação de informação (BAEZA-YATES; RIBEIRO-NETO, 1999). Foram computadas as médias de revocação, precisão e F1-*measure*, considerando todas as consultas para cada base de dados. O Teste T foi utilizado para verificar a significância da diferença de revocação, precisão e F1-*measure* entre o *baseline* e TPI. Para clareza do texto, as médias de revocação, precisão e F1-*measure*, considerando todas as consultas para cada

base de dados, são referenciadas simplesmente como revocação, precisão e F1-*measure*, respectivamente.

A Figura 7.1 apresenta a revocação, precisão e F1-*measure* para as bases Mondial, Pubmed e Lattes. Na base Mondial, TPI obteve 94% de revocação, enquanto o *baseline* obteve 88%. O valor de P bi-caudal calculado pelo Teste T foi 0,16. TPI obteve 74% de revocação na base Pubmed e 93% na base Lattes, enquanto o *baseline* obteve 57% na base Pubmed e 77% na base Lattes. O valor de P bi-caudal calculado pelo Teste T foi menor que 0,05 para a revocação na base Pubmed e na base Lattes. A precisão de TPI na base Mondial foi 88%, na base Pubmed foi 77% e na base Lattes foi 80%, enquanto a precisão do *baseline* na base Mondial foi 56%, na base Pubmed foi 44% e na base Lattes foi 42%. O valor de P bi-caudal calculado pelo Teste T foi menor que 0,05 para a precisão em todas as bases. O F1-*measure* de TPI na base Mondial foi 91%, na base Pubmed foi 75% e na base Lattes foi 86%, enquanto que o F1-*measure* do *baseline* na base Mondial foi 68%, na base Pubmed foi 50% e na base Lattes foi 54%. O valor de P bi-caudal calculado pelo Teste T foi menor que 0,05 para o F1-*measure* em todas as bases.

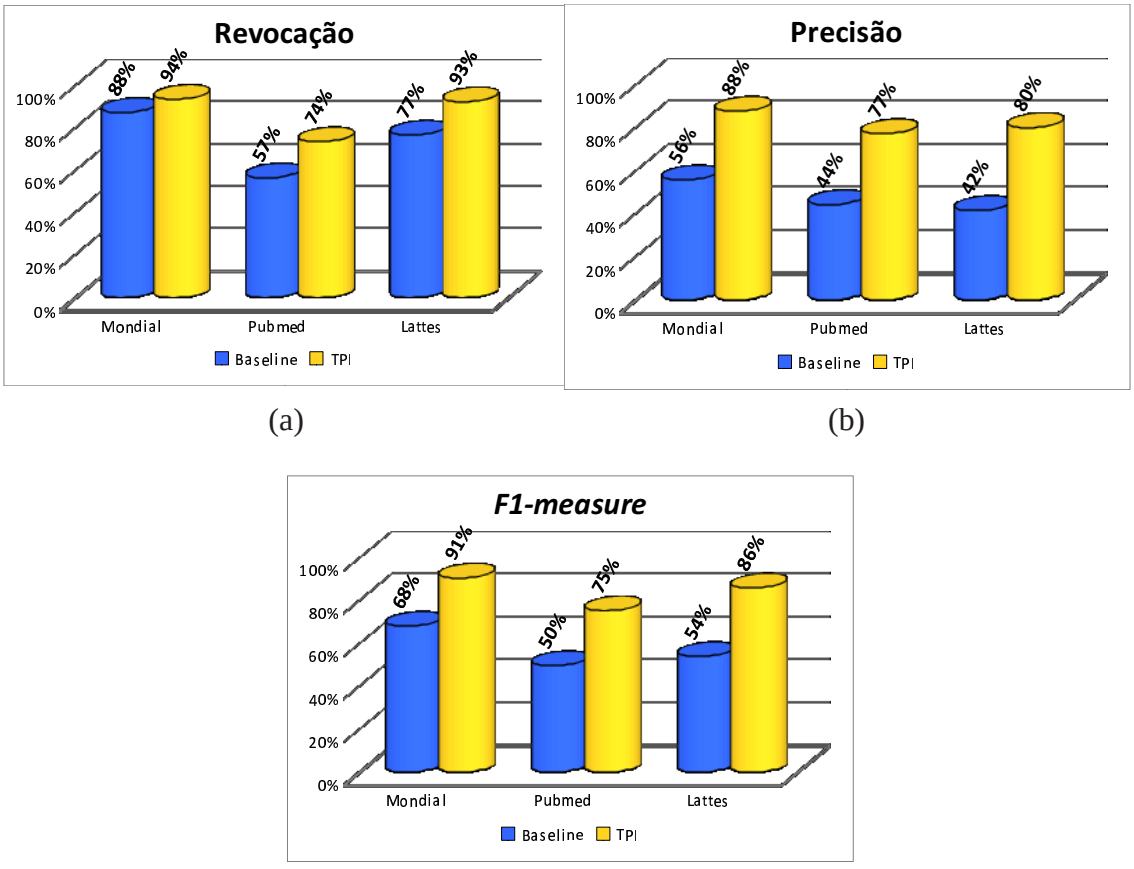


Figura 7.1: Revocação, Precisão e F1-*measure* para as bases Mondial, Pubmed e Lattes

TPI aumenta significativamente (valor de p do Teste T $< 0,05$) a revocação, a precisão e o *F1-measure* para as consultas temporais em todas as bases de dados, exceto a revocação da base Mondial (valor de p do Teste T $> 0,05$). Esse aumento na qualidade dos resultados ocorre porque o *baseline* trata as expressões temporais da mesma forma que trata as demais palavras-chave da consulta. TPI aplica o predicado temporal apenas nos nodos temporais mais próximos da consulta, suporta vários operadores temporais e granularidades, utiliza o comportamento do caminho temporal a fim de desambiguar o

formato de valores temporais e identifica instantes fragmentados e intervalos temporais.

Na base Mondial não há diferença estatística significante de revocação, pois essa base não contém nenhum instante fragmentado nem intervalo temporal. No entanto, TPI aumenta significativamente a precisão e o *F1-measure* porque essa base possui vários nodos com valor numérico que não são nodos temporais. Por exemplo, `inflation`, `total_area` e `population`. O *baseline* considera esses nodos como casamentos para as expressões temporais da consulta, o que diminui a sua precisão. TPI aplica a condição temporal apenas em nodos temporais, não sendo afetado por esses nodos.

TPI obteve o maior valor de *F1-measure* na base Lattes, uma vez que a base Lattes contém vários intervalos temporais. Essa é uma característica que afeta fortemente a revocação dos motores de busca XML convencionais uma vez que esses não estão habilitados para identificar, por exemplo, que 2007 é um valor válido para um nodo temporal cujo ano inicial é 2006 e o ano final é 2008. Além disso, não possuem o conhecimento que um nodo temporal que representa um intervalo temporal com o instante final aberto indica que a informação é válida atualmente. A existência de intervalos temporais também afeta a precisão dos motores de busca XML convencionais. Por exemplo, considere a consulta "`RESEARCH-PROJECT INITIAL-YEAR 2002 (PROJETO_PESQUISA ANO_INICIO 2002)`", cujo objetivo é retornar os projetos de pesquisa que iniciaram em 2002. Um motor de busca XML convencional retorna, para essa consulta, os projetos de pesquisa cujo ano de início é igual a 2002, considerando `INITIAL-YEAR` como um predicado da consulta. Além disso, retorna os projetos de pesquisa cujo ano de término (nodo `FINAL-YEAR`) é igual a 2002, considerando `INITIAL-YEAR` como um nodo de retorno explícito. Em TPI, a mesma consulta é escrita da seguinte forma: "`RESEARCH-PROJECT starts 2002`". TPI aplica o predicado temporal apenas no ano de início (nodo `INITIAL-YEAR`) do projeto de pesquisa, uma vez que TPI está habilitado para identificar e tratar adequadamente intervalos temporais.

Baseline e TPI tiveram a revocação, a precisão e o F1-*measure* afetados por algumas consultas que não retornavam nenhum valor relevante devido a uma inferência errônea dos nodos de retorno. Esse comportamento ocorreu principalmente na base Pubmed porque há um elemento `title` referindo-se ao título do periódico e um elemento `articletitle` referindo-se ao título do artigo. Em algumas consultas, os usuários utilizaram a palavra-chave `title` com o objetivo de retornar o título do artigo. Nesse caso, o motor de busca XML inferiu que os usuários desejavam o título do periódico e nenhum título de artigo foi retornado. Na consulta "`country name government republic indep_date last 3 decades`", cujo objetivo é retornar os nomes dos países cujo governo é república e cuja independência ocorreu nas últimas três décadas, alguns países cujo governo não é república foram retornados quando o termo "`Republic`" faz parte de seus nomes, tal como `Czech Republic`. Esses problemas de inferência errônea são inerentes ao motor de busca XML utilizado. É importante ressaltar que a abordagem proposta é independente do motor de busca XML utilizado. Enfatiza-se que, mesmo nesses casos, TPI aplica o predicado temporal nos nodos corretos. O problema está exclusivamente na identificação dos nodos de retorno, o que está fora do escopo desse trabalho.

7.2.2 Tempo de Processamento

Este experimento avaliou o tempo gasto para manipular a parte temporal das consultas realizadas na avaliação da qualidade dos resultados. Apenas foi avaliado o tempo de manipulação da parte temporal, pois as demais etapas são comuns entre o *baseline* e TPI.

O experimento calculou a média do tempo de processamento considerando todas as consultas para cada base de dados. O Teste T foi utilizado para verificar a significância da diferença de tempo entre o *baseline* e TPI. Para clareza do texto, a média de tempo de processamento, considerando todas as consultas para cada base de dados, é referenciada simplesmente como tempo de processamento.

A Figura 7.2 apresenta o tempo gasto para manipular a parte temporal da consulta. Na base Mondial, TPI levou 985 ms e o *baseline* levou 397. O valor de P bi-caudal para a base Mondial, calculado pelo Teste T, foi menor que 0,05. Na base Pubmed, TPI levou 374 ms e o *baseline* levou 477 ms. O valor de P bi-caudal para a base Pubmed, calculado pelo Teste T, foi 0,4. Na base Lattes, TPI levou 530 ms e o *baseline* levou 716 ms. O valor de P bi-caudal para a base Lattes, calculado pelo Teste T, foi 0,49.

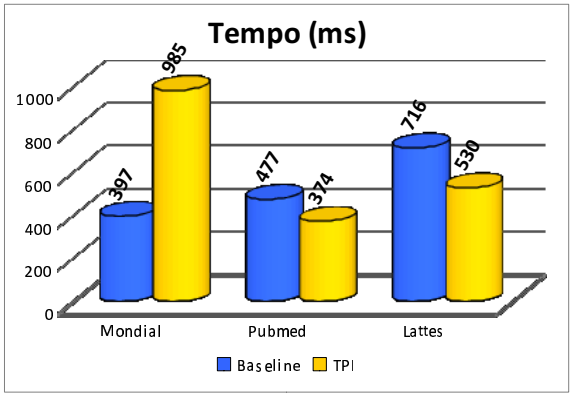


Figura 7.2: Tempo para manipular a parte temporal da consulta

Embora TPI adicione uma nova camada para o processamento da consulta do motor de busca XML, o que normalmente aumenta o tempo de processamento, nas bases Pubmed e Lattes o tempo de processamento foi menor (embora sem diferença estatística significante, pois $P > 0{,}05$). Isso ocorreu porque TPI utiliza um índice temporal enquanto o *baseline* utiliza um índice convencional.

Na base Mondial, TPI obteve um resultado pior. Esse resultado se deve ao fato de algumas palavras-chave possuírem vários casamentos (devido a estrutura da base de dados). Por exemplo, na base Mondial, `country` é um elemento filho do elemento `mondial` e um atributo dos elementos `city`, `border`, `province`, `members`, `located`, etc. Com isso, as consultas que possuem o termo `country` contêm um grande número de casamentos. Essa característica aumenta o tempo de TPI, pois ele analisa os casamentos das palavras-chave não temporais para definir em qual nodo aplicar a condição temporal.

7.2.3 Escalabilidade

O aspecto `escalabilidade` foi avaliado através de três experimentos, que verificaram o comportamento do *baseline* e de TPI, em relação ao tempo gasto para manipular a parte temporal, com o aumento crescente de determinadas variáveis. O Experimento 1 e o Experimento 2 variam o número de palavras-chave não temporais da consulta. O Experimento 3 varia o tamanho do intervalo de tempo desejado. Os três experimentos foram executados sobre a base Pubmed.

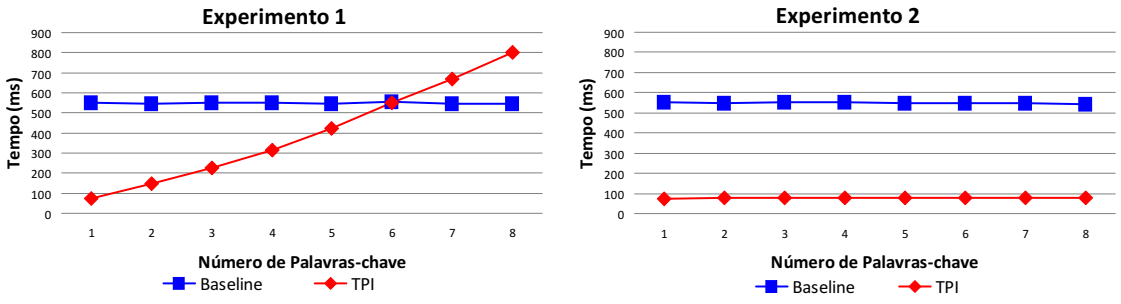
Os experimentos 1 e 2 utilizaram consultas com um número de palavras-chave não temporais crescente, um operador temporal e um segundo operando fixos. Além disso, todas as consultas referem-se ao mesmo nodo temporal e todos os casamentos para as

palavras-chave não temporais estão à mesma distância do nodo temporal referente a consulta. Foram utilizadas 8 consultas:

- `ArticleTitle intersect 2010`
- `ArticleTitle Language intersect 2010`
- `ArticleTitle Language Affiliation intersect 2010`
- `ArticleTitle Language Affiliation Abstract intersect 2010`
- `ArticleTitle Language Affiliation Abstract Pagination intersect 2010`
- `ArticleTitle Language Affiliation Abstract Pagination Journal intersect 2010`
- `ArticleTitle Language Affiliation Abstract Pagination Journal AuthorList intersect 2010`
- `ArticleTitle Language Affiliation Abstract Pagination Journal AuthorList PublicationTypeList intersect 2010`

As consultas no *baseline* não possuíam o operador temporal `intersect` explícito, pois motores de busca XML convencionais não apresentam suporte a operadores temporais. No entanto, a forma como encontram os casamentos para as expressões temporais pode ser comparada com a aplicação do operador temporal `intersect`.

O diferencial entre o Experimento 1 e o Experimento 2 é que no Experimento 1 todas as **palavras-chave** não temporais da consulta possuíam o mesmo número de casamentos (720). Isso significa que a medida que o número de palavras-chave não temporais aumenta, o número total de casamentos da consulta também aumenta. No Experimento 2, todas as **consultas** possuem o mesmo número total de casamentos para as palavras-chave não temporais (720). Isso significa que na consulta com 1 palavra-chave não temporal, essa palavra-chave possui 720 casamentos. Por outro lado, na consulta com 8 palavras-chave não temporais, cada uma dessas palavras-chave possui 90 casamentos.



(a) (b)

Figura 7.3: Escalabilidade na Pubmed variando o número de palavras-chave não temporais

O Experimento 1 (Figura 7.3 (a)) mostra que o tempo de processamento do *baseline* é constante enquanto o tempo de processamento do TPI aumenta linearmente quando o

número de casamentos total da consulta aumenta. O Experimento 2 (Figura 7.3 (b)) mostra que o tempo de processamento das duas abordagens é constante quando o número de palavras-chave aumenta, mantendo constante o número total de casamentos da consulta. Isso ocorre porque na avaliação da parte temporal o *baseline* apenas procura casamentos para cada expressão temporal da consulta, enquanto que TPI analisa todos os casamentos para as palavras-chave não temporais a fim de identificar o nodo temporal mais próximo da consulta.

O Experimento 3 utilizou consultas com um tamanho de intervalo de tempo desejado crescente, um número constante de palavras-chave não temporais e um número constante de casamentos para cada palavra-chave não temporal. Todas as consultas referem-se ao mesmo nodo temporal e todas as palavras-chave da consulta estão a mesma distância desse nodo. Foram utilizadas 8 consultas:

- `ArticleTitle Language intersect [2000,2000]`
- `ArticleTitle Language intersect [2000,2001]`
- `ArticleTitle Language intersect [2000,2002]`
- `ArticleTitle Language intersect [2000,2003]`
- `ArticleTitle Language intersect [2000,2004]`
- `ArticleTitle Language intersect [2000,2005]`
- `ArticleTitle Language intersect [2000,2006]`
- `ArticleTitle Language intersect [2000,2007]`

Para submeter as consultas ao *baseline*, foi necessário alterar a estrutura da consulta, retirando o operador temporal e adicionando uma expressão temporal para cada instante do período desejado. As consultas submetidas ao *baseline* são:

- `ArticleTitle Language 2000`
- `ArticleTitle Language 2000 2001`
- `ArticleTitle Language 2000 2001 2002`
- `ArticleTitle Language 2000 2001 2002 2003`
- `ArticleTitle Language 2000 2001 2002 2003 2004`
- `ArticleTitle Language 2000 2001 2002 2003 2004 2005`
- `ArticleTitle Language 2000 2001 2002 2003 2004 2005 2006`
- `ArticleTitle Language 2000 2001 2002 2003 2004 2005 2006 2007`

O Experimento 3 (Figura 7.4) mostra que o tempo de processamento do TPI é constante enquanto o tempo de processamento do *baseline* é linear quando o tamanho do intervalo temporal aumenta. Isso ocorre porque, para qualquer tamanho de intervalo temporal, TPI percorre o índice temporal uma única vez. Por outro lado, o *baseline* percorre o seu índice de nodos XML uma vez para cada instante do intervalo temporal. Por exemplo, considere a consulta "`ArticleTitle intersect [2000,2002]`", com o objetivo de retornar os artigos publicados entre 2000 e 2002. Na manipulação da parte temporal, TPI percorre o índice temporal uma única vez, procurando os intervalos temporais cujo ano do instante inicial é menor ou igual a 2002 e o ano do instante final é maior ou igual a 2000. Para o *baseline*, a mesma consulta teria a seguinte estrutura: "`ArticleTitle Language 2000 2001 2002`". Com isso, o *baseline* necessita, para manipular a parte temporal, percorrer o seu índice de nodos XML três vezes: (i) procurando as ocorrências de 2000; (ii) procurando as ocorrências de 2001; e (iii) procurando as ocorrências de 2002.

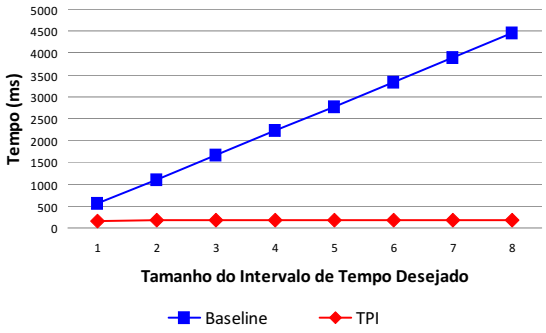


Figura 7.4: Escalabilidade na PubMed variando o tamanho do intervalo de tempo desejado

O fato da avaliação da escalabilidade ter sido realizada em somente uma base não prejudica o resultado do experimento, pois o objetivo dessa avaliação foi verificar se o motor de busca XML possuía um comportamento constante, linear ou exponencial em relação ao tempo de processamento da parte temporal com o aumento de determinadas variáveis. O que define esse comportamento é o algoritmo utilizado, e não a base utilizada, pois todas as características temporais das bases (instantes temporais, instantes fragmentados e intervalos temporais) são indexadas da mesma forma e possuem as características e as operações desejadas.

7.3 Análise Geral dos Experimentos

Os experimentos mostraram que TPI aumenta significativamente a qualidade dos resultados de consultas temporais por palavras-chave quando comparado com um motor de busca XML convencional. O aumento da qualidade dos resultados se deve ao fato que TPI realiza um tratamento especial da informação temporal presente nas consultas e no conteúdo dos documentos XML. Esse tratamento inclui o suporte de vários operadores temporais, algumas palavras reservadas e diferentes granularidades. Além disso, TPI leva em conta o padrão de valores nos caminhos XML para decidir se um nodo é temporal ou não, desambiguar formatos, identificar instantes fragmentados e intervalos temporais. Motores de busca XML convencionais tratam as expressões temporais da mesma forma que as demais palavras-chave da consulta. Isso significa que cada nodo texto que contém a

palavra-chave é considerado um casamento, incluindo datas, preços, códigos, medidas, ou qualquer outro valor numérico. Além disso, esses motores apenas consideram cada valor isolado para computar os casamentos para as expressões temporais sem desambiguação de formato, não havendo identificação de instantes fragmentados e intervalos temporais.

As extensões de linguagens estruturadas, que adicionam suporte ao tempo, apresentadas nos trabalhos relacionados (TXPath, TXQuery e T-Yago) recuperam todos os resultados relevantes e todos os resultados recuperados são relevantes. Entretanto, essas linguagens estruturadas exigem que o usuário aprenda uma linguagem de consulta complexa e necessitam de um grande conhecimento da estrutura dos dados. Por outro lado, consultas por palavras-chave, como proposto por TPI, representam uma abordagem amigável para descoberta de informação que tem sido amplamente estudada para documentos texto (BHALOTIA et al., 2002; HRISTIDIS et al., 2006; LIU; CHEN, 2007).

Considerando a escalabilidade da manipulação da parte temporal, TPI escala constante quando o tamanho do intervalo de tempo desejado aumenta e linear quando o número total de casamentos da consulta aumenta. Esse fato, aliado ao aumento significativo da qualidade dos resultados, permite a constatação de que a abordagem TPI pode ser aplicada na prática. Além disso, o fato da solução proposta realizar uma interceptação da consulta sobre um motor de busca adicionando uma camada adicional de software torna a abordagem potencialmente aplicável para outros mecanismos de consulta por palavras-chave.

8 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho apresenta TPI, uma abordagem proposta para suportar consultas temporais por palavras-chave sobre documentos XML. Essa abordagem realiza duas interceptações no processamento de consultas, realizado por um motor de busca XML, para incorporar um tratamento adequado das informações temporais presentes na consulta. TPI é constituído de quatro componentes principais: um motor de busca XML, um processador temporal, um índice temporal e um índice convencional. A principal contribuição de TPI é realizar o tratamento adequado das informações temporais presentes nas consultas e nos documentos XML, melhorando significativamente a qualidade dos resultados.

TPI foi avaliado através de experimentos de qualidade dos resultados, tempo de processamento e escalabilidade em três bases XML reais, comparando com o motor de busca XML convencional XSeek. Os resultados indicam que TPI aumenta significativamente a qualidade dos resultados e mantêm o desempenho geral do motor de busca, comparando com os resultados de um motor de busca XML convencional.

O trabalho também especifica TKC, uma classificação genérica proposta para servir de guia para qualquer mecanismo de consulta temporal por palavras-chave. TKC foi definida a partir de uma análise de consultas Web e é base para TPI. TKC trata uma consulta temporal como um predicado temporal composto por três partes: (i) um primeiro operando, caracterizado pelas palavras-chave não temporais; (ii) um operador temporal, que define a relação temporal entre o primeiro e o segundo operando; e (iii) um segundo operando, caracterizado pelo tempo especificado como filtro da consulta. TKC traz contribuições tanto para a área de gerenciamento de dados temporais, quanto para a área de recuperação de informação. Enquanto para a área de gerenciamento de dados temporais a contribuição é a especificação dos requisitos temporais em consultas por palavras-chave, para recuperação de informação a contribuição são os algoritmos de mapeamento que servem de guia para a implementação dos requisitos temporais em mecanismos de consulta por palavras-chave, como por exemplo, motores de busca Web.

Detalhes do trabalho incluem ainda a definição dos algoritmos de mapeamento dos predicados temporais, definidos por palavras-chave, para expressões relacionais a fim de orientar a implementação do processamento das consultas temporais definidas em TKC. É proposto um índice temporal que permite um acesso rápido e consistente dos dados durante o processamento das consultas. Para a criação desse índice, foi definido um processo que utiliza um arquivo de configuração com regras para identificação de caminhos temporais e formatos de valores temporais. Além disso, são apresentadas algumas heurísticas para identificação de instantes fragmentados e intervalos temporais. O grande diferencial desse processo é a análise do comportamento do caminho XML para a realização dessas identificações.

Como produção científica foram publicados os seguintes artigos:

1. (MANICA; GALANTE, 2009) **Suporte a Consultas Temporais por Palavras-chave em XML**. In: SESSÃO DE PÔSTERES, 3., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 24. Anais... Fortaleza: SBC, 2009. p.17-20. Esse artigo descreve a proposta inicial da dissertação;

2. (MANICA; GALANTE; DORNELES, 2010a) **Mapeamento Automático de Modelos de Dados XML Temporais Ad-hoc para um Modelo de Dados XML Temporal Padrão**. In: ESCOLA REGIONAL DE BANCOS DE DADOS, ERBD, 6. Anais... [S.l.: s.n.], 2010. Esse artigo apresenta detalhes do processo de indexação temporal;

3. (MANICA; GALANTE; DORNELES, 2010b) **Two-Phase Interception: uma abordagem para suporte a consultas temporais por palavras-chave em XML**. In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCOS DE DADOS, WTDBD, 9., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 25. Anais... Belo Horizonte: SBC, 2010. p.25-30. Esse artigo apresenta uma visão geral de TPI;

4. (MANICA; DORNELES; GALANTE, 2010) **Supporting Temporal Queries on XML Keyword Search Engines**. JIDM, [S.l.], v.1, n.3, p.471-486, 2010. Esse artigo descreve a versão completa deste trabalho, incluindo TKC (sem os algoritmos de mapeamento), TPI e os experimentos realizados;

5. (FROZI et al., 2010) **Chronos - Um Motor de Busca Temporal**. In: SESSÃO DE DEMOS, 7., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 25. Anais... Belo Horizonte: SBC, 2010. p.13-18. Esse artigo descreve o suporte a consultas temporais por palavras-chave, de acordo com TKC, em um motor de busca de páginas Web.

Um trabalho foi desenvolvido em paralelo com essa dissertação:

- (FROZI, 2010) **Um mecanismo de consulta temporal por palavras-chave em páginas Web**. 2010. Monografia (Graduação em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre. Desenvolvida paralelamente a esse trabalho, essa monografia descreve o suporte a consultas temporais por palavras-chave, de acordo com TKC, em um motor de busca de páginas Web. Essa monografia foi co-orientada pelo autor dessa dissertação.

Ao longo desse trabalho, alguns aspectos de TKC foram identificados como passíveis de melhorias ou extensões:

- suportar operadores de agregação. Por exemplo, recuperar a média de salário de um empregado nos últimos 2 anos;

- suportar a operação *coalesce*, de modo a eliminar duplicatas ocasionadas pela existência de intervalos sobrepostos;

- suportar consultas de saída temporal, nas quais o usuário deseja descobrir quando determinado evento ocorreu;

- suportar consultas sobre períodos de pausa. Recuperar as pausas de determinadas durações, a n-ésima pausa, a quantidade de pausas, etc. Essa não é uma tarefa trivial, uma vez que não é exigido um modelo temporal específico. Com isso, torna-se necessário, primeiramente, identificar nodos que referem-se a informações de um mesmo objeto, porém em tempos diferentes.

Ao longo desse trabalho, alguns aspectos de TPI, também, foram identificados como passíveis de melhorias ou extensões:

- permitir a criação semi-automática do arquivo de configuração utilizado na indexação temporal. Essa semi-automatização permitirá retirar a necessidade de um usuário especialista para criar o arquivo de configuração. Para essa semi-automatização podem ser utilizadas técnicas de inteligência artificial, como por exemplo, algoritmos de árvore de decisão, como C4.5 (QUINLAN, 1993), uma vez que o arquivo de configuração define uma árvore de decisão;

- expandir a identificação de instantes fragmentados e intervalos temporais para contemplar todas as possibilidades oferecidas pelo modelo de dados XML. Atualmente, essa identificação é realizada por lógicas pré-definidas que consideram uma visão simplificada do modelo de dados XML.

Outro trabalho está em desenvolvimento tendo como base TPI:

- uma dissertação de mestrado que tem por objetivo inferir datas relativas e implícitas em consultas por palavras-chave em documentos XML. Uma consulta relativa contém expressões que indicam indiretamente a noção de tempo, como por exemplo, ontem, hoje, amanhã. Uma consulta implícita não possui um operador temporal, mas essa informação pode ser inferida pelo processamento da consulta. Por exemplo, a consulta "`miss universo`" possivelmente o usuário quer saber sobre o último concurso que ocorreu. Em resumo, essa dissertação estende TPI para processar consultas na qual a temporalidade depende da intenção do usuário, cujo resultado pode mudar dependendo do posicionamento da consulta no tempo.

REFERÊNCIAS

ALLEN, J. F. Maintaining Knowledge about Temporal Intervals. **Commun. ACM**, [S.l.], v.26, n.11, p.832–843, 1983.

ALONSO, O.; GERTZ, M.; BAEZA-YATES, R. A. On the value of temporal information in information retrieval. **SIGIR Forum**, [S.l.], v.41, n.2, p.35–41, 2007.

ALONSO, O.; GERTZ, M.; BAEZA-YATES, R. A. Search results using timeline visualizations. In: SIGIR. **Anais...** ACM, 2007. p.908.

ANNIE. **Open Source Information Extraction**. Disponível em: <http://www.aktors.org/technologies/annie/>. Acesso em: 11 nov. 2010.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval**. [S.l.]: ACM Press / Addison-Wesley, 1999.

BHALOTIA, G. et al. Keyword Searching and Browsing in Databases using BANKS. In: ICDE. **Anais...** IEEE Computer Society, 2002. p.431–440.

CLARK, J.; DEROSE, S. **XML Path Language (XPath) 1.0**. Disponível em: <http://www.w3.org/TR/XPATH>. Acesso em: 17 nov. 2010.

COHEN, S. et al. XSEarch: a semantic search engine for xml. In: VLDB. **Anais...** [S.l.: s.n.], 2003. p.45–56.

CRAVEIRO, O.; MACEDO, J.; MADEIRA, H. Use of Co-occurrences for Temporal Expressions Annotation. In: SPIRE. **Anais...** Springer, 2009. p.156–164. (Lecture Notes in Computer Science, v.5721).

CUNNINGHAM, H. et al. A framework and graphical development environment for robust NLP tools and applications. In: ACL. **Anais...** [S.l.: s.n.], 2002. p.168–175.

EDELWEISS, N. Bancos de Dados Temporais: teoria e prática. In: XVII JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, XVIII CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, Belo Horizonte, Brasil. **Anais...** Sociedade Brasileira de Computação, 1998. v.2, p.225–282.

EDELWEISS, N.; OLIVEIRA, J. P. M. de. Modelagem de Aspectos Temporais de Sistemas de Informação. In: IX ESCOLA DE COMPUTAÇÃO, Recife, Brasil. **Anais...** [S.l.: s.n.], 1994.

EFENDIOGLU, D.; FASCHETTI, C.; PARR, T. J. Chronica: a temporal web search engine. In: ICWE. **Anais...** ACM, 2006. p.119–120.

FARFÁN, F. et al. XOntoRank: ontology-aware search of electronic medical records. In: ICDE. **Anais...** IEEE, 2009. p.820–831.

FENG, J. et al. Finding and ranking compact connected trees for effective keyword proximity search in XML documents. **Inf. Syst.**, [S.l.], v.35, n.2, p.186–203, 2010.

FERRO, L. et al. **TIDES Temporal Annotation Guidelines - Version 1.0.2**. MITRE Technical Report MTR 01W0000041. McLean, Virginia: The MITRE Corporation. June 2001.

FROZI, A. **Um mecanismo de consulta temporal por palavras-chave em páginas Web**. 2010. Monografia (Graduação em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

FROZI, A. et al. Chronos - Um Motor de Busca Temporal. In: SESSÃO DE DEMOS, 7., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 25. **Anais...** Belo Horizonte: SBC, 2010. p.13–18.

GAO, D.; SNODGRASS, R. T. Temporal Slicing in the Evaluation of XML Queries. In: VLDB. **Anais...** [S.l.: s.n.], 2003. p.632–643.

GUO, L. et al. XRANK: ranked keyword search over xml documents. In: SIGMOD CONFERENCE. **Anais...** ACM, 2003. p.16–27.

GUTIME. **Adding TIMEX3 Tags**. Disponível em: <http://www.timeml.org/site/tarsqi/modules/gutime/index.html>. Acesso em: 12 jun. 2010.

HRISTIDIS, V. et al. Keyword Proximity Search in XML Trees. **IEEE Trans. Knowl. Data Eng.**, [S.l.], v.18, n.4, p.525–539, 2006.

HULL, D. A. Using Statistical Testing in the Evaluation of Retrieval Experiments. In: SIGIR. **Anais...** ACM, 1993. p.329–338.

JENSEN, C. S. et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In: TEMPORAL DATABASES, DAGSTUHL. **Anais...** [S.l.: s.n.], 1997. p.367–405.

JIN, P. et al. TISE: a temporal search engine for web contents. In: IITA '08, Washington, USA. **Anais...** IEEE Computer Society, 2008. p.220–224.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. **Soviet Physics Doklady.**, [S.l.], v.10, n.8, p.707–710, February 1966.

LI, G. et al. Effective keyword search for valuable lcas over xml documents. In: CIKM. **Anais...** ACM, 2007. p.31–40.

LI, Y.; YU, C.; JAGADISH, H. V. Schema-Free XQuery. In: VLDB. **Anais...** Morgan Kaufmann, 2004. p.72–83.

LI, Y.; YU, C.; JAGADISH, H. V. Enabling Schema-Free XQuery with meaningful query focus. **VLDB J.**, [S.l.], v.17, n.3, p.355–377, 2008.

LIU, Z.; CHEN, Y. Identifying meaningful return information for XML keyword search. In: SIGMOD CONFERENCE. **Anais...** ACM, 2007. p.329–340.

LIU, Z.; CHEN, Y. Reasoning and identifying relevant matches for XML keyword search. **PVLDB**, [S.l.], v.1, n.1, p.921–932, 2008.

LIU, Z.; CHEN, Y. Return specification inference and result clustering for keyword search on XML. **ACM Trans. Database Syst.**, [S.l.], v.35, n.2, 2010.

MANI, I.; WILSON, D. G. Robust Temporal Processing of News. In: ACL. **Anais...** [S.l.: s.n.], 2000.

MANICA, E.; DORNELES, C. F.; GALANTE, R. Supporting Temporal Queries on XML Keyword Search Engines. **JIDM**, [S.l.], v.1, n.3, p.471–486, 2010.

MANICA, E.; GALANTE, R. Suporte a Consultas Temporais por Palavras-chave em XML. In: SESSÃO DE PÔSTERES, 3., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 24. **Anais...** Fortaleza: SBC, 2009. p.17–20.

MANICA, E.; GALANTE, R.; DORNELES, C. F. Mapeamento Automático de Modelos de Dados XML Temporais Ad-hoc para um Modelo de Dados XML Temporal Padrão. In: ESCOLA REGIONAL DE BANCOS DE DADOS, ERBD, 6. **Anais...** [S.l.: s.n.], 2010.

MANICA, E.; GALANTE, R.; DORNELES, C. F. Two-Phase Interception: uma abordagem para suporte a consultas temporais por palavras-chave em xml. In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCOS DE DADOS, WTDBD, 9., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 25. **Anais...** Belo Horizonte: SBC, 2010. p.25–30.

MARKOWETZ, A.; YANG, Y.; PAPADIAS, D. Keyword search over relational tables and streams. **ACM Trans. Database Syst.**, [S.l.], v.34, n.3, 2009.

NAVATHE, S.; NAVATHE, S. **Sistemas de Banco de Dados**. São Paulo, Brasil: Pearson/Addison Wesley, 2005.

NEGRI, M.; MARSEGLIA, L. **Recognition and normalization of time expressions**: itc-irst at tern 2004. Technical report, ITC-irst, Trento.

NUNES, S.; RIBEIRO, C.; DAVID, G. Use of Temporal Expressions in Web Search. In: ECIR. **Anais...** Springer, 2008. p.580–584. (Lecture Notes in Computer Science, v.4956).

PASCA, M. Towards temporal Web search. In: SAC. **Anais...** ACM, 2008. p.1117–1121.

QUINLAN, J. R. **C4.5**: programs for machine learning. [S.l.]: Morgan Kaufmann, 1993.

RIZZOLO, F.; VAISMAN, A. A. Temporal XML: modeling, indexing, and query processing. **VLDB J.**, [S.l.], v.17, n.5, p.1179–1212, 2008.

SANTOS, D. et al. Second HAREM: new challenges and old wisdom. In: PROPOR. **Anais...** Springer, 2008. p.212–215. (Lecture Notes in Computer Science, v.5190).

SNODGRASS, R. T.; AHN, I. A Taxonomy of Time in Databases. In: SIGMOD CONFERENCE. **Anais...** ACM Press, 1985. p.236–246.

TARSQI. **Temporal Awareness and Reasoning Systems for Question Interpretation**. Disponível em: <http://www.timeml.org/site/tarsqi/index.html>. Acesso em: 22 mar. 2010.

TATARINOV, I. et al. Storing and querying ordered XML using a relational database system. In: SIGMOD CONFERENCE. **Anais...** ACM, 2002. p.204–215.

TIMEML. **Markup Language for Temporal and Event Expessions**. Disponível em: <http://www.timeml.org>. Acesso em: 10 nov. 2010.

TREETAGGER. **A Language Independent Part-of-Speech Tagger**. Disponível em: <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/>. Acesso em: 10 jun. 2010.

VICENTE-DÍEZ, M. T.; MARTÍNEZ, P. Temporal Semantics Extraction for Improving Web Search. In: DEXA WORKSHOPS. **Anais...** IEEE Computer Society, 2009. p.69–73.

W3C. **XQuery 1.0**: an xml query language. Disponível em: <http://www.w3.org/TR/xquery>. Acesso em: 17 nov. 2010.

WANG, Y. et al. Timely YAGO: harvesting, querying, and visualizing temporal knowledge from wikipedia. In: EDBT. **Anais...** ACM, 2010. p.697–700. (ACM International Conference Proceeding Series, v.426).

WIKIPEDIA. **Student's t-test**. Disponível em: <http://en.wikipedia.org/wiki/Student's_t-test>. Acesso em: 12 mar. 2010.

XU, Y.; PAPAKONSTANTINOU, Y. Efficient Keyword Search for Smallest LCAs in XML Databases. In: SIGMOD CONFERENCE. **Anais...** ACM, 2005. p.537–538.

APÊNDICE A ESPECIFICAÇÃO COMPLETA DE TKC

```
tempQuery ::= {kws} tempOp op
kws ::= string
tempOp ::= ( After | Before | Contains | During | Equals | FinishedBy |
             Finishes | Intersect | Meets | MetBy | OverlappedBy |
             Overlaps | StartedBy | Starts | Após | Depois de |
             A partir de | Following | Antes de | Contém | Durante |
             Igual | Termina em | É terminado por | Intersecciona | Encontra |
             É encontrado por | Sobrepõem | É sobreposto por | Inicia |
             É iniciado por | Em | No(a) | De(a|o) | Para | Entre)
op ::=  ( instant | interval | compositeInterval )
instant ::= [ number / ] [ number / ] number
interval ::= [ instant , instant ]
compositeInterval ::= word  [value] granularity
word ::= ( Current | Last | Next | Atual | Último(s) | Próximo(s))
granularity ::= ( Day(s) | Week(s) | Month(s) | Year(s) | Decade(s) |
                  Century(ies) | Dia(s) | Semana(s) | Mês(es) |
                  Semestre(s) | Ano(s) | Década(s) | Século(s))
value ::= integer
```

APÊNDICE B DIAGRAMA DE CLASSES DE TKC