INSTITUTO MILITAR DE ENGENHARIA

MIRIAM OLIVEIRA DOS SANTOS

ARMAZENAMENTO E RECUPERAÇÃO DE DOCUMENTOS XML HETEROGÊNEOS : APLICANDO TÉCNICAS DE KDD PARA APOIAR O PROJETO FÍSICO EM SGBD's XML NATIVOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Orientador: Profa. Maria Cláudia Reis Cavalcanti - D.Sc.
Co-orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.

Rio de Janeiro

2007

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

S237a Santos, Miriam Oliveira dos
Armazenamento e Recuperação de Documentos XML Heterogêneos: Aplicando Técnicas de KDD para Apoiar o Projeto Físico em SGBD's XML Nativos / Miriam Oliveira dos Santos - Rio de Janeiro: Instituto Militar de Engenharia, 2007.

173 p.: il., graf., tab.

Dissertação (mestrado) - Instituto Militar de Engenharia – Rio de Janeiro, 2007.
1. XML. 2. Dados Semi-Estruturados. 3. Datamining. 4. SGBD's XML Nativos

I. Titulo II. Instituto Militar de Engenharia

CDD 005.3

INSTITUTO MILITAR DE ENGENHARIA

MIRIAM OLIVEIRA DOS SANTOS

ARMAZENAMENTO E RECUPERAÇÃO DE DOCUMENTOS XML HETEROGÊNEOS: APLICANDO TÉCNICAS DE KDD PARA APOIAR O PROJETO FÍSICO EM SGBD's XML NATIVOS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Orientador: Profa. Maria Cláudia Reis Cavalcanti - D.Sc.

Co-orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.

Aprovada em 06 de junho de 2007 pela seguinte Banca Examinadora:

_____

Profa. Maria Cláudia Reis Cavalcanti – D.Sc. do IME – Presidente

_____

Prof. Ronaldo Ribeiro Goldschmidt, D.Sc. do IME

_____

Profa. Cláudia Maria Garcia Medeiros de Oliveira, PhD do IME

_____

Profa. Fernanda Araújo Baião Amorim, D.Sc da UNIRIO

Rio de Janeiro

2007

Aos meus pais Heloisa Maria Oliveira dos Santos e José dos Santos

A minha irmã Betânia Oliveira dos Santos

AGRADECIMENTOS

A Deus pelas oportunidades, conquistas e vitórias.

A minha família, em especial meus pais, Heloisa e José e minha irmã Betânia, que acompanharam em todas as fases de minha vida, desde a infância, a vontade de um dia poder chegar onde cheguei. E que junto comigo compartilharam mais de dois anos de desafios, conquistas, provações, vitórias e grandes esforços para que fosse possível atingir este objetivo. Sem o apoio da minha família nada disto teria sido possível.

A professora Maria Claúdia Reis Cavalcanti, pela orientação, dedicação, paciência, compreensão, confiança e incentivo que mesmo diante de vários obstáculos nunca deixou de acreditar no meu potencial como aluna e como ser humano, tornando possível a concretização do meu sonho.

Ao professor Ronaldo Ribeiro Goldschmidt, pela orientação e atenção ao longo do curso.

Aos professores Sérgio Côrtes e Elvira Maria Antunes Uchoa, por despertarem em mim o interesse pela área de Banco de Dados nas aulas de graduação do curso Tecnólogo em Processamento de Dados na PUC-Rio.

Aos meus amigos, alunos do curso de mestrado em Sistemas e Computação, pela convivência, troca de experiências e amizade.

Ao Instituto Militar de Engenharia, em especial a Seção de Engenharia de Sistemas, pela excelente qualidade do curso de mestrado e a oportunidade ímpar de passar por esta instituição. E a todos os professores e funcionários que fizeram e sempre irão fazer parte da minha vida.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

API	*Aplication Programming Interface*
ASC II	*American Standard Code for Information Interchange*
BI	*Business Intelligence*
BDB XML	*Berkeley DB XML*
BLOB	*Binary large Objects*
CLOB	*Character Large Objects*
DOM	*Document Object Model*
DTD	*Document Type Definition*
GAIDoX	*Guia para Armazenamento e Indexação de Documentos XML*
HTML	*Hyper Text Markup Languagesl*
KDD	*Knowledge Discovery in Databases*
LOB	*Large Objects*
PCDATA	*Parsed Character Data*
RDF	*Resource Description Format*
SAX	*Simple API for XML*
SGBD	*Sistemas Gerenciadores de Bancos de Dados*
SQL	*Structured Query Language*
XML	*Extensible Markup Language*
XPATH	*XML Path Language*
XQUERY	*XML Query*
XSLT	*Extensible Stylesheet Language Transformation*
W3C	*World Wide Web Consortium*

RESUMO

Os avanços tecnológicos têm contribuído para um expressivo aumento do volume e da diversidade de informações que circulam atualmente pela Web. Muitas dessas informações encontram-se organizadas em documentos XML e provêm de várias fontes. O gerenciamento de conteúdo envolvendo documentos XML heterogêneos carece de mecanismos que orientem o processo de armazenamento desses documentos, de forma a facilitar sua posterior recuperação.

Assim sendo, este trabalho apresenta uma estratégia baseada em princípios de Descoberta de Conhecimento e Mineração de Dados para orientar o processo de armazenamento de documentos XML heterogêneos em SGBD's XML Nativos.

Avalia-se o potencial da estratégia proposta, através de um estudo de caso utilizando o SGBD XML Nativo Berkeley DB XML.

ABSTRACT

Technological advances have contributed for an expressive increase of Web information, in terms of volume and diversity. Much of this information is organized as XML documents and come from many different sources. Content management for heterogeneous XML documents does not provide efficient mechanisms for guiding the storage of these documents, in such a way that facilitates their retrieval.

Therefore, this paper presents a strategy based on Knowledge Discovery and Data Mining to guide the storage of heterogeneous XML documents.

The potential of the proposed strategy is analysed by a case study using the Native XML Database Berkeley DB XML.

1 INTRODUÇÃO

Os avanços tecnológicos ocorridos, sobretudo na última década, têm contribuído para um expressivo aumento do volume e da diversidade de informações que circulam atualmente pela Web. São inúmeras e heterogêneas as fontes de dados, assim como os dispositivos de saída a que tais informações se destinam. Assim sendo, as organizações que lidam com este tipo de informação necessitam de mecanismos voltados para o gerenciamento de conteúdo (PEREIRA; BAX, 2002) que apóiem a captação, organização e distribuição de informação corporativa. Os Sistemas de Gerenciamento Eletrônico de Documentos (GED's) e os Sistemas Gerenciadores de Bancos de Dados (SGBD's) são exemplos de recursos voltados ao gerenciamento de conteúdo. Os GED's têm sido utilizados no armazenamento, localização e recuperação de informações não estruturadas (MACEDO, 2003). Já os SGBD's tradicionais, por outro lado, têm sido aplicados na gestão de informações estruturadas e requerem portanto, a definição de estruturas de dados prévias. Os SGBD's tradicionais já oferecem soluções para o tratamento da hetereogeneidade destas informações, como por exemplo, o mapeamento de esquemas (FLORESCU; KOSSMANN, 1999), (VIEIRA, 2002) e arquiteturas baseadas em mediadores (RUBERG et al., 2002), entre outras.

As informações semi-estruturadas são definidas por (FLORESCU, 2005) como uma categoria de informação que não pode ser facilmente e eficientemente modelada por apresentarem uma estrutura irregular. Há uma demanda crescente no sentido de que os SGBD's tradicionais se voltem para o gerenciamento de documentos semi-estruturados e heterogêneos, que vêm sendo representados através da linguagem XML. Em geral, nos SGBD's relacionais o armazenamento consiste na fragmentação do documento em diversas tabelas através de técnicas de mapeamento (FLORESCU; KOSSMANN, 1999), (VIEIRA, 2002). No mapeamento direto, tabelas específicas são criadas a partir dos elementos (*tags*) e atributos do documento XML. Já no mapeamento indireto existem estruturas genéricas que armazenam o conteúdo dos elementos e atributos presentes em qualquer documento XML. Uma das desvantagens do mapeamento direto é a necessidade de se criar uma estrutura específica. Já no mapeamento indireto, mesmo que não seja preciso a criação de estruturas específicas, ainda há a necessidade de um conhecimento prévio do esquema genérico.

Nos SGBD's XML Nativos, os documentos são armazenados sem que haja necessidade de se realizar mapeamentos. Os SGBD's XML Nativos, próprios para o armazenamento desta categoria de documentos, oferecem maior flexibilidade em relação aos SGBD's relacionais tradicionais, porque não há obrigatoriedade da definição de um esquema. Os documentos são armazenados dentro de coleções que são repositórios de documentos XML. Não há restrição para os "tipos" de documentos armazenados em uma mesma coleção, que pode conter documentos com diferentes estruturas.

1.1 MOTIVAÇÃO

Embora existam diversas abordagens para o armazenamento de documentos XML, o tratamento de um acervo de grandes proporções composto por informações oriundas de diversas fontes exige uma solução mais elaborada.

O interesse em monitorar diferentes fontes de documentos está presente em várias áreas, entre as quais podem ser citadas: governo, gerência de bibliotecas e prospecção tecnológica.

No governo, por exemplo, através da Receita Federal, tem-se interesse em confrontar as informações declaradas no imposto de renda com documentos provenientes de outras fontes, em busca de dados conflitantes.

Um outro exemplo que pode ilustrar a demanda por uma análise integrada de informações de diferentes fontes, é quando um bibliotecário faz prospecção de informações para realizar a compra de novos livros. Para isto, resolve verificar o acervo de outras bibliotecas, que pode, através de convênios, ceder ou disponibilizar estes livros para esta biblioteca. Havendo ou não um livro específico, o bibliotecário irá necessitar pesquisar em diferentes livrarias eletrônicas, a disponibilidade e os preços destes livros. A partir da definição do livro, é preciso verificar maiores informações sobre o autor disponibilizadas por meio de seu currículo acadêmico ou profissional. Pode ainda surgir interesse em se pesquisar artigos e outras obras científicas publicadas pelo autor. Para isto, o bibliotecário precisa ter acesso a catálogos de artigos e até mesmo aos artigos em sua forma integral. Convém observar que todas estas informações se completam e podem ser provenientes de documentos de diversas fontes.

Na área de prospecção tecnológica (COELHO et al., 2004), os esforços estão

concentrados na construção de conhecimento através da análise sistemática de grandes volumes de dados. Os exercícios de prospecção tecnológica visam identificar oportunidades, ameaças e impactos potenciais no futuro, estando relacionados a atividades como: criatividade, monitoramento e métodos estatísticos.

Embora a maioria destas informações possa ser encontrada na Internet, a busca de forma pouco sistemática nos diversos sistemas de busca disponíveis, pode levar a decisões erradas, como por exemplo, a compra indevida de um livro, ou a decisão, pouco embasada, por investir em uma dada tecnologia, ou ainda, a aprovação de declarações de renda suspeitas. No entanto, ao coletar e armazenar, de modo mais sistemático, tais documentos, permite-se criar um ambiente onde a busca pode ser feita de forma mais eficiente.

Muitos destes documentos, embora freqüentemente em formatos diferentes, poderiam ser disponibilizados em formato XML. Para tornarmos o problema de armazenar os diferentes documentos mais tratável, neste trabalho supomos que tais documentos estejam disponíveis na forma de documentos XML.

1.2 CARACTERIZAÇÃO DO PROBLEMA

O gerenciamento de conteúdo envolvendo documentos XML heterogêneos, para tratar um acervo de grandes proporções e com documentos de diferentes fontes, carece de mecanismos que orientem o processo de armazenamento desses documentos. Fazendo uma analogia com a mesa de um burocrata, como resolver o problema de se localizar um dado documento a partir de uma pilha de documentos, já que estes documentos podem estar organizados: (i) sem critério algum, em uma única pilha; (ii) sem critério algum, em pilhas separadas; (iii) com algum critério baseado no conteúdo de cada documento, em pilhas separadas, que mantenham documentos similares em cada uma destas pilhas. Intuitivamente, (iii) parece ser a melhor opção. Neste sentido, um tratamento prévio deste acervo poderia ser adotado, de modo a viabilizar o seu armazenamento e a recuperação de documentos.

Várias das abordagens encontradas na literatura (ABITEBOUL et al., 2005), (FLESCA et al., 2005), (YANG et al., 2005), (NIERMAN; JAGADISH, 2002), (LEE et al., 2001) propõem soluções voltadas para acervos com baixo grau de heterogeneidade, onde a intenção é extrair um esquema único. Já (LI et al., 2004), embora facilite a formulação de consultas sobre documentos heterogêneos, não provê solução para o

armazenamento destes documentos. Assim sendo, este trabalho tem como objetivo propor uma estratégia baseada em princípios de Descoberta de Conhecimento e Mineração de Dados para orientar o processo de armazenamento de documentos XML heterogêneos.

1.3 VISÃO GERAL DA PROPOSTA E CONTRIBUIÇÕES

Neste trabalho, propomos uma estratégia que envolve o uso de técnicas de KDD, como clusterização, sumarização e associação, no sentido de minimizar a heterogeneidade dos documentos, através da separação do acervo em grupos mais homogêneos.

As contribuições propostas neste trabalho estão relacionadas a seguir:

- Especificação de uma estratégia, composta por etapas bem definidas, que auxilie no armazenamento e na recuperação mais eficientes de documentos XML heterogêneos em SGBD's XML Nativos;

- Implementação de um protótipo para apoiar a estratégia proposta que permite auxiliar o usuário a desenvolver um projeto físico de banco de dados adequado e individualizado a cada conjunto de documentos que se deseje armazenar.

- Aplicação de técnicas de KDD, técnicas comumente utilizadas em Recuperação de Informação, no projeto físico de SGBD's XML Nativos.

- A realização de um estudo de caso envolvendo a construção de um acervo heterogêneo significativo que ilustre o potencial da estratégia proposta.

1.4 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado da seguinte forma: No Capítulo 2, é

apresentada uma visão geral da tecnologia XML, conceituando-a como uma tecnologia voltada para a representação do dado semi-estruturado, descrevendo brevemente a estrutura do documento XML, mostrando como estes documentos podem ser classificados, as tecnologias e formas de armazenamento, além das oportunidades e desafios que os norteiam. No Capítulo 3, são descritos os conceitos básicos e tarefas de KDD com ênfase nas tarefas de clusterização, associação e sumarização que fazem parte da estratégia proposta. No Capítulo 4, é apresentada detalhadamente a estratégia proposta neste trabalho, descrevendo todas as etapas necessárias. O capítulo 4 contém ainda uma análise comparativa dos trabalhos correlatos em relação à estratégia proposta. No capítulo 5, é descrita a arquitetura do processo implementado, as plataformas utilizadas, e é realizada uma descrição completa do uso do protótipo implementado, através da exemplificação de sua utilização. Ainda neste capítulo foi realizado um detalhamento do SGBD XML Nativo utilizado no protótipo e no estudo de caso. O capítulo 6 apresenta um estudo de caso e uma análise comparativa que ilustra o potencial da estratégia proposta. Os resultados do estudo de caso realizado são apresentados e analisados em detalhe. No capítulo 7, é colocado em prática um retrospecto sobre toda a estratégia, apresentando as conclusões sobre o trabalho, as contribuições obtidas e são sugeridas algumas possibilidades de estudos futuros.

2 ARMAZENAMENTO DE DOCUMENTOS XML

Uma das linguagens mais comuns para exibição de dados na Internet é o HTML, (*Hypertext Markup Language*). O HTML é utilizado apenas para formatação e estruturação dos documentos na Web (ELMASRI; NAVATHE, 2005). Quando esta linguagem foi criada os sites eram utilizados para exibir informações estáticas não havendo necessidade de exibir conteúdo dinâmico. Com o passar do tempo a Internet passou a comportar aplicações em que a existência de conteúdo dinâmico se tornaria fundamental ampliando a necessidade de armazenamento e recuperação. As informações representadas em HTML, por não possuírem nenhuma estrutura, não podiam ser traduzidas para um modelo relacional (FLORESCU, 2005). Assim, outras linguagens surgiram para suprir esta necessidade. A linguagem XML *(Extensible Markup Language)* é uma destas linguagens, que permitiu a representação e troca de documentos (ELMASRI; NAVATHE, 2005).

Segundo (VAKALI et al., 2005, p.1), "XML é uma linguagem hierárquica baseada em *tags* que aplicam relacionamento do tipo pai-filho". A linguagem XML apresenta duas características marcantes. A primeira característica, é que um documento expresso em XML é auto-descritivo, pois permite associar significado aos dados através das etiquetas (*tags)* ou elementos presentes no próprio documento. A segunda característica é que os documentos XML têm estrutura lógica definida e flexível que pode ser representada por um DTD (*Document Type Definition*) ou XML *Schema* (SUN et al., 2002). Estas duas características facilitam o armazenamento e recuperação dos dados armazenados, diferente da linguagem HTML onde as *tags* são utilizadas apenas para formatação e não descrevem o conteúdo a ser exibido.

O armazenamento de documentos XML, no entanto, não é uma tarefa simples. Em (FLORESCU, 2005), a autora afirma o seguinte:

> *"As metodologias para armazenamento em bancos de dados tradicionais necessitam que haja conhecimento prévio do que será manipulado para que a partir deste conhecimento seja gerado um esquema obedecendo tais características e só então os dados possam ser armazenados. Freqüentemente a estrutura final para*

> *armazenamento de determinadas categorias de informação não pode ser antecipada. Estas informações passam por sucessivos estágios de processamento e refinamento até que possam ser melhor compreendidas"*.

Esta categoria de informação, que não pode ser facilmente e eficientemente modelada, é conhecida como dados semi-estruturados. Os dados semi-estruturados apresentam uma estrutura irregular, isto é, os documentos que se deseja "colecionar", em sua grande maioria, não apresentam uma estrutura idêntica. Os dados semi-estruturados também podem ser vistos como um caso extremo de evolução de esquema, onde os dados têm um complexo relacionamento com os esquemas que os descrevem *(FLORESCU, 2005).*

As duas definições apresentadas sugerem que os dados semi-estruturados requerem maior liberdade de representação de informação e maior fidelidade na representação da realidade, onde a individualidade da estrutura de cada documento é preservada e por isto há irregularidades que levam a não uniformidade das informações. Além disso, a informação é mutável e o seu modelo também. A linguagem XML vem atender a esta demanda não impondo a existência de esquemas nem conhecimento prévio do que se quer armazenar. Além disso, o projeto de esquemas XML flexíveis permite verificar que um conjunto de documentos pouco uniformes a serem armazenados e manipulados estejam em conformidade com algumas regras.

Resumindo, segundo *(FLORESCU, 2005)*, o papel dos esquemas XML é assegurar significado para o dado, permitir buscas automáticas sobre os dados, comparação e processamento. No entanto, a sua utilização deve ser cuidadosa, de modo a não impor a rigidez do modelo de dados relacional. O ideal é que haja um equilíbrio, isto é, não se deve eliminar completamente o uso de esquemas XML, mas também não se deve depender de sua existência ou permitir que restrinjam a manipulação correta dos fatos (informação).

Nas subseções seguintes, é apresentada uma visão geral da tecnologia XML com ênfase nos conceitos e tecnologias associadas e nas abordagens para o armazenamento de documentos XML, discutindo o armazenamento em SGBD's habilitados e em SGBD's XML Nativos.

2.1 CONCEITOS E TECNOLOGIAS ASSOCIADAS

Há uma diferença entre os conceitos empregados em documentos XML bem formados e documentos XML válidos. O documento XML é considerado bem formado se respeitar algumas condições descritas em (ELMASRI; NAVATHE, 2005, p.610):

> *"Estes documentos devem seguir as diretrizes sintáticas do modelo de árvore, deve começar com uma declaração XML, deve ter um único elemento raíz e cada elemento precisa incluir um par correspondente de tags de início e fim. Isto assegura que os elementos aninhados especificam uma estrutura de árvore bem formada."*

Já para o documento XML ser considerado válido, há um critério mais rigoroso. Neste caso, além de bem formado, o documento deve seguir a estrutura de um esquema de validação previamente especificado.

Em torno desta linguagem existem vários conceitos e tecnologias relacionadas que serão apresentados nesta seção.

2.1.1 LINGUAGENS E APIS XML

Entre as diversas linguagens de definição de esquemas ou estruturas para documentos XML podemos citar *DTD* ou *XML Schema* (ELMASRI; NAVATHE, 2005) como as mais utilizadas. O objetivo destas linguagens é especificar como o documento deve estar estruturado, reforçando que os documentos XML obedeçam a estas características para que possam ser armazenados e/ou manipulados. Através de um esquema ou estrutura é possível impor a ordem dos elementos e tipos de dados, como é feito nos esquemas de bancos de dados tradicionais.

Independente da existência de esquemas, o conteúdo dos documentos XML pode ser consultado, isto é, é possível selecionar subconjuntos de dados dentro destes documentos. Para realizar estas consultas foram especificadas duas linguagens de

consulta XML: a Xpath[1], que se baseia na estrutura de caminhos, e a Xquery[2], considerada mais completa.

A XPath (*XML Path Language)* utiliza expressões de caminhos para acessar qualquer item de dado no documento XML. Uma expressão Xpath retorna uma coleção de nós de elementos que satisfazem certos padrões especificados na expressão. Os nós na expressão Xpath são nós da árvore de documentos que são nomes de *tags* (elemento), ou de atributo. Dois separadores principais são utilizados durante a especificação de um caminho: barra única (/) e barras duplas (//). Uma barra única antes de uma *tag* especifica que a *tag* deve aparecer como filho direto da *tag* anterior (pai); já as *tags* duplas indicam que a *tag* pode aparecer como um descendente da *tag* anterior de qualquer nível. As barras duplas são utilizadas quando não sabemos o caminho completo (ELMASRI; NAVATHE, 2005).

A XQuery (*XML Query Language)* é uma tentativa da W3C de disponibilizar uma linguagem de consulta que provê a mesma funcionalidade que o SQL *(Structured Query Language)* apresenta para os bancos de dados relacionais. O Xquery permite a especificação de consultas mais genéricas em um ou mais documentos XML. A forma típica de uma consulta Xquery é conhecida por expressão FLWR que representa as quatro cláusulas principais do Xquery (ELMASRI; NAVATHE, 2005). A Tabela 2.1 especifica os comandos FLWR.

TAB. 2.1 – Descrição dos comandos FLWR

Expressão	**Descrição**
For	Variáveis ligadas a nodos individuais (elementos)
Let	Variáveis ligadas a coleção de nodos (elementos)
Where	Condições qualificadoras
Return	Especificação do resultado da consulta

Para manipulação programática de documentos XML foram especificadas APIs de manipulação XML, de modo a facilitar o desenvolvimento e a portabilidade de aplicações que lidam com estes documentos. As APIs *DOM* e *SAX* (CHAUDHRI, 2003) são as duas principais API's para manipular documentos XML em aplicações DOM. *Document Object Model* é um modelo para armazenar e manipular em memória documentos com estrutura hierárquica, independente da linguagem de programação ou

[1] http://www.w3.org/TR/xpath
[2] http://www.w3.org/TR/xquery/

sistema operacional. O Analisador DOM analisa um documento XML e constrói uma árvore a qual pode ser utilizada para percorrer os vários nós. O SAX (*Simple API for XML),* processa o documento por partes a cada abertura e fechamento de *tag*. Como resultado, o documento é analisado passo a passo e pode ser manipulado antes de ter sido percorrido por inteiro (CHAUDHRI, 2003).

2.1.2 CLASSIFICAÇÃO DOS DOCUMENTOS XML

Os documentos XML podem ser caracterizados de acordo com o tipo de conteúdo que transportam. Podendo ser classificados em documentos centrados em dados e documentos centrados em textos.

Os documentos XML centrados em dados (*data-centric*) são documentos que utilizam XML como transporte de dados. Estes documentos têm itens de dados pequenos que seguem uma estrutura específica e conseqüentemente podem ser extraídos de bancos de dados (ELMASRI; NAVATHE, 2005).

Os documentos XML centrados em textos (*document-centric*) são documentos que utilizam XML para grandes quantidades de textos não estruturados, como artigos de jornais, revistas ou livros. Existem poucos ou nenhum elemento de dados estruturados nesses documentos (ELMASRI; NAVATHE, 2005).

As fig.2.1 e 2.2 exemplificam as classificações descritas.

```
<?xml version="1.0" ?>
- <CURRICULUM>
  - <DADOSPESSOAIS>
      <NOME>MIRIAM</NOME>
    </DADOSPESSOAIS>
  - <FORMACAO>
      <INSTITUICAO>IME</INSTITUICAO>
      <NIVEL>MESTRADO</NIVEL>
      <AREA>BANCO DE DADOS</AREA>
    </FORMACAO>
</CURRICULUM>
```

FIG. 2.1. Representação de documento XML centrado em dados

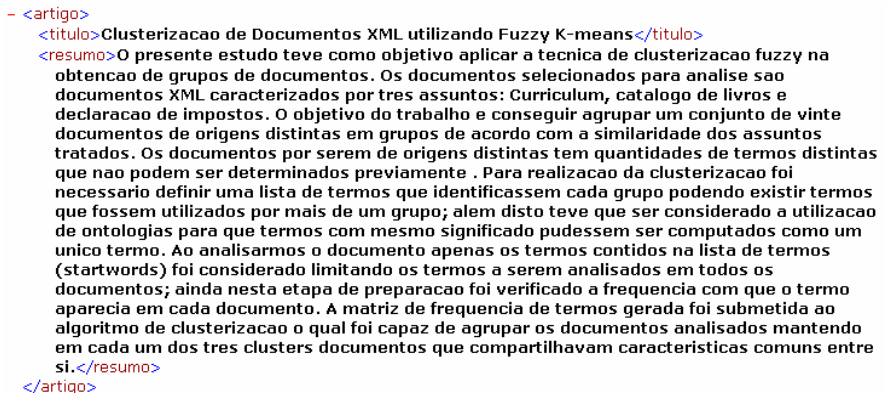


FIG. 2.2. Representação de documento XML centrado em textos.

Independente do tipo de documento XML sendo tratado, estes documentos precisam ser armazenados, consultados e manipulados. Inicialmente estes documentos eram armazenados em SGBD's tradicionais tais como SGBD's relacionais até que surgissem os primeiros SGBD's XML Nativos para atender às características específicas deste tipo de documento.

2.2 TECNOLOGIAS E ABORDAGENS PARA O ARMAZENAMENTO XML

Existem algumas abordagens para armazenamento e recuperação de documentos XML que atualmente são adotadas pelos SGBD's Relacionais, Objeto Relacionais e SGBD's XML Nativos.

A escolha da abordagem para armazenamento depende do tipo de aplicação e o fator mais importante nesta escolha é identificar o que será armazenado: documentos centrados em dados ou documentos centrados em textos (BOURRET, 2004). Um outro fator importante é saber como este documento será usualmente recuperado, isto é, se em sua forma íntegra ou se em fragmentos do mesmo.

As duas principais abordagens para o armazenamento de documentos XML envolvem SGBD's Relacionais e SGBD's XML Nativos. Em geral, nos SGBD's Relacionais o armazenamento consiste na fragmentação do documento em diversas tabelas através de técnicas de mapeamento. No mapeamento direto, tabelas específicas são criadas a partir dos elementos (*tags*) e atributos do documento XML. Já no mapeamento indireto existem estruturas genéricas que armazenam o conteúdo dos

elementos e atributos presentes em qualquer documento XML. Uma das desvantagens do mapeamento direto é a necessidade de se criar uma estrutura específica. Já no mapeamento indireto, mesmo que não seja preciso a criação de estruturas específicas, ainda há a necessidade de um conhecimento prévio do esquema genérico. Além disso, outra desvantagem dos SGBD's relacionais está na forma de armazenamento que é realizada de forma particionada, o documento é fragmentado em várias tabelas, o que pode prejudicar a recuperação do mesmo no formato original (VAKALI et al., 2005).

Já nos SGBD's XML Nativos, os documentos são armazenados sem que haja necessidade de se realizar mapeamentos. Os SGBD's XML Nativos, próprios para o armazenamento desta categoria de documentos, oferecem maior flexibilidade em relação aos SGBD's relacionais tradicionais, porque não há obrigatoriedade da definição de um esquema. Os documentos são armazenados dentro de coleções que são repositórios de documentos XML. Não há restrição para os tipos de documentos armazenados em uma mesma coleção, que pode conter documentos com diferentes estruturas.

Há ainda os SGBD´s Relacionais chamados habilitados, que permitem um armazenamento similar ao oferecido pelos SGBD´s XML Nativos.

Nas subseções seguintes, são detalhadas as formas de armazenamento em SGBD's habilitados, na maior parte representados por SGBD's Relacionais, e o armazenamento em SGBD's XML Nativos.

2.2.1 ARMAZENAMENTO EM SGBD'S HABILITADOS

O Armazenamento em SGBD's habilitados pode ser realizado de duas formas: ou se fragmenta os documentos em várias tabelas ou se mantém os documentos armazenados de forma contígua, íntegra, sendo mantidos em uma única tabela e em alguns casos em uma única coluna utilizando-se de tipos específicos. O Armazenamento em SGBD's habilitados pode ser realizado utilizando-se ou não de mapeamentos.

2.2.1.1 ARMAZENAMENTO EM SGBD'S HABILITADOS SEM MAPEAMENTO

Neste caso, o SGBD é utilizado como um mero repositório de dados. Por exemplo, no SGBD Oracle há tipos de variáveis responsáveis por armazenar grandes blocos de dados não estruturados. Inicialmente, os documentos XML eram armazenados neste SGBD em colunas do tipo CLOB, sem nenhum recurso adicional para consultas sobre os dados armazenados, a não ser os já oferecidos para dados textuais. A recuperação destes documentos só era possível recuperando todo o documento, não sendo possível realizar consultas ou alterações buscando-se parte do documento. Posteriormente foi criado um tipo especial *XMLTYPE*, que encapsula o modelo de armazenamento CLOB provendo acesso através da linguagem de consulta *Xpath*, o que facilita a manipulação do documento XML (CHAUDHRI, 2003).

Nas subseções seguintes apresentamos brevemente as diferentes abordagens para o armazenamento em SGBD's habilitados com mapeamento, porém uma explicação mais aprofundada pode ser encontrada em (FLORESCU e KOSSMANN, 1999), (TIAN et al., 2002), (VIEIRA,2002).

2.2.1.2 ARMAZENAMENTO EM SGBD'S HABILITADOS COM MAPEAMENTO

Existem dois tipos de mapeamentos que os SGBD's habilitados se utilizam. São eles: mapeamento direto e mapeamento indireto.

Para exemplificação das formas de mapeamento será utilizada a fig. 2.3a que contém um exemplo de documento XML. A fig. 2.3b exemplifica graficamente a estrutura do documento XML adotado.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SigmodRecord>
 <issue>
    <volume>15</volume>
    <number>2</number>
    <articles>
       <article>
              <title articleCode="152033">A DBMS prototype to support...</title>
              <authors>
                        <author>F Andersen</author>
                        <author>H Blanken</author>
                        <author>K Kuespert</author>
                        <author>P Dadam</author>
                        <author>R Erbe</author>
              </authors>
       </article>
   </articles>
 </issue>
</SigmodRecord>
```

FIG. 2.3a documento XML

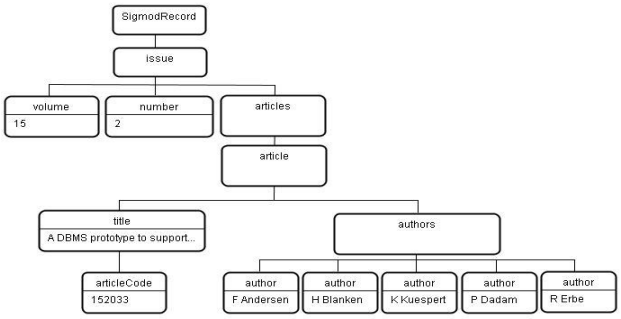


FIG. 2.3b representação do documento XML

MAPEAMENTO DIRETO

Para exemplificar a forma de mapeamento direto, podemos nos servir da representação específica para os elementos (VIEIRA, 2002). Nesta abordagem, cada elemento XML possui uma estrutura própria no SGBD, de forma que atributos de um elemento são atributos da estrutura equivalente. O maior problema desta abordagem é identificar a ordem em que os elementos aparecem no documento. Existem dois tipos de abordagens para os elementos em SGBD's Relacionais: a abordagem binária ou atributo e a abordagem específica ou DTD.

A representação binária ou atributo é descrita da seguinte forma: cada aresta do gráfico que representa uma instância do documento XML é representada por uma tabela. Todas as arestas com mesmo nome irão pertencer a uma mesma tabela. A fig 2.4 demonstra a estrutura necessária para realização deste mapeamento.

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
SIGMODRECORD	1	1	Null	Null	2

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
ISSUE	2	1	Null	Null	3

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
VOLUME	3	1	15	Null	0

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
NUMBER	3	2	2	Null	0

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
ARTICLES	3	3	Null	Null	4

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
ARTICLE	4	1	Null	Null	5

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
TITLE	5	1	Null	A DBMS prototype to support...	0

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
AUTHORS	5	2	Null	Null	6

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
ARTICLECODE	6	1	Null	152033	0

TABELA	COLUNAS				
	Origem	Ordem	VInt	VString	Destino
AUTHOR	6	2	Null	F Andersen	0
	6	3	Null	H Blanken	0
	6	4	Null	K Kuespert	0
	6	5	Null	P Dadam	0
	6	6	Null	R Erbe	0

FIG. 2.4 Representação específica (binária ou atributo)

Já na representação específica (DTD), são criadas tabelas de acordo com os relacionamentos existentes entre os elementos do DTD. Pressupõe-se que cada dupla na tabela possua um identificador e uma coluna que identifica o elemento pai.

MAPEAMENTO INDIRETO

Para exemplificar a forma de mapeamento indireto, iremos nos utilizar da *Representação única para os elementos*, da *Representação genérica para os elementos* (VIEIRA, 2002) e da *Representação única com armazenamento orientado a objetos* (TIAN et al., 2002).

Em ambas as representações *única para elementos* e *genérica para elementos*, há identificadores que permitem associar os elementos e atributos a informações como sua *origem*, a *ordem* em que aparecem no documento, sua descrição representada pelo campo *tag,* o *destino* (caso não esteja no último nível da estrutura) e o *dado* propriamente dito.

A representação única faz uso de uma única estrutura (tabela) para representar os elementos de um documento XML, conforme ilustra a fig. 2.5.

Edge Inline				
ORIGEM	ORDEM	TAG	DESTINO	DADO
1	1	SigmodRecord	2	null
2	1	issue	3	null
3	1	volume	0	15
	2	number	0	2
	3	articles	4	null
4	1	article	5	null
5	1	title	0	A DBMS prototype to support...
	2	authors	6	null
6	1	articleCode	0	152033
	2	author	0	F Andersen
	3	author	0	H Blanken
	4	author	0	K Kuespert
	5	author	0	P Dadam
	6	author	0	R Erbe

FIG. 2.5 Representação *edge inline*

Enquanto que a representação genérica, proposta por (FLORESCU e KOSSMANN, 1999), assume que cada elemento XML tem um identificador e que não há diferenciação entre elementos e atributos. Neste tipo de mapeamento, além de existir uma tabela que associa elementos e atributos a estruturas pré-definidas que os identificam, é gerada uma tabela para cada tipo de dado encontrado nos documentos, conforme ilustra a fig. 2.6.

Edge				
ORIGEM	ORDEM	TAG	FLAG	DESTINO
1	1	SigmodRecord	ref	2
2	1	issue	ref	3
3	1	volume	int	v1
	2	number	int	v2
	3	articles	ref	4
4	1	article	ref	5
5	1	title	string	v3
	2	authors	ref	6
6	1	articleCode	string	v4
	2	author	string	v5
	3	author	string	v6
	4	author	string	v7
	5	author	string	v8
	6	author	string	v9

Vint	
vID	Valor
v1	15
v2	2

Vstring	
vID	Valor
v3	A DBMS prototype to support...
v4	152033
v5	F Andersen
v6	H Blanken
v7	K Kuespert
v8	P Dadam
v9	R Erbe

FIG. 2.6 Representação mapeamento *Edge* em tabelas separadas

Em TIAN et al.(2002), é citada a representação única com armazenamento orientado a objetos. A tabela 2.2 ilustra os atributos utilizados nesta estratégia e a fig. 2.7 exemplifica o armazenamento do documento nesta estrutura.

TAB. 2.2 - Descrição dos atributos utilizados na estratégia

Denominação	**Descrição**
Offset	Deslocamento, funciona como chave do elemento, representado no topo da estrutura
Lenght	Tamanho ocupado
Tag	Nome do elemento
Parent	*Offset,* chave do elemento pai
Prev	*Offset* do elemento anterior no mesmo nível
Next	Próximo elemento no mesmo nível
Opt_child	Parâmetro opcional, somente utilizado se existir elementos filhos.
Opt_attr	Parâmetro opcional, somente elementos que possuem atributo(s) o utilizarão.
Opt_text	Somente elementos folhas possuem conteúdo (contéudo dos elementos)

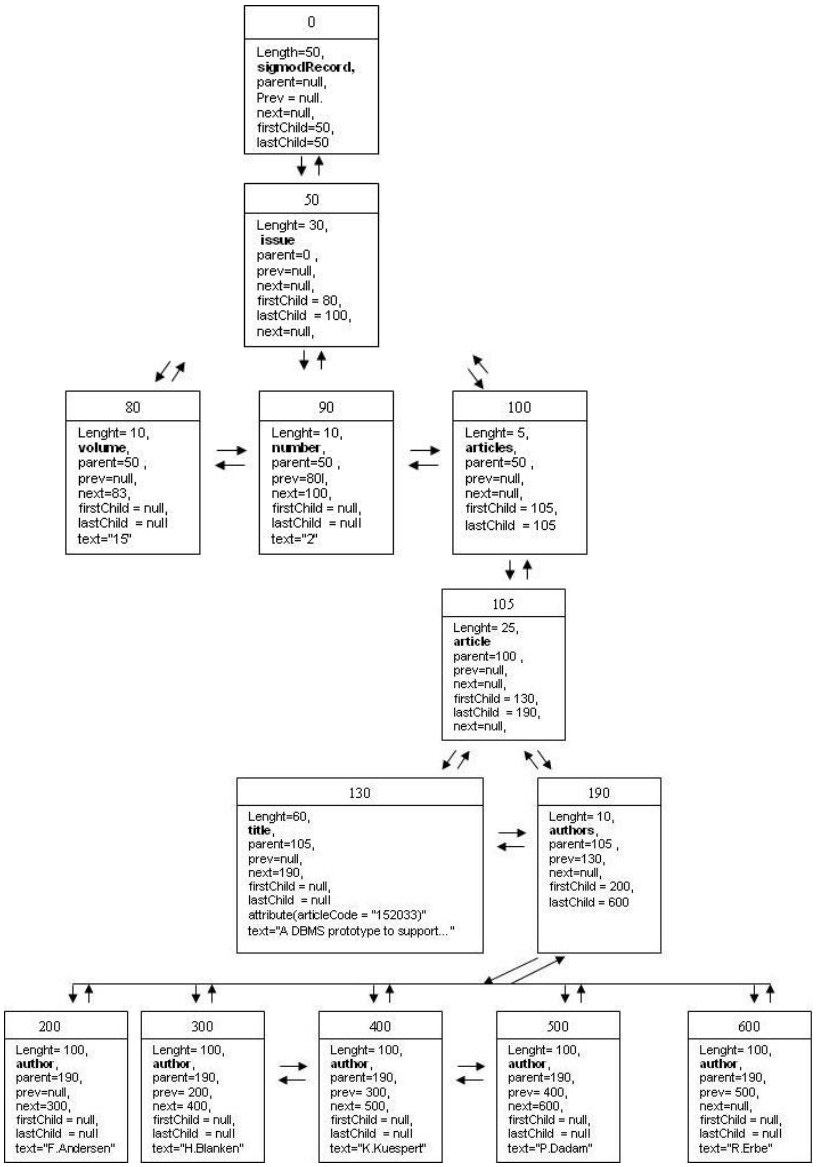


FIG. 2.7 Representação única com armazenamento orientado a objetos

2.2.2 ARMAZENAMENTO EM SGBD'S XML NATIVOS

A empresa Software AG lançou a primeira versão do Servidor XML Tamino em 1999, o qual se classifica como um SGBD XML nativo. Desde então o termo *nativo* tornou-se popular sendo utilizado em diferentes escopos. O termo nativo conforme definido pela Software AG, determina que para um SGBD ser considerado nativo, o sistema deve ser construído e desenhado para manipulação de documentos XML e não somente como uma extensão de funcionalidade construída em uma camada a parte de um sistema de banco de dados (CHAUDHRI et al.,2003).

Segundo VAKALI et al. (2005), o armazenamento em SGBD's XML Nativos traz três benefícios principais. São eles: Escalabilidade, Acesso Rápido a dados e fidedignidade. A Escalabilidade está relacionada ao fato de que os SGBD's XML Nativos são baseados em um formato interno; o Acesso Rápido a dados ocorre por que não há sobrecarga de tempo para mapear, trocar informações e realizar joins entre o XML e outras estruturas internas tais como tabelas; a Fidedignidade está relacionada à habilidade dos SGBD's XML nativos de apresentar o documento XML da mesma forma como foi armazenado, o que não ocorre em SGBD's tradicionais.

Além destes benefícios, os SGBD's XML Nativos oferecem maior flexibilidade em relação SGBD's tradicionais principalmente no armazenamento dos documentos XML. Isto porque na maioria dos SGBD's XML Nativos, diferentemente dos SGBD's tradicionais, não há obrigatoriedade de definição de um esquema. Existem algumas situações em que a utilização de *XML Schemas* ou DTD's pode ser útil, principalmente quando é preciso validar regras, definir índices e quando se deseja intercambiar documentos com aplicações externas.

Nos SGBD's XML Nativos, assim como em bancos de dados tradicionais, os índices são responsáveis por melhorar o desempenho na execução de uma consulta. Alguns SGBD's XML Nativos realizam indexação automática, enquanto que outros apenas disponibilizam este recurso para que o próprio usuário o utilize.

A recuperação das informações dos documentos armazenados é realizada através de linguagens padrão definidas pela W3C, tais como Xquery e XPath. A atualização destes documentos é definida também pela W3C, através da linguagem XUpdate, porém esta ainda está em estágio de definição e é pouco adotada.

Foram selecionados alguns SGBD's XML Nativos que serão detalhados a seguir, para que fossem investigadas as suas principais características. Alguns SGBD's relacionais como o Oracle já oferecem várias das características encontradas nos SGBD's XML Nativos. No entanto, não está no escopo deste trabalho analisá-lo.

Ao final desta seção, realizamos uma análise comparativa sobre os SGBD's Berkeley DB XML, Tamino, OrientX e Exist.

2.2.2.1 BERKELEY DB XML

O Berkeley DB XML teve origem a partir do SGBD Relacional Berkeley DB, cujo código é aberto, e foi desenvolvido pela empresa Sleepycat, tendo sido adquirido recentemente pela Oracle[3]. É um SGBD XML Nativo que se utiliza da linguagem Xquery para acessar documentos armazenados. O Berkeley DB XML (BDB XML) herdou as principais características já existentes no SGBD Relacional Berkeley DB. Uma característica importante é que o BDB XML não funciona como um servidor, pois é usado como uma biblioteca que é ligada (embutida) diretamente dentro de aplicações. Um outro recurso oferecido é a utilização de uma interface de acesso via linha de comando, que permite que o usuário acesse os dados fora da linguagem de programação normalmente utilizada para interagir com o BDB XML. No BDB XML, todos os documentos são armazenados em coleções chamadas de *containers*. Os *containers* são dispositivos que contêm uma coleção de documentos XML e informações sobre estes documentos. O Berkeley DB XML trabalha com índices, provê no controle de transação as propriedades ACID, normalmente providas por SGBD's relacionais, utiliza as linguagens Xquery e Xpath , suporta alterações parciais do documento e oferece integração com diversas linguagens; além de suportar diversas plataformas (SLEEPYCAT, 2005). O SGBD XML Nativo Berkeley DB XML encontra-se descrito com mais detalhes no capítulo 5.

[3] http://www.oracle.com/database/berkeley-db/xml/index.html

2.2.2.2 TAMINO

O Tamino, tem caráter comercial, foi lançado em 1999 pela empresa Software AG[4]. O servidor Tamino oferece algumas das funcionalidades existentes em SGBD's tradicionais tais como segurança, controle de transações, acesso multiusuário e escalabilidade (CHAUDHRI et al.,2003).

O Servidor Tamino pode armazenar e processar qualquer documento XML bem-formado e criar uma definição correspondente quando não existir DTD associado. Não há obrigatoriedade do documento estar associado a um DTD ou SCHEMA para ser armazenado, mas caso faça referência a uma destas estruturas, deverá estar de acordo com sua definição para que seja armazenado e mantido dentro do banco (SCHÖNING, 2001).

Segundo SCHÖNING(2001), a utilização de *schemas* no Tamino é recomendada quando uma das seguintes situações ocorre:

- A definição de índice para o elemento ou atributo;
- O mapeamento de um elemento ou atributo para um aplicativo externo;
- É preciso conceder direitos de acesso no nível de atributos ou elementos;
- A presença ou multiplicidade do elemento deve ser forçada.

O repositório de dados do Tamino consiste de múltiplas coleções de documentos denominados *collections*. Estas coleções são chamadas de *containers* que agrupam documentos. Cada coleção pode estar associada a um XML Schema, que permite validar o documento (CHAUDHRI et al.,2003).

O Servidor XML Tamino suporta três tipos de índices: o índice padrão é o índice baseado em valor, que é o mesmo tipo de índice existente em SGBD's relacionais e que serve para agilizar a pesquisa quando se procura por elementos ou atributos que possui um determinado valor. O segundo tipo de índice oferecido é sobre textos longos que também costuma ser oferecido por SGBD's relacionais. Na indexação de textos, é

[4] http://www.softwareag.com/de/products/tamino/

possível indexar o documento inteiro ou sub-árvores do documento. Por fim, existe um terceiro tipo de índice denominado indexação por estrutura, que beneficia consultas sobre documentos que não estão associados a um esquema (SCHÖNING,2001)(CHAUDHRI et al.,2003).

2.2.2.3 ORIENTX

O SGBD XML Nativo OrientX , Orient Ruc Idke Native XML cujo código é aberto, foi desenvolvido pela Universidade de Renmin da China[5]. Diferentemente de outros SGBD's XML Nativos, força a existência de esquemas por considerar que são fundamentais para o gerenciamento de regras em dados XML. Segundo os idealizadores do OrientX, a obrigatoriedade dos esquemas para o armazenamento dos documentos melhoram a eficiência do armazenamento e a recuperação da informação. O OrientX utiliza-se de um esquema nativo para armazenar, consultar e gerenciar dados XML (MENG et al.,2003).

Dentre as principais características do OrientX destaca-se: suporte a estratégia de clusterização baseado em esquema. O esquema também é utilizado como um guia para escolher uma estratégia apropriada para armazenamento automaticamente. A informação contida no esquema é utilizada para validar o processo de alteração e consultas (MENG et al.,2003).

2.2.2.4 EXIST

O Exist é um SGBD XML Nativo, cujo código é aberto, lançado em 2001, por Wolfgang Meier, da instituição exist-db.org[6]. Tem como característica, o uso de técnicas avançadas como pesquisas de palavras no texto, consultas sobre termos aproximados e expressões regulares. O Exist foi escrito em Java e pode ser executado dentro de um *servlet* ou dentro de uma aplicação (CHAUDHRI et al.,2003). Outras

[5] http://idke.ruc.edu.cn/orientx/
[6] http://exist.sourceforge.net/

características podem ser destacadas, como a não obrigatoriedade do documento XML estar associado a um *Schema XML* ou DTD. Diferentes tipos de documentos podem ser armazenados em uma mesma coleção.

Seu principal foco tem sido suportar o armazenamento e recuperação de documentos XML centrados em texto. O Exist provê uma extensão ao Xpath com objetivo de processar eficientemente consultas sobre os documentos e possui uma estrutura de indexação que consegue recuperar ocorrências de palavras que auxiliam o usuário na consulta do conteúdo textual (CHAUDHRI et al.,2003). O Exist utiliza um esquema de indexação numérica para identificar nós. A indexação provê rápida identificação de possíveis relacionamentos entre nós em uma árvore de documentos. A indexação é aplicada para todos os nós do documento, incluindo-se elementos, atributos, texto e comentários. Ao contrário de outras abordagens, não é necessário explicitar a criação de índices. Todos os índices são gerenciados pelo próprio SGBD que é capaz de criar índices de forma automática (CHAUDHRI et al.,2003).

2.2.2.5 QUADRO COMPARATIVO

Procuramos sintetizar no quadro comparativo presente na tabela 2.3, as principais características e abordagens dos SGBD's XML Nativos citados nesta seção. Pode-se perceber que o uso de SGBD's XML Nativos é uma tecnologia recente e que a maior parte dos SGBD's analisados têm licenciamento *Open Source*, com exceção do Tamino que tem caráter comercial. As linguagens de desenvolvimento adotadas são: *Java* e *C*. Os SGBD's analisados oferecem suporte a estruturas de validação, sem que seu uso torne-se obrigatório, com exceção para o Orient X. Dois destes SGBD's apresentam características peculiares como a criação de metadados pelo Berkeley DB XML e o oferecimento de suporte específico para documentos centrados em textos, que é o caso do Exist. A indexação pode ser realizada de forma manual ou automática e os tipos de índices estão voltados para pesquisas sobre valores exatos, pesquisas sobre textos e dados irregulares. Quanto a linguagens de consulta, o SGBD Exist acusa deficiências para utilização da linguagem Xpath.

Tabela 2.3 - Quadro Comparativo Principais SGBD's XML Nativos

SGBD XML NATIVO	**Berkeley DB XML**	**Tamino**	**Orient X**	**Exist**
Fabricante	Sleepycat	Software Ag	Renmin University	Wolfgang Meier
Ano	-	1999	-	2001
Licença	código aberto	comercial	código aberto	código aberto
Linguagem	C	-	-	Java
APIS	C++, Java , Perl, Python, PHP	-	-	-
Obrigatoriedade de esquema	Não	não	sim	não
Foco/Característica particular	Permite criação de metadados	-	-	Foco em documentos xml do tipo *document-centric*
Indexação	Manual	-	-	Automática
Tipos de índices	Valores exatos, pesquisa em textos, dados irregulares (*tags* que podem ou não existir)	Valores exatos, pesquisa em textos e indexação por estrutura	-	pesquisa em textos
Consulta	-	-	-	Não oferece suporte adequado para Xpath

Os SGBD's em que não se pode apurar determinadas informações aparecem com uma marcação indicativa "-" no quadro correspondente a informação.

3 KDD E MINERAÇÃO DE DADOS

A área de *KDD (Knowledge Discovery in Databases*) denominada Descoberta de Conhecimento em Bases de Dados, surgiu da necessidade de se abstrair conhecimento a partir de grandes bases de dados.

Uma das definições encontradas para o termo KDD diz que: "KDD é um processo de várias etapas, não trivial, interativo e iterativo, para identificação de padrões compreensíveis, válidos, novos e potencialmente úteis a partir de grandes conjuntos de dados" (FAYYAD et al., 1996).

Em GOLDSCHMIDT e PASSOS (2005, p.4), são fornecidos alguns comentários relacionados à definição dada por (FAYYAD et al., 1996).

> *"A expressão não trivial alerta para a complexidade normalmente presente na execução de processos de KDD. O termo interativo indica a necessidade de atuação do homem como responsável pelo controle do processo (...). O termo iterativo, por outro lado, sugere a possibilidade de repetições integrais ou parciais do processo de KDD na busca de resultados satisfatórios por meio de refinamentos sucessivos."*

Ainda sobre a definição dada em (FAYYAD et al., 1996), temos as expressões: padrões compreensíveis, válidos, novos e potencialmente úteis, que se referem respectivamente à possibilidade de interpretação, à confiabilidade e à apresentação de novos conhecimentos antes não visíveis e que possam ser efetivamente aplicáveis (GOLDSCHMIDT; PASSOS, 2005, p.4).

As técnicas para mineração de dados podem ser subdivididas em técnicas tradicionais, técnicas específicas e técnicas híbridas. As técnicas tradicionais são citadas como tecnologias que independem do contexto da mineração de dados. São consideradas tecnologias tradicionais a utilização de redes neurais, lógica nebulosa, algoritmos genéticos, estatísticas, entre outras. As técnicas específicas são citadas como técnicas desenvolvidas exclusivamente para aplicações em tarefas de KDD. Como

exemplo, é citado o algoritmo Apriori, desenvolvido especificamente para descoberta de regras de associação .Por fim, as técnicas híbridas são definidas como a junção de duas ou mais técnicas tradicionais e específicas (GOLDSCHMIDT;PASSOS,2005, p.17).

Este capítulo discute os principais conceitos relacionados às tarefas de KDD utilizadas na estratégia proposta e detalha alguns dos recursos utilizados em cada uma das etapas operacionais, incluindo os algoritmos específicos de mineração de dados aplicados à nossa estratégia.

3.1 ETAPAS OPERACIONAIS DE KDD

A descoberta de conhecimento em bases de dados é composta por três etapas operacionais: pré-processamento, processamento ou mineração de dados e pós-processamento, conforme ilustra a fig. 3.1.

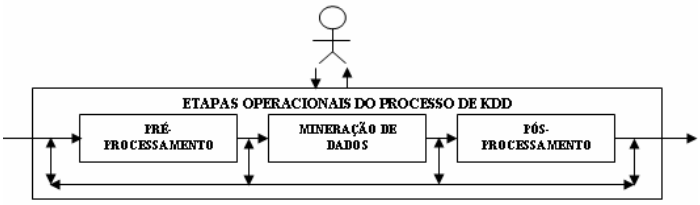


FIG. 3.1 Etapas do processo de KDD

A etapa de pré-processamento é responsável pela captação, organização e tratamento dos dados. A etapa de processamento ou mineração de dados é responsável pela busca efetiva de conhecimento através da aplicação de algoritmos de mineração. A etapa pós-processamento é responsável pela avaliação e pela aplicação do conhecimento obtido (GOLDSCHMIDT; PASSOS, 2005, p.3).

3.1.1 PRÉ-PROCESSAMENTO

Em GOLDSCHMIDT e PASSOS (2005, p.11), a etapa de pré-processamento é

descrita da seguinte forma.

> *"A etapa de pré-processamento compreende todas as funções relacionadas à captação, à organização e ao tratamento dos dados. Esta etapa tem como objetivo a preparação dos dados para os algoritmos da etapa de Mineração de Dados."*

Esta etapa faz-se necessária, porque nem sempre os dados a serem submetidos à etapa de mineração de dados estão prontos para serem utilizados. Para o tratamento dos dados são sugeridas as seguintes funções: seleção de dados, redução de valores, limpeza, codificação de valores, enriquecimento e normalização dos dados.

SELEÇÃO DE DADOS

A função de seleção de dados tem como objetivo, identificar quais informações, dentre as bases de dados existentes, devem ser efetivamente consideradas durante o processo de *KDD*, podendo ser utilizada para cumprir esta função a redução de dados horizontal que consiste em eliminar registros que não atendam a determinada condição ou a redução de dados vertical que consiste em eliminar atributos que contêm informações irrelevantes para descoberta de conhecimento (GOLDSCHMIDT; PASSOS, 2005, p.26).

GOLDSCHMIDT e PASSOS (2005, p.29) enumeram um conjunto de motivações para aplicação de seleção de dados, em particular para redução de dados vertical.

1. Um conjunto de atributos bem selecionado pode conduzir a modelos de conhecimento mais precisos e mais concisos.
2. O tempo de processamento do algoritmo de mineração de dados pode ser inferior se comparado ao processamento realizado sobre todo o conjunto de atributos;

3. A redução de tamanho aplicada a conjuntos de dados através da eliminação de um atributo é mais significativa do que a exclusão de um registro.

A primeira motivação está relacionada à relevância ou não do atributo para o modelo de conhecimento que se deseja obter. O atributo pode conter valores com baixa ou nenhuma variabilidade, pode conter informações que são utilizadas para identificação do registro dentro da base de dados como um seqüencial ou pode conter informações realmente irrelevantes. Estes atributos podem distorcer os resultados ou até mesmo não afetar de forma significativa o resultado, estando presente ou não.

A segunda motivação está relacionada ao tempo de processamento. Quanto menor a quantidade de dados a serem processados, menor o tempo de processamento desses dados pelos algoritmos de mineração de dados.

A terceira motivação está na comparação entre os benefícios em tempo de processamento da redução vertical em relação à redução horizontal. A redução vertical sempre afetará todos os registros da base de dados, enquanto a redução horizontal afeta apenas os registros que atendem a determinada condição. Portanto, a redução vertical terá uma economia de tempo de processamento mais significativa do que a redução horizontal.

REDUÇÃO DE VALORES

A função de redução de valores tem como objetivo diminuir o número de valores distintos em determinados atributos, proporcionando um melhor desempenho na execução de algoritmos de mineração de dados (GOLDSCHMIDT;PASSOS, 2005, p.33)

GOLDSCHMIDT e PASSOS (2005, p.33) citam dois métodos de redução de valores, sendo um aplicado a valores nominais e outro aplicado a valores contínuos ou discretos. Esses métodos são denominados redução de valores nominais e redução de valores contínuos ou discretos respectivamente.

No caso da redução de valores nominais, há dois enfoques:

- Identificação de hierarquia entre atributos, onde se mapeia conjuntos de valores relacionados entre si e traça-se uma hierarquia entre estes valores. A partir de um determinado ponto na hierarquia traçada, pode-se eliminar registros que estejam acima ou abaixo deste ponto.

- Identificação de hierarquia entre valores, onde se generaliza os valores dos atributos para grupos de valores distintos que têm o mesmo significado.

No caso da redução de valores contínuos ou discretos, os valores são substituídos por um único valor, em um grupo de valores em comum, através de métodos tais como clusterização e arredondamento de valores.

LIMPEZA

A função de limpeza de dados tem como objetivo tratar os dados a partir dos quais se deseja extrair informações. Dados incompletos, ruidosos ou inconsistentes são alguns dos principais problemas a serem considerados durante o processo de limpeza. São apresentadas três funções de limpeza de dados que são usualmente utilizadas: limpeza de informações ausentes, limpeza de inconsistências e limpeza de valores não pertencentes ao domínio (GOLDSCHMIDT; PASSOS, 2005, p.37).

CODIFICAÇÃO DE VALORES

A função de codificação de valores consiste em adequar as informações a serem processadas, de acordo com o tipo de informação esperada pelo algoritmo de mineração ao qual o conjunto de dados será submetido (GOLDSCHMIDT; PASSOS, 2005, p.40). Esta adequação é responsável por mapear dados nominais em valores numéricos e dados numéricos em valores nominais, dependendo das restrições impostas pelo algoritmo a ser utilizado.

ENRIQUECIMENTO

A função de enriquecimento consiste em agregar mais informações ao conjunto de dados existentes. O enriquecimento é realizado por meio da criação de novos atributos não existentes na base de dados original. Através de informações obtidas na própria base de dados ou em bases externas, o conjunto de dados pode ser estendido (GOLDSCHMIDT; PASSOS, 2005, p.46).

NORMALIZAÇÃO DOS DADOS

GOLDSCHMIDT e PASSOS (2005, p.44) referem-se à normalização dos dados, afirmando que:

> *"A operação de normalização de dados consiste em ajustar a escala dos valores de cada atributo, de forma que os valores fiquem em pequenos intervalos, tais como de -1 a 1, ou de 0 a 1. Tal ajuste faz-se*

necessário para evitar que alguns atributos, por apresentarem uma escala de valores maiores que outros, influenciem de forma tendenciosa em determinados métodos de mineração de dados."

Em GOLDSCHMIDT e PASSOS (2005, p.45) são citados alguns métodos de normalização, tais como: Normalização Linear, Normalização por Desvio Padrão, Normalização pela Soma dos Elementos, Normalização pelo Valor Máximo dos Elementos e Normalização por Escala Decimal.

Escolhemos o método de normalização pelo valor máximo dos elementos para exemplificação. A normalização pelo valor máximo dos elementos, consiste em dividir cada valor do atributo que esteja sendo normalizado pelo maior valor dentre os valores de tal atributo (GOLDSCHMIDT; PASSOS, 2005, p.47).

A tabela 3.1 exemplifica os resultados da normalização obtidos para o atributo `tempoExperiencia`, cujo valor máximo, dentre os valores existentes, é igual a nove. A coluna `tempoExperiencia` indica o valor original contido neste atributo e a coluna `tempoExperienciaNormalizado` indica o valor normalizado a ser utilizado no processamento do algoritmo.

TAB. 3.1 - Normalização pelo valor máximo

	TempoExperiência	TempoExperiênciaNormalizado
Registro1	9	1.00
Registro2	2	0.22
Registro3	5	0.55
Registro4	0	0.00

Dentre as funções de pré-processamento citadas, utilizamos neste trabalho: a função de seleção de dados, através da redução de dados vertical, a redução de valores para valores nominais com enfoque na hierarquia entre valores, e a normalização de dados. No Capítulo 4, veremos como estas funções foram utilizadas.

3.1.2 PROCESSAMENTO

A etapa de processamento, também denominada como mineração de dados, é descrita como a etapa responsável pela busca efetiva por conhecimentos úteis no contexto da aplicação de KDD. Trata-se da etapa principal deste processo e envolve a aplicação de algoritmos sobre os dados em busca de conhecimentos implícitos e úteis (GOLDSCHMIDT; PASSOS, 2005, p.12).

Dentre as principais tarefas de KDD citamos neste trabalho: Tarefa de Associação, Tarefa de Clusterização e Tarefa de Sumarização.

3.1.2.1 TAREFA DE ASSOCIAÇÃO

"A tarefa de descoberta de associação consiste em encontrar conjuntos de itens que ocorram simultaneamente e de forma freqüente em um banco de dados" (GOLDSCHMIDT; PASSOS, 2005, p.59).

Alguns conceitos relacionados à tarefa de associação são definidos em GOLDSCHMIDT e PASSOS (2005, p.61) e são apresentados a seguir.

Uma regra de associação é uma implicação da forma $\mathbf{X} \rightarrow \mathbf{Y}$, onde **X** e **Y** são conjuntos de itens tais que $\mathbf{X} \cap \mathbf{Y} = \varnothing$**.** Os itens podem ser interpretados como objetos. A ocorrência dos objetos está associada a *transações*. Por exemplo: em um supermercado, cada compra é representada por uma nota fiscal, cada nota fiscal representa uma transação e os objetos que fazem parte da compra são os itens da transação. Podemos descobrir que conjuntos de itens são freqüentes e que objetos estão relacionados entre si. Desta forma, pode-se perceber objetos cuja compra implica a compra de outros objetos. Para tal, utiliza-se os conceitos de *suporte* e *confiança*. O *suporte* tem por objetivo identificar que associações surgem em uma quantidade expressiva a ponto de ser destacada das demais, enquanto que a medida de confiança procura expressar a qualidade de uma regra, indicando o quanto a ocorrência do antecedente da regra pode assegurar a ocorrência do conseqüente desta regra. A união de conjuntos de objetos $\mathbf{X} \cup \mathbf{Y}$ é considerada freqüente se o número de vezes em que ocorrer em relação ao total de transações for superior a uma freqüência mínima

estabelecida, chamada *suporte mínimo*. Para formulação da regra, uma associação somente pode ser considerada válida se o número de vezes em que **X** ∪ **Y** ocorrer em relação ao número de ocorrências de **X** for superior a um valor, chamado *confiança mínima*. Este valor assegura a qualidade da regra, indicando o quanto a ocorrência do antecedente da regra **X** assegura a ocorrência do conseqüente da regra **Y**. Para cálculo do valor da confiança deve ser utilizada a expressão confiança (**X**, **Y**) = (suporte (**X** ∪ **Y**) / suporte(**X**)), e depois compará-lo com a confiança mínima estabelecida previamente. Os valores de suporte mínimo e confiança mínima devem ser estabelecidos pelo usuário ou especialista no problema.

Em REIS (2005, p.44) foram relacionados alguns algoritmos para descoberta de regras de associação, estes algoritmos representam a evolução do algoritmo APRIORI (AGRAWAL; SRIKANT, 1994), que por sua vez foi criado como uma extensão do algoritmo AIS. A fig. 3.2 ilustra a seqüência de algoritmos. Na figura encontram-se resumidas as principais características de cada algoritmo relacionadas à velocidade de processamento ganha com cada um deles.

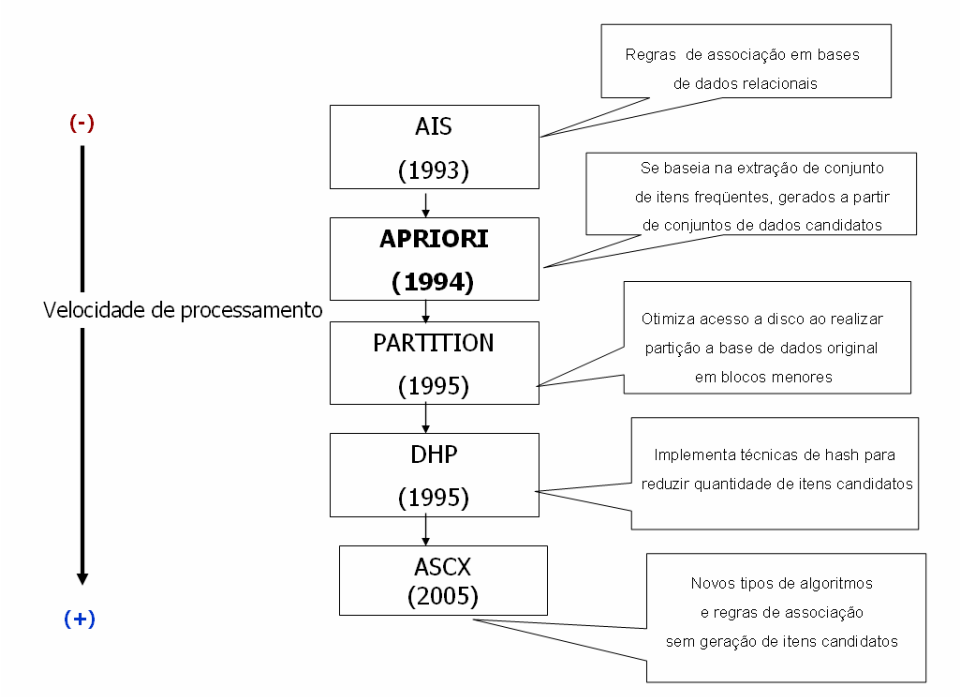


FIG. 3.2 Evolução dos algoritmos de associação

ALGORITMO APRIORI

"O algoritmo APRIORI se baseia na extração de conjunto de itens freqüentes

gerados a partir de conjuntos candidatos de itens do conjunto de dados da aplicação“ (REIS, p.44).

O algoritmo APRIORI constitui-se basicamente de duas grandes etapas que devem ser alcançadas. Primeiramente, devem ser encontrados os conjuntos de itens freqüentes que atendam ao suporte mínimo estabelecido, conforme ilustra o exemplo da fig. 3.3. Em seguida, a partir destes conjuntos geram-se as regras possíveis, filtrando aquelas que atendam à confiança mínima estabelecida conforme ilustra a fig. 3.4.

De acordo com o suporte mínimo de 60% e a confiança mínima de 80%, são descritos os seguintes passos a partir das transações existentes nas fig. 3.3 e 3.4:

- É verificada a quantidade de transações existentes. No exemplo, são quatro as transações. Deve ser calculada para todos os objetos a porcentagem de transações nas quais eles participam;

- Aplicando o filtro associado ao suporte mínimo de 60% pré-estabelecido, somente os itens `b, c` e `e` atendem a este suporte e, portanto, permanecem no processamento. Os demais são descartados.

- O processo é reinicializado, agora tentando-se formar pares de objetos que atendam ao suporte mínimo estabelecido. Os itens `b, c` e `e` são combinados dois a dois e é verificado quais atendem a 60% das transações. Neste exemplo, os itens `(b,c)` e `(b,e)` que possuem suporte de 75% atendem ao suporte mínimo estabelecido. Os itens `(c,e)` com suporte de 50% são descartados por não atenderem ao suporte mínimo estabelecido.

- O processo é interrompido, quando não se consegue formar novas combinações que atendam ao suporte mínimo estabelecido. No exemplo, a combinação `(b, c, e)` não atende. Neste momento, devemos partir para geração das regras, onde os conjuntos que atendem ao suporte mínimo estabelecido serão submetidos a verificações de confiança para formação das regras.

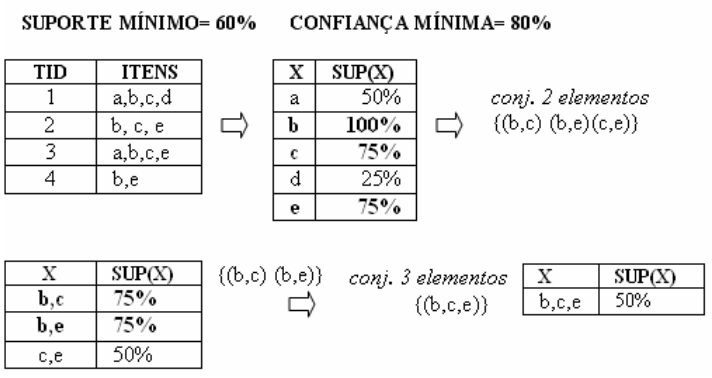


FIG. 3.3 Verificação de suporte

- Para verificação das regras formadas, os objetos `{(b,c);(b,e)}` devem ter os papeis de antecedente e conseqüente trocados entre si para verificação sobre quais implicações são verdadeiras, ou seja, atendem ao critério de confiança mínima.

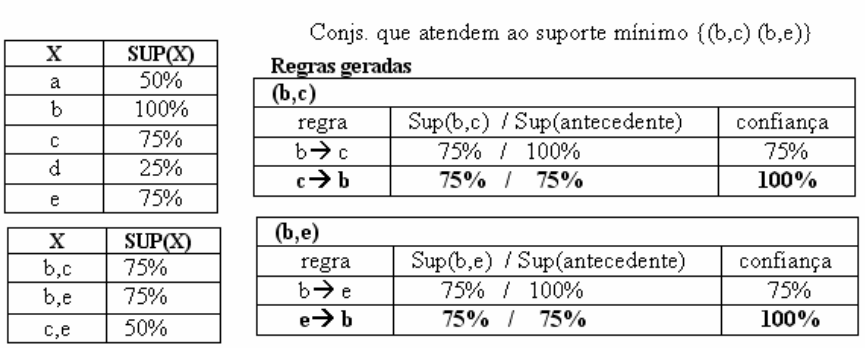


FIG. 3.4 Verificação de confiança – formação de regras

- Após verificações das combinações dos itens `b → c`; `c→b`; `b → e`; `e→b` conforme fig. 3.4, verifica-se como regras válidas, as implicações `c→b`; `e→b`, pois atendem à confiança mínima estabelecida de 80% .

3.1.2.2 TAREFA DE CLUSTERIZAÇÃO

A definição encontrada em GOLDSCHMIDT e PASSOS (2005, p.73) diz que:

> *"A tarefa de clusterização, também chamada de agrupamento, é utilizada para separar os registros de uma base de dados em subconjuntos ou clusters, de tal forma que os elementos de um cluster compartilhem de propriedades comuns que os distingam de elementos em outros clusters. O objetivo desta tarefa é maximizar a similaridade intracluster e minimizar a similaridade intercluster."*

A clusterização tem como objetivo identificar grupos de dados homogêneos entre si. Os objetos pertencentes a um mesmo cluster sempre terão características em comum, mas isto não descarta a possibilidade de termos objetos que sejam mais compatíveis que outros. Esta tarefa depende de uma série de fatores que influenciam diretamente o resultado.

A tarefa de clusterização envolve a organização de um conjunto de padrões de acordo com alguma medida de similaridade. Diferentemente da classificação que tem rótulos pré-definidos, a clusterização precisa automaticamente identificar os rótulos. Em geral, rótulos são definidos após a análise das características dos elementos pertencentes ao grupo (GOLDSCHMIDT; PASSOS, 2005, p.74). A identificação destes rótulos pode ser realizada através de outra técnica de mineração de dados chamada sumarização.

Para se chegar ao objetivo final, que são os *clusters* formados, de uma forma geral e independente do algoritmo a ser utilizado, segue-se a seqüência de passos exposta por GOLDSCHMIDT e PASSOS (2005, p.74): supondo-se a existência de **n** pontos de dados $\mathbf{x_1, x_2,..., x_n}$, de maneira que cada ponto pertença a um espaço **p** dimensional $\mathbf{R^p}$. A tarefa de clusterização desses pontos de dados, separando-os em **k** *clusters*, consiste em encontrar **k** pontos em $\mathbf{m_j}$ **e** em $\mathbf{R^p}$, de tal forma que a expressão contida na fig. 3.5 seja minimizada, onde $\mathbf{d^2\ (x_i, m_j)}$ denota uma distância entre $\mathbf{x_i}$ e $\mathbf{m_j}$. Os pontos $\mathbf{m_j}$ são denominados centróides ou médias dos *clusters*.

$$\frac{\sum_i \min_j d^2(x_i, m_j)}{N}$$

FIG. 3.5 Expressão de minimização da tarefa de clusterização

Em CARLANTÔNIO (2001), são citadas algumas métricas usualmente utilizadas para cálculo da distância entre dois pontos. São elas: Distância Euclidiana,

Distância de Manhattan e Distância de Minkowski, entre outras

A tarefa de clusterização é executada sobre estruturas de dados capazes de armazenar os objetos a serem processados. As estruturas comumente utilizadas são: matriz de dados e matriz de similaridade. Em uma matriz de dados, as linhas representam cada um dos objetos a serem clusterizados e as colunas, os atributos ou características de cada objeto (GOLDSCHMIDT; PASSOS, 2005, p.75).

Considerando **n** objetos cada qual com **p** atributos, obtém-se uma matriz de dados **n x p**, conforme ilustra a fig. 3.6.

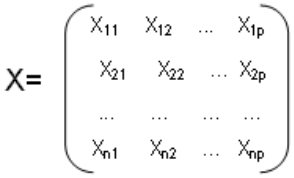


FIG. 3.6 Matriz de dados

A matriz de similaridade é construída a partir da matriz de dados. Cada elemento da matriz representa a distância entre pares de objetos. Quanto mais próximo de zero for esta distância, mais similares serão os objetos (HAN; KEMBER, 2001),(GOLDSCHMIDT; PASSOS, 2005, p.75).

As figs. 3.7a e 3.7b exemplificam a clusterização de objetos a partir da análise dos atributos despesa e renda, de acordo com os valores exibidos na tabela 3.2. Foram também definidas as variáveis quantidades de *clusters* iguais a dois (2) e os centróides iniciais (10,10) e (40,20).

TAB. 3.2 – clusterização de objetos

	atributos	
	renda	**despesa**
Objeto1	10	10
Objeto2	10	30
Objeto3	20	20
Objeto4	40	20
Objeto5	40	30
Objeto6	50	10
Objeto7	50	20

A partir da definição da quantidade de *clusters* e da identificação dos

centróides, a associação dos demais objetos é dada ao *cluster* em relação ao qual apresentam menor distância. Com isto, no exemplo, obtivemos o seguinte resultado: o primeiro grupo formado pelos objetos 1,2 e 3 e o segundo grupo formado pelos demais objetos 4,5,6,7, conforme ilustra a fig. 3.7b.

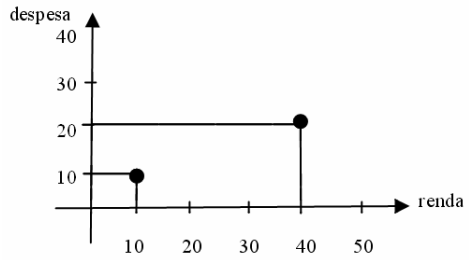


FIG. 3.7a Definição dos centróides

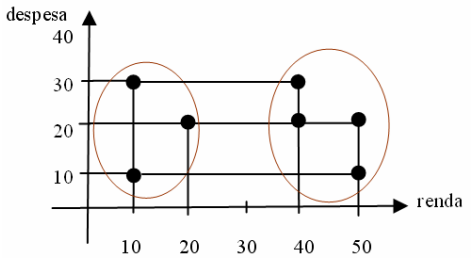


FIG.3.7b Associação dos objetos aos *clusters*

Segundo (HAN e KEMBER, 2001), os métodos de clusterização de dados estão divididos em cinco categorias: métodos de particionamento, métodos hierárquicos, métodos baseados em densidade, métodos baseados em grades e métodos baseados em modelos.

Segundo (GOLDSCHMIDT; PASSOS, 2005, p.77), os métodos de clusterização mais conhecidos e utilizados são os métodos de particionamento e os métodos hierárquicos. Enquanto nos métodos de particionamento é necessário definir o número de *clusters*, nos métodos hierárquicos este parâmetro não precisa ser definido. Nos métodos hierárquicos a determinação de números de *clusters* tem duas abordagens: aglomerativa (*bottom-up*) e divisiva (top-down). Enquanto a primeira abordagem inicia com cada objeto pertencendo a um *cluster,* ou seja, se há *n* objetos, serão *n clusters* iniciais, e a cada iteração ocorre a junção dos *clusters* dois a dois. A segunda abordagem inicia com todos os objetos pertencendo a um único *cluster* e a cada iteração estes *clusters* vão se subdividindo. Em ambos os casos, o processo é interrompido quando alguma condição de parada é alcançada. Normalmente dois critérios podem ser utilizados como condição de parada: quantidade de iterações ou alguma condição que indica tolerância a erros.

ALGORITMO *K-MEANS*

O algoritmo *K-Means* é um método de particionamento. Seu funcionamento encontra-se explicado por GOLDSCHMIDT e PASSOS (2005, p.102), da seguinte forma.

> *"O algoritmo K-means é um método popular da tarefa de clusterização. Tomam-se, randomicamente, k pontos de dados (dados numéricos) como sendo os centróides (elementos centrais) dos clusters. Em seguida, cada ponto (ou registro da base de dados) é atribuído ao cluster cuja distância deste ponto em relação ao centróide de cada cluster é menor dentre todas as distâncias calculadas. Um novo centróide para cada cluster é computado pela média dos pontos do cluster, (...). O processo termina quando os centróides do cluster param de se modificar, ou após um número limitado de iterações que tenha sido especificado pelo usuário".*

A expressão "*os centróides param de se modificar*" significa que durante um número finito de iterações o objeto que representa o centróide do *cluster*, continua o mesmo, indicando a ausência de deslocamentos de registros de dados entre *clusters*. A fig. 3.8 representa o conjunto de passos utilizados na execução deste algoritmo.

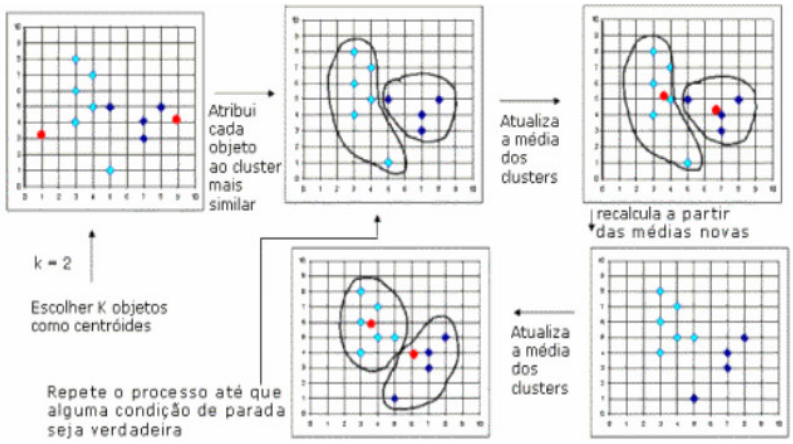


FIG. 3.8 Conjunto de passos executados pelo algoritmo *k-Means*

O algoritmo *k-Means* é um algoritmo determinístico (desde que o conjunto de centróides iniciais seja o mesmo). Em seu resultado final é apresentado a que *cluster*

pertence cada objeto e cada objeto pode pertencer somente a um único *cluster*. O objeto pertence ao *cluster* em relação ao qual o objeto apresenta menor distância. Convém ressaltar, no entanto, que ocorrem situações em que o objeto é quase tão próximo a um centróide de um *cluster* do que a outro, mas pela característica deste algoritmo, o objeto é associado ao *cluster* que apresenta menor distância não deixando vestígios de qualquer relação de compatibilidade ou pertinência aos demais *clusters.*

Segundo (HUANG, 1998, p.288) existem variações do algoritmo *k-Means* e cita o algoritmo *ISODATA* que se difere do *k-Means* por não necessitar que seja informado o número de *clusters* a serem formados, pois inclui um procedimento para pesquisar pelo melhor número de *clusters*. O espaço de busca aumenta, aumentando o custo deste procedimento que apresenta problemas de desempenho em grandes bases de dados; o segundo algoritmo citado é o *FUZZY K-MEANS*.

ALGORITMO *FUZZY K-MEANS*

O Algoritmo *Fuzzy K-Means* teve origem a partir do algoritmo *K-Means*, que, conforme comentado anteriormente é um algoritmo clássico de clusterização (DOBSA, 2003).

A clusterização a partir do algoritmo *Fuzzy K-Means* se difere dos métodos usuais de clusterização porque permite que um objeto pertença a mais de um *cluster* com diferentes graus de pertinência, desde que a soma dos graus de pertinência do objeto aos clusters seja igual a 1 (ROSA et al., 2001), conforme especifica a fig. 3.9. Além disto, o grau de pertinência de um determinado objeto ao *cluster* deve ser um valor entre 0 e 1, conforme especifica a fig. 3.10.

$$\sum_{j=1}^{k} \mu_{ij} = 1$$

FIG. 3.9 Soma dos graus de pertinência de um objeto **(i)** aos *clusters* **(j)** deve ser igual a 1

$$\mu_{ij} \in [0,1]$$

FIG. 3.10 Grau de pertinência de um objeto **(i)** ao *cluster* **(j)** deve ser um valor entre 0 e 1.

Na função objetivo, os valores que minimizam esta função, assim como o algoritmo propriamente dito, são descritos segundo definição de ROSA et al, (2001).

A função objetivo expressa na fig. 3.11 é utilizada para obtenção dos *clusters*, onde **m** é o coeficiente *fuzzy* responsável pelo grau de fuzzificação dos elementos de $\mathbf{x_i}$ e $\mathbf{v_{k.}}$, e o centróide do k-ésimo *cluster*.

$$J(\mu_{ij}, v_k) = \sum_{i=1}^{c} \sum_{j=1}^{n} \mu_{ij}^{m} \left\| x_j - v_i \right\|^2$$

FIG.3.11 Função objetivo

O resultado da clusterização *fuzzy* pode ser expresso através da matriz **U= [μ_{ij}],** para **i = 1, ..., c** e **j=1, ..., n.**

Os valores que minimizam a função objetivo são obtidos a partir das equações expressas na fig. 3.12a e 3.12b.

$$\mu^{ij} = \frac{\left(1 \Big/ \left\| x^j - v^i \right\| \right)^{\frac{1}{m-1}}}{\sum_{k=}^{c} \left(1 \Big/ \left\| x^j - v^k \right\| \right)^{\frac{1}{m-1}}}$$

FIG.3.12a Cálculo dos graus de pertinência

$$v^i = \frac{1}{\sum_{j=1}^{n} \left(\mu^{ij} \right)^m x^j}$$

FIG.3.12b Cálculo dos centróides

Os dados necessários para execução dos cálculos presentes nas figs. 3.11, 3.12a e 3.12b estão listados abaixo :

- Conjunto de **(n)** dados a serem clusterizados associados a um vetor de atributos **x = [x_1, x_2, x_3,..., x_n]**. Cada atributo representa as características de cada registro ou objeto.Os valores expressos por estes atributos são utilizados na execução do algoritmo.
- Número de *clusters* **(k)**. Quantidade de grupos a serem formados.

- Taxa de nebulosidade **(m)** , sendo m $\in$ (1, ∞).
- Os critérios de parada são normalmente determinados por um número máximo de iterações e por uma taxa de erro mínimo (ε) a serem determinadas.

O algoritmo é inicializado através de um conjunto de valores iniciais para a matriz **U**. O processamento do algoritmo segue a seqüência descrita abaixo.

1) Inicializa-se :
- **c** (número de *clusters*)
- **ε** (critério de parada)
- **m** (coeficiente nebuloso)
- $\mathbf{U^0}$ (matriz inicial com os graus de pertinência);

2) Calcula-se os centróides dos *clusters* (conforme cálculo expresso na fig. 3.12b)

3) Atualiza-se a matriz $\mathbf{U^{k+1}}$ com os graus de pertinências calculados (conforme cálculo expresso na fig. 3.12a)

4) A cada execução do algoritmo, verificar se atende aos critérios de parada (iteração e taxa de erro).

Calcular $\nabla = \| \mathbf{U^{k+1}} - \mathbf{U^k} \|$ (representa o cálculo da função de erro)

Se $\nabla < \varepsilon \Rightarrow$ fim do algoritmo

Senão

K = K+1

Voltar ao passo 2

3.1.2.3 TAREFA DE SUMARIZAÇÃO

Segundo GOLDSCHMIDT e PASSOS (2005, p.73):

> *"A tarefa de sumarização, também denominada descrição de conceitos, consiste em identificar e apresentar, de forma concisa e compreensível, as principais características dos dados contidos em um conjunto de dados"*.

A descrição de conceitos pode ser interpretada como uma generalização dos dados a partir das características mais relevantes dentro dos registros analisados. A tarefa de sumarização é muito utilizada em conjunto com outras tarefas de mineração de dados, principalmente a tarefa de clusterização. Como já explicado anteriormente, a tarefa de clusterização é aplicada a dados em que não há nenhuma identificação prévia, produzindo como resultado grupos de objetos similares entre si. Sabemos os objetos em um *cluster* são similares, mas, em geral, não sabemos o que representa cada grupo. Nesse ponto, a tarefa de sumarização pode ser realizada, procurando descrever as principais características dos objetos do *cluster.* Este processo pode ser automatizado, utilizando-se de algoritmos próprios para sumarização. Em GOLDSCHMIDT e PASSOS (2005, p.73), é citado o algoritmo de sumarização C4.5.

Independente do algoritmo de mineração de dados utilizado nesta etapa, após a finalização do seu processamento há uma última etapa no processo de KDD denominada pós-processamento, responsável por apresentar estas informações de forma legível. Será nesta etapa que o conhecimento obtido será apresentado.

3.1.3 PÓS-PROCESSAMENTO

A etapa de Pós-Processamento envolve a visualização, a análise e a interpretação dos modelos de conhecimento gerados pela etapa de mineração de dados. Um modelo de conhecimento obtido pode ser simplificado removendo detalhes que não afetem a avaliação do modelo, no caso de grandes volumes de dados (GOLDSCHMIDT; PASSOS, 2005, p.55). O modelo de conhecimento pode ser transformado de forma a facilitar sua análise. Esta etapa é ainda responsável pela organização e apresentação dos resultados por meio de gráficos, planilhas, relatórios, árvores de decisão, geração de regras etc. (GOLDSCHMIDT; PASSOS, 2005, p.56).

O pós-processamento, na verdade, é uma etapa em que é decidido onde e como aplicar o conhecimento obtido. Dependendo do objetivo da aplicação, não haverá necessidade de se realizar quaisquer tipos de tratamento no modelo de conhecimento durante a etapa de pós-processamento.

No próximo capítulo, discutiremos o propósito do presente trabalho que envolve as etapas do processo de KDD, adaptadas para atender o problema em questão.

4 ESTRATÉGIA DE ARMAZENAMENTO E RECUPERAÇÃO APLICADA A DOCUMENTOS XML HETEROGÊNEOS

Como já citado anteriormente, este trabalho tem como objetivo encontrar mecanismos que auxiliem o armazenamento e recuperação mais eficiente de documentos XML heterogêneos em SGBD's XML Nativos, propondo uma estratégia para armazenamento e indexação destes documentos. A estratégia proposta se utiliza das informações obtidas como resultado do processo de KDD sobre a estrutura dos documentos XML e sobre os recursos existentes nos SGBD's XML Nativos, de modo a guiar o usuário no desenvolvimento do projeto físico.

Conforme exposto no capítulo sobre mineração de dados, o processo de KDD apresenta três etapas. Inspirada nesta organização, a estratégia proposta também se divide nestas três etapas conforme ilustra a fig. 4.1. Estas etapas envolvem desde a captação e tratamento de documentos XML até o armazenamento dos mesmos.

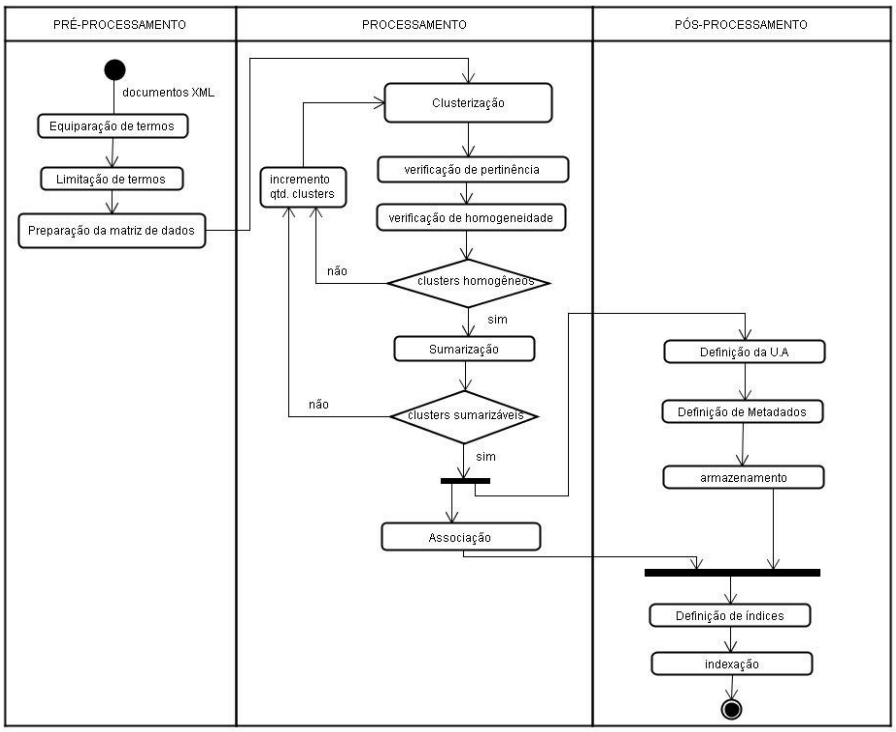


FIG. 4.1 Etapas do processo proposto

Apesar dos documentos XML poderem estar associados a padrões pré-definidos e a termos estruturas de validação que garantem que estes padrões sejam

seguidos, nossa estratégia parte do pressuposto que os documentos provêm de origens distintas, o que nos leva a possibilidade de ter documentos com estruturas heterogêneas.

Para alcançar o objetivo geral proposto neste trabalho, existem dois objetivos específicos que esperamos atingir por meio destas etapas que são: o armazenamento em separado de documentos similares e a criação de índices adequados para agilizar o futuro acesso a tais documentos.

Na etapa de Pré-processamento foram identificadas três sub-etapas que visam extrair informações sobre os documentos para a etapa central, denominada Processamento. As atividades destas sub-etapas se ocupam da identificação e tratamento dos termos para montagem de uma matriz de dados.

Na etapa de Processamento são aplicados algoritmos que possibilitam: a identificação de grupos de documentos similares através da sub-etapa de clusterização, a descrição, através da sub-etapa de sumarização, dos grupos formados, e, por fim, a sugestão de índices através da sub-etapa de associação.

A etapa Pós-processamento consiste em aplicar o conhecimento obtido através dos resultados da etapa anterior no armazenamento dos documentos e na definição dos índices. Nesta etapa também são realizadas atividades de definição da unidade de armazenamento orientadas ao SGBD XML Nativo a ser utilizado.

Nas subseções seguintes apresentaremos o detalhamento das etapas que compõem a estratégia proposta.

4.1 ETAPA PRÉ-PROCESSAMENTO

Nossa estratégia inicia-se a partir da análise da estrutura dos documentos XML. A estrutura destes documentos é composta por etiquetas, denominadas *tags*. Cada etiqueta pode ser vista como um termo que indica ou descreve a natureza da informação por ela "encapsulada". Um dos objetivos deste trabalho é verificar a similaridade entre os documentos que na estratégia proposta baseia-se na utilização destes termos pelos documentos. Por isso, deste ponto em diante a palavra termo será utilizada no lugar da palavra "etiqueta", isto é, os termos tratados pelas sub-etapas que compõem a etapa de pré-processamento são provenientes das "etiquetas" dos documentos XML.

A seguir descrevemos cada sub-etapa que compõe a etapa de pré-processamento.

4.1.1 EQUIPARAÇÃO DE TERMOS

Na sub-etapa de equiparação de termos, inicialmente é preciso identificar todos os termos distintos presentes em todos os documentos.

Como apropriadamente colocado por FLORESCU (2005):

> *"Atualmente, um problema comum no gerenciamento de informação é a falta de concordância no vocabulário e schemas. Metodologias para processamento de informação requerem que toda a comunidade envolvida na geração, processamento e consumo da mesma informação concordem com um dado schema e vocabulário, infelizmente, diferentes pessoas, organizações e comunidades tem utilizado caminhos distintos para modelar a mesma informação."*

Assim sendo, no sentido de minimizar este problema, esta sub-etapa tem como objetivo substituir termos equivalentes por um termo representante, reduzindo a quantidade de termos a tratar. Desta forma, evitamos diferenciar documentos similares

só por que utilizam termos diferentes ignorando que os mesmos têm igual significado. Isto pode ser resolvido utilizando um vocabulário controlado de termos denominado tesauro (UNESCO, 1973).

Como destaca CAMPOS et al.(2006, p.6) :

> *"O uso de tesauros tem se destacado como ponto de apoio para a organização e acesso multifacetado da informação, bem como para a recuperação de conceitos relacionados".*

Esta sub-etapa é de suma importância, já que estamos trabalhando com documentos heterogêneos e sem este tratamento teríamos dificuldade em identificar a similaridade entre os documentos, por não haver uma padronização no uso dos termos (etiquetas).

A fig. 4.2 ilustra o funcionamento da etapa de equiparação de termos. Dado um termo de entrada, denotado como termo original, através da utilização de um tesauro será possível identificar um termo relacionado que melhor representa o termo em análise.

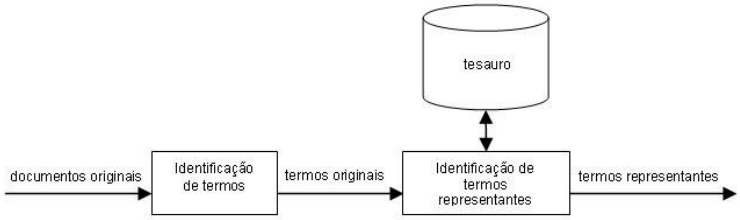


FIG. 4.2 Equiparação de termos

Um recurso simples que pode ser utilizado quando as origens dos documentos são previamente conhecidas, é a construção de uma tabela de equivalência de termos que armazenará os termos representantes e seus termos equivalentes. Sendo assim para cada documento, se o termo original for encontrado na tabela o substituiremos pelo termo representante, caso o termo não seja encontrado o próprio termo será utilizado. A tabela 4.1 fornece alguns exemplos de termos representantes e seus termos equivalentes.

TAB. 4.1 - Representação de termos representantes e termos equivalentes

Termos representantes	Termos equivalentes		
instituição	faculdade	universidade	
Cargo	profissão		
especialização	especialidade		
curriculum	curriculo	curriculo-vitae	cv
Sumário	sinopse	descrição	
categoria	assunto	tema	

4.1.2 LIMITAÇÃO DE TERMOS

A sub-etapa de limitação de termos consiste em restringir a quantidade de termos a serem analisados. Em KDD, a limitação de termos é realizada pela função de Seleção de dados, através da redução de dados vertical, que tem como objetivo eliminar atributos irrelevantes (GOLDSCHMIDT; PASSOS, 2005, p.26). A limitação de termos também pode ser realizada pelas técnicas conhecidas como *startword* e *stopword* (BRASIL, 2004). Enquanto a técnica de *startword* indica que termos devem ser considerados, a técnica *stopword* indica que termos devem ser ignorados.

As técnicas de *startword* e *stopword* estão exemplificadas conforme ilustram as figs. 4.3a e 4.3b.

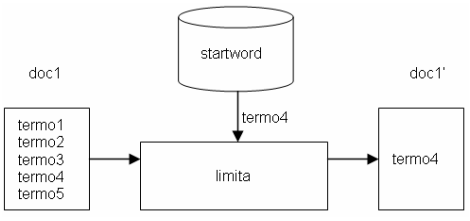


FIG. 4.3a Técnica *startword*

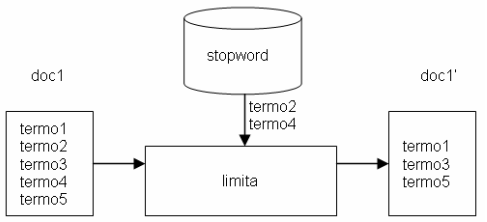


FIG. 4.3b Técnica *stopword*

De forma a facilitar a escolha dos termos para execução de uma destas técnicas, pode-se calcular a quantidade de vezes que os termos ocorrem nos documentos através de uma atividade denominada ranqueamento de termos, que irá levar em conta a freqüência com que os termos aparecem nos documentos. No ranqueamento não são consideradas as repetições de termos dentro de cada documento. A fig. 4.4 ilustra o ranqueamento de termos a partir da análise de vinte e sete documentos. Por exemplo,

suponha que o termo `article` ocorre em vinte e seis documentos dos vinte e sete documentos existentes. Neste caso, o termo `article` seria ranqueado na proporção 26/27, ou seja, o termo aparece em 96% dos documentos. O objetivo desta sub-etapa é mostrar ao usuário quais termos são mais freqüentes e quais termos são menos freqüentes. Dependendo da aplicação, os termos mais freqüentes podem ser utilizados como *startword* enquanto os termos menos freqüentes podem ser utilizados como *stopword.* No entanto, isto não pode ser uma regra já que haverá termos que serão mais freqüentes e deverão ser tratados como *stopword.* Por exemplo, considere um conjunto de documentos distintos com, digamos, três contextos: artigos, livros e revistas, e que estes contextos possuem termos diferenciados que caracterizam cada grupo mas também possuem um conjunto de termos em comum, tais como `título, autor, seção e parágrafo`. Estes termos podem vir a ser desconsiderados, pois estando presentes na estrutura destes três grupos de documentos, não servem para distinguir os grupos. Por outro lado, manter estes termos ajuda a ficar claro que um documento pode ser pertinente a mais de um grupo, ou ainda, pode ser importante para análises que levam em consideração a hierarquia em que os termos aparecem no documento. Um outro exemplo para utilização de *stopword* é a presença de termos que funcionam como identificadores, tais como `código, id`. A utilização destes termos pode influenciar de forma tendenciosa na indicação de similaridade entre documentos que são contextualmente diferentes.

article	26/27	96%
prolog	26/27	96%
title	26/27	96%
authors	24/27	89%
author	24/27	89%
name	24/27	89%
contact	24/27	89%
email	24/27	89%
phone	24/27	89%
dateline	24/27	89%
city	24/27	89%
country	24/27	89%
date	24/27	89%
genre	15/27	56%
keywords	17/27	63%

FIG. 4.4 Ranqueamento dos termos

Uma outra medida que pode ser aplicada nesta sub-etapa é adotar limites configuráveis para seleção automática destes termos sob aprovação do usuário.

Cabe ressaltar que o usuário deverá optar por uma destas técnicas na limitação dos termos. A fig. 4.5 representa as atividades envolvidas nesta sub-etapa.

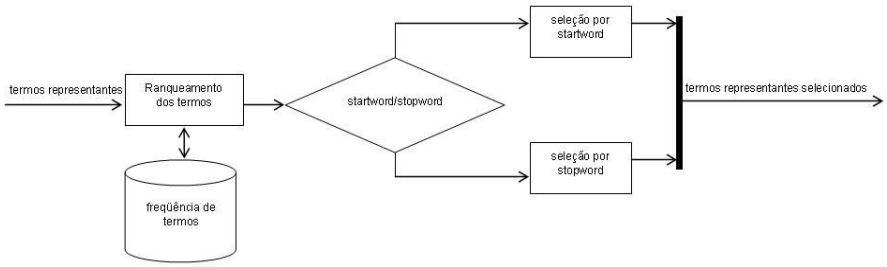


FIG. 4.5 Limitação por *startword/stopword*

4.1.3 PREPARAÇÃO DA MATRIZ DE DADOS

A penúltima sub-etapa da etapa de pré-processamento é a criação de uma matriz de dados apropriada para "clusterização", seguida da normalização dos dados que a compõem conforme ilustra a fig. 4.6.

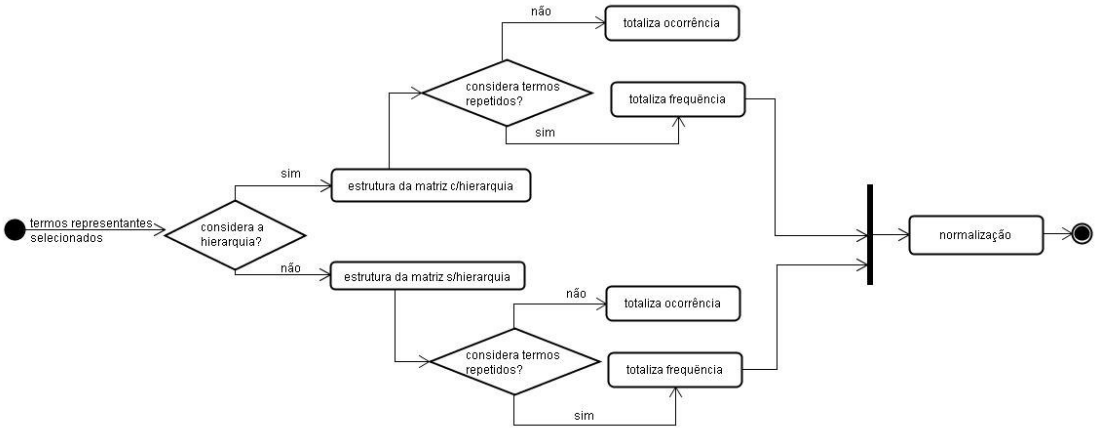


FIG. 4.6 Escolha da estrutura da matriz de dados seguida da normalização

> *"Uma matriz de dados é composta por linhas e colunas. As linhas representam cada um dos objetos a serem clusterizados e as colunas, os atributos ou características de cada objeto" (*GOLDSCHMIDT; PASSOS*, 2005, p.75).*

A matriz é composta por valores numéricos associados ao cruzamento de cada termo referenciando cada documento. Estes valores numéricos resultam da análise da presença dos termos na estrutura dos documentos podendo representar a sua:

- Ocorrência
- Freqüência

Para efeito de utilização no algoritmo de clusterização, apenas uma única matriz é utilizada.

Independente da abordagem adotada, inicialmente montamos um vetor de termos distintos, com os termos identificados nas etapas anteriores. A fig. 4.7 ilustra um vetor de termos.

$$Vetor = (termo_1, termo_2, termo_3, ..., termo_n)$$

FIG. 4.7 Vetor de termos

A seguir detalhamos cada tipo de representação e as características de cada uma destas abordagens.

OCORRÊNCIA DO TERMO

Para cada documento/termo lido deve ser atribuído o valor 0 ou 1. O valor *zero* indica que o termo não existe no documento analisado e o valor *um* indica que o termo existe no documento. A tabela 4.2 ilustra uma matriz de ocorrência de termos.

TAB. 4.2 - Matriz de ocorrência de termos

Documentos xml	**termo$_1$**	**termo$_2$**	**termo$_3$**	**...**	**termo$_n$**
doc$_1$.xml	1	0	0	...	1
doc$_2$.xml	1	1	0	...	0
doc$_3$.xml	0	0	1	...	0
...	...	...	...	...	...
doc$_N$.xml	0	1	1	...	1

FREQÜÊNCIA DO TERMO

Neste caso não é analisada apenas a ocorrência do termo, mas também sua freqüência. Para cada documento/termo lido é atribuído o valor *zero* caso o termo não faça parte do documento. Caso o termo faça parte do documento, é verificado quantas vezes o termo aparece no documento, ou seja, com que freqüência o termo aparece no

documento. A tabela 4.3 ilustra uma matriz de freqüência de termos.

TAB. 4.3 - Matriz de freqüência de termos

Documentos xml	**termo$_1$**	**termo$_2$**	**termo$_3$**	**...**	**termo$_n$**
doc$_1$.xml	3	0	0	...	1
doc$_2$.xml	2	1	0	...	0
doc$_3$.xml	0	0	1	...	0
...	...	...	...	...	...
doc$_N$.xml	0	3	7		1

Há uma terceira abordagem que pode ser considerada na análise da estrutura do documento XML. Nesta abordagem levamos em consideração a hierarquia do termo no documento e podemos calcular a sua freqüência ou ocorrência. Este tipo de análise pode ser realizada em documentos XML pois suas *tags* estão organizadas em uma estrutura aninhada. Esta abordagem permite uma maior precisão se tivermos como objetivo identificar documentos estruturalmente e hierarquicamente similares, pois os documentos, além de serem compostos pelas mesmas *tags*, têm que ter estas *tags* aninhadas em uma mesma estrutura hierárquica para serem considerados similares. A tabela 4.4 ilustra uma matriz hierárquica de termos baseada nas informações fornecidas pela fig. 4.8.

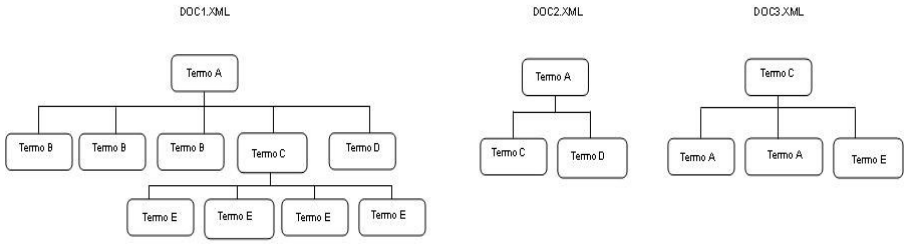


FIG. 4.8 Representação da hierarquia de termos em documentos XML

Nas etapas anteriores, além de ter sido identificado que termos estavam presentes nos documentos, também foi identificado para cada documento que termos possuíam sub-elementos, ou seja, sub-termos, e assim pode-se identificar que termos participam da função pai e que termos participam da função filho. O termo filho tem como termo pai o elemento imediatamente superior a ele, ou seja, o termo que se encontra um nível acima.

Antes da montagem da matriz, deve ser construído um vetor de termos com as combinações termopai →termofilho como ilustra a fig. 4.9.

$$Vetor = (\ termoPai_1termoFilho_{1,}\ termoPai_2termoFilho_{2,}\ termoPai_2termoFilho_{1,\ldots,}\ termoPai_ntermoFilho_n)$$

FIG. 4.9 vetor de termos hierárquico

Este elemento do vetor constituirá uma coluna da matriz de dados. Em seguida, pode-se adotar a abordagem de ocorrência ou freqüência para compor as linhas com os valores numéricos do relacionamento entre os documentos e cada um destes elementos. A tabela 4.4 mostra a exemplificação da montagem de uma matriz hierárquica de termos considerando a ocorrência.

TAB. 4.4 - Matriz hierárquica de termos – ocorrência

DOCUMENTOS XML	TERMO A → TERMO B	TERMO A → TERMO C	TERMO C→ TERMO E	TERMO A→ TERMO D	TERMO C→ TERMO A
doc_1.xml	1	1	1	1	0
doc_2.xml	0	1	0	1	0
doc_3.xml	0	0	1	0	1

A tabela 4.5 exemplifica a montagem de uma matriz hierárquica de termos considerando a freqüência, de acordo com os documentos e termos presentes na fig. 4.8.

TAB. 4.5 - Matriz hierárquica de termos – freqüência

DOCUMENTOS XML	TERMO A → TERMO B	TERMO A → TERMO C	TERMO C→ TERMO E	TERMO A→ TERMO D	TERMO C→ TERMO A
doc_1.xml	3	1	4	1	0
doc_2.xml	0	1	0	1	0
doc_3.xml	0	0	1	0	2

Por termos um conjunto de documentos heterogêneos a serem analisados, algumas considerações devem ser feitas em relação à escolha da matriz mais apropriada para o problema em questão. Existem termos que são comuns mesmo em documentos que são inerentemente diferentes. Um mesmo termo pode aparecer em diferentes contextos. Com isso, uma matriz de ocorrência pode não ser apropriada já que traria uma falsa similaridade entre documentos presentes em diferentes contextos. Já a adoção da matriz de freqüência reduziria este problema, já que contabiliza quantas vezes o termo aparece no documento e não simplesmente a verificação de sua

ocorrência, pois provavelmente documentos que pertençam a um mesmo contexto tendem a ter *tags* com a mesma denominação com freqüências numericamente mais próximas. A construção de uma matriz hierárquica deve ser aplicada quando se deseja agrupar documentos que são estruturalmente similares. Estamos trabalhando com documentos heterogêneos e conseqüentemente com estruturas diferentes, portanto, a aplicação direta de uma matriz hierárquica impossibilitaria identificar documentos similares se estes não possuíssem a mesma estrutura hierárquica. A matriz hierárquica pode ser aplicada quando o contexto já é conhecido e deseja-se identificar documentos estruturalmente e hierarquicamente semelhantes.

No nosso caso, desejamos utilizar documentos XML heterogêneos de fontes distintas, e com isto, consideramos que a matriz de freqüência é mais adequada, pois mesmo que existam documentos com termos comuns que não pertençam ao mesmo contexto, a matriz de freqüência irá possibilitar uma melhor precisão no momento de identificar conjuntos de documentos similares. A escolha da matriz de freqüência não desconsidera as demais, que poderiam ser utilizadas na identificação de sub-grupos dentro dos grupos principais identificados num primeiro processamento do algoritmo de clusterização.

Os valores constantes na matriz que será submetida à etapa de clusterização deverão ser normalizados.

Como definido por GOLDSCHMIDT e PASSOS (2005, p.44),

> *"A operação de normalização de dados consiste em ajustar a escala dos valores de cada atributo de forma que os valores fiquem em pequenos intervalos, tais como de -1 a 1, ou de 0 a 1. Tal ajuste faz-se necessário para evitar que alguns atributos, por apresentarem uma escala de valores maiores que outros, influenciem de forma tendenciosa em determinados métodos de mineração de dados."*

No caso de uma matriz de ocorrência, os valores já estão dentro do intervalo citado tornando desnecessária a normalização, mas no caso de uma matriz de freqüência estes valores deverão ser normalizados antes de seguirmos para a etapa de processamento.

Existem alguns métodos de normalização, tais como: Normalização Linear, Normalização por Desvio Padrão, Normalização pela soma dos Elementos, Normalização pelo Valor Máximo dos Elementos e Normalização por Escala Decimal (*GOLDSCHMIDT; PASSOS, 2005, p.44)*.

Neste trabalho, adotamos o método de Normalização pelo valor máximo dos elementos. Este método consiste em dividir cada valor do termo que esteja sendo normalizado pelo maior valor dentre os valores de tal termo. Ao final da normalização utilizada temos valores entre 0 e 1.

4.2 ETAPA DE PROCESSAMENTO

Como já mencionado anteriormente, a idéia nesta etapa é encontrar grupos de documentos XML para facilitar sua distribuição por diferentes unidades de armazenamento e ainda identificar possíveis candidatos à indexação. Assim, as sub-etapas clusterização, verificação de pertinência, verificação de homogeneidade, sumarização e associação estão voltadas a identificar tais grupos e sugerir índices sobre os mesmos. A fig. 4.10 ilustra as sub-etapas da etapa de processamento.

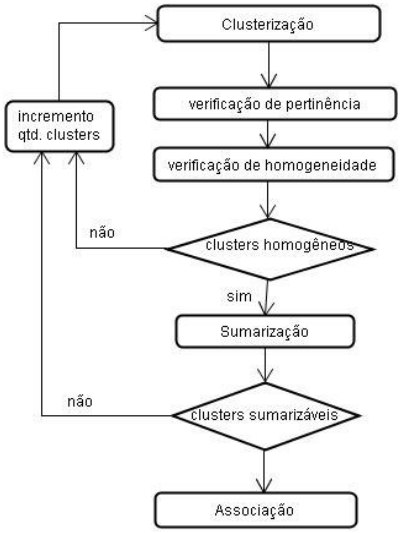


FIG. 4.10 Etapa processamento

A sub-etapa de clusterização é voltada para identificar grupos de documentos similares. Assim, cada grupo identificado sugere a utilização de uma unidade de

armazenamento contendo documentos similares. Em seguida, tais grupos devem ser validados de acordo com a expectativa do usuário quanto à pertinência e à homogeneidade dos *clusters* formados. Para reforçar a validação de cada grupo, a sub-etapa de sumarização irá descrever o conteúdo de cada *cluster* e, por fim, a sub-etapa de associação irá identificar termos freqüentes que poderão ser utilizados para a escolha de candidatos a índices.

Quanto mais similares forem os elementos de um *cluster*, melhor serão os resultados obtidos na definição deste cluster e na escolha do tipo de UA – Unidade de Armazenamento adequada para suportar estes documentos.

O resultado da etapa processamento é usado como subsídio para a etapa seguinte onde ocorre a definição das UA e a indexação dos termos.

A seguir explicamos cada uma das sub-etapas da etapa de processamento.

4.2.1 CLUSTERIZAÇÃO DOS DOCUMENTOS

A definição encontrada em GOLDSCHMIDT e PASSOS (2005, p.73), diz que:

> *"A tarefa de clusterização, também chamada de agrupamento, é utilizada para separar os registros de uma base de dados em subconjuntos ou clusters, de tal forma que os elementos de um cluster compartilhem de propriedades comuns que os distingam de elementos em outros clusters."*

No contexto deste trabalho, os registros da base de dados correspondem a documentos XML e a matriz de dados construída na etapa anterior é que contém as características a serem consideradas para identificar as semelhanças e diferenças entre os documentos a agrupar. Esta matriz é submetida ao algoritmo de clusterização. A abordagem adotada para clusterização de documentos heterogêneos se utiliza do algoritmo *fuzzy k-means* (GOLDSCHMIDT; PASSOS, 2005, p.116). O algoritmo *fuzzy k-means* é um algoritmo de clusterização que permite que um objeto pertença a mais de um grupo com diferentes graus de pertinência (ROSA, 2001, p.3). Utilizando-se desta abordagem, será possível identificar documentos que compartilhem características comuns com documentos de mais de um grupo.

4.2.2 VERIFICAÇÃO DE PERTINÊNCIA

A sub-etapa de verificação de pertinência tem por objetivo definir a que grupos pertencem cada documento.

Após o processo de clusterização ter sido concluído, os documentos apresentam graus de pertinência a todos os *clusters*. Por isto, neste momento, os grupos ainda não estão definidos de fato. Os graus de pertinência apresentados variam, o documento pode ser mais ou menos pertinente a um *cluster* do que a outro. Neste caso, é preciso definir se realmente o documento apresenta um grau de pertinência significativo com relação a cada *cluster*.

Sejam μ o grau de pertinência de um documento ***X*** a um *cluster* ***C*** e ℓ um limiar mínimo de pertinência a ser considerado que tenha sido previamente definido pelo usuário. Dizemos que o documento **X** pertence ao *cluster* **C** se o grau de pertinência μ **(X,C)** for maior ou igual ao limiar ℓ estabelecido. Conforme especifica a expressão da fig. 4.11.

$$\mu(X,C) \geq \ell \Rightarrow X \in C$$

FIG. 4.11 Pertinência do documento ao *cluster*

Assim, uma vez definido o limiar ℓ, de forma geral, ou para cada grupo, é possível definir os grupos de documentos.

Convém enfatizar que, após a sub-etapa de verificação de pertinência, um mesmo documento pode estar associado a mais de um *cluster.*

4.2.3 VERIFICAÇÃO DE HOMOGENEIDADE

Esta sub-etapa analisa se os documentos que compõem cada *cluster* formam um grupo homogêneo. Este tipo de análise deve ser realizada, pois nem sempre é conhecida a quantidade de contextos distintos, de onde provêm os documentos. Por isso, não é possível em um primeiro momento determinar a quantidade de *clusters*

adequada. Uma clusterização com quantidades de *clusters* inadequadas pode resultar em grupos onde é possível identificar subgrupos. Assim, mesmo que o usuário tenha uma idéia de quantos *clusters* devem ser formados, é necessário validar se o *cluster* é homogêneo. Isso pode ser feito pela análise dos graus de pertinência dos documentos que compõem o *cluster*.

Dado um cluster **C**, se o maior grau de pertinência subtraído do menor grau de pertinência dos documentos que compõem o *cluster* for menor ou igual a um limiar de homogeneidade **ℓ',** que foi previamente definido pelo usuário, então o cluster **C** é considerado homogêneo. Senão, o cluster **C** é considerado heterogêneo, conforme especifica a expressão da fig. 4.12.

$$Se \quad \mu_{max}(c) - \mu_{min}(c) \leq \ell' \quad Então \;\; C \text{ é homogêneo}$$
$$Senão \;\; C \text{ é heterogêneo}$$

$$\mu_{max}(c) = Max \{\mu(x_i, c)\} \qquad \text{Onde } 1 \leq i \leq k$$

$$\mu_{min}(c) = Min \{\mu(x_i, c)\}$$

FIG. 4.12 Avaliação da homogeneidade do cluster

Caso tenha sido verificada a homogeneidade do *cluster*, devemos seguir para o processo de sumarização, onde será descrito cada *cluster*.

No caso de ter sido verificada a heterogeneidade de pelo menos um *cluster*, volta-se à sub-etapa de clusterização. O processo de clusterização deve ser refeito incrementando a quantidade de *clusters* (de 1 em 1), procurando caracterizar melhor cada grupo formado. Cabe ressaltar que o processo deve ser refeito para todos os documentos e não apenas para os documentos pertencentes ao *cluster* no qual foi constatada a heterogeneidade.

4.2.4 SUMARIZAÇÃO

Segundo GOLDSCHMIDT e PASSOS (*2005, p.73):*

> *"A tarefa de sumarização, também denominada descrição de conceitos, consiste em identificar e apresentar, de forma concisa e compreensível, as principais características dos dados contidos em um conjunto de dados".*

Na sub-etapa de sumarização, para cada grupo formado serão descritas suas características de forma a identificar a que contexto pertencem os documentos de cada grupo. A impossibilidade de se ter uma descrição única dos documentos que compõem o *cluster* pode indicar que há heterogeneidade no grupo de documentos em análise. Assim, da mesma forma que na sub-etapa anterior, se pelo menos um *cluster* não puder ser caracterizado, volta-se à sub-etapa de clusterização.

A sumarização dos grupos servirá para identificar posteriormente as unidades de armazenamento que conterão estes documentos, possibilitando ao usuário direcionar as consultas para a unidade de armazenamento correta.

A sumarização pode ser realizada, escolhendo-se aleatoriamente documentos do *cluster* e avaliando as características destes documentos. Este processo pode ser automatizado utilizando-se de algoritmos próprios para sumarização. Em GOLDSCHMIDT e PASSOS (2005, p.73), é citada a ferramenta WizRule® que, baseada em Lógica Indutiva, gera regras que descrevem o conjunto de dados.

A conclusão das sub-etapas anteriores que compõem a etapa de processamento, permitem neste ponto que a etapa de pós-processamento seja iniciada, restando nesta etapa apenas a execução da sub-etapa de associação, que pode ser realizada em paralelo e que possibilitará a indicação de índices.

4.2.5 ASSOCIAÇÃO

Quando se trata de documentos XML homogêneos, o uso de uma estrutura de validação como o *XML Schema* ou *DTD* é bastante útil para identificar possíveis índices. Mas no contexto deste trabalho, os documentos XML heterogêneos não estão obrigatoriamente associados a estas estruturas e portanto precisa-se fazer uso de outras técnicas que possibilitem a indicação de índices. Optamos pela tarefa de associação para

nos ajudar nesta identificação.

Como explica GOLDSCHMIDT e PASSOS (2005, p.59):

> *"A Tarefa de Associação consiste em encontrar conjuntos de itens que ocorram simultaneamente e de forma freqüente em um banco de dados."*

No trabalho proposto, a tarefa de associação ficará responsável por identificar termos que ocorram freqüentemente juntos nos documentos de um grupo. Para tal serão utilizadas duas abordagens:

- Termos independentes (1 a 1 ou 2 a 2);
- Termos hierarquicamente dependentes (pai, filho).

A fig. 4.13 mostra que estas abordagens são realizadas por uma seqüência de atividades que podem ser executadas paralelamente.

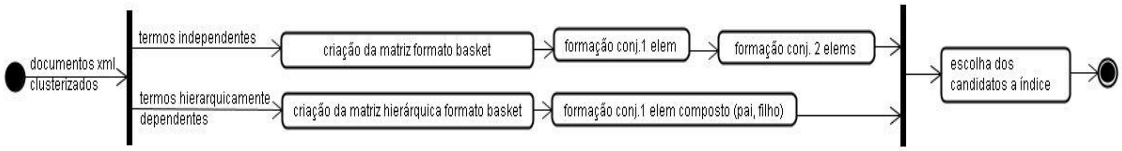


FIG. 4.13 Abordagens para definição de candidatos a índice

A sub-etapa de associação é iniciada após termos conhecimento dos documentos que compõem os *clusters.* No caso da abordagem de análise dos termos independentes é montada uma matriz no formato *basket* (GOLDSCHMIDT; PASSOS, 2005, p.60). Como já explicado no capítulo de mineração de dados, o formato *basket* é composto por transações e itens da transação. No contexto deste trabalho, cada documento será identificado como uma transação e cada termo será identificado como um item, conforme ilustra a tabela 4.6 como base das informações presentes na fig. 4.14. Após a matriz no formato *basket* ter sido criada, são realizadas duas atividades inspiradas no funcionamento do algoritmo clássico de mineração de regras de associação chamado Apriori (GOLDSCHMIDT; PASSOS, 2005, p.105): a formação de conjuntos de 1 elemento, responsável por identificar termos que ocorram

freqüentemente, e a formação de conjuntos de 2 elementos gerados pela combinação dos termos resultantes da atividade anterior (conjuntos de 1 elemento), dois a dois, que atendam a um suporte estabelecido pelo usuário.

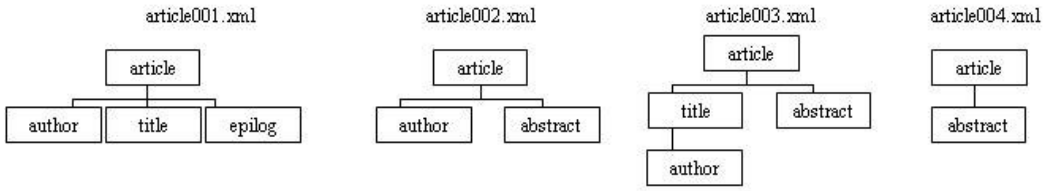


FIG. 4.14 Representação hierárquica documentos XML

TAB. 4.6 - Exemplificação formato *basket* não hierárquico

seq. transação	transação (documento xml)	item (termo)
1	article001.xml	title
1	article001.xml	article
1	article001.xml	author
1	article001.xml	epilog
2	article002.xml	article
2	article002.xml	author
2	article002.xml	abstract
3	article003.xml	title
3	article003.xml	article
3	article003.xml	author
3	article003.xml	abstract
4	article004.xml	article
4	article004.xml	abstract

No caso da combinação de termos, dois a dois, é importante considerar a definição dada em GOLDSCHMIDT e PASSOS (2005, p.61).

> *"Uma associação é considerada freqüente, se o número de vezes em que a união de conjuntos de itens (X U Y) ocorrer em relação ao número total de transações do banco de dados for superior a uma freqüência mínima (denominada suporte mínimo) que é estabelecida em cada aplicação. Busca-se por meio do suporte, identificar que associações surgem em uma quantidade expressiva a ponto de ser destacada das demais existentes".*

Para executar a sub-etapa de associação somente a definição de suporte mínimo será necessária. Nosso objetivo é apenas identificar os termos freqüentes de acordo com a expectativa do usuário, calculada através da indicação de um suporte mínimo, não sendo necessária outras informações normalmente requeridas quando se deseja formar regras.

A fig. 4.15 ilustra a execução da sub-etapa de associação, a partir das "transações" existentes na tabela 4.6 para termos independentes.

Supondo que foi estabelecido um suporte mínimo de 60%, inicialmente verifica-se que itens (X) atendem ao suporte mínimo estabelecido. Neste caso os itens que atendem ao suporte (60%) são: `article`, `author` e `abstract`. Em uma próxima fase, realiza-se a combinação destes itens dois a dois e uma nova verificação do suporte é realizada. Os itens compostos que atendem ao suporte (60%) previamente estabelecido são: (`article`, `author`) e (`article`, `abstract`).

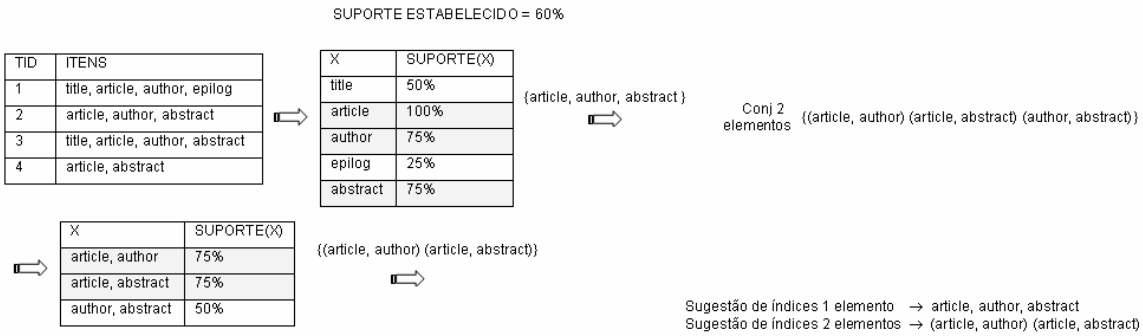


FIG. 4.15 Exemplificação do processo para identificação de termos independentes (1 a 1 ou 2 a 2).

O conjunto de passos necessários para identificação de pares de termos hierarquicamente dependentes (termoPai, termoFilho) que ocorram freqüentemente, é similar ao processo executado para termos independentes.

A identificação dos pares (termoPai, termoFilho) é realizada na sub-etapa identificação de termos na etapa pré-processamento. A tabela 4.7 mostra um exemplo do modelo *basket* com base na informação de termos hierarquicamente dependentes ilustrados na fig. 4.16.

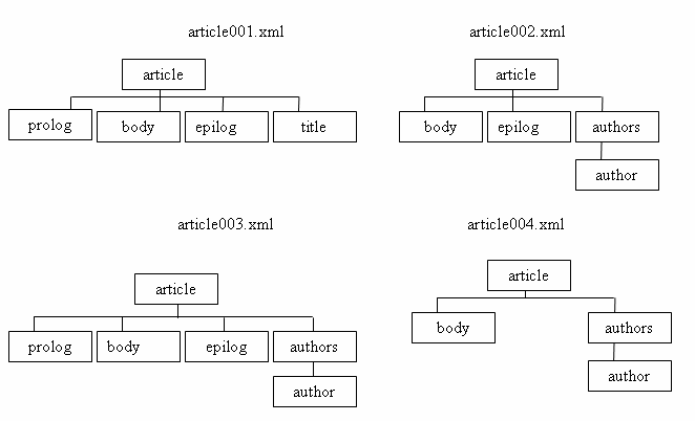


FIG. 4.16 Representação hierárquica documentos XML

TAB. 4.7 - Exemplificação formato *basket*-hierarquico

Seq. transação	Transação (documento xml)	Item (termo)
1	Article001.xml	artitle, prolog
1	Article001.xml	article, body
1	Article001.xml	article, epilog
1	Article001.xml	article, title
2	Article002.xml	article, body
2	Article002.xml	article, epilog
2	Article002.xml	authors, author
2	Article002.xml	article, authors
3	Article003.xml	artitle, prolog
3	Article003.xml	article, body
3	Article003.xml	article, epilog
3	Article003.xml	authors, author
3	Article003.xml	article, authors
4	Article004.xml	article, body
4	Article004.xml	authors, author
4	Article004.xml	article, authors

Supondo que foi estabelecido um suporte mínimo de 60%, verifica-se que itens (X) atendem ao suporte mínimo estabelecido. Neste caso os itens que atendem ao suporte (60%) são: (`article`, `body`); (`article`,`epilog`); (`authors`, `author`).

A fig. 4.17 ilustra a execução da sub-etapa de associação a partir das "transações" existentes na tabela 4.7 para termos hierarquicamente dependentes.

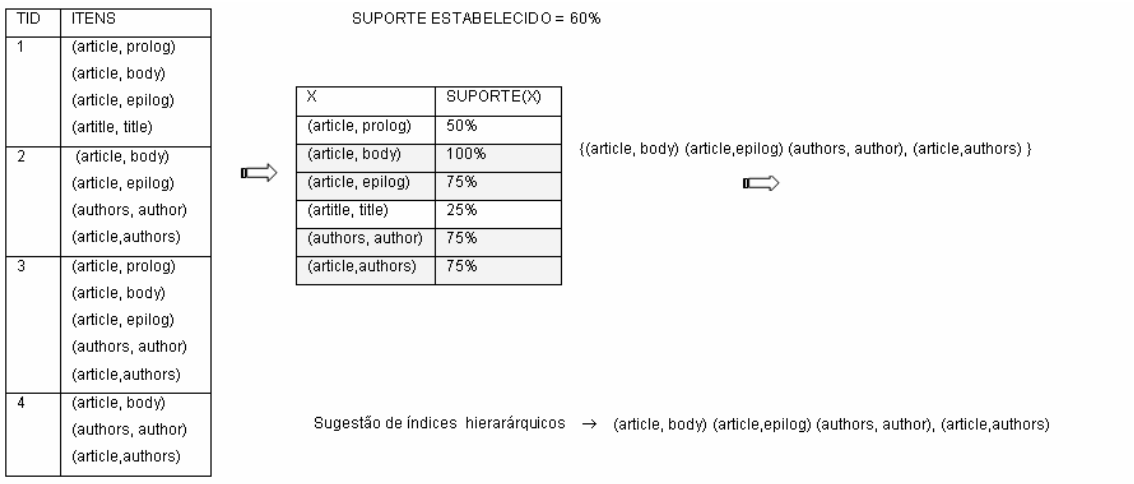


FIG. 4.17 Exemplificação do processo para identificação de termos hierarquicamente dependentes

Após identificado os termos que ocorrem freqüentemente segundo as duas abordagens citadas, deve ser realizada a sub-etapa de definição de índices na etapa de pós-processamento.colha dos candidatos a índice. Esta atividade consiste em selecionar, com base nas informações resultantes da atividade anterior, com base nas consultas que serão realizadas e com bases nos recursos de indexação do SGBD XML Nativo, que termos ou pares de termos devem ser indexados.

Na etapa de processamento, vimos todas as sub-etapas desde o processo de clusterização ao processo de associação. Cabe lembrar que as tarefas de clusterização, sumarização e associação estão detalhadas no capítulo de mineração de dados.

4.3 ETAPA PÓS-PROCESSAMENTO

Nesta etapa, o usuário é guiado na elaboração de um projeto físico de banco de dados com base nas informações geradas nas etapas anteriores somado aos recursos normalmente oferecidos por SGBD's XML Nativos. Nesta etapa, são definidas quantas Unidades de Armazenamento (UA's) serão necessárias para o armazenamento dos documentos, de que tipos devem ser estas UA's e os índices que devem atuar sobre as mesmas.

Nesta etapa, inicialmente define-se o tipo e a quantidade das UA's. Em seguida, define-se quais metadados serão associados aos documentos. Uma vez

identificados os metadados, os documentos são então armazenados nas unidades correspondentes. Uma vez criadas e populadas as UA's, a partir da sugestão de índices fornecida pela etapa de processamento e das alternativas oferecidas pelo SGBD, procede-se a definição e criação dos índices.

As sub-etapas presentes na etapa de processamento estão ilustradas na fig. 4.18.

FIG. 4.18 Etapa pós-processamento

4.3.1 DEFINIÇÃO DA UNIDADE DE ARMAZENAMENTO

Após a definição dos *clusters*, partimos para verificações que envolvem a UA. Esta sub-etapa visa definir quantas UA's serão necessárias para armazenar os *clusters* definidos. A princípio, a quantidade mínima de UA's corresponde a quantidade de *clusters* estabelecida. A quantidade máxima de UA's depende de outros fatores que vão desde a capacidade física de cada UA a estratégias diferenciadas que podem ser aplicadas de acordo com as características dos documentos.

A definição da UA engloba duas atividades conforme ilustra a fig. 4.19. A Análise da UA quanto à estratégia de armazenamento e a Análise da UA quanto à capacidade física.

A análise da UA quanto à estratégia de armazenamento avalia os recursos que o SGBD XML Nativo oferece, enquanto a análise da UA, quanto a capacidade física, verifica a capacidade do mesmo.

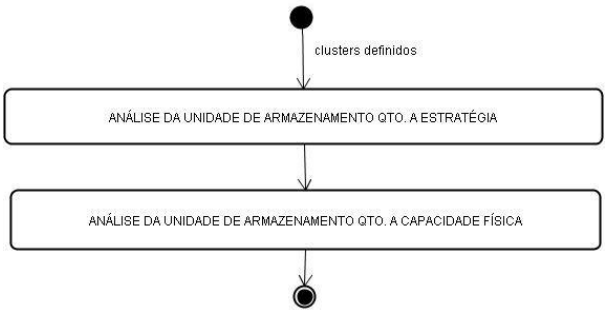


FIG. 4.19 Atividades da definição da UA

ANÁLISE DA UNIDADE DE ARMAZENAMENTO QUANTO À ESTRATÉGIA

Esta atividade está relacionada aos recursos que o SGBD oferece. Existem SGBD's XML Nativos que definem estratégias de armazenamento diferenciadas de acordo com o tamanho dos documentos. Por exemplo, há SGBD's que orientam que documentos muito grandes sejam armazenados de forma fragmentada, enquanto documentos pequenos sejam armazenados de forma contígua, sem que haja a fragmentação (SLEEPYCAT, 2005). As figs. 4.20a e 4.20b ilustram as formas de armazenamento citadas.

Um mesmo cluster pode conter documentos com diferentes tamanhos. Neste caso, sugere-se que os documentos sejam distribuídos de acordo com suas características em UA's apropriadas.

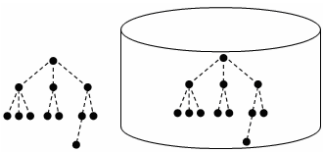


FIG 4.20a Armazenamento contíguo

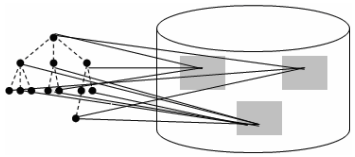


FIG 20b Armazenamento fragmentado

Se houver documentos que apresentem características diferenciadas, cada documento será armazenado na UA que melhor atenda a estas características.

ANÁLISE DA UNIDADE DE ARMAZENAMENTO QUANTO À CAPACIDADE FÍSICA

Sejam **U** a capacidade da UA e $\mathbf{S(X_i)}$ o tamanho de cada documento $\mathbf{X_i}$ que compõe o *cluster* e **Q** a quantidade de UA's necessárias. Se a soma do tamanho dos documentos que pertencem a um mesmo *cluster* ultrapassa a capacidade máxima de

armazenamento suportada pela UA, estes devem ser subdivididos em mais de uma UA conforme especifica a expressão da fig. 4.21. Se a UA suportar o armazenamento de todos os documentos de cada *cluster,* estes devem ser mantidos em uma única UA.

$$\text{Se } \sum_{i=1}^{n} S(x_i) \geq U \text{ Então } Q = \left\lceil \frac{\sum_{i=1}^{n} S(x_i)}{U} \right\rceil \text{ Senão } Q = 1$$

FIG. 4.21 Quantidade de UA quanto à capacidade da UA

No caso do particionamento em mais de uma UA pode ser levado em conta o grau de pertinência destes documentos ao *cluster* na hora de decidir quais os documentos devem ficar em qual UA, ou seja, documentos que possuem graus de pertinência mais próximos devem ser armazenados em uma mesma UA. A fig. 4.22 ilustra a subdivisão de *clusters* em mais de uma unidade de armazenamento de acordo com a capacidade da UA.

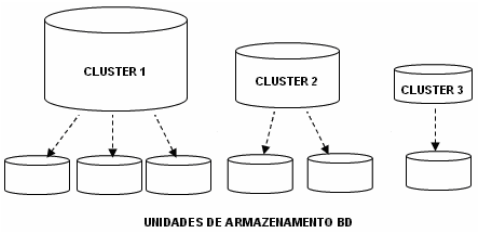


FIG 4.22 Capacidade da UA

Após termos as UA's definidas, parte-se para a definição dos metadados.

4.3.2 DEFINIÇÃO DE METADADOS

Alguns SGBD's permitem o armazenamento de metadados (ELMASRI; NAVATHE, 2005, p.7), que são informações adicionais que podem ser associadas ao documento. Estas informações não alteram o conteúdo do documento, já que são dados sobre dados e podem ser utilizadas posteriormente no momento da recuperação do documento, servindo como filtro em consultas.

Na estratégia proposta, vale ressaltar que o documento sendo armazenado é um documento modificado. Diferente do original, nestes documentos os termos originais foram substituídos pelos termos representantes na etapa de pré-processamento. Havendo necessidade de se manter o documento original, pode ser feito uma referência ao mesmo através da utilização de metadados. Esta referência irá indicar a localização do arquivo original, que pode ser um diretório, uma unidade de armazenamento ou até mesmo a Internet.

Além da referência ao documento original, há outras informações que podem ser armazenadas como metadados, tais como: nome do documento que na maioria dos SGBD's XML Nativos já é utilizado, pertinência do documento ao *cluster*, nome de quem criou a unidade de armazenamento, data de criação, tamanho do documento, dentre outras informações que podem ser consideradas. Como já mencionado anteriormente, podemos utilizar consultas que envolvam estas informações. Por exemplo: em uma determinada consulta deseja-se saber os livros que foram cadastrados por uma determinada pessoa, em um determinado dia e que tenham um determinado valor de pertinência ao grupo. Enfim, através destas informações conseguimos ter informações agregadas à informação original, o documento XML, sem que para isto sua estrutura ou conteúdo tenham que ser modificadas.

Sugerimos que as informações geradas nas etapas do projeto sejam aproveitadas como metadados ao armazenar os documentos. Outras informações também podem ser consideradas a critério do usuário.

4.3.3 ARMAZENAMENTO

Uma vez definidas as UA's e os metadados, os documentos já podem ser armazenados. A sub-etapa armazenamento consiste em realizar o armazenamento físico dos documentos XML no SGBD XML Nativo.

Nesta sub-etapa é criada a UA de acordo com as definições configuradas na sub-etapa definição da UA. Os documentos que compõem cada UA serão armazenados de acordo com o resultado da etapa de processamento e os metadados presentes em cada UA serão criados de acordo com o que foi configurado na sub-etapa de definição de metadados.

É importante armazenar os documentos tratados, resultado da etapa de pré-processamento, já que desta forma a recuperação das informações contidas nestes documentos será facilitada. Outras técnicas para recuperação de documentos heterogêneos, como a criação de visões (FLORESCU, 2005) também são encontradas na literatura. No entanto, em acervos com grande volume de documentos, a indexação é uma prática necessária para agilizar o acesso, e, a não ser que sejam materializadas, as "visões" não podem ser indexadas.

4.3.4 DEFINIÇÃO DE ÍNDICES

Esta sub-etapa visa aplicar estratégias de indexação sobre as UA's definidas. É possível criar índices sobre a estrutura dos documentos XML, isto é, os elementos (*tags*) ou atributos. A partir da escolha dos candidatos a índice, última atividade da sub-etapa de associação, temos a informação de quais *tags* podem ser indexadas.

Além de existir estratégias para indexação de *tags* isoladas, existem estratégias que permitem a combinação de dois ou mais *tags* para realizar a indexação. A indexação geralmente é permitida para elementos, atributos e metadados e a estratégia de indexação permite inferirmos algumas regras baseadas na estrutura e no conteúdo dos documentos.

Na etapa de processamento foi realizada a indicação de que índices poderiam ser utilizados, através da sub-etapa de associação. Nesta etapa discutiremos que tipos de índices deverão ser fisicamente criados de acordo com os recursos normalmente existentes nos SGBD's XML Nativos. A fig. 4.23 detalha a relação entre as sub-etapas da etapa de processamento, responsável pela indicação de índices e da etapa de pós-processamento responsável pela definição dos índices.

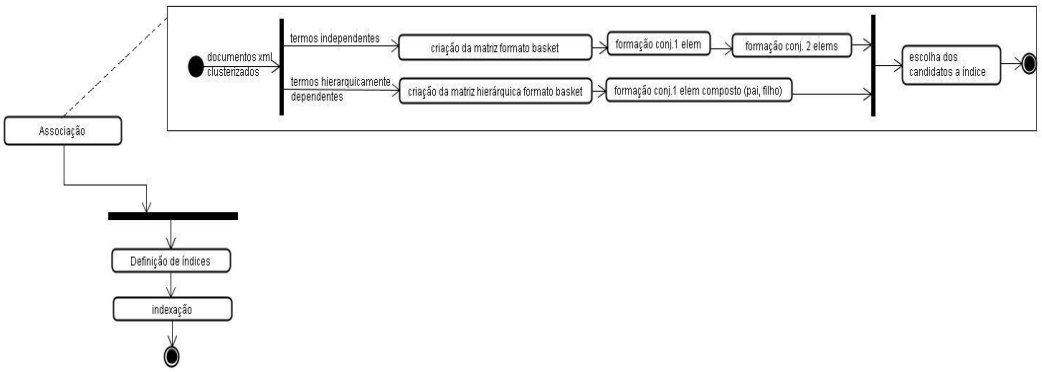


FIG. 4.23 Relação entre as sub-etapas responsáveis pela indicação e definição de índices

DEFINIÇÃO DE ÍNDICES UNITÁRIOS

O objetivo desta sub-etapa é a definição de estratégias de indexação. Pela simples avaliação da ocorrência das *tags* nos documentos podemos verificar que *tags* são constantes, ou seja, elementos que aparecem em grande parte dos documentos, e que *tags* raramente aparecem, ou seja, elementos específicos que mesmo dentro de um grupo de documentos similares aparecem em um documento ou em outro, sem nenhuma regularidade. A sub-etapa de associação provê esta informação.

Tradicionalmente, no projeto físico de BD, os índices são inicialmente definidos sobre atributos chave. Para atributos que não são chave, supondo que há uma distribuição uniforme de valores distintos, pode-se estimar a seletividade dos atributos (ELSMARI; NAVATHE, 2005, p.377) para escolha de índices. Muitos SGBD's oferecem recursos para obter esta informação de modo mais preciso. Quando não se tem noção das consultas mais críticas, pode-se utilizar da informação sobre a frequência das *tags*, combinada à seletividade(ELSMARI; NAVATHE, 2005, p.377) estimada para encontrar os elementos (*tags*) candidatos à indexação.

O fato de uma *tag* ser freqüente sugere que pode ser utilizada na maior parte das consultas, porém indexá-la pode melhorar ou não a recuperação dos documentos, dependendo da seletividade da *tag*. Por exemplo, em grande parte dos documentos, a *tag* `nome` pode aparecer com alta ocorrência, neste caso sua indexação é importante por que esta *tag* sempre trará valores distintos. Da mesma forma a *tag* `Pais` irá aparecer com alta ocorrência, mas a variabilidade de valores é baixa se os documentos forem originários de um único país. Neste caso, a *tag* `Pais` não deveria ser indexada pois sua seletividade não justificaria.

Uma característica dos documentos XML é a sua irregularidade quanto à

estrutura. Há *tags* que aparecem raramente. Uma estratégia específica de indexação sugere manter índices sobre *tags* que apresentam tais características. Por outro lado não se deve indexar todas as *tags* raras, já que há *tags* que dificilmente serão utilizadas na formulação de uma consulta. Para exemplificar esta situação, considere o contexto de análise de currículo onde algumas informações nunca são consideradas para avaliar o grau de experiência e instrução dos candidatos. Por exemplo, a *tag* `primeiroGrau` que se refere a formação primária do candidato, é uma *tag* rara, pois dificilmente as pessoas que criam um currículo incluem esta informação, apesar de já terem tido esta formação. Além de ser uma *tag* rara, esta *tag* traz uma informação que pode ser irrelevante e nunca ser utilizada na formulação de uma consulta. Por outro lado, a *tag* `doutorado`, que também é uma *tag* rara pode ser bastante relevante quando se pretende buscar um diferencial em um candidato. Neste caso, haverá consultas em que se buscará esta *tag* mesmo que a maioria dos currículos armazenados não a tenha.

Há estratégias de indexação específicas tanto para *tags* muito freqüentes quanto para *tags* raras, cujo uso deve ser cuidadosamente analisado para que o objetivo de melhora da performance das consultas seja alcançado.

Assim sendo, nesta etapa o usuário deve definir que estratégias, recursos ou técnicas serão utilizados para cada caso. No capítulo que fala sobre a Implementação do protótipo (Capítulo 5), na seção Detalhamento Berkeley DB XML (seção 5.5), apresentamos recursos oferecidos por este banco de dados que incluem estratégias de indexação diferenciadas.

DEFINIÇÃO DE ÍNDICES COMPOSTOS

Assim como nos SGBD's relacionais, onde um índice pode ser composto por um ou mais atributos, em SGBD's XML Nativos pode-se criar índices compostos com base em 2 ou mais elementos.

Há duas estratégias para definição de índices compostos: sem considerar o caminho e considerando o caminho. Enquanto a primeira estratégia não considera a hierarquia entre as *tags* nos documentos, a segunda verifica que pares de *tags* dentro de um mesmo caminho são freqüentes.

Este tipo de verificação é importante já que existem estratégias de indexação específicas que consideram o caminho. A utilização de índices que consideram o caminho torna-se essencial quando existe um elemento de mesmo nome que é *filho* de elementos diferentes.

Por exemplo, supondo que há vários documentos com a seguinte estrutura (`dadosPessoais`,`nome`); (`instituicao`,`nome`), como ilustrado na fig. 4.24. A criação de um índice sobre a *tag* (`nome`) pode não ser tão eficiente quanto a criação de dois índices compostos (`dadosPessoais`,`nome`) e (`instituicao`,`nome`). Isto porque a *tag* (`nome`) aparece em diferentes contextos. Neste caso, o importante é identificar as consultas mais freqüentes, se estas demandariam pelo contexto ou não. A indexação da *tag* `nome` sem o seu contexto não traria grandes ganhos na execução de uma consulta que determina o contexto da *tag* `nome`, já que ao utilizar tal índice seria necessário percorrer dois caminhos distintos, mesmo que apenas um estivesse correto.

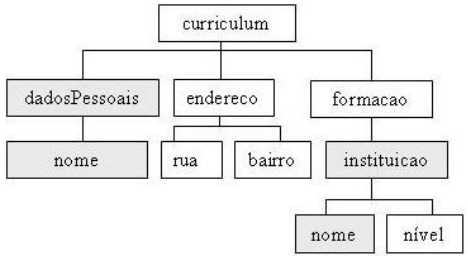


Fig. 4.24 Representação de *tags* da mesma descrição em diferentes caminhos

A Tabela 4.8. representa uma matriz hierárquica de visualização de termos, com base nas informações do documento XML da fig. 4.24.

TAB. 4.8 - Matriz hierárquica de termos da figura 4.24

FILHO / PAI	CURRICULUM	DADOSPESSOAIS	NOME	ENDERECO	RUA	BAIRRO	FORMACAO	INSTITUICAO	NIVEL
CURRICULUM									
DADOSPESSOAIS			1						
NOME									
ENDERECO					1	1			
RUA									
BAIRRO									
FORMACAO								1	
INSTITUICAO			1						1
NIVEL									

Os índices compostos correspondem a estratégias diferenciadas de indexação. No primeiro caso, consultas que envolvessem uma seleção baseada em dois ou mais elementos poderiam se beneficiar desta estratégia. No segundo caso, é importante

ressaltar que há estratégias diferentes e que estas provêm alternativas para a criação de índices. O exemplo da *tag* `nome`, que ocorre em mais de um ponto em um documento e que é freqüente através dos documentos, sugere que a criação de índices, considerando ou desconsiderando o caminho, sejam criados. A segunda alternativa não seria útil para consultas que consideram o contexto da *tag* `nome`, porém poderia ser útil quando a consulta fosse mais genérica. Utilizar ou não os índices é uma decisão do SGBD.

DEFINIÇÃO DE ÍNDICES SOBRE TEXTOS

Para os documentos centrados em dados, as consultas são direcionadas a valores específicos. A estratégia de indexação que deve ser utilizada é a estratégia que trabalha com expressões de igualdade.

Os documentos centrados em textos usualmente definem elementos que descrevem textos. Nestes casos, é comum surgir a necessidade de recuperar estes documentos utilizando parte do texto. Este tipo de consulta pode se tornar muito oneroso se não houver um índice criado para os elementos que mantêm estes textos. Existem estratégias de indexação previstas pelos SGBD's XML Nativos, que visam melhorar o desempenho de consultas deste tipo.

Mas como identificar se um documento é centrado em textos ou centrado em dados? Se o documento não está associado a um *XML Schema* ou a um *DTD* esta tarefa fica mais difícil. Neste caso, pode-se fazer uma análise com base no tamanho do conteúdo dos campos. Considerando esta informação é possível sugerir o tipo de índice mais apropriado.

4.3.5 INDEXAÇÃO

Nesta sub-etapa será realizada a indexação fisicamente no SGBD XML Nativo para cada UA, de acordo com o que foi definido na sub-etapa de definição de índice.

4.4 CONSIDERAÇÕES SOBRE A ESTRATÉGIA

Optou-se nesta estratégia, a identificação de documentos similares, observando, dentre outros fatores, apenas sua estrutura. Portanto, não foi utilizada a Técnica de *Textmining*, que poderia ser empregada se o objetivo fosse identificar documentos que compartilhassem o mesmo conteúdo. No nosso caso, o objetivo foi identificar e documentar similares que pertencessem a um mesmo contexto.

Os documentos submetidos a esta estratégia são documentos que normalmente não sofrem alterações, tais como: livros, artigos, revistas e outros documentos, nos quais uma alteração gera novas versões destes documentos, e portanto, outros documentos. Com isto, a tendência é que os documentos, uma vez armazenados e identificados dentro de unidades que mantêm documentos similares, não sofram alterações e assim as unidades de armazenamento mantêm sua homogeneidade.

Neste estudo, não foi considerada a entrada de novos documentos em UA's já populadas. A entrada de novos documentos isoladamente ou mesmo em um número considerável requer uma nova avaliação sobre que UA receberia o(s) novo(s) documentos. De qualquer forma, cada novo documento teria que passar pelas etapas de pré-processamento, processamento e pós-processamento, podendo haver alterações na estratégia. Diante deste cenário, outras questões podem ser levantadas, já que o novo documento pode não ser similar a nenhum dos grupos existentes e com isto surge a necessidade de se criar um novo grupo e portanto uma nova UA, a fim de manter a homogeneidade e o conjunto de regras aplicadas a esta UA.

A criação de UA's homogêneas somada a criação de índices, já nos fornece uma idéia do ganho de tempo a ser obtido na recuperação destes documentos. Para avaliar a estratégia proposta será mostrado um estudo de caso que apresenta uma comparação com outras abordagens.

4.5 CONSIDERAÇÕES TRABALHOS CORRELATOS

Diversas abordagens baseadas em KDD têm sido propostas para tratamento de um conjunto de documentos XML heterogêneos. Em (ABITEBOUL et al., 2005),

(FLESCA et al., 2005), (YANG et al., 2005), (NIERMAN e JAGADISH, 2002) e (LEE et al., 2001) é proposta a criação de um esquema único a partir da percepção da similaridade entre os documentos. Em alguns destes trabalhos são descritas técnicas para preparar os documentos XML para mineração de dados através da equiparação terminológica dos termos presentes nos documentos. Entretanto, a idéia de derivar um esquema único a partir de um acervo heterogêneo de documentos pode se tornar uma tarefa difícil, se não inviável, dependendo do grau de heterogeneidade do mesmo. Em (LI et al., 2004) é proposta uma estratégia para se trabalhar com coleções heterogêneas, baseada na formulação de consultas a diferentes esquemas que mantenham conteúdos similares. Diferente da nossa estratégia neste caso, os documentos não são agrupados por similaridade, permanecendo nas unidades de armazenamento originais, e portanto, sem os benefícios do armazenamento em separado e da indexação, que agilizariam as consultas.

Neste trabalho, propomos uma estratégia que envolve o uso de técnicas de KDD, como clusterização, sumarização e associação, no sentido de minimizar a heterogeneidade dos documentos, através da separação do acervo em grupos mais homogêneos. Em (LIAN et al., 2004), a clusterização dos documentos também é utilizada, porém com objetivo diferente. Neste caso, a clusterização é realizada através da análise da hierarquia dos termos para minimizar o problema de fragmentação em bancos de dados relacionais. A tabela 4.9 resume através de um quadro comparativo as abordagens da estratégia proposta e das demais estratégias encontradas nos trabalhos correlatos citados.

TAB. 4.9 – Tabela comparativa estratégia proposta e trabalhos correlatos

	Estratégia proposta	**Demais estratégias**
Trabalha com documentos heterogêneos	X	X
Aplicam técnicas para equiparação terminológica	X	X
Utilizam-se técnicas de clusterização para identificar grupos de documentos similares	X	X
Clusterização por conteúdo		X
Clusterização por estrutura	X	X
Resultado é aplicado no armazenamento	X	
Resultado é aplicado na criação de esquema único		X
Utilizam-se outras técnicas de mineração(além da clusterização)	X	
Prevê a criação de índices sobre acervos heterogêneos	X	
Foco do trabalho está nas consultas a documentos heterogêneos		X
Propõe um conjunto de etapas para orientar armazenamento e indexação	X	

Diferentes técnicas de clusterização foram utilizadas nestes trabalhos, sendo que todas produzem um resultado determinístico, enquanto a técnica de clusterização *fuzzy*, adotada neste trabalho, confere resultados mais precisos já que o objeto submetido à clusterização pode pertencer a mais de um grupo com diferentes graus de pertinência.

5 GAIDoX

O GAIDoX (Guia para Armazenamento e Indexação de Documentos XML) tem por objetivo dar suporte ao cumprimento das etapas definidas na estratégia proposta para o desenvolvimento do projeto físico para armazenamento de documentos XML heterogêneos.

Inicialmente, iremos discutir o que foi implementado no protótipo, em relação ao que foi proposto no capítulo 4. Em seguida, será apresentada a arquitetura do protótipo bem como as plataformas utilizadas. Por fim, será fornecido um detalhamento do Berkeley DB XML, SGBD XML Nativo utilizado no armazenamento dos documentos.

5.1 REDUÇÃO DO PROCESSO

No processo proposto, conforme discutido no capítulo 4, há uma série de etapas, sub-etapas e atividades que devem ser seguidas para o armazenamento e indexação dos documentos XML. Nem todas as sub-etapas foram implementadas. A seguir será revisto o modelo proposto e especificado do que não foi implementado. Na fig. 5.1, que ilustra as etapas da estratégia proposta, estão assinaladas as sub-etapas que não foram implementadas.

A etapa de pré-processamento, sub-etapa Equiparação de termos, tem como objetivo reduzir a heterogeneidade dos documentos. Existem várias técnicas como o uso de *tesauros* que poderiam ter sido empregadas. No entanto, como o foco do nosso trabalho está no armazenamento, esta implementação não foi priorizada. Portanto, pressupõe-se que os documentos que possam vir a ser utilizados por este protótipo são documentos que já receberam algum tratamento prévio.

Na etapa processamento, a sub-etapa Sumarização não utiliza nenhum algoritmo para obtenção da descrição de cada *cluster* de forma automática ou semi-automática. O GAIDoX, sugere ao usuário que sejam escolhidos aleatoriamente alguns documentos de cada *cluster* para a descrição dos mesmos.

Por fim, a sub-etapa Definição de índices não foi implementada. Para a criação dos índices, os usuários deverão se apoiar nos resultados fornecidos pela etapa de associação e realizar a criação dos índices diretamente no SGBD XML Nativo, no nosso caso, o Berkeley DB XML.

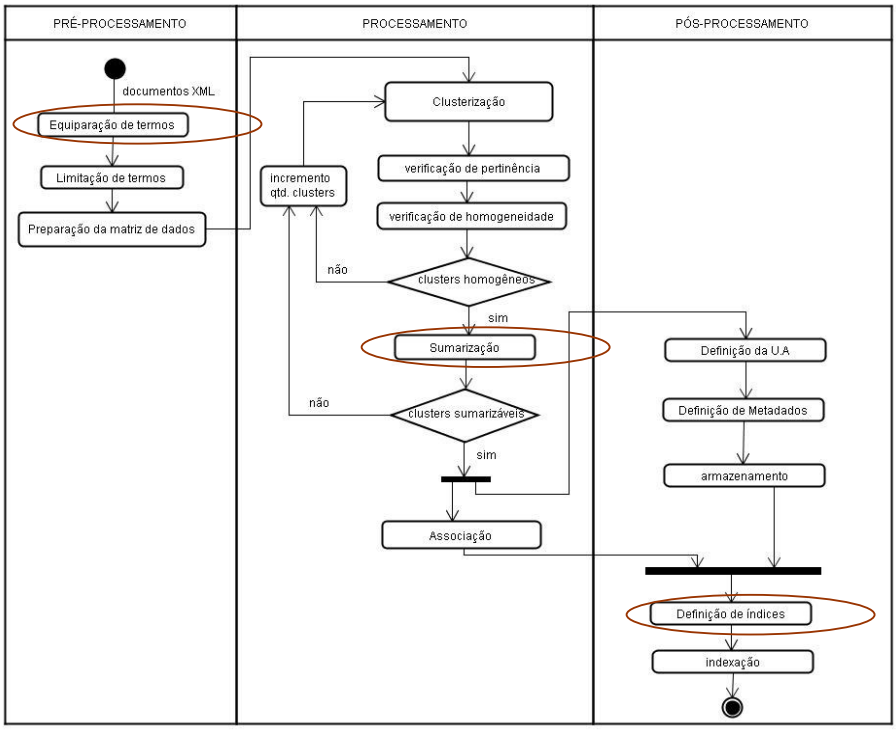


FIG. 5.1 Redução do processo

A seguir, será discutida a arquitetura do projeto proposto, evidenciando algumas características da implementação empregada.

5.2 ARQUITETURA DO GAIDoX

A arquitetura do protótipo, exposta na fig. 5.2, mostra os principais módulos, a interface com o MATLAB e os bancos de dados empregados.

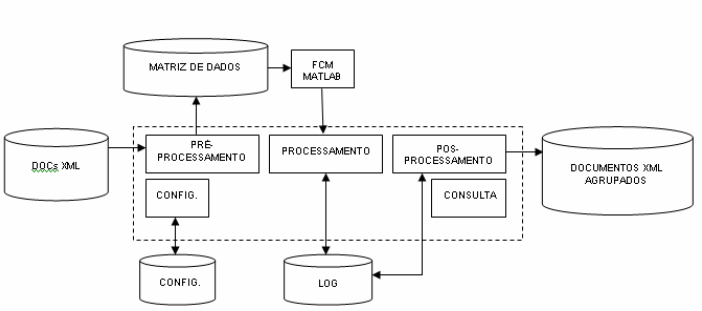


FIG. 5.2 Arquitetura do protótipo

A implementação do protótipo foi feita em Java JDK versão 1.5, utilizando a versão 2.2.13 da API fornecida pelo SGBD XML Nativo Berkeley DB XML (http://www.oracle.com/database/berkeley-db.html). Este SGBD é utilizado para armazenar os grupos de documentos XML, após a etapa de processamento.

O módulo de *configuração* faz o cadastramento de informações básicas, como a indicação de diretórios onde estão armazenados os documentos, a parametrização de valores (e.g. número de *clusters*, limiar de homogeneidade, etc.) e informações relacionadas ao SGBD utilizado (e.g. capacidade máxima de cada UA). Estas informações são armazenadas no BD de configuração e são consultadas pelos demais módulos.

O módulo de *pré-processamento* cobre as tarefas de identificação de termos e composição da matriz de dados. Neste módulo ainda é possível limitar a quantidades de *tags* que irão compor a matriz de dados.

No módulo de *processamento*, a tarefa de clusterização, que utiliza o algoritmo *Fuzzy K-Means* é realizada pelo MatLab (http://www.mathworks.com/), constituindo o módulo FCM MATLAB. Os resultados obtidos são importados para o módulo de processamento e repassados para as tarefas de verificação de pertinência e verificação de homogeneidade. Após constatada a homogeneidade dos *clusters* , é solicitada uma descrição para os mesmos. Este módulo cobre ainda as tarefas de associação cujos resultados são utilizados para a sugestão dos índices.

No módulo de *pós-processamento*, os documentos são armazenados de acordo com os resultados da clusterização obtidos na etapa de processamento, que além do armazenamento propriamente dito, cobre ainda as atividades de definição das UA's, onde é indicada a quantidade de UA's necessárias de acordo com a sua capacidade e o tamanho dos *clusters* formados; e a atividade de definição de metadados.

Os módulos de processamento e pós-processamento mantêm um *log* para

armazenar informações sobre as características do armazenamento e decisões do usuário. Basicamente, as informações do *log* contêm para cada *cluster*: o tamanho ocupado, a quantidade de UA's necessárias, os metadados selecionados, a data de criação e informações sobre a estratégia de armazenamento.

No protótipo há ainda um módulo para realização de consultas sobre os documentos já agrupados e armazenados.

5.2.1 MÓDULO DE CONFIGURAÇÃO

Neste módulo são definidos todos os parâmetros que são necessários para a configuração do sistema. Estes parâmetros de uma forma geral são responsáveis pela indicação de diretórios, parametrização de valores limite e disponibilização de informações relacionadas ao SGBD XML Nativo, no qual os documentos XML serão armazenados. A fig. 5.3 mostra a tela de configuração de parâmetros.

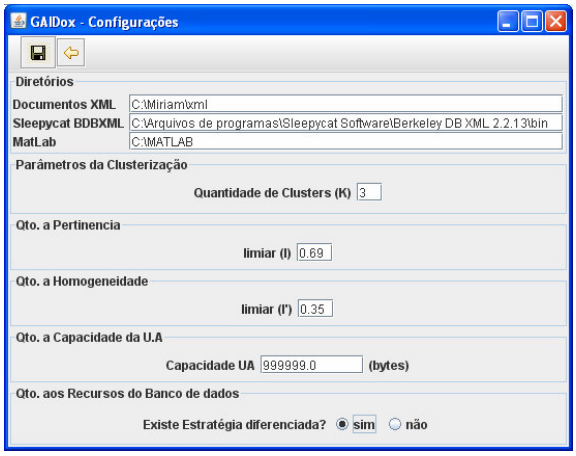


FIG. 5.3 Tela de configuração de parâmetros

A seguir serão explicados cada um dos parâmetros e sua função.

DIRETÓRIO XML

Diretório onde os documentos XML a serem tratados e posteriormente armazenados estão localizados.

DIRETÓRIO BERKELEY DB XML

Diretório onde está localizado o SGBD XML Nativo Berkeley DB XML.

DIRETÓRIO MATLAB

Diretório de instalação do MatLab.

QUANTIDADE DE *CLUSTERS*

O usuário deverá colocar um valor inicial de quantidade de *clusters* nos quais os documentos serão agrupados. Se o usuário tem uma idéia da quantidade de contextos distintos, este valor deverá corresponder a esta expectativa. Caso contrário, recomenda-se que este parâmetro seja inicializado com valor dois (2).

PERTINÊNCIA PADRÃO

O usuário deverá definir um limiar de pertinência que considere satisfatório para aplicação na verificação da pertinência dos documentos aos *clusters* realizada na fase de pós-processamento. Este valor deverá estar no intervalo entre 0.1 a 1.0.

HOMOGENEIDADE PADRÃO

O usuário deverá definir um limiar de homogeneidade a ser aplicado ao *cluster* que ajudará a definir se o *cluster* encontra-se homogêneo. Este limiar será aplicado na fase de pós-processamento para verificação da homogeneidade do *cluster*. Este valor deverá estar no intervalo entre 0.1 a 1.0.

CAPACIDADE DA UA

Capacidade da Unidade de Armazenamento, este valor deve ser determinado em *bytes*. Esta informação será utilizada para verificar se há espaço suficiente nas UA's disponibilizadas pelo SGBD para manter os documentos que serão armazenados, ou se fará necessário a criação de mais de uma unidade.

ESTRATÉGIA DIFERENCIADA

Se o SGBD apresenta estratégias diferenciadas de armazenamento, deverá ser indicado o valor 'S', caso contrário o valor 'N'. Por exemplo, para o SGBD XML Nativo Berkeley DB XML há duas formas de se manter os documentos armazenados em suas unidades de armazenamento: o documento pode ser mantido de forma fragmentada, *node*, ou de forma íntegra, *whole*. Neste caso, para este SGBD deveria ser indicado que o mesmo apresenta estratégias diferenciadas.

5.2.2 MÓDULO DE PRÉ-PROCESSAMENTO

Na etapa de pré-processamento ocorre a captação dos documentos a partir de um diretório previamente definido no módulo de configuração. Em seguida, é montada uma matriz de dados e é oferecida ao usuário uma opção para limitação de termos.

A aplicação inicia-se apontando para um diretório previamente configurado que contém os documentos XML. Os documentos e seus termos são lidos e é contabilizada e armazenada a quantidade de vezes em que o termo aparece no documento. Neste mesmo instante, são armazenadas as relações existentes entre os termos na hierarquia do documento (o termo filho e seu termo pai). Esta informação não será utilizada neste momento porque o sistema trabalha com uma matriz de freqüência e não considera a hierarquia dos termos nos documentos. No entanto, esta informação é armazenada para ser utilizada em outras etapas. Quando a leitura do último documento é concluída, é exibida a tela de pré-processamento. A fig. 5.4 mostra a tela de pré-processamento.

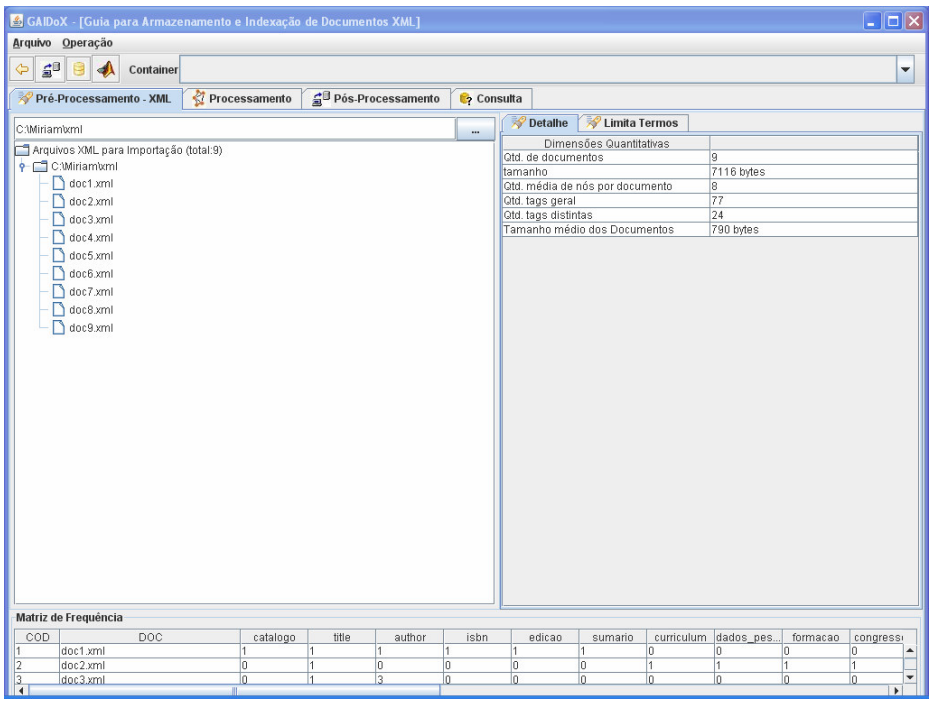
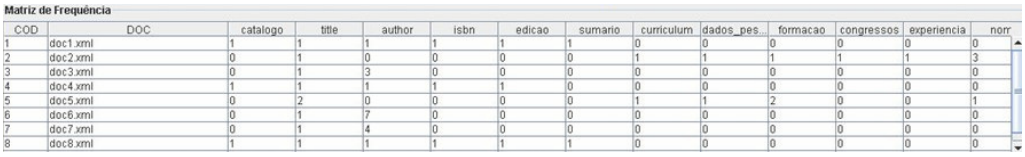


FIG. 5.4 Tela de pré-processamento

No canto superior esquerdo são exibidos os nomes dos documentos lidos e a

quantidade de documentos existente no diretório conforme mostra a fig. 5.4.

No canto inferior central é exibida a matriz de freqüência de dados. A matriz é composta por colunas e linhas. Nas colunas, vemos todos os termos presentes nos documentos e, nas linhas de termos, os documentos e sua relação com estes termos. Note que para cada documento é exibida a freqüência com que o termo aparece no documento, conforme mostra a fig. 5.5.



Matriz de Frequência

COD	DOC	catalogo	title	author	isbn	edicao	sumario	curriculum	dados_pes...	formacao	congressos	experiencia	norr
1	doc1.xml	1	1	1	1	1	1	0	0	0	0	0	0
2	doc2.xml	0	1	0	0	0	0	1	1	1	1	1	3
3	doc3.xml	0	1	3	0	0	0	0	0	0	0	0	0
4	doc4.xml	1	1	1	1	1	0	0	0	0	0	0	0
5	doc5.xml	0	2	0	0	0	0	1	1	2	0	0	1
6	doc6.xml	0	1	7	0	0	0	0	0	0	0	0	0
7	doc7.xml	0	1	4	0	0	0	0	0	0	0	0	0
8	doc8.xml	1	1	1	1	1	1	0	0	0	0	0	0

FIG. 5.5 Matriz de freqüência de dados

Conforme mostra a fig. 5.4., no canto superior direito, além de ser disponibilizado um detalhamento dos documentos lidos através da "aba" denominada detalhamento, há uma segunda "aba" denominada "limita termos" que será utilizada no auxilio à utilização de técnicas de *startword* ou *stopword*. Para limitação dos termos, são listados todos os termos presentes nos documentos e sua porcentagem de ocorrência em relação a todos os documentos analisados. A utilização deste recurso influencia diretamente a matriz de freqüência de dados gerada. A fig. 5.6 ilustra a tela pré-processamento com a opção de limitação de termos.

Inicialmente, a técnica de *startword* está selecionada como *default* e todos os termos estão selecionados. A utilização da "aba" limita termos e influi diretamente na construção da matriz final que será submetida à próxima etapa. No caso da utilização de *startword*, se algum dos termos for desmarcado, a tabela é reconstruída e somente os termos que continuam selecionados farão parte da matriz. No caso da escolha da técnica de *stopword,* os termos que estiverem marcados serão desconsiderados na reconstrução/atualização da matriz.

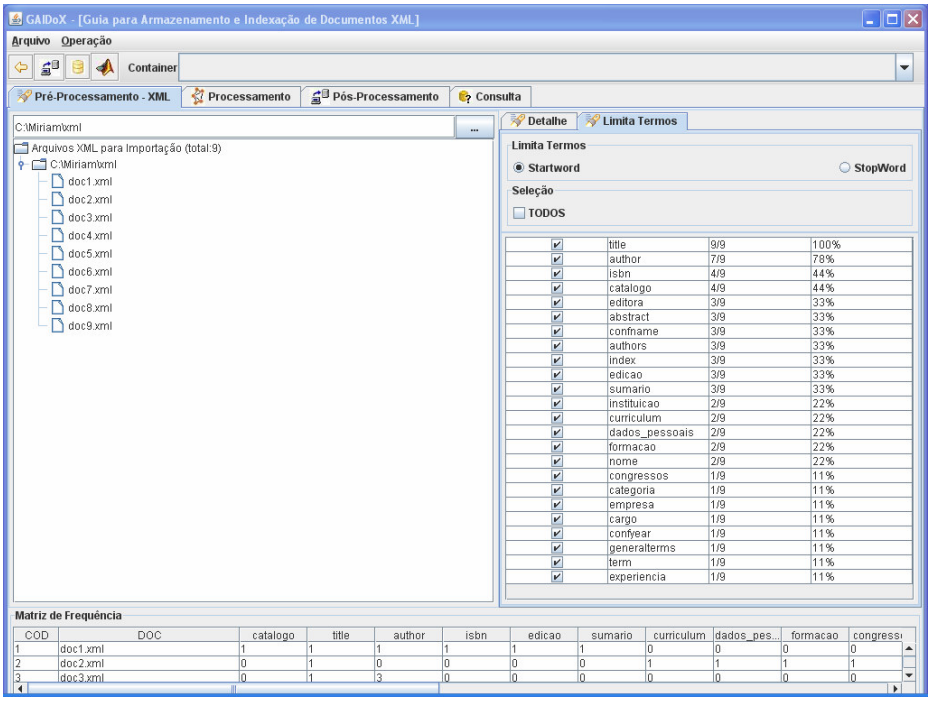


FIG. 5.6 Tela de pré-processamento – Limitação de termos

A matriz de freqüência de dados é exportada para um diretório, este diretório deverá ser o mesmo onde encontra-se o *script* para submissão ao MatLab. O arquivo responsável por executar o método de clusterização no MATLAB importa a matriz de freqüência de dados para utilização no algoritmo, como será mostrado posteriormente. As figs. 5.7 e 5.8 mostram a seqüência de passos para exportação da matriz.

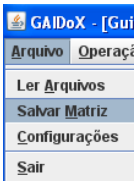


FIG. 5.7 Menu de salvar matriz

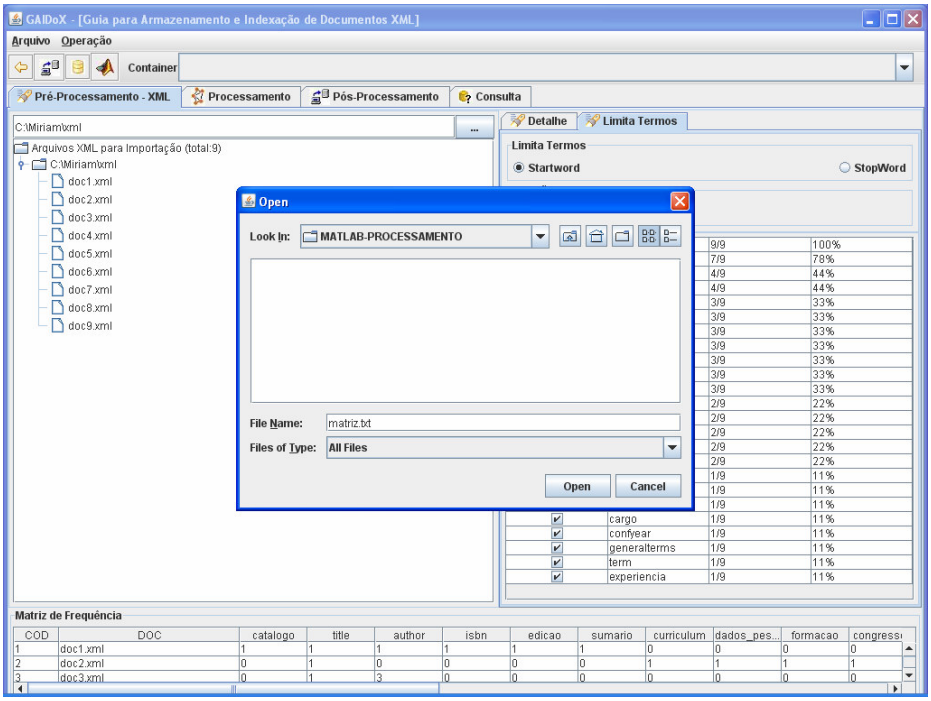


FIG. 5.8 Escolha do diretório para exportação da matriz

Após realizada a exportação da matriz de freqüência, a etapa de pré-processamento é finalizada.

5.2.3 MÓDULO DE PROCESSAMENTO

Neste módulo, foram implementadas as seguintes sub-etapas de processamento: Indicação dos *clusters*, Indicação de índices e Indicação de índices compostos hierárquicos.

5.2.3.1 INDICAÇÃO DOS *CLUSTERS*

O algoritmo de processamento não foi implementado neste módulo. Conforme exposto anteriormente, a última atividade na etapa de pré-processamento é a exportação

da matriz de freqüência. Esta matriz de freqüência será submetida ao algoritmo de clusterização *Fuzzy K-Means* do MATLAB. A fig. 5.9 mostra a tela de processamento.

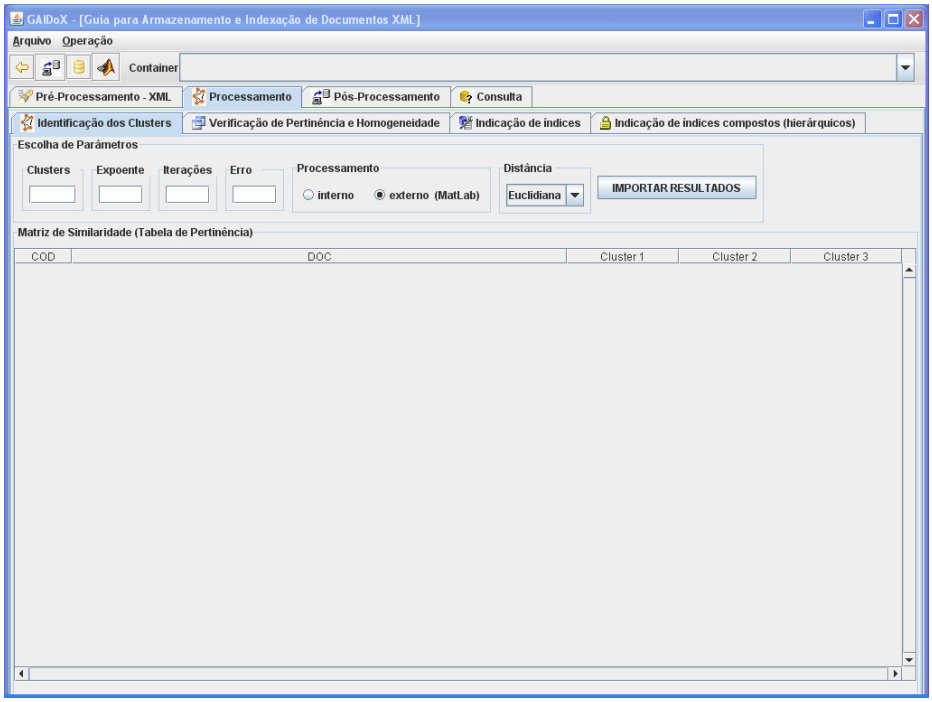


FIG. 5.9 Tela de processamento

A integração entre o módulo de processamento e o MATLAB é realizada através do acionamento do botão de execução do MATLAB conforme ilustra a fig. 5.10.

FIG. 5.10 chamada ao executável MATLAB

A fig. 5.11 mostra a interface do MATLAB e a chamada ao código que será responsável por executar o algoritmo.

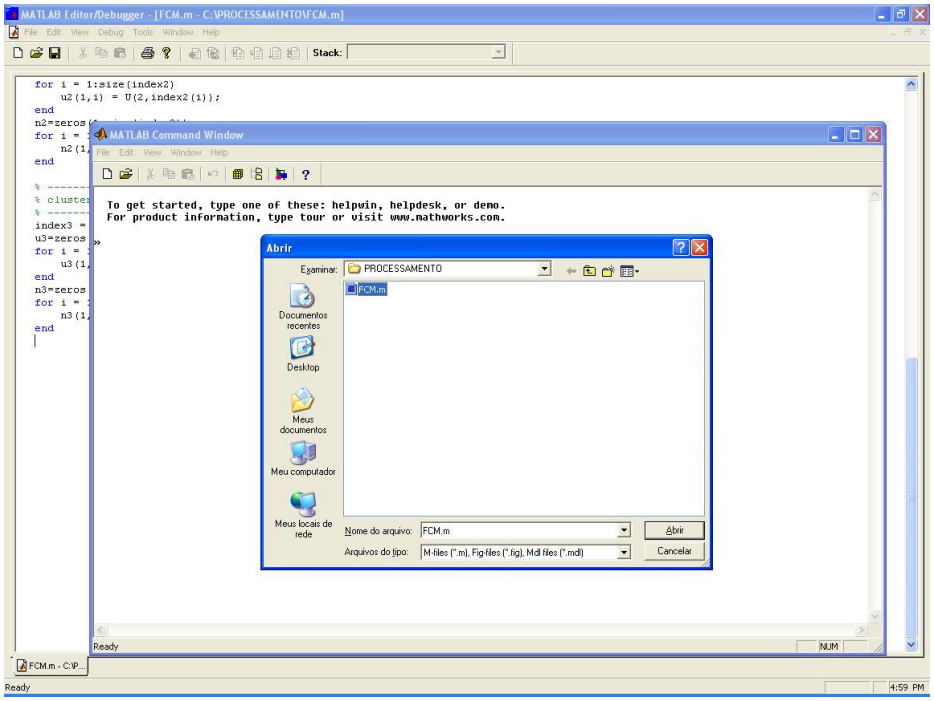


FIG. 5.11 Interface MATLAB

Foi gerado um script a ser executado denominado *FCM.m*. A fig. 5.12 mostra a estrutura deste arquivo, integralmente disponível no anexo 1 desta dissertação.

Na implementação do protótipo, a normalização dos dados foi realizada neste momento, mas também poderia ter sido realizada ao final da etapa de pré-processamento. O importante é que no momento da submissão ao método de clusterização, a matriz já esteja normalizada.

A estrutura do arquivo *FCM.m* está dividida em quatro partes:

- Leitura do arquivo que contém a matriz de freqüência de dados;
- Normalização da matriz;
- Chamada ao método FCM que encontra-se encapsulado no MATLAB;
- Exibição dos resultados.

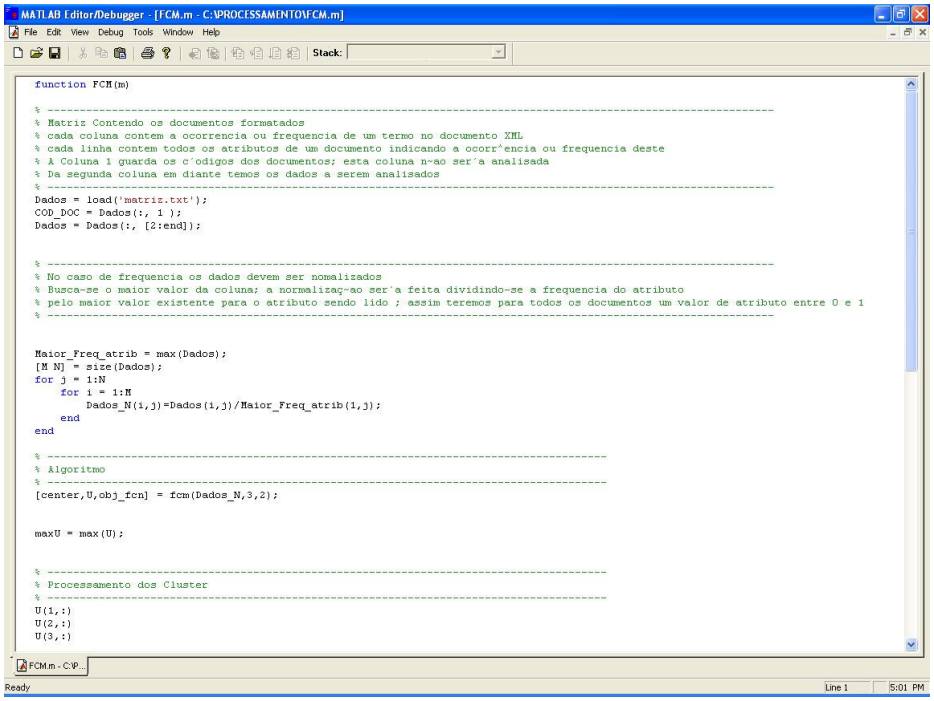


FIG. 5.12 Estrutura do Arquivo FCM.M

LEITURA DO ARQUIVO

A matriz está sendo carregada em uma variável denominada *dados.* A primeira posição/ coluna da matriz contém a identificação do documento e as demais posições contêm as freqüências relacionadas a cada termo. O código que identifica cada um dos documentos estão presentes na primeira coluna da matriz de freqüência de dados presente na fig. 5.5, e as freqüências aparecem a partir da terceira coluna da mesma figura. Estes códigos foram gerados automaticamente no momento em que os documentos estavam sendo lidos.

NORMALIZAÇÃO DA MATRIZ

Neste momento, é executada a normalização pelo valor máximo dos elementos. As freqüências são substituídas pelo seu valor normalizado.

EXECUÇÃO DO ALGORITMO

É realizada a chamada ao método FCM, que recebe três parâmetros: a matriz normalizada, a quantidade de *clusters* e a taxa de nebulosidade a ser utilizada no algoritmo.

EXIBIÇÃO DOS RESULTADOS

Por fim, os resultados do processamento são exibidos. A fig. 5.13 mostra os resultados obtidos na execução do algoritmo de clusterização para os nove documentos contidos na matriz de freqüência.

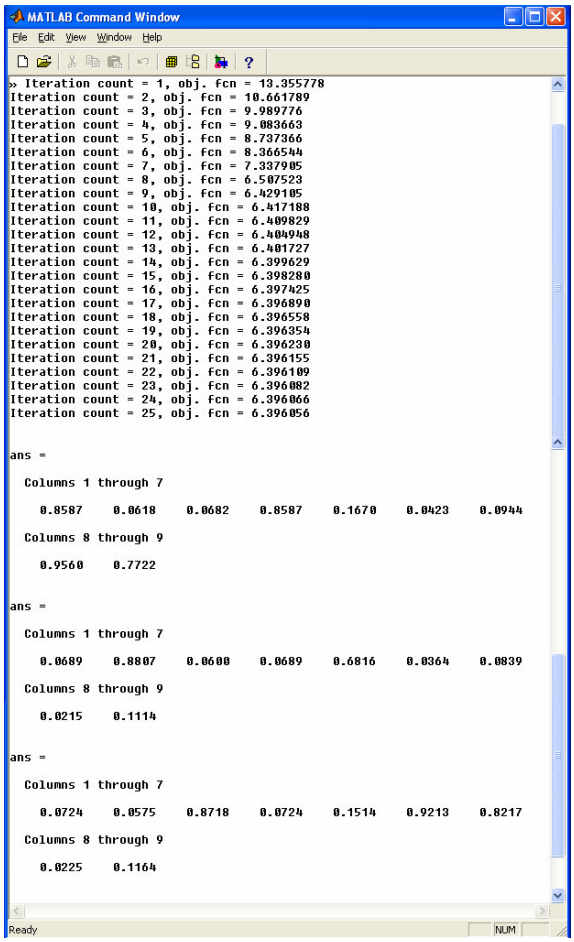


FIG. 5.13 Resultado da execução do algoritmo FCM

Os resultados mostram de forma seqüencial, a pertinência dos documentos a cada *cluster*. Por exemplo, os resultados 0.8587, 0.0689 e 0.0724 referem-se à pertinência em relação ao documento 1, que é o primeiro documento da matriz, aos clusters 1, 2 e 3 respectivamente.

O único processo realizado fora do sistema é a execução do algoritmo de clusterização, realizado através do MATLAB. Os resultados devem ser salvos em um arquivo onde devem ser mantidos os resultados efetivos, conforme ilustra a fig. 5.14.

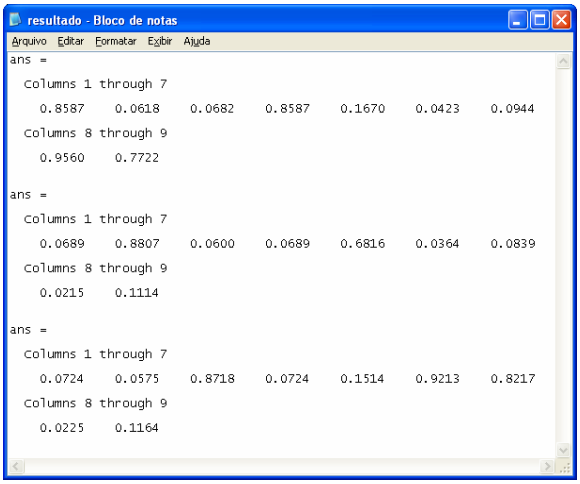


FIG. 5.14 Arquivo resultado para exportação

No GAIDoX, importamos os resultados através do botão "Importar resultados". Para tal, localizamos o arquivo que mantém o resultado. A fig. 5.15 mostra a interface de importação dos resultados.

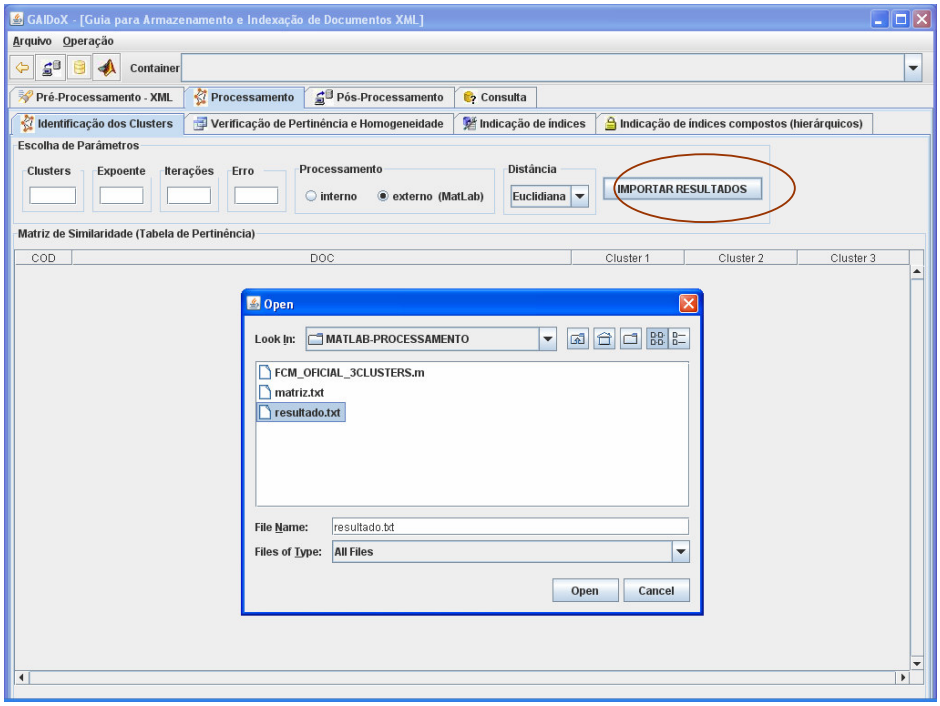


FIG. 5.15 Importação de resultados

Com os resultados importados, é gerada automaticamente uma matriz de similaridade que contém os valores de pertinência auferidos, conforme mostra a fig. 5.16. Estes resultados serão utilizados nas sub-etapas seguintes de *Verificação de*

Pertinência e *Homogeneidade*.

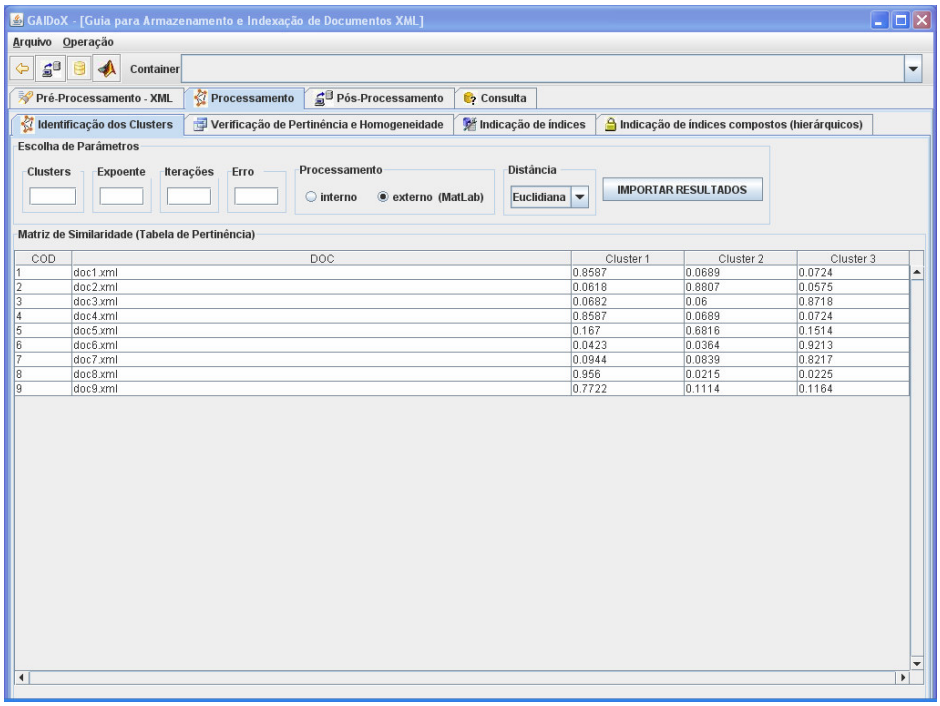


FIG. 5.16 Matriz de similaridade

5.2.3.2 VERIFICAÇÃO DE PERTINÊNCIA E HOMOGENEIDADE

Antes que seja realizado o armazenamento físico no SGBD, algumas atividades ainda devem ser cumpridas. São elas: verificação de pertinência e verificação de homogeneidade.

VERIFICAÇÃO DE PERTINÊNCIA

Inicialmente, todos os documentos são pertinentes a todos os *clusters*, com diferentes valores de pertinência. Cabe ao usuário definir que valor é considerado expressivo para que o documento seja considerado daquele *cluster*. A fig. 5.17 exibe as pertinências dos documentos ao *cluster* em análise.

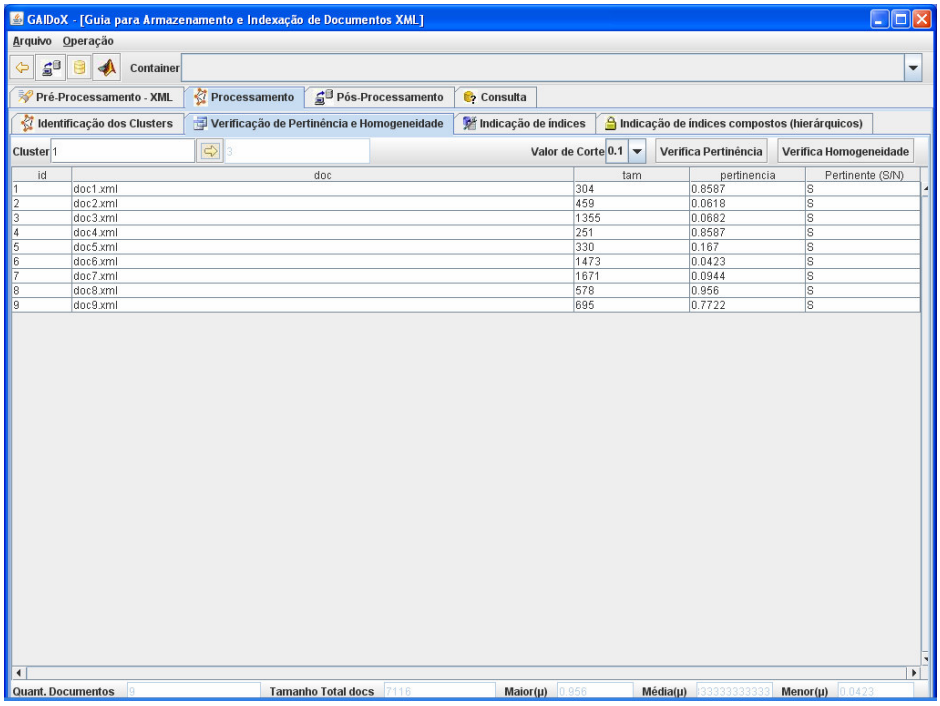


FIG. 5.17 Verificação de pertinência

Este valor previamente configurado, que aparece no protótipo como valor de corte, chama-se limiar de pertinência. A todos os documentos que têm o valor de pertinência superior ao valor de corte, é atribuído o valor "S", significando que estes documentos são pertinentes ao *cluster* em questão. Os demais recebem o valor "N". A fig. 5.18 ilustra a pertinência ao *cluster* 1, de acordo com o valor de corte atribuído, 0.6.

Este processo deve ser realizado para todos os *clusters*. Note que na tela é exibido o *cluster* sendo analisado. As figs. 5.19 e 5.20 mostram os resultados do processo de avaliação de pertinência para os *clusters* 2 e 3 respectivamente.

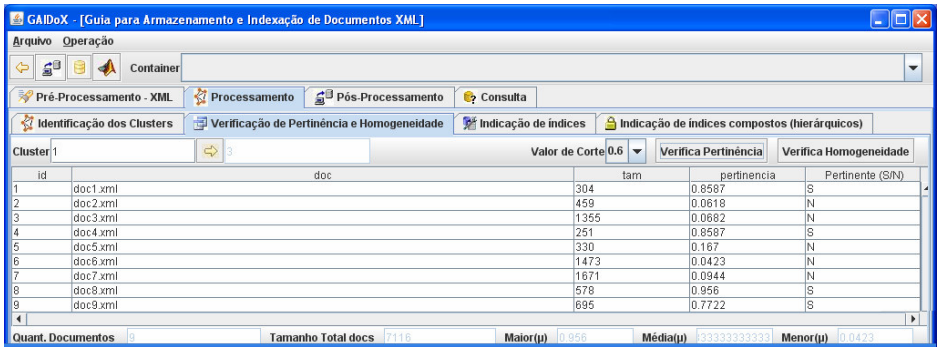


FIG. 5.18 Verificação de pertinência – *cluster* 1

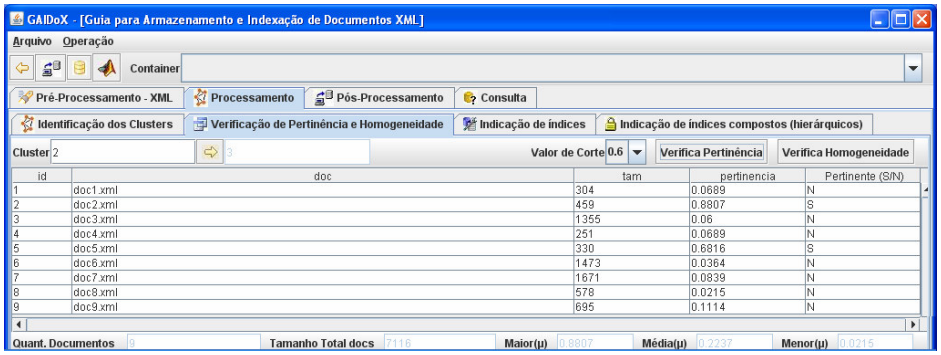


FIG. 5.19 Verificação de pertinência – *cluster* 2

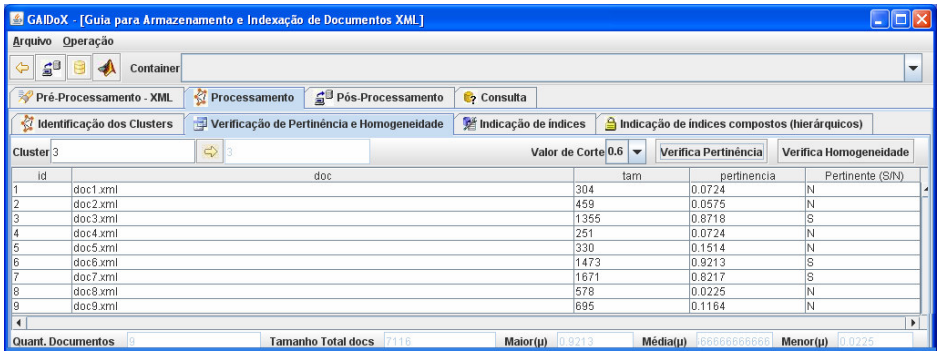


FIG. 5.20 Verificação de pertinência – *cluster* 3

Depois de definida a pertinência dos documentos ao *cluster,* deve ser verificado se os documentos que compõem cada *cluster* formam um *cluster* homogêneo.

VERIFICAÇÃO DE HOMOGENEIDADE

A Verificação de Homogeneidade consiste em identificar se o *cluster* é homogêneo. Para tal, é verificado o maior e o menor grau de pertinência dentre os documentos do *cluster* e segundo um limiar previamente configurado é atestado a homogeneidade ou não do *cluster*. Todos os clusters foram considerados homogêneos diante de um limiar que permitia uma diferença de até 0.35, conforme configuração prévia.

A sub-etapa de sumarização não foi implementada. Neste caso sugerimos a escolha aleatória de alguns documentos que compõem o *cluster* para realizarmos a descrição dos mesmos.

A fig. 5.21 mostra um dos documentos que compõem o *cluster* 1. A fig. 5.22 mostra a descrição deste *cluster*.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <catalogo>
    <title>Informatica para Concursos</title>
    <author>Jadiel Franca</author>
    <isbn>8573934018</isbn>
    <edicao>1 edicao</edicao>
    <sumario>Voce precisa aprender informatica para ontem,
      mas nao sabe a diferenca entre HD e PHD</sumario>
  </catalogo>
```

FIG. 5.21 Documento pertencente ao *cluster* 1 (doc1.xml)

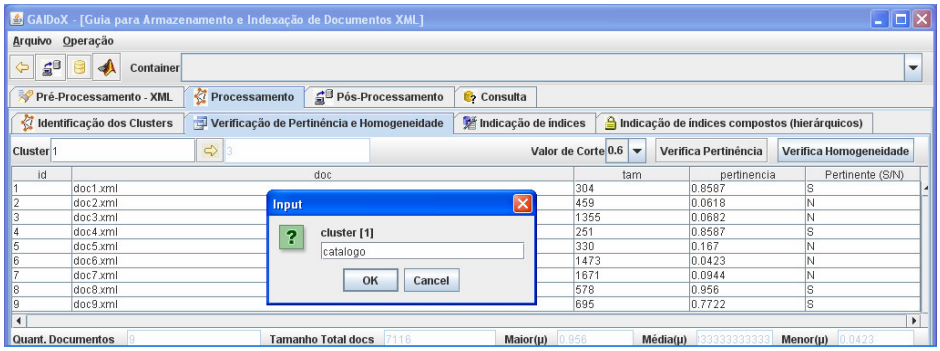


FIG. 5.22 Sumarização do *cluster* 1

Este processo é repetido para os demais *clusters.* A fig. 5.23 mostra um dos documentos que compõem o *cluster* 2. A fig. 5.24 mostra a descrição deste *cluster.*

```
<?xml version="1.0" encoding="UTF-8" ?>
- <curriculum>
  - <dados_pessoais>
      <nome>MIRIAM</nome>
    </dados_pessoais>
  - <formacao>
      <instituicao>IME</instituicao>
      <title>mestrado em ciencias e computacao</title>
    </formacao>
  - <congressos>
      <nome>XIX Congresso SBC/ 99</nome>
      <nome>Microsoft Developer Days 2000</nome>
    </congressos>
  - <experiencia>
      <empresa>XYZ</empresa>
      <cargo>Analista de Sistemas</cargo>
    </experiencia>
  </curriculum>
```

FIG. 5.23 Documento pertencente ao *cluster* 2 (doc2.xml)

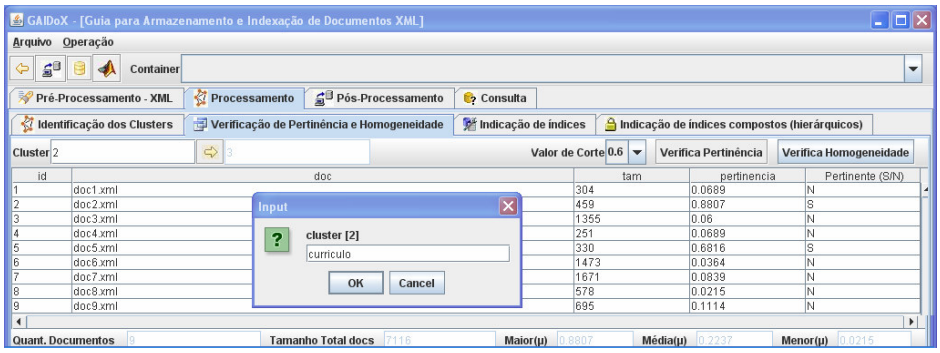


FIG. 5.24 Sumarização do *cluster* 2

A fig. 5.25 mostra um dos documentos que compõem o *cluster* 3. A fig. 5.26 mostra a descrição deste *cluster*.

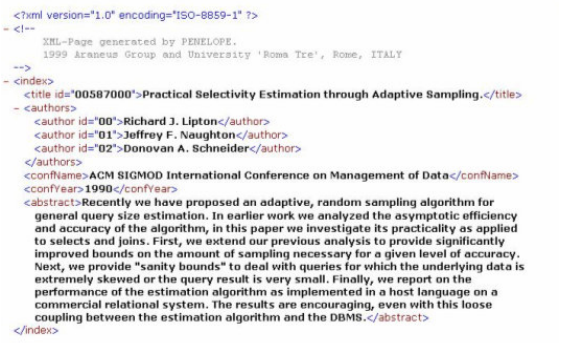


FIG. 5.25 Documento pertencente ao *cluster* 3 (doc3.xml)

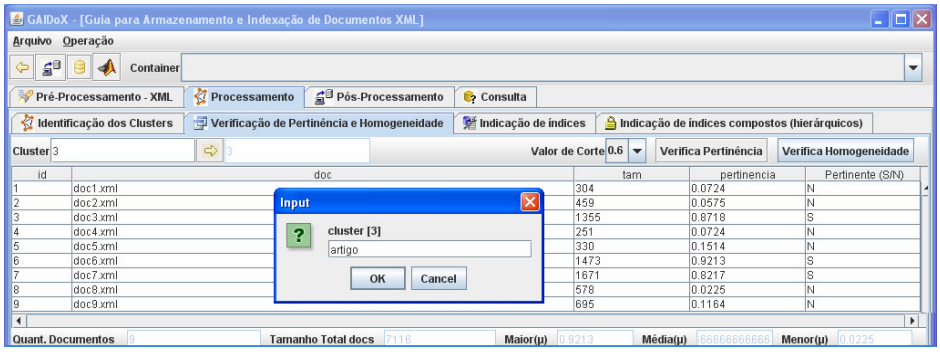


FIG. 5.26 Sumarização do *cluster* 3

Após realizada a avaliação de pertinência e homogeneidade, o armazenamento pode ser realizado. Caso fosse constatada a não homogeneidade destes *clusters* , o processo deveria ser refeito a partir da etapa de processamento, englobando novamente a clusterização com um número maior de *clusters*.

5.2.3.3 INDICAÇÃO DE ÍNDICES SEM CONSIDERAR A HIERARQUIA

Na tela presente na fig. 5.27, são definidos os termos *tags*, que apresentam uma freqüência superior ao suporte estabelecido, seja separadamente, ou na combinação de dois *tags*. Para executar esta funcionalidade foi implementado parte do algoritmo de associação, conhecido como APRIORI. No capítulo de Mineração de dados, a descrição deste algoritmo encontra-se detalhada.

A indicação de índices deve ser realizada por *cluster*. Para cada um, são listados os documentos que representam a transação e suas *tags*, que representam os seus itens. Na tela, estas informações aparecem à esquerda constituindo uma matriz no formato *basket*. A partir da definição de um suporte e do *cluster* em análise, o botão responsável por gerar a associação, já pode ser acionado. Os resultados referentes a um suporte de 90% para conjuntos com 1 elemento (uma única *tag*) e para 2 elementos (combinação de duas *tags*) são exibidos, conforme ilustrado nas figs. 5. 27 e 5.28 respectivamente.

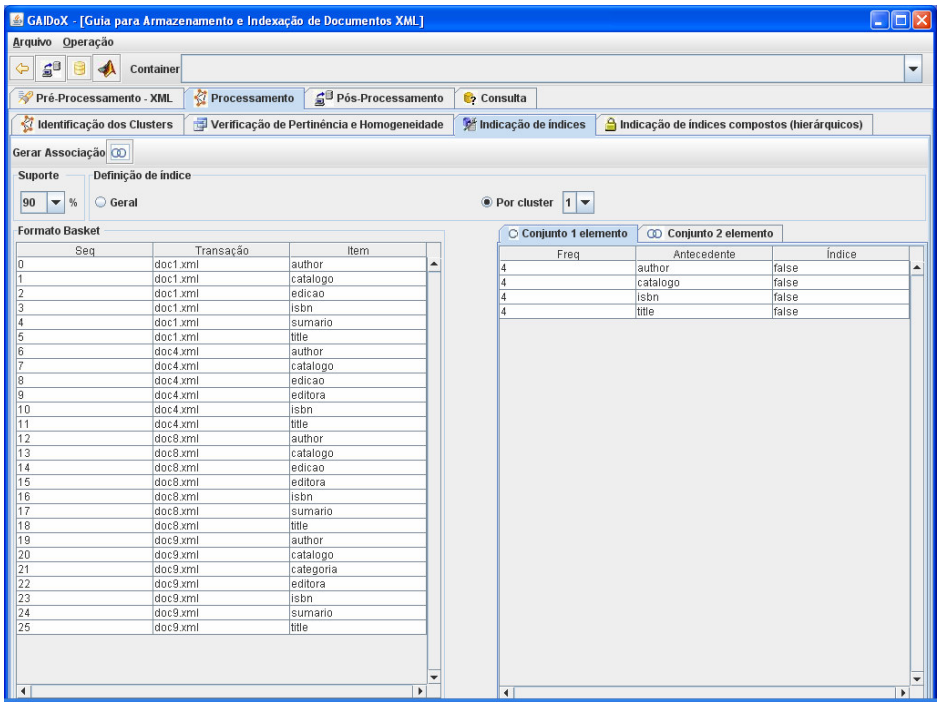


FIG. 5.27 Indicação de índices simples sem considerar a hierarquia

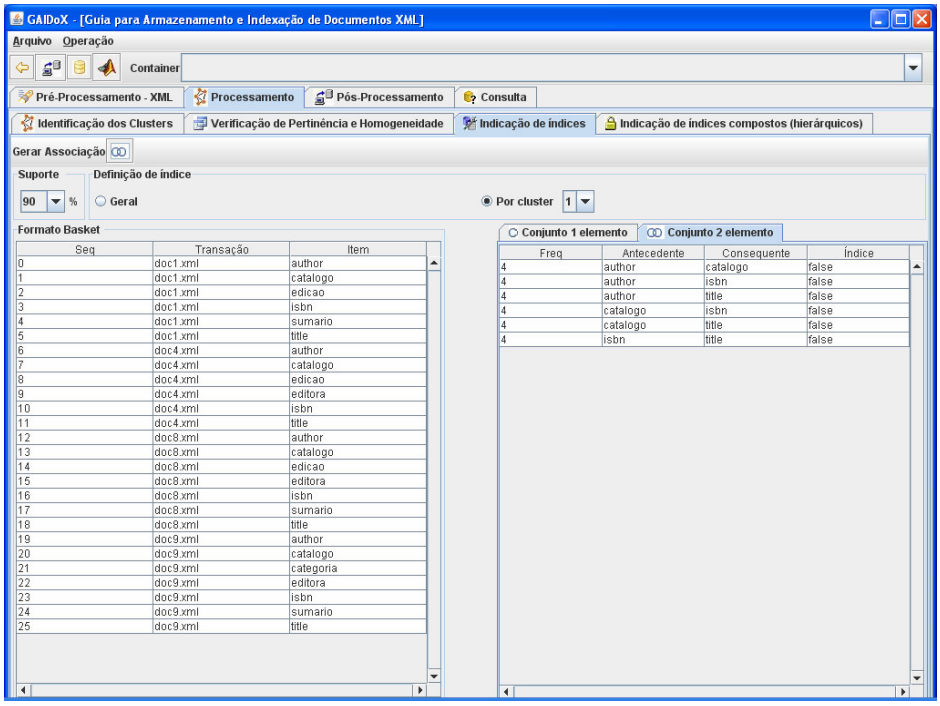


FIG. 5.28 Indicação de índices compostos sem considerar a hierarquia

Este processo deve ser realizado para todos os *clusters*. Vale lembrar que o processo de indicação de índices implementado, apenas exibe os candidatos a índice. A escolha dos termos, *tags*, que deverão ser indexadas, fica a cargo do usuário, assim como a aplicação de estratégias de indexação mais adequadas para cada *tag*.

5.2.3.4 INDICAÇÃO DE ÍNDICES CONSIDERANDO A HIERARQUIA

Segue-se a mesma lógica da indicação de índices sem considerar a hierarquia. A única diferença está nas informações que serão lidas para montagem da matriz no formato *basket*. Neste caso, para cada documento são listados os termos observando a relação de dependência hierárquica *termo-pai* e *termo-filho*. A fig. 5.29 mostra os resultados da indicação de índices para o primeiro *cluster* tendo como suporte um valor de 90%.

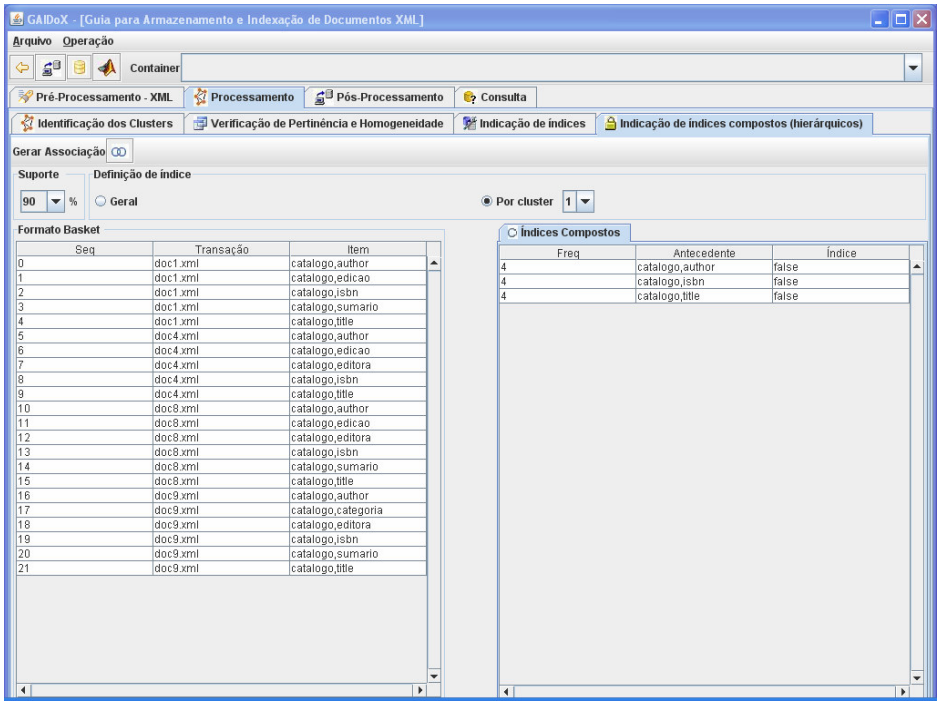


FIG. 5.29 Indicação de índices compostos considerando a hierarquia

5.2.4 MÓDULO DE PÓS-PROCESSAMENTO

Na etapa de Pós-Processamento foi implementada somente a sub-etapa de armazenamento e um módulo a parte para visualização da hierarquia dos termos nos documentos denominada Matriz Hierárquica de Termos.

5.2.4.1 ARMAZENAMENTO

O armazenamento dos documentos XML é realizado através de interface com o Berkeley DB XML.

O processo de armazenamento inicia-se pelo acionamento do botão de armazenamento ilustrado na fig. 5.30.

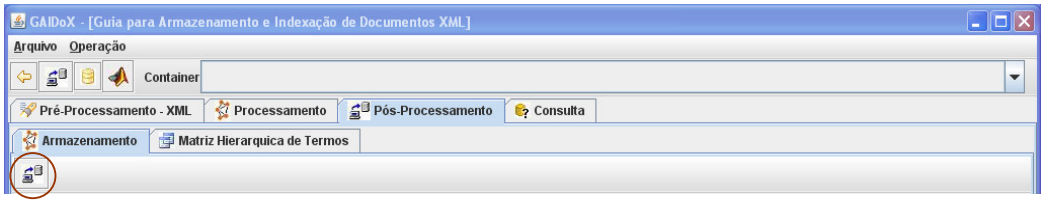


FIG. 5.30 Armazenamento

Antes que o documento possa ser armazenado, é preciso cumprir algumas etapas. Para isto, foi implementado um assistente para realização do armazenamento cujas etapas estão ilustradas na fig. 5.31.

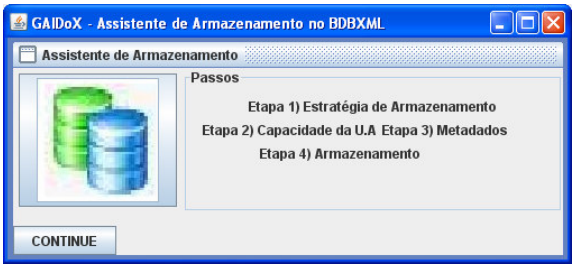


FIG. 5.31 Assistente de armazenamento

Na primeira etapa, denominada Estratégia de Armazenamento, são exibidas informações do *cluster* em análise e é informado se existe estratégia de armazenamento diferenciada para a unidade de armazenamento. A existência ou não de estratégia diferenciada já foi informada anteriormente na tela de configuração. A fig. 5.32 mostra a tela que representa a primeira etapa.

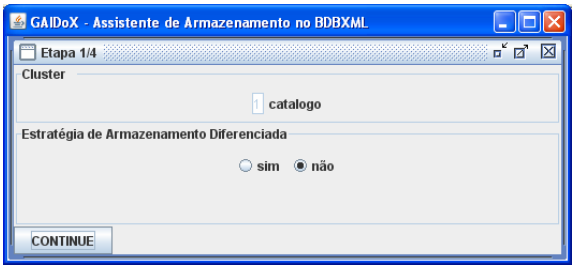


FIG. 5.32 Assistente de armazenamento – Etapa 1/4

Na segunda etapa, Capacidade da Unidade de armazenamento, além das informações do *cluster*, também são exibidas informações como a capacidade da unidade de armazenamento e o tamanho total dos documentos que compõem o *cluster*. Neste caso, é realizada a soma dos tamanhos de todos os documentos que compõem o

cluster. Por fim, é informada a quantidade de unidades de armazenamento necessárias para manter os documentos do *cluster*. Se a capacidade da unidade de armazenamento for maior que a totalização dos tamanho dos documentos do *cluster*, somente uma unidade é utilizada. Caso contrário, a relação entre a soma dos tamanhos dos documentos do *cluster* e a capacidade da UA fornecerão a quantidade de UA necessárias. A fig. 5.33 mostra a tela que representa a segunda etapa.

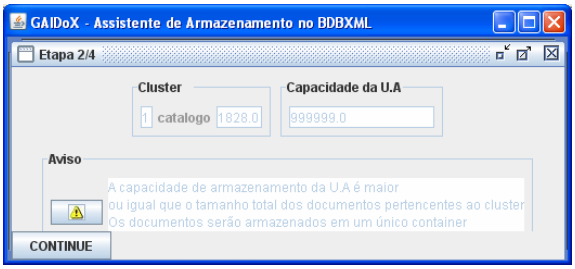


FIG. 5.33 Assistente de armazenamento – Etapa 2/4

Na terceira etapa, denominada Metadados, são definidas que informações serão mantidas como metadados sobre os documentos. Cabe ao usuário decidir os metadados a utilizar. Vale lembrar, que o metadado nome, que guarda o nome do documento, é um metadado *default*. Os demais precisam ser selecionados para serem armazenados. Nesta etapa também são exibidas informações do *cluster* em análise. A fig. 5.34 mostra a tela que representa a terceira etapa.

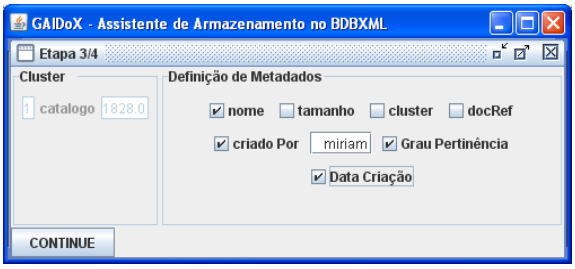


FIG. 5.34 Assistente de armazenamento – Etapa 3/4

Na quarta etapa, denominada Armazenamento, são exibidas informações do *cluster* da UA. A estratégia de armazenamento é definida nesta fase. A fig. 5.35 ilustra a etapa de armazenamento.

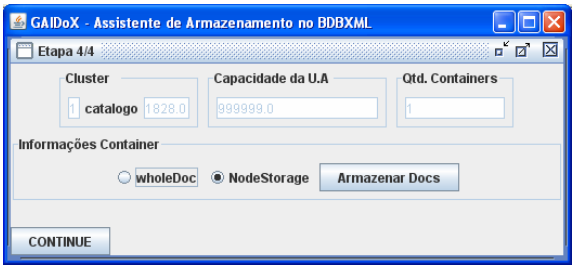


FIG. 5.35 Assistente de armazenamento – Etapa 4/4

Para conclusão desta etapa, é necessário acionar o botão *Armazenar docs*. Após o acionamento do botão, é realizada uma interface com o Berkeley DB XML e todos os documentos do *cluster* em análise são armazenados.

No momento do armazenamento, é necessário definir o nome do *container* que representa a unidade de armazenamento que manterá os documentos, conforme ilustra a fig. 5.36.

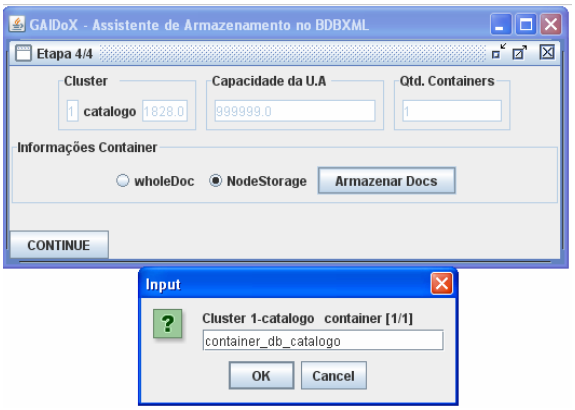


FIG. 5.36 Assistente de armazenamento – Etapa 4/4 – definição do *container*

A fig. 5.37 exibe, a título ilustrativo, o nome dos documentos armazenados na UA, *container_db_catalogo.dbxml.*

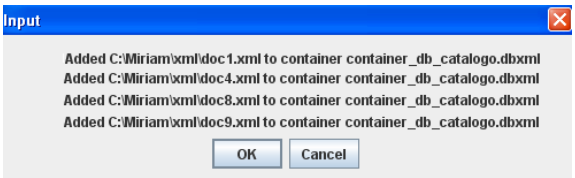


FIG. 5.37 conclusão do armazenamento

O acionamento do botão *continue*, presente na fig. 5.36, dá prosseguimento ao processo para armazenamento dos demais *clusters* nas unidades de armazenamento

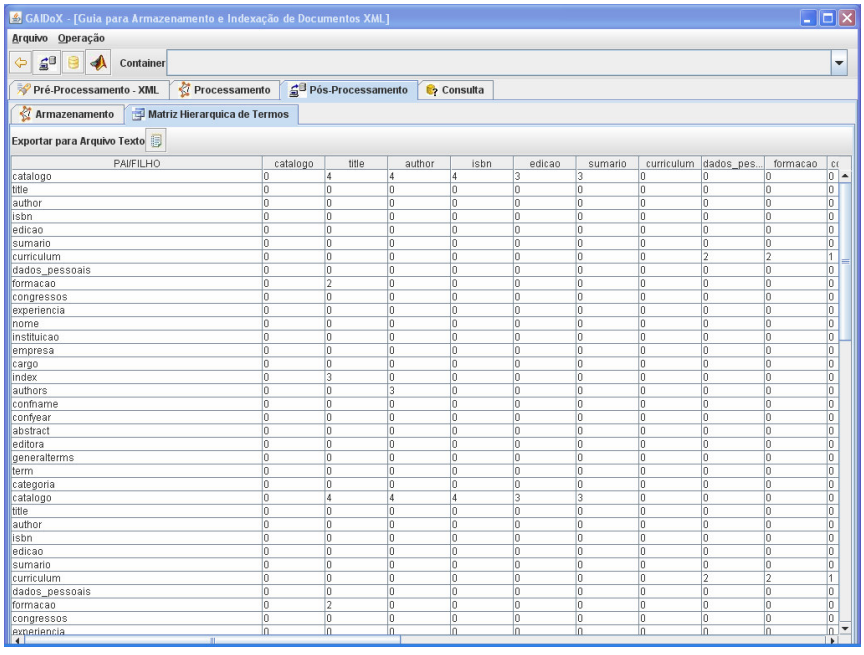
correspondentes. As mesmas etapas são executadas para os demais *clusters* a partir da etapa 1/4 do assistente de armazenamento. Após todos os *clusters* terem sido armazenados, é exibida uma mensagem que informa a conclusão do processo com sucesso.

Com todos os documentos armazenados, partimos para definição de índices. Vale lembrar que todas as decisões tomadas pelo usuário durante o armazenamento são armazenadas em *log*, conforme indicado na arquitetura do protótipo.

5.2.5 MATRIZ HIERÁRQUICA

Para facilitar a visualização de índices que atendem a forma de indexação *edge* do SGBD XML Nativo Berkeley DB XML (recomendada para indexação de *tags* de mesma descrição em diferentes caminhos) foi montada uma matriz que mostra a relação entre as *tags* representadas como termo-pai e termo-filho.

Se em uma mesma coluna houver um valor de ocorrência diferente de zero em diferentes linhas, a combinação destes termos, termo-pai e termo-filho, deve ser indexada segundo esta estratégia, que está descrita na próxima seção. A fig. 5.38 ilustra uma matriz hierárquica de termos.



PAI/FILHO	catalogo	title	author	isbn	edicao	sumario	curriculum	dados_pes...	formacao
catalogo	0	4	4	4	3	3	0	0	0
title	0	0	0	0	0	0	0	0	0
author	0	0	0	0	0	0	0	0	0
isbn	0	0	0	0	0	0	0	0	0
edicao	0	0	0	0	0	0	0	0	0
sumario	0	0	0	0	0	0	0	0	0
curriculum	0	0	0	0	0	0	0	2	2
dados_pessoais	0	0	0	0	0	0	0	0	0
formacao	0	2	0	0	0	0	0	0	0
congressos	0	0	0	0	0	0	0	0	0
experiencia	0	0	0	0	0	0	0	0	0
nome	0	0	0	0	0	0	0	0	0
instituicao	0	0	0	0	0	0	0	0	0
empresa	0	0	0	0	0	0	0	0	0
cargo	0	0	0	0	0	0	0	0	0
index	0	3	0	0	0	0	0	0	0
authors	0	0	3	0	0	0	0	0	0
confname	0	0	0	0	0	0	0	0	0
confyear	0	0	0	0	0	0	0	0	0
abstract	0	0	0	0	0	0	0	0	0
editora	0	0	0	0	0	0	0	0	0
generalterms	0	0	0	0	0	0	0	0	0
term	0	0	0	0	0	0	0	0	0
categoria	0	0	0	0	0	0	0	0	0
catalogo	0	4	4	4	3	3	0	0	0
title	0	0	0	0	0	0	0	0	0
author	0	0	0	0	0	0	0	0	0
isbn	0	0	0	0	0	0	0	0	0
edicao	0	0	0	0	0	0	0	0	0
sumario	0	0	0	0	0	0	0	0	0
curriculum	0	0	0	0	0	0	0	2	2
dados_pessoais	0	0	0	0	0	0	0	0	0
formacao	0	2	0	0	0	0	0	0	0
congressos	0	0	0	0	0	0	0	0	0
experiencia	0	0	0	0	0	0	0	0	0

FIG. 5.38 Matriz Hierárquica de termos

5.2.6 MÓDULO DE CONSULTA

Neste módulo é possível definir consultas, executá-las e visualizar seus resultados. As consultas são processadas e seus resultados retornados, a partir da conexão com o SGBD Berkeley DB XML.

As figs. 5.39 e 5.40 mostram a tela de consulta e os resultados retornados para as consultas especificadas.

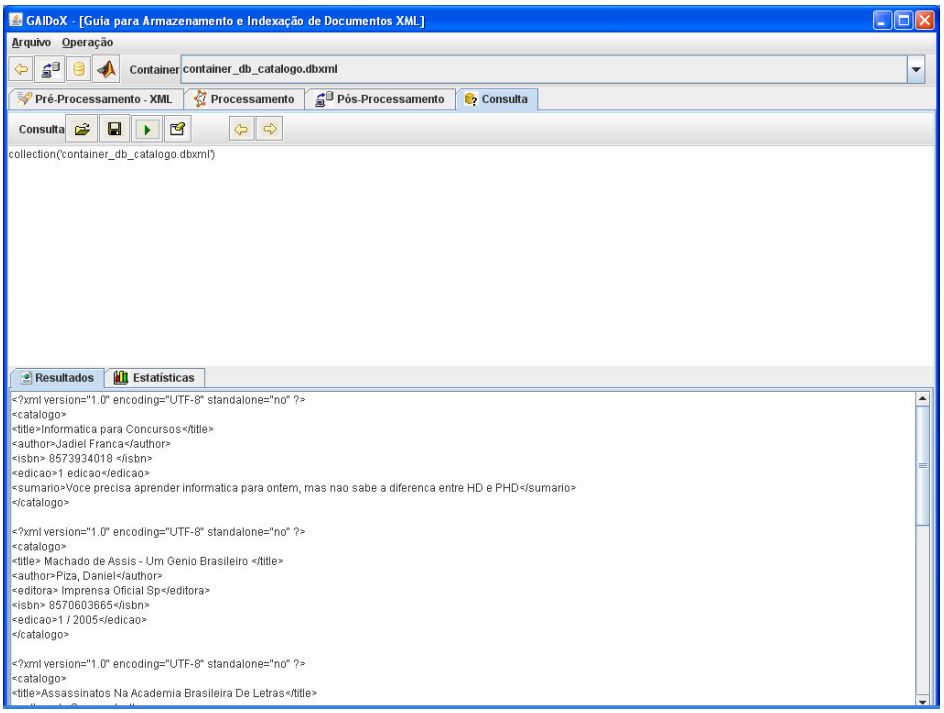


FIG. 5.39 Módulo de Consulta - consulta I

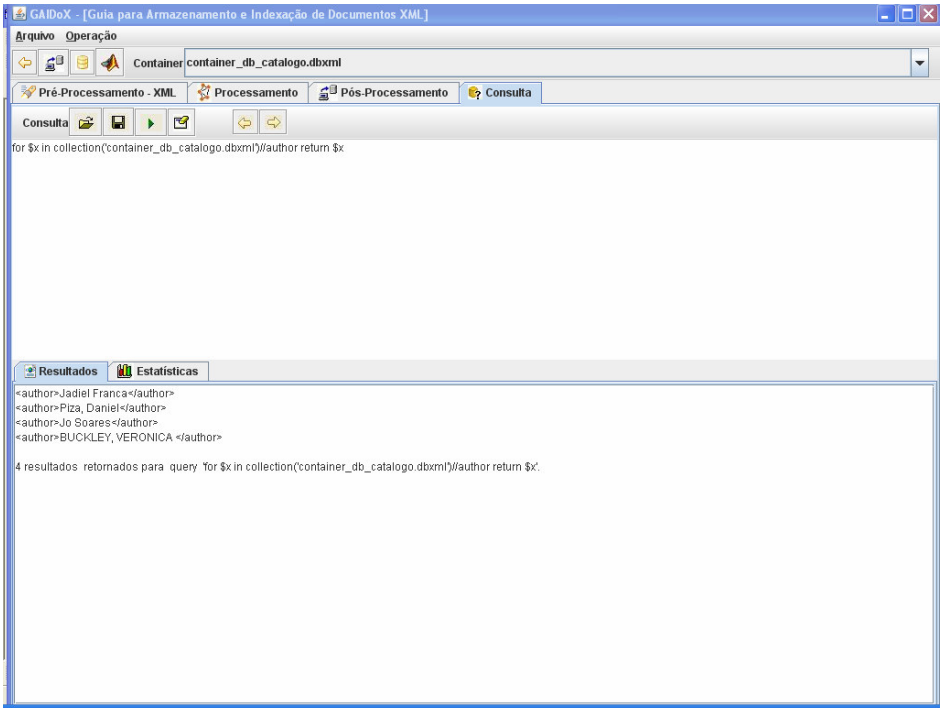


FIG. 5.40 Módulo de Consulta - consulta II

Neste módulo também foi implementada a seleção de consultas que simula o uso de *views*, desde que previamente cadastradas em um documento XML reservado para o armazenamento destas consultas. A fig. 5.41 mostra o documento XML *consultas.xml* , responsável por guardar as consultas pré-cadastradas.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <consultas>
  - <container nome="container_db_artigos.dbxml">
      <consulta id="1">collection('container_db_artigos.dbxml')//autor</consulta>
      <consulta id="2">collection('container_db_artigos.dbxml')//componente</consulta>
      <consulta id="3">collection('container_db_artigos.dbxml')//orientadores</consulta>
      <consulta id="4">for $x in collection('container_db_artigos.dbxml')//autor return $x</consulta>
    </container>
  </consultas>
```

FIG. 5.41 Módulo de consulta - views

As consultas pré-cadastradas são selecionadas para execução através dos botões que exibem as consultas cadastradas. As consultas são exibidas através dos botões de navegação exibidos na fig. 5.42.

FIG. 5.42 Navegação entre as consultas da *view*

Através da apresentação do GAIDoX, que se encontra implementado na forma de um protótipo, foi possível verificar a viabilidade da estratégia proposta. Na próxima

seção, discutiremos as características e recursos do Berkeley DB XML.

5.3 DETALHAMENTO DO BERKELEY DB XML

Tendo em vista que no capítulo 2 já foram descritas as principais características do SGBD XML Nativo Berkeley DB XML, focamos nesta seção o detalhamento deste SGBD, no que diz respeito às formas de armazenamento empregadas e às opções de indexação existentes, que são foco do nosso trabalho.

Quanto às Formas de Armazenamento, são suportadas duas formas distintas: *wholedoc* e *node* (SLEEPYCAT, 2005);(BRIAN, 2006). Na forma de armazenamento *wholedoc,* os documentos são armazenados em sua forma íntegra. Na forma de armazenamento *node,* os documentos são armazenados de forma fragmentada. Assim, as unidades de armazenamento que tenham sido criadas utilizando-se da forma de armazenamento *node,* tornam a recuperação do documento mais rápida, em se tratando de consultas que buscam parte do documento, enquanto que as unidades de armazenamento criadas a partir da estratégia *whole* somente tornam-se mais eficientes quando os documentos são constantemente recuperados de forma íntegra, já que não há reconstrução do documento (BRIAN, 2006).

Quanto às Opções de Indexação, o BDB XML disponibiliza vários tipos de índices. Os índices podem ser definidos sobre a estrutura dos documentos ou ainda sobre metadados (SLEEPYCAT, 2005); (BRIAN, 2006). O Berkeley DB XML gera e indexa automaticamente o metadado `name`, que guarda o nome do documento, sendo para os demais índices necessário explicitar sua necessidade através da criação manual, segundo as opções disponibilizadas que serão descritas a seguir.

Conforme explicitado em (SLEEPYCAT, 2005); (BRIAN, 2006), os índices devem ser criados observando os tipos de índices especificados abaixo:

[uniqueness] – {path type} – {node type} – {key type} – {sytax type}

UNIQUENESS

Indica que o valor do índice deve ser único por *container*. Somente deve ser utilizado para elementos em que não há repetições de valor. Por *default*, um índice

nunca será *uniqueness*, com exceção para o metadado `name`.

PATHTYPES

Identifica o tipo de *path* no qual se encontra o elemento que se espera indexar. Há dois tipos de indexação possíveis: *node* e *edge*. O tipo *node* indexa um único nó no *path*, não sendo adequado utilizar *node* para atributos que aparecem repetidamente em diferentes contextos. O tipo *edge*, é recomendável quando existe dois ou mais nós envolvendo um elemento de mesma descrição; isto facilita a execução da *query* fazendo com que não haja necessidade de se examinar dois índices.

NODE TYPE

Esta opção de índice especifica a categoria de nó a ser indexado. Há três tipos de nós possíveis: *elementos, atributos* ou *metadados*. O tipo *elemento* é utilizado para melhorar consultas que testam o valor dos elementos; o tipo *atributo* é utilizado para melhorar consultas que testam o valor dos atributos; o tipo *metadados* é encontrado nas informações de metadado dos documentos, se especificado.

KEY TYPE

É um tipo de índice que otimiza a indexação de um certo tipo de consulta a ser testada. Há três tipos *: equality, presence e substring*. O tipo *equality* melhora da performance da consulta quando se procura nós com valores específicos, utilizado em expressões de igualdade; o tipo *presence* melhora a performance da consulta quando se procura nós que podem existir ou não no documento; o tipo *substring* melhora a performance da consulta quando se procura nós cujo valor contém uma dada substring. A *Xquery* utiliza *contains*() para funções de *substring*.

SYNTAX TYPE

Determina o formato dos valores esperados para cada índice. É preciso considerar que diferentes comparações e funções têm diferentes implementações dependendo do tipo de dados, da consulta ou argumento. Um dado é comparado com outros dados com diferentes conversões específicas para o dado sendo consultado (BRIAN, 2006, p.6). A tabela 5.1 mostra uma lista de tipos de dados permitidos.

TAB. 5.1 - Tipos de dados no BDB XML

None	Datetime	Float	hexBinary	YearMonthDuration
anyURI	dayTimeDuration	gDay	Notation	untypedAtomic
Base64Binary	Decimal	gMonth	Qname	
Boolean	Doublé	Gyear	String	
Date	Duration	gYearMonth	Time	

Para avaliação dos índices criados, o BDB XML provê uma aplicação chamada *QPLAN* (SLEEPYCAT, 2005), que determina o efeito dos índices sobre as consultas executadas, e neste caso pode ser verificado se os índices estão tendo efeito sobre cada consulta e que caminhos estão sendo percorridos para execução das mesmas, possibilitando a otimização da consulta e criação de novos índices.

6 ESTUDO DE CASO

Nesta seção, descreve-se o estudo de caso realizado para avaliar a estratégia proposta. Neste estudo de caso, as etapas de clusterização, armazenamento e indexação foram os principais alvos de avaliação. Em um primeiro momento, buscou-se avaliar a etapa de clusterização, verificando se, a partir de um acervo conhecido de documentos heterogêneos provenientes de diferentes fontes, seria possível gerar *clusters* com documentos similares. Em um segundo momento, o objetivo foi a estratégia como um todo, verificando se, através de consultas diversificadas sobre as unidades de armazenamento criadas e indexadas, a estratégia proposta apresentaria um diferencial em termos de tempo de resposta. São apresentados os resultados auferidos nos testes realizados fazendo-se uma comparação entre a forma de armazenamento sugerida pela estratégia proposta e as outras duas formas de armazenamento que serão descritas oportunamente.

Esta seção está organizada como descrito a seguir. Inicialmente, detalha-se a *construção do acervo*, descrevendo as fontes de documentos XML utilizados para o estudo de caso. Na sub-seção seguinte, discute-se a formação dos grupos a partir desta tarefa sobre o acervo selecionado. Na sub-seção *6.3* (consultas), descreve-se as consultas que foram selecionadas para execução dos testes, assim como seus objetivos. Na sub-seção 6.4 (plano de testes), descreve-se as diferentes estratégias em que se baseia o estudo de caso, a metodologia e o ambiente de testes utilizados para avaliar o tempo de execução das consultas. Na sub-seção 6.5 (resultados e gráficos), discute-se os resultados auferidos nos testes realizados, comparando-se o resultado das diferentes estratégias. Na sub-seção 6.6 (considerações) são relatados alguns problemas encontrados na utilização do SGBD XML Nativo escolhido e realizado uma análise geral sobre os resultados esperados e os resultados alcançados.

6.1 CONSTRUÇÃO DO ACERVO

Como já foi dito, a estratégia proposta está voltada para guiar o armazenamento dos documentos similares entre si, esperando que haja um melhor tempo de resposta na recuperação dos mesmos. Nesta estratégia, os documentos não precisam estar associados a quaisquer estruturas de validação (*XMLSCHEMA* ou *DTD)*. Através do conjunto de passos da estratégia, é possível identificar grupos de documentos similares tendo como base apenas as *tags* presentes na estrutura de cada um dos documentos XML em análise. Sendo assim, quaisquer documentos bem formados podem ser submetidos à estratégia proposta.

Para realizar este estudo de caso, de forma a avaliar a estratégia proposta, foi preciso reunir um grande volume de documentos, pois assim, os problemas de desempenho tornam-se mais visíveis. Além disso, preocupou-se também com a construção de um acervo heterogêneo, com documentos de diferentes fontes, que variassem significativamente em termos estruturais, porém que guardassem alguma similaridade. Deste modo, foi possível avaliar a etapa de clusterização, permitindo antever a formação dos grupos de documentos similares, e ainda, avaliar as etapas de armazenamento e indexação através de consultas que visam recuperar documentos dos diversos grupos de documentos.

Assim sendo, optou-se por trabalhar com três conjuntos de documentos heterogêneos, sendo que um destes conjuntos, artigos completos, é formado por documentos gerados automaticamente por um utilitário. O segundo conjunto, resumos de artigos, contém documentos reais, provenientes de uma revista eletrônica, e o terceiro grupo, currículos, contém documentos criados manualmente. Cada um dos grupos de documentos são descritos em mais detalhes a seguir.

ARTIGOS COMPLETOS

A montagem do acervo intitulado *artigos completos*, foi realizada a partir de documentos XML gerados automaticamente pelo utilitário do *benchmark* Xbench toxgene[7]. Este utilitário gera documentos com informações aleatórias, e oferece várias opções quanto ao tipo de documento que está sendo gerado e o tamanho do acervo desejado.

[7] http://www.cs.toronto.edu/tox/toxgene/

O *benchmark* Xbench trabalha com diferentes conjuntos de dados, levando-se em conta a combinação de quatro características. São elas: documentos centrados em texto (CT), documentos centrados em dados (CD), documento único (UD), e múltiplos documentos (MD). Para este estudo de caso, escolheu-se a opção de documentos centrados em texto em múltiplos documentos (CT/MD). Nesta opção, os documentos gerados são artigos completos. Neste tipo de acervo, os documentos gerados, apesar de não estarem associados a uma estrutura de validação, seguem uma estrutura pré-definida que está ilustrada na fig. 6.1.

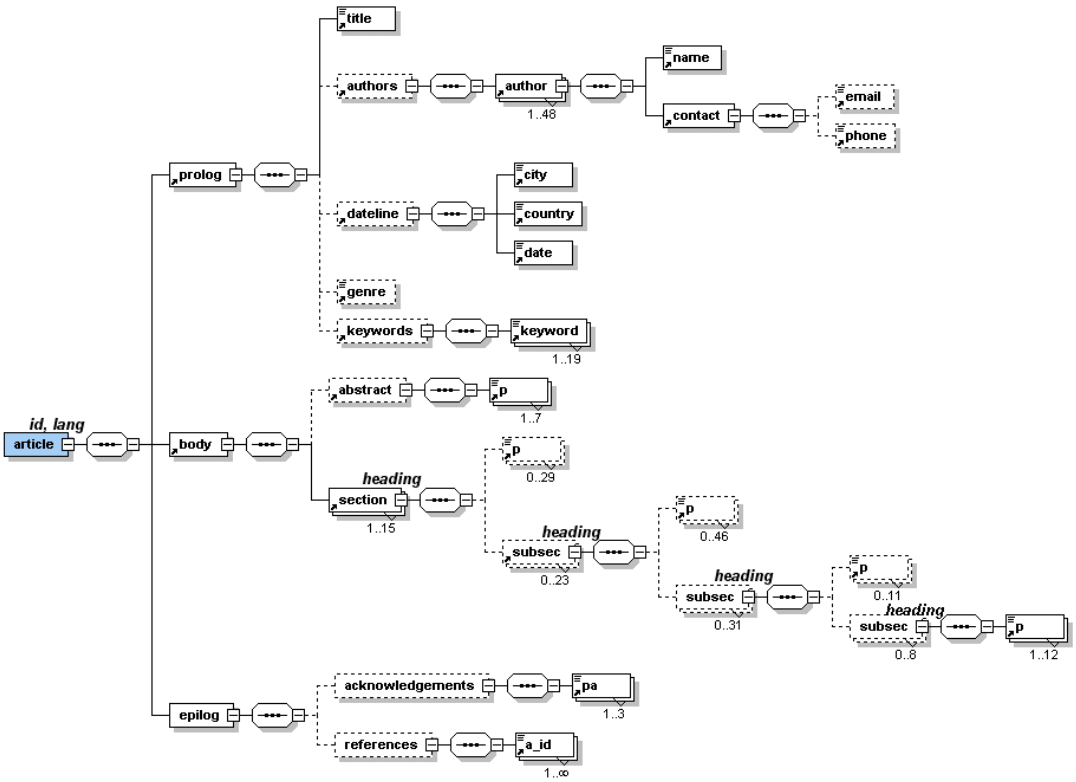


FIG. 6.1 Estrutura Xbench

Do acervo gerado, que originalmente continha 1Gb e 2666 documentos, foram selecionados 405 documentos, totalizando 250Mb em tamanho. A fig. 6.2 mostra um exemplo de documento XML deste tipo de acervo.

FIG. 6.2 documentos XML do acervo Xbench

RESUMO DE ARTIGOS

A montagem do acervo, intitulado *resumos de artigos*, baseou-se em documentos XML originários da revista eletrônica Sigmod Record[8], que mantém artigos reais, catálogos de artigos e resumos de artigos no formato XML. Escolheu-se trabalhar com resumos de artigos.

Como o SGBD XML Nativo escolhido não provê suporte para o uso de DTD, foi necessário retirar esta referência de cada um dos documentos originados da Sigmod Record. A fig. 6.3 mostra o DTD deste conjunto de documentos.

[8] http://www.sigmod.org/record/xml/

```
<!ENTITY % carSet SYSTEM 'CarSet.cfg'>
%carSet;
 <!ELEMENT IndexTermsPage
(title,authors,confName,confYear,volume,number,initPage,endPage,fullText,abstract,generalTerms,categoryAndSubjectDescriptors)>
 <!ELEMENT title (#PCDATA)>
 <!ATTLIST title id  CDATA  #IMPLIED>
 <!ELEMENT authors (author)*>
 <!ELEMENT author (#PCDATA)>
 <!ATTLIST author id  CDATA  #IMPLIED>
 <!ELEMENT confName (#PCDATA)>
 <!ELEMENT confYear (#PCDATA)>
 <!ELEMENT volume (#PCDATA)>
 <!ELEMENT number (#PCDATA)>
 <!ELEMENT initPage (#PCDATA)>
 <!ELEMENT endPage (#PCDATA)>
 <!ELEMENT fullText (size)?>
 <!ELEMENT size (#PCDATA)>
 <!ENTITY % Xlink
   " xml:link CDATA #FIXED 'simple'
    href  CDATA  #IMPLIED
    inline (true | false) #FIXED 'true'
   "
 >
 <!ATTLIST fullText   %Xlink;>
 <!ELEMENT abstract (#PCDATA)>
 <!ELEMENT generalTerms (term)*>
 <!ELEMENT term (#PCDATA)>
 <!ELEMENT categoryAndSubjectDescriptors (categoryAndSubjectDescriptorsTuple)*>
 <!ELEMENT categoryAndSubjectDescriptorsTuple (category,content)>
 <!ELEMENT category (#PCDATA)>
 <!ELEMENT content (#PCDATA)>
```

FIG. 6.3 DTD Sigmod

Este acervo contém em sua maioria, documentos bem similares com poucas variações em termos estruturais, já que seguem uma estrutura pré-definida. As variações encontradas se restringem ao uso de *tags* opcionais e a diferentes cardinalidades no uso de algumas *tags.*

O acervo *resumos de artigos* conta com 911 documentos, totalizando 1,96 Mb em tamanho. A fig. 6.4 mostra um exemplo de documento XML deste acervo.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--   XML-Page generated by PENELOPE.
   1999 Araneus Group and University 'Roma Tre', Rome, ITALY
   -->
- <IndexTermsPage>
    <title id="00585000">Query Flocks: A Generalization of Association-Rule Mining.</title>
  - <authors>
      <author id="00">Dick Tsur</author>
      <author id="01">Jeffrey D. Ullman</author>
      <author id="02">Serge Abiteboul</author>
      <author id="03">Chris Clifton</author>
      <author id="04">Rajeev Motwani</author>
      <author id="05">Svetlozar Nestorov</author>
      <author id="06">Arnon Rosenthal</author>
    </authors>
    <confName>ACM SIGMOD International Conference on Management of Data</confName>
    <confYear>1998</confYear>
    <volume>27</volume>
    <number>2</number>
    <initPage>1</initPage>
    <endPage>12</endPage>
  - <fullText>
      <size />
    </fullText>
    <abstract>Association-rule mining has proved a highly successful technique for extracting useful information from very large
      databases. This success is attributed not only to the appropriateness of the objectives, but to the fact that a number of new
      query-optimization ideas, such as the "a-priori" trick, make association-rule mining run much faster than might be expected.
      In this paper we see that the same tricks can be extended to a much more general context, allowing efficient mining of very
      large databases for many different kinds of patterns. The general idea, called "query flocks," is a generate-and-test model for
      data-mining problems. We show how the idea can be used either in a general-purpose mining system or in a next generation
      of conventional query optimizers.</abstract>
    <generalTerms />
    <categoryAndSubjectDescriptors />
</IndexTermsPage>
```

FIG.6.4 Exemplo documento XML do acervo *resumo de artigos*

CURRÍCULOS

A montagem do acervo intitulado *currículos*, reuniu documentos XML

originários de três fontes. Uma delas foi a base de currículos Lattes[9] , do CNPq, que apesar de seguir um esquema pré-definido, costuma apresentar documentos com variabilidade estrutural significativa. Uma segunda fonte, foi um curso de pós-graduação que gerou uma coleção de currículos dos alunos criados por cada aluno individualmente, sem seguir uma estrutura pré-definida. Uma terceira fonte foi a própria Web, onde buscou-se por currículos profissionais que foram transformados em XML.

O acervo conta com 15 documentos, totalizando 58,9 kb em tamanho. A fig. 6.5 mostra um exemplo de documento XML deste tipo de acervo.

Apesar de ser um acervo pequeno, é o acervo que apresenta maior diversidade entre suas estruturas.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <curriculum>
  - <dados_pessoais>
      <nome>Miriam Oliveira dos Santos</nome>
      <nascimento>11/11/1977</nascimento>
      <nacionalidade>Brasileira</nacionalidade>
      <naturalidade>Rio de Janeiro</naturalidade>
      <estado_civil>Solteira</estado_civil>
    </dados_pessoais>
  - <formacao>
    - <mestrado>
        <instituicao>Instituto Militar de Engenharia - IME</instituicao>
        <curso>Mestrado em Ciencias e Computacao (cursando)</curso>
      - <tese>
          <title>Armazenamento e Recuperação de documentos XML heterogêneos</title>
          <subtitle>Avaliando Alternativas para armazenamento em bancos de dados XML Nativos</subtitle>
        </tese>
        <linha_Pesquisa>Banco de Dados</linha_Pesquisa>
      </mestrado>
    - <graduacao>
        <instituicao>Pontificia Universidade Catolica do Rio de Janeiro - PUC/RJ</instituicao>
        <curso>Tecnologo em Processamento de Dados (1997-2000)</curso>
      </graduacao>
    </formacao>
  </curriculum>
```

FIG. 6.5 Exemplo de documento XML do acervo *currículo*

Em resumo, o acervo de documentos utilizado neste estudo de caso, apresenta documentos que variam tanto em tamanho, quanto em estrutura. No acervo de artigos completos encontram-se documentos muito grandes, e no acervo dos currículos, encontram-se documentos bem pequenos. A estratégia apresentada no capítulo 4, proposta, foi elaborada para atender acervos de quaisquer tamanhos e estruturas.

A tabela 6.1 totaliza a quantidade de documentos do acervo utilizado neste estudo de caso, seus tamanhos e suas origens.

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

TAB. 6.1 - Acervo XML

Tipo	Origem	Qtd	Tamanho
resumos de artigos	Sigmod record	911	1,96Mb
artigos completos	Xbench	405	250Mb
curriculos	curriculos Lattes Web Outros	15	58,9Kb
	Total	**1331**	**252Mb**

Na próxima seção, falaremos dos resultados obtidos na etapa de clusterização a qual estes documentos foram submetidos.

6.2 RESULTADOS OBTIDOS NA ETAPA DE CLUSTERIZAÇÃO

Conhecendo o acervo preparado, foi possível verificar se o algoritmo de clusterização utilizado se comportaria de forma correta, distribuindo os documentos corretamente por um dos seguintes grupos: artigos completos, resumo de artigos ou currículo.

Para os acervos, artigos completos e resumo de artigos, nenhum tratamento foi empregado, já que as *tags* que formam estes documentos já apresentavam homogeneidade em sua descrição. Para os documentos *currículos*, foi empregado um tratamento para que as *tags* que representavam a mesma informação, mesmo que descritas de formas distintas, fossem representadas por uma única descrição.

Após computada a freqüência com que as *tags* apareciam em cada documento, foi montada uma matriz de freqüência de dados para ser submetida ao algoritmo de clusterização.

Uma característica do algoritmo é atribuir diferentes graus de pertinência dos documentos aos *clusters* formados. É claro que, como esperado, o acervo de resumo de artigos e artigos completos têm características comuns, mas os graus de pertinência tornam-se mais ou menos significativos dependendo da quantidade de *clusters* empregada. Optamos por gerar três grupos de documentos, e de acordo com as etapas propostas, nos coube definir após ter os resultados da clusterização, a que grupos o documento seria considerado pertinente. A tabela 6.2 mostra o maior e menor graus de pertinência obtidos em cada grupo formado.

TAB. 6.2 – Resultados da Clusterização

	Graus de Pertinência	
Grupos	**Maior**	**Menor**
resumos de artigos	0,9692	0,8594
artigos completos	0,9355	0,8503
Currículos	0,6833	0,4615

O primeiro *cluster* agrupou documentos pertencentes a artigos completos, o segundo *cluster* agrupou documentos pertencentes a resumos de artigos, e o último *cluster* agrupou currículos. Esta informação é de suma importância para o armazenamento e posteriormente à aplicação de consultas.

Independente da quantidade de documentos empregados, o algoritmo de clusterização se comportou corretamente, mantendo documentos com características em comum no mesmo grupo.

A partir dos grupos definidos, estes documentos foram armazenados. Na próxima seção, detalharemos as consultas que foram utilizadas para avaliação da nossa estratégia.

6.3 CONSULTAS

Para este trabalho foram selecionadas algumas consultas com base nas consultas[10] disponibilizadas pelo *benchmark* Xbench. O objetivo aqui foi selecionar consultas que pudessem avaliar de forma diversificada a recuperação dos documentos armazenados. Buscou-se eleger consultas que recuperam documentos que pudessem estar armazenados em uma única UA (com documentos similares agrupados).

As consultas selecionadas que atendem ao grupo de documentos gerado pelo Xbench foram adaptadas para atender aos demais grupos de documentos.

A seguir são apresentadas as consultas selecionadas, por grupo de documentos e os índices empregados. Independente da característica da consulta, priorizou-se o emprego de índices sobre os atributos ou elementos que aparecem na *seleção* da consulta, ou que estão sendo testados segundo alguma condição.

Quatro funcionalidades estão sendo avaliadas pelas consultas

[10] http://db.uwaterloo.ca/ddbms/projects/xbench/Specification.html

selecionadas. São elas: consultas por valores exatos, ordenação, verificação de dados irregulares, referências e junções. Na tabela 6.3 são listadas as consultas selecionadas e os índices empregados em cada uma delas.

TAB. 6.3 - Consultas utilizadas nos testes e seus respectivos índices

Consultas para o grupo Artigos Completos	**Indices utilizados**
1. for $a in collection("container")/article[@id="1"] return $art/prolog/title	id (node-attribute-equality-string)
2. for $prolog in collection("container ")/article/prolog where $prolog/authors/author/name="Ben Yang" return $prolog/title	Name(node-element-equality-string)
3. for $a in collection("container.dbxml")/article/prolog where $a/dateline/country="Canada" order by $a/dateline/date return {$a/title} {$a/dateline/date}	country(node-element-equality-string)
4. collection("container")/article/prolog[not(genre)]/title	Genre(node-element-presence-none)
5. collection("container.dbxml")/article/prolog/genre/title	Genre(node-element-presence-none)
6. for $a in collection("container")/article[@id="7"]/epilog/references/a_id, $b in collection("container")/article where $a = $b/@id return {$b/prolog/title}	id (node-attribute-equality-string)
Consultas para o grupo Resumos de Artigos	**Indices utilizados**
1. for $art in collection("container ")/IndexTermsPage/title[@id="00585"] return $art/title	id (node-attribute-equality-string)
2. for $art in collection("container")/IndexTermsPage where $art/authors/author="Dick Tsur" return $art/author	author(node-element-equality-string)
3. for $a in collection("container")/IndexTermsPage where $a/confyear="1993" order by $a/title return {$a/title}	confyear(node-element-equality-string)
4. collection("container")/IndexTermsPage/categoryAndSubjectDescriptors [not(categoryAndSubjectDescriptorsTuple)]	categoryAndSubjectDescriptorsTuple (node-element-presence-none)
5. collection("container")/IndexTermsPage/categoryAndSubjectDescriptors /categoryAndSubjectDescriptorsTuple/category	categoryAndSubjectDescriptorsTuple (node-element-presence-none)
6. for $a in collection("container")/IndexTermsPage/title[@id="00585"]/authors/author, $b in collection("container")/IndexTermsPage/authors/author where $a = $b return {$b/IndexTermsPage/title}	id (node-attribute-equality-string)
Consultas para o grupo Currículos	**Indices utilizados**
1. for $cur in collection("container") where contains($cur//instituicao,"IME") return $cur//dados_pessoais	instituicao(node-element-substring-string)
2. collection("container ")[//doutorado]//nome	doutorado(node-element-presence-none)
3. collection("container ")[//cidade="Rio de Janeiro"]//experiencia_profissional	cidade(node-element-equality-string)
4. for $tese in collection("container")//tese where contains ($tese/title,"XML") return $tese	title(node-element-substring-string)
5. collection("container")[//instituicao/nome="IME"]	Nome(node-element-equality-string)
6. for $cur in collection("container") where contains ($cur//dados_pessoais/nome,"ana") order by $cur/dados_pessoais/nome return $cur	Dados_pessoais.nome(edge-element-equality-string)

Em anexo, encontram-se os planos de execução correspondentes a cada consulta, sendo possível identificar os índices utilizados.

UTILIZAÇÃO DE ÍNDICES (GRUPO ARTIGOS COMPLETOS)

As consultas 1 e 6 se utilizam do índice criado para o atributo `id`. O índice empregado para este atributo *node-attribute-equality-string* atende as características de consultas de busca por valores exatos. Embora o atributo `id` tenha um valor numérico, o mesmo foi descrito na consulta como *string*, portanto na criação do índice, o tipo de dados empregados atende a este tipo de dados.

A segunda consulta se utiliza do índice criado para o elemento `name`. O índice empregado para este atributo *node-element-equality-string* atende as características da consulta de busca por valores exatos.

A terceira consulta se utiliza do índice criado para o elemento `country`. O índice empregado para este elemento *node-element-equality-string* atende as características da consulta de busca por valores exatos. Não foram empregados índices para o elemento *date* presente na ordenação.

Para as consultas 4 e 5, que testam o uso de dados irregulares, foi incluído um índice para o elemento `genre`. O índice empregado para este elemento *node-element-presence-none* é empregado a *tags* que não são obrigatórias, atendendo à característica destas consultas.

UTILIZAÇÃO DE ÍNDICES (GRUPO RESUMO DE ARTIGOS)

As consultas 1 e 6 se utilizam do índice criado para o atributo `id`. O índice empregado para este atributo *node-attribute-equality-string* atende às características da consulta de busca por valores exatos. Embora o atributo `id` tenha um valor numérico, o mesmo foi descrito na consulta como *string*. Portanto, na criação do índice, o tipo de dados empregados atende a este tipo de dados.

A segunda consulta se utiliza do índice criado para o elemento `author`. O índice empregado para este atributo *node-element-equality-string* atende às características da consulta de busca por valores exatos.

A terceira consulta se utiliza do índice criado para o elemento `confyear`. O índice empregado para este elemento *node-element-equality-string* atende às características da consulta de busca por valores exatos. Não foram empregados índices para o elemento *title* presente na ordenação.

Para as consultas 4 e 5, que testam o uso de dados irregulares, foi incluído um índice para o elemento `categoryAndSubjectDescriptorsTuple`. O índice empregado para este elemento *node-element-presence-none* é empregado a *tags* que não são obrigatórias, atendendo à característica destas consultas.

UTILIZAÇÃO DE ÍNDICES (GRUPO CURRÍCULOS)

Como já falado anteriormente, o acervo currículo é o que apresenta maior diversidade de documentos em termos estruturais.

Sendo assim, outras características estão sendo avaliadas, privilegiando

consultas do tipo "//", que procuram um elemento *tag* em qualquer ponto do documento. Além da utilização deste recurso que se encontra presente em todas as consultas, ainda foram avaliados:

- Pesquisa de textos (consultas 1;4): consultas do tipo *contains,* que verifica se uma dada *string* aparece no texto de uma dada *tag;*
- Consulta por dados irregulares (consulta 2): testa o retorno de *tags* que não são obrigatórias;
- Consulta por valores exatos (consultas 3;5;6).

As consultas 1 e 4 se utilizam do índice *node-element-substring-string* , que atende as características de pesquisa de uma dada substring em um texto. Os elementos indexados para cada uma destas consultas são: `instituição` e `title`, respectivamente.

A segunda consulta testa o uso de dados irregulares. Foi incluído um índice para o elemento `doutorado`. O índice empregado para este elemento *node-element-presence-none* é empregado a *tags* que não são obrigatórias, atendendo à característica desta consulta.

A terceira consulta se utiliza do índice criado para o elemento `cidade`. O índice empregado para este elemento *node-element-equality-string* atende as características da consulta de busca por valores exatos.

As consultas 5 e 6 se utilizam de índices criados para o elemento `nome`. A consulta 5 foi executada considerando o índice para o elemento `nome`, sem considerar que o mesmo aparece em mais de um caminho. Já a consulta 6 foi executada com a existência do índice, considerando o caminho. Neste caso, `dadosPessoais.nome` foi indexado.

6.4 PLANO DE TESTES

Há três situações principais que estarão sendo verificadas sobre as Unidades de Armazenamento - UA :

- UA's Múltiplas Homogêneas: a estratégia proposta que realiza o armazenamento por UA somente de documentos similares. Nesta estratégia temos uma UA para cada *cluster* formado.

- UA Única Heterogênea: a estratégia que realiza o armazenamento de diferentes documentos em uma única UA.

- UA's Múltiplas Heterogêneas: a estratégia que se utiliza de múltiplas UA's para distribuir os documentos, sendo que esta distribuição é realizada de forma aleatória, sem considerar qualquer tipo de relação entre os documentos.

Para cada uma destas situações foi avaliada a utilização ou não de índices. Os resultados obtidos foram confrontados entre as três estratégias.

Não foram exploradas estratégias diferenciadas oferecidas pelo SGBD XML Nativo Berkeley DB. A não utilização dos mesmos em nada afetam a avaliação das estratégias macro, que são: Armazenamento em UA múltiplas homogêneas, Armazenamento em UA única heterogênea e Armazenamento em UA múltiplas heterogêneas.

Para avaliar a estratégia proposta e compará-la com outras estratégias, inicialmente planejou-se uma avaliação que considerava a estratégia de armazenamento *node* e *whole*. Porém a estratégia *whole* não suportou a quantidade de documentos sendo armazenados.

Com base nas três situações apresentadas e na utilização de índices ou não. Para comparação das estratégias, foi executado o plano de testes presente na fig.6.6.

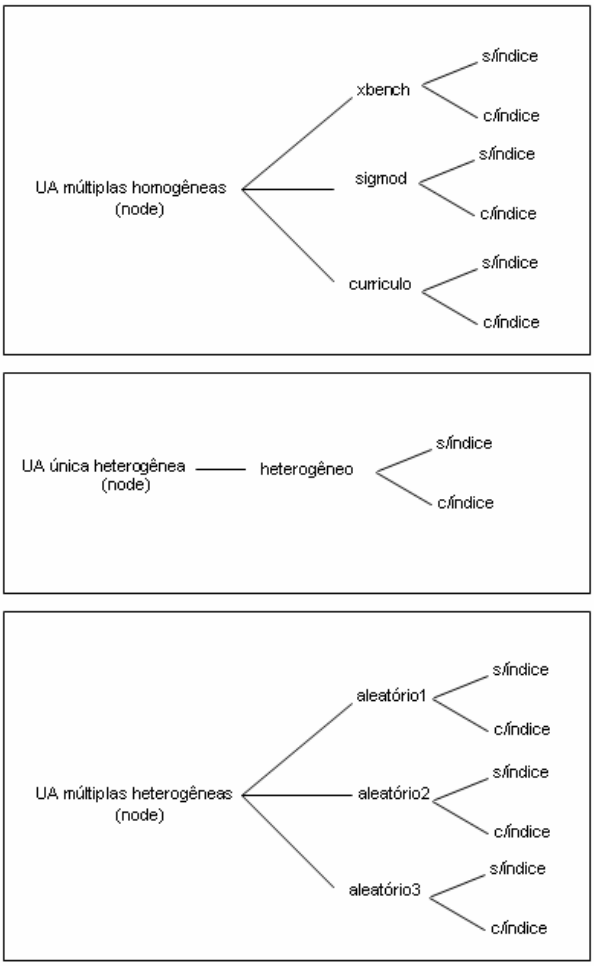


FIG. 6.6 Plano de testes

UA MÚLTIPLAS HOMOGÊNEAS

Através de uma identificação prévia, conferida através das etapas propostas pela sub-etapa de sumarização, pode-se direcionar as consultas para as UA específicas que contém cada categoria de documentos.

A tabela 6.4 mostra detalhes dos documentos que compõem esta unidade.

TAB. 6.4 - Distribuição homogênea – node

Acervo	container	qtd. docs	tam. total	menor	maior	tam.armaz	tam c/ índices
sigmod	container_db_sigmod.dbxml	911	1,96Mb	1kb	7kb	5,24Mb	22,8Mb
xbench	container_db_xbench.dbxml	405	250Mb	3kb	1,96Mb	510Mb	899Mb
currículo	container_db_curriculum.dbxml	15	58,9Kb	1Kb	10Kb	32Kb	488Kb

UA ÚNICA HETEROGÊNEA

A UA única heterogênea foi carregada com todos os documentos existentes conforme ilustra a tabela 6.5.

Todas as consultas, referentes a qualquer um dos acervos, são direcionadas a uma única UA, que contém todos os documentos e portanto está preparada para atender toda e qualquer consulta.

TAB. 6.5 - Distribuição heterogênea – node

Acervo	container	qtd. Docs	tam. total	menor	maior	tam.armaz	tam c/ índices
sigmod xbench curriculo	container_db_heterogeneo_unico.dbxml	1331	252Mb	1kb	1,96Mb	516Mb	520Mb

UA MÚLTIPLAS HETEROGÊNEAS

As UA múltiplas heterogêneas foram carregadas com documentos aleatórios. Dividiu-se os documentos a armazenar por três UA distintas e a seleção dos documentos que compõem estas UA foi sorteada aleatoriamente através de uma função de escolha randômica.

As UA múltiplas heterogêneas foram carregadas de modo que nas três UA's houvesse documentos provenientes dos três grupos (artigos/xbench, resumos/sigmod e currículos). Desta forma, qualquer consulta a ser realizada deve ser executada para as três UA's de armazenamento para que os documentos que atendem a pesquisa sejam recuperados.

A tabela 6.6 mostra detalhes dos documentos que compõem esta unidade.

TAB. 6.6 - Distribuição heterogênea aleatória – node

Acervo	Container	qtd. docs	tam armaz	tam c/ índice
sigmod xbench currículo	container_db_aleatorio1.dbxml	444	184Mb	186Mb
sigmod xbench currículo	container_db_aleatorio2.dbxml	443	160Mb	162Mb
sigmod xbench, currículo	container_db_aleatorio3.dbxml	444	171Mb	172Mb

Em um primeiro momento, não foram utilizados índices para realização dos testes. Posteriormente foram acrescentados os índices e verificadas as diferenças nos tempos de execução. A escolha dos índices foi feita com base nas consultas selecionadas, de modo a permitir uma melhora no desempenho destas consultas. Os índices criados estão listados no anexo *índices*.

O SGBD XML Nativo Berkeley DB XML oferece duas estratégias diferenciadas : armazenamento contíguo (*whole*) e armazenamento fragmentado (*node*). Os testes foram realizados utilizando a estratégia *node*.

6.4.1 METODOLOGIA DE TESTE

Os testes foram realizados de acordo com outros estudos de casos encontrados na literatura. Em TAVARES et al (2000), foi considerado como tempo de resposta a média entre a execução a frio (a primeira execução) e a execução a quente (que considera as execuções subsequentes). Cada consulta foi executada 20 vezes para que fosse obtida cada média.

Em RISHE et al (2000), os tempos a quente e a frio são descritos da seguinte forma:

> *"As transações de um benchmark são executadas em dois modos: quente e frio. Ambos 'tempo a quente' e 'tempo a frio' são coletadas para cada transação. Ambos os resultados são incluídos no resultado final. O tempo a frio é o tempo requerido para execução de uma transação imediatamente após a inicialização do SGBD no sistema com a cache vazia. Isto é normalmente conseguido reinicializando o sistema após a execução de cada transação. O tempo a quente é o tempo requerido para executar uma transação imediatamente após a execução de uma transação idêntica sem limpar a cache ou reinicializar o SGBD. Para coletar o tempo a quente, roda-se a mesma transação em um mesmo loop até que o tempo de execução se estabilize, o que tipicamente ocorre na terceira ou quarta execução. Uma vez que o tempo de execução se estabilize, nós computamos a*

média aritmética das cinco transações e isto é considerado o tempo a quente final de execução para essa transação."

Em RISHE et al.(2000), considera-se que a quantidade de execuções está relacionada a uma estabilização dos tempos de execução, mas não foram encontradas referências sobre o que pode ser considerado uma unidade de tempo estabilizada, já que sempre haverá diferenças mesmo que pequenas entre uma execução e outra.

Assim, como em PRAKASH e BHOWMICK (2006), no trabalho proposto, consideramos como tempo de execução final, a média entre os tempos de execução a quente. Para computação dos tempos foram realizadas 6 (seis) execuções, sendo descartado o tempo da primeira execução.

O ambiente utilizado para os testes foi o SGBD XML Nativo Berkeley DB XML. Os testes foram realizados utilizando-se um microcomputador com processador Pentium 4.3, memória RAM de 2 Gbytes, disco rígido de 160 Gbytes e sistema operacional Windows XP Professional.

6.5 RESULTADOS E GRÁFICOS

Os resultados computados no estudo de caso e seus respectivos gráficos estão incluídos nos apêndices 1, 2 e 3. Nestes resultados estão os tempos de execução auferidos em cada uma das consultas.

Foram avaliados os tempos de execução com e sem índice em cada uma das estratégias propostas no estudo de caso.

Podemos perceber através da análise do gráfico das consultas realizadas sobre os três acervos, sem a utilização de índices, que a estratégia proposta apresenta os menores tempos de processamento e, portanto, os melhores resultados em todas as consultas realizadas.

RESULTADOS CONSULTAS ACERVO ARTIGOS COMPLETOS (XBENCH) – S/ÌNDICE

A fig. 6.7 apresenta um gráfico comparativo entre a estratégia proposta e as

outras estratégias, na recuperação dos documentos a partir das consultas selecionadas sem utilização de índices.

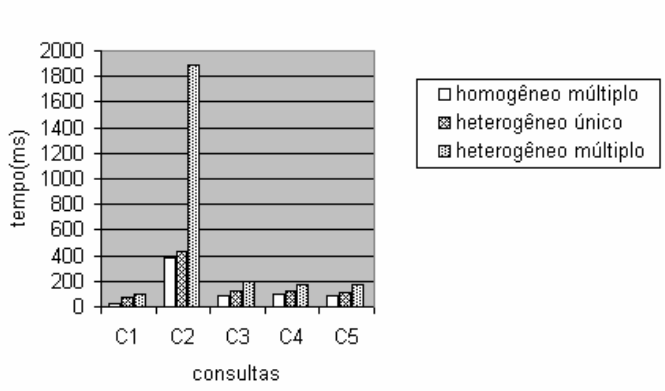


FIG. 6.7 Tempos de Recuperação dos documentos por consulta s/índice – artigos completos

A sexta consulta não foi colocada no gráfico, já que não se pode computar os resultados nas três estratégias. De qualquer forma, na nossa estratégia, esta consulta foi computada em menos tempo em relação às outras duas estratégias. Uma delas teve o tempo excedido e a outra não foi finalizada após mais de uma hora de processamento.

RESULTADOS CONSULTAS ACERVO ARTIGOS COMPLETOS (XBENCH) – C/ÍNDICE

Pode-se perceber através da análise dos gráficos da fig. 6.8, que a estratégia proposta apresenta os menores tempos de processamento na maior parte das consultas. Os ganhos de desempenho são significativamente maiores para as consultas C2, C3. Os índices usados nestas consultas são baseados em igualdade de valores sobre elementos (*tags*), diferentemente dos índices usados nas demais consultas que são índices para dados irregulares (presença) e para atributos. Em relação à estratégia Heterogêneo Múltiplo, exceto nas consultas C1 e C6, os ganhos são sempre significativamente maiores. A consulta C6, que envolve o uso de índices para realizar junção, apresentou para a estratégia Homogêneo Múltiplo, piores resultados do que em relação à estratégia Heterogêneo Múltiplo.

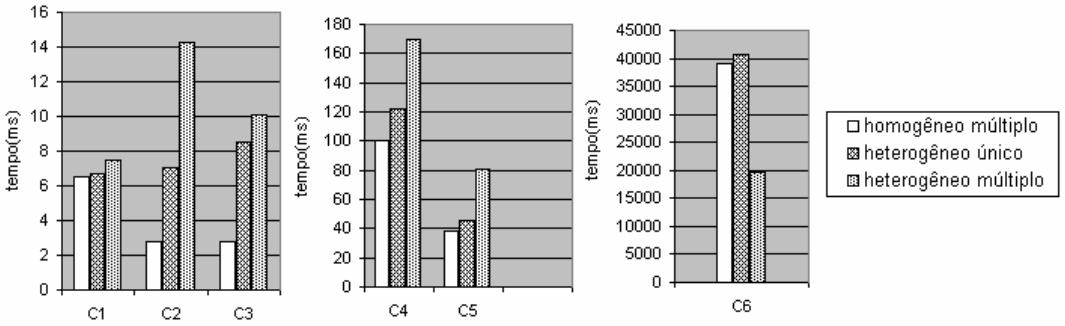


FIG. 6.8 Tempos de Recuperação dos documentos por consulta c/índice – artigos completos

RESULTADOS CONSULTAS ACERVO RESUMO DE ARTIGOS (SIGMOD) – S/ÌNDICE

A fig. 6.9 apresenta um gráfico comparativo entre a estratégia proposta e as outras estratégias, na recuperação dos documentos a partir das consultas selecionadas sem utilização de índices.

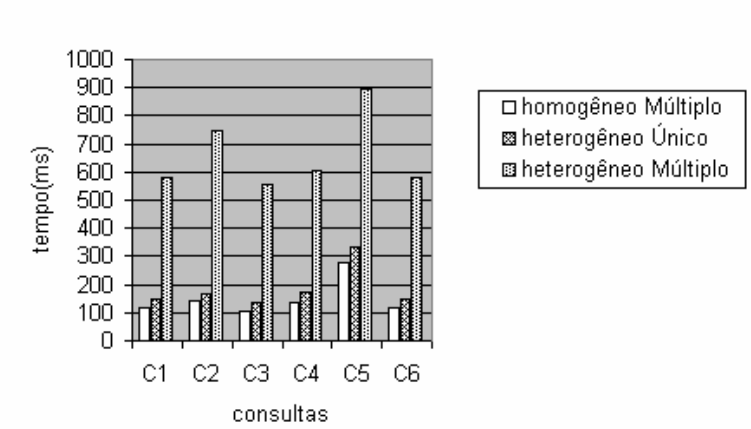


FIG. 6.9 Tempos de Recuperação dos documentos por consulta s/índice – resumos de artigos

RESULTADOS CONSULTAS ACERVO RESUMO DE ARTIGOS (SIGMOD) – C/ÌNDICE

Pode-se perceber através da análise dos gráficos da fig. 6.10, que a estratégia proposta apresenta os menores tempos de processamento em todas as consultas, exceto na consulta C5 que perde para a estratégia Heterogêneo Único e não apresenta grandes diferenças em relação a estratégia Heterogêneo Múltiplo. O índice usado nesta consulta oferece suporte a dados irregulares (presença).

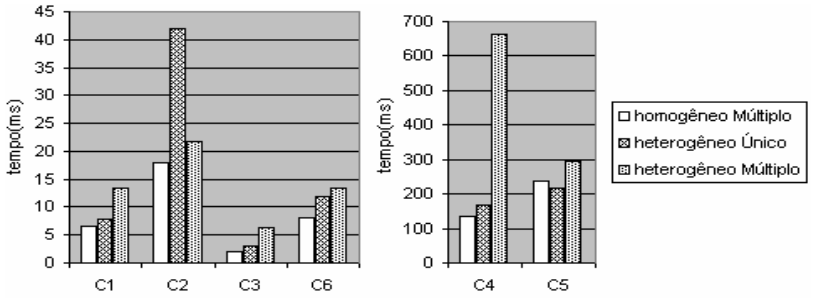


FIG. 6.10 Tempos de Recuperação dos documentos por consulta c/índice – resumos de artigos

RESULTADOS CONSULTAS ACERVO CURRICULO – S/ÌNDICE

A fig. 6.11 apresenta um gráfico comparativo entre a estratégia proposta e as outras estratégias, na recuperação dos documentos a partir das consultas selecionadas sem utilização de índices.

Para este acervo, não conseguimos computar os tempos para a forma de armazenamento Heterogêneo Múltiplo por excesso de tempo na computação que foi interrompida.

Já os tempos de processamento para a estratégia Homogêneo Múltiplo apresentaram resultados próximos de zero. O maior tempo computado nestas consultas foi de aproximadamente 13 ms.

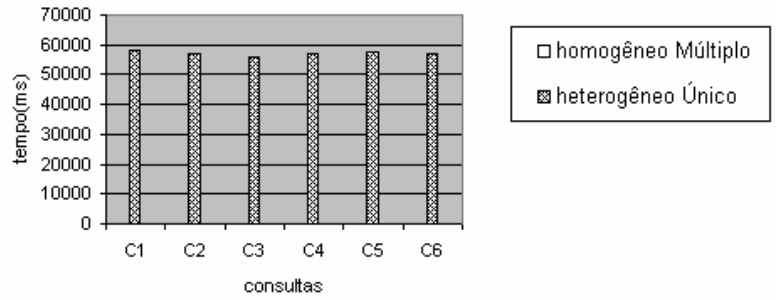


FIG. 6.11 Tempos de Recuperação dos documentos por consulta s/índice – currículo

RESULTADOS CONSULTAS ACERVO CURRICULO – C/ÌNDICE

A fig. 6.12 não apresenta os tempos para a estratégia Heterogêneo Múltiplo pois ocorreram diversos erros no processamento destas consultas sobre este grupo de documentos. Todas as consultas aplicadas a este grupo utilizam-se do recurso de barras duplas, que indica que a *tag* subseqüente pode aparecer como um descendente da *tag* precedente. Este recurso foi utilizado, já que há grandes diferenças estruturais neste grupo de documentos.

A consulta C2 apresentou o maior ganho na estratégia proposta, Homogêneo Múltiplo, em relação às demais estratégias. Esta consulta envolve o uso de índices para dados irregulares (presença). Já a consulta C3 foi a que apresentou tempos de execução mais próximos entre as estratégias Homogêneo Único e Heterogêneo Múltiplo. O índice utilizado nesta consulta é baseado em igualdade de valores sobre elementos.

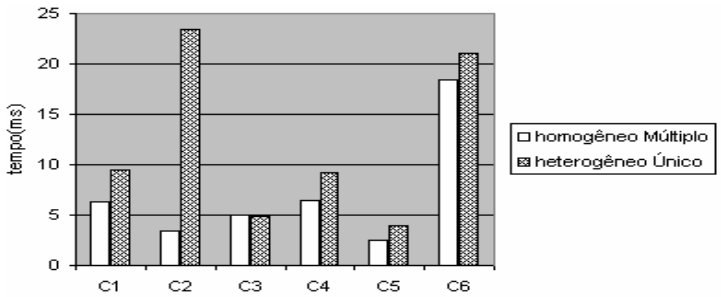


FIG. 6.12 Tempos de Recuperação dos documentos por consulta c/índice – currículo

6.6 CONSIDERAÇÕES

Durante a realização dos testes, foram encontrados alguns problemas no SGBD XML escolhido, Berkeley DB XML. Um dos principais problemas foi que algumas consultas tiveram que ser reescritas para que o otimizador reconhecesse os índices. Um outro problema importante diz respeito às alternativas de indexação oferecidas pelo BDB XML. Pela análise do plano de consulta, percebe-se que os índices foram utilizados de forma diferenciada pelo otimizador de consulta, ora beneficiando, ora prejudicando o desempenho das consultas, o que tornou a comparação, nestes casos, inconclusiva. Com isto, concluímos que a otimização de consultas do BDB XML depende de outros fatores que precisam ser melhor explorados.

Traçando-se um paralelo entre as estratégias de armazenamento *node* e *whole,* verificamos que a partir de uma mesma quantidade de documentos a armazenar, a estratégia *node* gasta quase que o dobro de tempo para realizar o mesmo armazenamento em relação a um *container* criado pela estratégia *whole*. Em relação a estas duas estratégias de armazenamento, também foi verificado que o armazenamento *node* ocupa um espaço de armazenamento em disco maior que a estratégia *whole.* Em compensação, verificamos que a estratégia *node,* na recuperação dos documentos, tem um tempo de retorno muito mais rápido que a estratégia *whole*. Não foram apresentados os resultados obtidos pela estratégia *whole*, pois esta apresentou problemas após o

armazenamento do acervo.

Apesar de terem sido avaliadas poucas consultas, o fato de se ter UA's homogêneas, mantendo documentos com características comuns, proporciona algumas vantagens sobre as outras estratégias. Uma delas, é saber para onde as consultas serão direcionadas, o que não acontece na forma de armazenamento heterogêneo múltiplo, onde para cada pesquisa, várias UA's precisam ser consultadas. Isto não acontece na forma de armazenamento heterogêneo único, que mantém todos os documentos em uma única UA, mas por outro lado, não se pode aplicar estratégias de armazenamento diferenciadas. A estratégia de armazenamento proposta possibilita o emprego adequado dos recursos de armazenamento e acesso ao tipo de documento que está sendo armazenado. Nos três conjuntos de documentos apresentados, foi possível perceber claramente os ganhos, em especial para o grupo de *currículos*.

Identificamos ainda, outros testes importantes que poderiam ser realizados para avaliar melhor a estratégia proposta, como por exemplo, para avaliar acesso concorrente, ou para avaliar consultas com resultados possíveis em todos os grupos. Por fim, não foi levado em consideração nos testes, o tempo para a inclusão de índices, que em alguns casos pode chegar a mais de uma hora.

7 CONCLUSÃO

As informações que hoje circulam pela Web são provenientes de inúmeras e heterogêneas fontes de dados. Assim sendo, torna-se necessária a aplicação de estratégias que permitam a captação, organização e distribuição destas informações para a aplicação da gerência de conteúdo. Embora existam diversas abordagens para apoiar o armazenamento e recuperação destas informações, concluiu-se que o tratamento de um acervo de grandes proporções composto por informações oriundas de diversas fontes exigiria uma solução mais elaborada. Partindo-se do pressuposto que estes documentos sejam disponibilizados no formato XML, este trabalho propôs uma estratégia que visa apoiar a construção de um projeto físico de banco de dados que envolve o armazenamento dos documentos de forma sistemática em SGBD's XML Nativos próprios para o armazenamento desta categoria de documentos. A principal característica da estratégia proposta é o uso de técnicas de KDD sobre a estrutura dos documentos XML para apoiar a distribuição dos documentos em grupos homogêneos, o que possibilita a montagem de um ambiente de armazenamento que possa atender as características dos documentos a armazenar e a aplicação de regras de indexação e recuperação.

A fim de ilustrar o potencial da estratégia proposta, este trabalho incluiu a preparação e utilização de um acervo conhecido para a realização de um estudo de caso. A estratégia proposta mostrou um bom potencial, contribuindo para a criação de um projeto físico de banco de dados, que apresentou melhores resultados na busca e recuperação de documentos, quando comparado ao armazenamento não orientado (aleatório ou não agrupado).

7.1 CONTRIBUIÇÕES

Podem ser destacados como contribuições deste trabalho os seguintes itens:

- A especificação de uma estratégia que guie o usuário na criação de um projeto físico de banco de dados, para o armazenamento de um conjunto de documentos XML heterogêneos em SGBD's XML Nativos.

- A implementação da estratégia na forma de um protótipo, que cobre a maioria das etapas propostas, desde a identificação de grupos de documentos similares, até o seu posterior armazenamento e indexação.

- A construção de um acervo de documentos XML heterogêneos porém com possibilidades de inter-relação entre eles.

- A realização de um estudo de caso sobre o acervo heterogêneo construído, ilustrando ganhos de desempenho da estratégia proposta.

- Exploração do SGBD XML Nativo Berkeley DB XML, através do armazenamento dos documentos segundo a estratégia proposta, além da utilização de índices sobre o SGBD e avaliação do suporte oferecido às consultas do *benchmark Xbench*.

- Publicação do artigo "*Uma Estratégia baseada em Técnicas de KDD para apoiar o Projeto Físico em SGBDs XML Nativos"* (SANTOS et al., 2007).

7.2 MELHORIAS NO SISTEMA

No protótipo desenvolvido, algumas implementações ficaram pendentes e outras melhorias observadas podem ser realizadas futuramente. São elas:

- Apesar de ter sido avaliado apenas o SGBD BDB XML, vale ressaltar que a estratégia proposta é aplicável a qualquer SGBD XML Nativo, assim como poderiam ter sido aplicados outros algoritmos de mineração que implementam as técnicas de associação e clusterização utilizados. Para tornar o protótipo independente do SGBD a ser utilizado precisaríamos realizar pequenas adaptações principalmente na fase de pós-processamento onde o SGBD é acionado em atividades que envolvem a sub-etapa de armazenamento.

- Os resultados do processo de clusterização foram obtidos a partir do sistema MATLAB, que não é OpenSource. Apesar de ter sido realizada a integração com este sistema, e ter-se como vantagem a utilização de um algoritmo de mineração que já é amplamente utilizado, produzindo resultados confiáveis para a principal etapa da estratégia proposta, sugere-se que o algoritmo seja implementado dentro do protótipo. Assim, com todos os recursos disponíveis dentro da ferramenta, a solução poderá ser disponibilizada por completo no meio acadêmico. Vale ressaltar que atualmente, a clusterização por outros meios externos, pode ser executada por qualquer algoritmo que implemente essa tarefa, desde que o resultado seja gerado no formato que o protótipo está preparado para receber no momento de sua importação.

- A ferramenta não gera índices automaticamente. Os índices são apenas sugeridos, e é opção do usuário criá-los ou não. A própria ferramenta poderia dispor de uma funcionalidade para realizar a indexação.

- É disponibilizado um módulo para execução de consultas. O sistema está preparado para atender consultas armazenadas, similares ao conceito de *views*, sendo que as consultas ficam salvas em um arquivo XML. Para tal é necessário cadastrar todas as consultas neste arquivo. Como melhoria, sugere-se que toda a consulta executada possa ser salva automaticamente e além disto, toda vez que a consulta for executada, seja registrado o tempo de execução correspondente. Estes resultados poderiam ser utilizados para a sugestão de novos índices.

- O processo de equiparação terminológica para o acervo utilizado, foi realizado de forma manual. Para resolver este problema seria necessário realizar um estudo detalhado de técnicas de recuperação de informação, que tratam deste problema e que poderiam ser implementadas ou integradas a esta realidade.

- No protótipo não foi implementado nenhum algoritmo que pudesse realizar de forma automática a tarefa de sumarização. Sugere-se, portanto, a pesquisa e implementação de algum algoritmo com esse propósito.

- Não foi especificado e/ou implementado o tratamento voltado para a inclusão de novos documentos em unidades de armazenamento já formadas. Assim sendo, uma funcionalidade para inclusão de novos documentos mostra-se como uma melhoria desejável para o protótipo.

7.3 TRABALHOS FUTUROS

Como trabalhos futuros, podemos citar inicialmente, o desenvolvimento de mecanismos para a descoberta e geração de um esquema único para cada unidade de armazenamento. Esta geração poderá ser ou não viável dependendo dos graus de heterogeneidade dos documentos dentro dos grupos formados. A utilização de um esquema único contribuiria para aplicação de índices e para o próprio SGBD criar planos de execução mais eficientes tendo como referência o esquema do documento, portanto tendo um conhecimento prévio de sua estrutura.

Na estratégia proposta, a quantidade de *clusters* é determinada pelos próprios usuários. O ideal seria que a quantidade de *clusters* formados seja determinada automaticamente pela aplicação. Existem algoritmos de clusterização em que os grupos são gerados automaticamente sem a necessidade de se determinar a quantidade de *clusters*. Nesse cenário, o próprio algoritmo poderia se encarregar de otimizar a quantidade de clusters necessárias (GOLDSCHMIDT; PASSOS, 2005, P.76).

Quanto ao SGBD XML Nativo, sugere-se investigar e avaliar a aplicação da estratégia sobre outros SGBD's, explorando as características de cada um. Essa investigação teria como objetivo comprovar que a estratégia proposta pode ser utilizada independente do SGBD XML Nativo escolhido.

Quanto à diversidade semântica, que é um problema enfrentado em acervos heterogêneos em que as mesmas informações são descritas de diferentes formas, é necessário integrar o protótipo desenvolvido com técnicas de recuperação de informação, de forma que o processo de equiparação de termos seja realizado de forma

automática.

Quanto à característica do algoritmo de clusterização empregado, *fuzzy k-means*, faltou explorar melhor os recursos deste algoritmo, para um acervo que tivesse um alto grau de interseção entre documentos de mais de um grupo. Neste caso foi sugerido o armazenamento em quantas unidades quanto fossem apontadas pertinências significativas. Não foram avaliados os resultados através de aplicação de consultas, até por que no acervo utilizado, apesar de haver características comuns acentuadas no acervo artigos completos e resumos de artigos, não houve necessidade de replicá-los e/ou mantê-los em uma mesma unidade, apesar de terem características comuns, já que a quantidade de *clusters* determinada caracterizava bem cada grupo de documentos.

Outra avaliação a realizar seria a análise do comportamento da estratégia aplicada a acervos desconhecidos, bem como, qual seria o momento ideal para a aplicação da estratégia, isto é, qual o volume mínimo de documentos a partir do qual a estratégia teria vantagens em ser aplicada.

Quanto ao processamento das consultas, sugere-se avaliar a estratégia proposta com recursos de paralelismo ou até mesmo em SGBD's distribuídos, aproveitando parte da estratégia, principalmente a aplicação de técnicas de mineração de dados para realização da fragmentação destes documentos, visando uma melhor recuperação e acessos concorrentes.

Neste trabalho, optou-se por trabalhar na fase de preparação da matriz de dados a ser submetida ao algoritmo de mineração, com os dados fornecidos por uma matriz de freqüência, pois acredita-se que melhor retrataria o conjunto de documentos que estavam sendo tratados. Julgamos que dependendo do grau de heterogeneidade dos documentos a matriz de ocorrência e a matriz hierárquica tornam-se inadequadas, porém podem ser aplicadas para identificação de subgrupos, dentro dos grupos formados. Como trabalho futuro, sugerimos a inclusão de mecanismos para facilitar e/ou automatizar a identificação do tipo de matriz adequada para o conjunto de documentos em análise.

Um outro ponto que não foi tratado e que merece atenção em trabalhos futuros, é a inclusão de novos documentos. A estratégia é aplicável para montagem inicial das unidades de armazenamento, que após criadas e realizada uma carga inicial, devem receber novos documentos. O que não foi previsto é como aplicar a estratégia quando da inclusão de um novo documento ou de novos documentos. Neste caso, temos que identificar a que grupo este documento pertence e no caso deste documento não ser

compatível com os demais, apontar automaticamente a necessidade de se criar novos grupos.

A estratégia proposta tem aplicação real a ser empregada. Esperamos que a mesma contribua para que se facilite cada vez mais a gerência de documentos XML e que estes, por conseguinte, sejam cada vez mais usados e suas características de flexibilidade possam ser exploradas mais amplamente, sem que isto se torne um empecilho para sua recuperação.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ABITEBOUL, SERGE; MANOLESCU, IOANA; NGUYEN, BENJAMIN; PREDA, NICOLETA. **A Test Plataform for the INEX heterogeneous track**, Springer Berlin, 2005, Advances in XML Information Retrieval, v. 3493, p.358-371.

AGRAWAL, RAKESH; SRIKANT, RAMAKRISNAN. **Fast Algorithms for mining Association Rules**, Proceedings of the 20th VLDB Conference, 1994.

BOURRET, RONALD. **XML And Databases**, Disponível em: <http://www.rpbourret.com/xml/XMLAndDatabases.htm> acesso em: 28 nov. 2005.

BRASIL, CHRISTIANE REGINA. **Ferramenta Inteligente de apoio a pesquisa: mineração de artigos científicos na web** ,USP, 2004. Dissertação.

BRIAN, DANNY. **The Definitive Guide to Berkeley DB XML**, APRESS, 2006.

CAMPOS, MARIA LUIZA DE ALMEIDA; CAMPOS, MARIA LUIZA MACHADO; GOMES, HAGAR ESPANHA; CAMPOS, MARIA LINAIR; MARTINS, ALISSANDRA EVANGELISTA; SALES, LUANA FARIAS. **Estudo comparativo de Softwares de Construção de Tesauros**, Perspect. Ciênc.Inf., 2006, v.11 n.1, p.68-81.

CARLANTÔNIO, L.M. **Novas Metodologias para clusterização de dados**. COPPE-UFRJ, 2001. Dissertação.

CHAUDHRI, AKMAL B. ; RASHID, AWAIS; ZICARI, ROBERTO. **XML Data Management – Native XML and XML-Enabled Database Systems**, Addison-Wesley, 2003.

COELHO, GILDA MASSARI, SANTOS MARIO DE MIRANDA; SANTOS, DALCI MARIA; FILHO, LÉLIO FELLOWS. **Prospecção de Tecnologias de futuro: métodos, técnicas e abordagens**, Revista Parcerias Estratégicas edição Nº19, Brasília, 2004.

DOBSA, JASMINKA, **Concept Decomposition by Fuzzy K-means Algorithm**, IEEE, 2003.

ELMASRI, RAMEZ ; NAVATHE, SHAMKANT. **Sistemas de Banco de Dados**, Pearson Addison Wesley, 4º Edição, 2005.

FAYYAD, U.M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **From Data Mining to Knowledge Discovery : An Overview. Knowledge Discovery and Data Mining**, Menlo Park: AAAI Press, 1996.

FLESCA, SÉRGIO ; MANCO GIUSEPPE, MASCIARI, ELIO; PONTIERI, LUIGI, PUGLIESE, ANDREA. **Fast Detection of XML Structural Similarity**, IEEE, 2005.

FLORESCU, DANIELA, **Structured Data**, Oracle, 2005.

FLORESCU, DANIELA; KOSSMANN, DONALD. **Storing and Querying Data using RDMBS**, IEEE , 1999.

GOLDSCHMIDT, RONALDO ; PASSOS, EMMANUEL. **Data Mining Um Guia Prático**, Campus, 2005.

HAN, J.; KEMBER, M. **Data Mining: Concepts and techniques**. San Francisco: Morgan Kaufmann Publishers, 2001

HUANG, ZHEXUE, **Extensions to the K-means Algoritm for Clustering Large Data Sets with Categorical Values**, Kluwer Academic Publishers, Data Mining and Knowledge Discovery 2, 283-304, 1998.

LEE,JUNG-WON ; LEE, KIHO, KIM, WON , **Preparations for Semantics-Based XML Mining**, IEEE, 2001.

LI, YUNYAO; YU, CONG; JAGADISH, H.V MAMOULIS. **Schema-Free Xquery**, VLDB, 2004.

LIAN, WANG; CHEUNG DAVID; MAMOULIS, NIKOS; YIU, SIU-MING. **An efficient and scalable algorithm for Clustering XML Documents by Structure**, IEEE, 2004.

MACEDO, GERALDO MAJELA FERREIRA. **Bases Para Implantação de um Sistema de Gerenciamento Eletrônico de Documentos - GED: Um Estudo De Caso** , UFSC, 2003, Dissertação.

MENG, XIAOFENG; WANG, YU ; LUO, DAOFENG; LU, SHICHAO; AN, JING; CHEN, YAN ; OU, JIANBO; JIANG, YU. **OrientX : A Schema-based Native XML Database System**, VLDB, 2003.

NIERMAN, A.; JAGADISH, H. V. **Evaluating Structural Similarity in XML Documents**, Proceedings of the Int. Workshop on the Web and Databases (WebDB), Madison, WI, 2002

PEREIRA, J. C. L. E BAX, M. P. **Introdução a Gestão de Conteúdos**, Workshop Brasileiro de Inteligência Competitiva e Gestão do Conhecimento, 2002.

PRAKASH, SANDEEP; BHOWMICK, SOURAV S. **A Tale of Two Approaches: Query Performance Study of XML Storage Strategies in Relational Databases**, DEXA, 2006.

REIS, MAURICIO C., **Descoberta de conhecimento em Documentos XML**, IME, 2005, Dissertação.

RISHE, NAPHTALI; VASCHILLO, ALEXANDER; VASILEVSKY, DMITRY; SHAPOSHNIKOV, ARTYOM; CHEN, SHU-Ching. **A Benchmarking Technique for DBMS'S With Advanced Data Models**, ADBIS-DASFAA Symposium on Advances in Databases and Information Systems, 2000.

ROSA, J.M.C; TANSCHEIT, RICARDO; VELLASCO, MARLEY; ZANINI, A; KLEIN, C.H; BLOCH, K.V; NOGUEIRA, A.R; SALIS, L.H ; SOUZA E SILVA, N.A .**Aplicação Fuzzy Clustering a Banco de Dados de Amostra Domiciliar da População da Ilha do Governador**, SBAI, 2001.

RUBERG, N. ; RUBERG, G. ; MATTOSO, M. L. Q. . **DIG: Um Serviço de Custos e Estatísticas para o Processamento Distribuído de Consultas**. SBBD, 2002.

SANTOS, MIRIAM OLIVEIRA DOS; CAVALCANTI, MARIA CLÁUDIA; GOLDSCHMIDT, RONALDO. **Uma Estratégia baseada em Técnicas de KDD para apoiar o Projeto Físico em SGBDs XML Nativos**, SBBD, 2007.

SCHÖNING, HARALD, **Tamino- a DBMS designed for XML**, IEEE, 2001.

SLEEPYCAT, **Introduction for Berkeley DB XML**, Disponível em : <http://www.sleepycat.com/products/bdbxml.html> acesso em: 12 maio. 2005.

SUN WAN, KIM; YOUN HEE KIM; JAEHO, LEE ; HAECHULL, HIM. **Developing a Native Storage Structure For XML Repository System in Main Memory** , IEEE, 2002.

TAVARES, FLÁVIO; VICTOR, ANDRÉ; MATTOSO, MARTA. **Parallel Processing Evaluation of Path Expressions**, SBBD, 2000.

TIAN, FENG; DEWITT, DAVID J.; CHEN, JIANJUN; ZHANG, CHUN. **The Design and Performance Evaluation of Alternative XML Storage Strategies**, Sigmod Record , 2002.

UNESCO, **Guidelines for the establishment and development of monolingual thesauri**. Paris, 1973, 37p.

VAKALI, ATHENA ; CATANIA, BARBARA; MADDALENA, ANNA. **XML Data Stores: Emerging Practices**, IEEE, 2005.

VIEIRA JUNIOR, HUMBERTO JOSÉ. **XVERTER: Armazenamento e Consulta de Dados XML em SGBD's**, COPPE- UFRJ, 2002, Dissertação.

WAAS, FLORIAN; SCHMIDT, ALBRECHT ; MANOLESCU, IOANA; KERSTEN, MARTIN; BUSSE, RALPH; CAREY, MICHAEL J. **Xmark : A Benchmark for XML Data Management**, Proceedings of the 28th VLDB Conference, 2002

YANG, JIANWU; CHEUNG, WILLIAM K ; CHEN, XIAOOU . **Integrating Element and Term Semantics for Similarity-Based XML Document Clustering**, International Conference on Web Intelligence, 2005

9 APÊNDICES

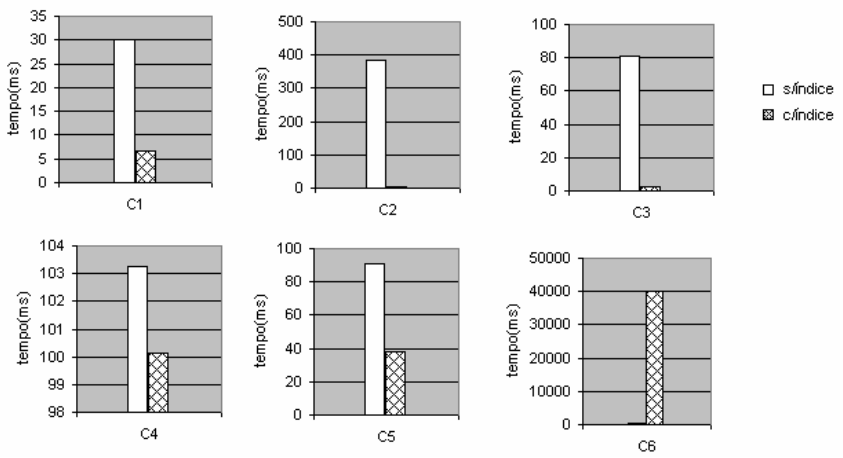
Nesta seção são detalhados através de tabelas os resultados obtidos no processamento das consultas em cada uma das estratégias, destacando os tempos de execução de cada consulta e mostrando graficamente as diferenças entre os tempos de execução segundo a metodologia a quente e a frio. Estes resultados foram utilizados no capítulo 6 -Estudo de Caso.

9.1 APÊNDICE 1: UNIDADES DE ARMAZENAMENTO MÚLTIPLAS HOMOGÊNEAS

ACERVO ARTIGOS COMPLETOS (XBENCH)

TAB. 9.1 - Resultado UA múltiplas homogêneas (XBENCH)

		à frio	à quente					resultado
		1 ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média à quente
C1	s/índice	61,417	29,762	30,543	30,19	30,217	30,151	30,1726
	c/índice	5,773	6,192	6,673	6,641	6,414	6,9	6,564
C2	s/índice	503,786	384,012	383,079	384,33	383,682	382,497	383,52
	c/índice	2,437	2,804	2,771	2,754	2,83	2,764	2,7846
C3	s/índice	214,161	81,552	81,225	80,755	80,889	81,01	81,0862
	c/índice	3,058	2,958	2,879	2,768	2,752	2,85	2,8414
C4	s/índice	89,443	106,246	103,422	101,587	102,531	102,417	103,2406
	c/índice	100,414	98,023	100,699	102,842	98,231	100,943	100,1476
C5	s/índice	148,357	110,744	85,472	86,08	85,49	85,465	90,6502
	c/índice	47,674	48,.184	48,37	47,62	47,31	48,306	38,3212
C6	s/índice	552.898,00	534.504,00	540.331,00	540.435,00	540.824,00	541.436,00	539506
	c/índice	37.002,00	38.257,00	38.895,30	39.313,40	39.117,70	39.347,90	38986,26



GRA. 9.1: Tempo de processamento das consultas com e sem índice UA múltiplas homogêneas (XBENCH)

A escolha dos índices foi realizada com base nas consultas a serem realizadas, observando-se a estrutura dos documentos. A tabela a seguir, mostra os índices adicionados, em seguida são mostrados o plano de execução das consultas, após a inclusão dos índices.

TAB. 9.2 - Índices da UA múltipla homogênea (XBENCH)

Index: unique-node-attribute-equality-string for node {}:id
Index: node-element-equality-string for node {}:name
Index: node-element-equality-string for node {}:country
Index: node-element-presence-none for node {}:genre
Index: unique-node-metadata-equality-string for node http://www.sleepycat.com/2002/dbxml}:name

Houve problema na inclusão do índice para o elemento `p`, que seria utilizado seguindo a estratégia *edge.* Este elemento aparece repetidamente em diferentes *paths*. Portanto as consultas não foram beneficiadas pela inclusão deste índice.

Erro na tentativa de incluir o índice edge

```
dbxml> addindex "" p    edge-element-substring-string
Adding index type: edge-element-substring-string to node: {}:p
Lock table is out of available locks
stdin:4: addIndex failed, Error: Not enough space
```

Para constatação dos índices utilizados, a seguir mostraremos os planos de execução associados a cada consulta.

C1: for $a in collection(“container”)/article[@id=”1”] return $art/prolog/title

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
      <OQPlan>V(node-attribute-equality-string,@id,=,'1')</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="article" nodeType="element">
      <OQPlan>V(node-attribute-equality-string,@id,=,'1')</OQPlan>
    </Step>
    <DbXmlFilter>
      <Navigation>
        <Step axis="attribute" name="id" nodeType="attribute"/>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="1" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
    <Step axis="child" name="prolog" nodeType="element"/>
    <Step axis="child" name="title" nodeType="element"/>
  </Navigation>
</XQuery>
```

C2: for $prolog in collection("container ")/article/prolog where $prolog/authors/author/name="Ben Yang" return $prolog/title

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
      <OQPlan>V(node-element-equality-string,name,=,'Ben Yang')</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="article" nodeType="element"/>
    <Step axis="child" name="prolog" nodeType="element"/>
    <DbXmlFilter>
      <Navigation>
        <Step axis="child" name="authors" nodeType="element"/>
        <Step axis="child" name="author" nodeType="element"/>
        <Step axis="child" name="name" nodeType="element">
          <OQPlan>V(node-element-equality-string,name,=,'Ben Yang')</OQPlan>
        </Step>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="Ben Yang" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
    <Step axis="child" name="title" nodeType="element"/>
  </Navigation>
</XQuery>
```

C3: for $a in collection("container.dbxml")/article/prolog where $a/dateline/country="Canada" order by $a/dateline/date return {$a/title} {$a/dateline/date}

```
<XQuery>
  <FLWOR>
    <ForBinding name="a">
      <Navigation>
        <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
          <OQPlan>V(node-element-equality-string,country,=,'Canada')</OQPlan>
        </QueryPlanFunction>
        <Step axis="child" name="article" nodeType="element"/>
        <Step axis="child" name="prolog" nodeType="element"/>
        <DbXmlFilter>
          <Navigation>
            <Step axis="child" name="dateline" nodeType="element"/>
            <Step axis="child" name="country" nodeType="element">
              <OQPlan>V(node-element-equality-string,country,=,'Canada')</OQPlan>
            </Step>
            <DbXmlCompare name="equal">
              <Sequence>
                <AnyAtomicTypeConstructor value="Canada" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
              </Sequence>
            </DbXmlCompare>
          </Navigation>
        </DbXmlFilter>
      </Navigation>
    </ForBinding>
    <Sort>
      <Specification modifier="ascending|empty_least">
        <Navigation>
          <Variable name="a"/>
          <Step axis="child" name="dateline" nodeType="element"/>
          <Step axis="child" name="date" nodeType="element"/>
        </Navigation>
      </Specification>
    </Sort>
    <DOMConstructor type="element">
      <Name>
        <Sequence>
          <AnyAtomicTypeConstructor value="Output" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
        </Sequence>
      </Name>
      <Children>
        <Navigation>
          <Variable name="a"/>
          <Step axis="child" name="title" nodeType="element"/>
        </Navigation>
        <Navigation>
          <Variable name="a"/>
          <Step axis="child" name="dateline" nodeType="element"/>
          <Step axis="child" name="date" nodeType="element"/>
        </Navigation>
      </Children>
    </DOMConstructor>
  </FLWOR>
</XQuery>
```

C4: collection("container")/article/prolog[not(genre)]/title

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
      <OQPlan>U</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="article" nodeType="element"/>
    <Step axis="child" name="prolog" nodeType="element">
      <Predicates>
        <Function name="{http://www.w3.org/2005/04/xpath-functions}:not">
          <Step axis="child" name="genre" nodeType="element">
            <OQPlan>P(node-element-presence-none,=,genre)</OQPlan>
          </Step>
        </Function>
      </Predicates>
    </Step>
    <Step axis="child" name="title" nodeType="element"/>
  </Navigation>
</XQuery>
```

C5: collection("container.dbxml")/article/prolog/genre/title

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
      <OQPlan>P(node-element-presence-none,=,genre)</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="article" nodeType="element"/>
    <Step axis="child" name="prolog" nodeType="element"/>
    <Step axis="child" name="genre" nodeType="element">
      <OQPlan>P(node-element-presence-none,=,genre)</OQPlan>
    </Step>
    <Step axis="child" name="title" nodeType="element"/>
  </Navigation>
</XQuery>
```

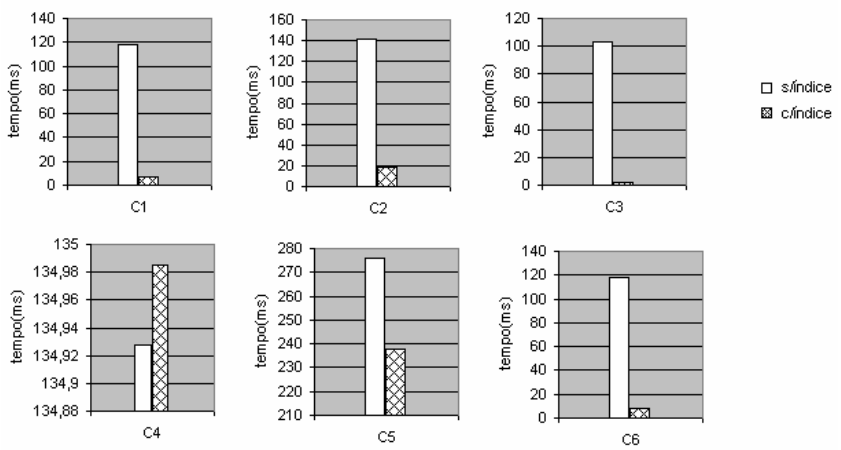
C6: for $a in collection("container")/article[@id="7"]/epilog/references/a_id, $b in collection("container")/article where $a = $b/@id return {$b/prolog/title}

```
<XQuery>
  <FLWOR>
    <ForBinding name="a">
      <Navigation>
        <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
          <OQPlan>V(node-attribute-equality-string,@id,=,'7')</OQPlan>
        </QueryPlanFunction>
        <Step axis="child" name="article" nodeType="element">
          <OQPlan>V(node-attribute-equality-string,@id,=,'7')</OQPlan>
        </Step>
        <DbXmlFilter>
          <Navigation>
            <Step axis="attribute" name="id" nodeType="attribute"/>
            <DbXmlCompare name="equal">
              <Sequence>
                <AnyAtomicTypeConstructor value="7" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
              </Sequence>
            </DbXmlCompare>
          </Navigation>
        </DbXmlFilter>
        <Step axis="child" name="epilog" nodeType="element"/>
        <Step axis="child" name="references" nodeType="element"/>
        <Step axis="child" name="a_id" nodeType="element"/>
      </Navigation>
    </ForBinding>
    <ForBinding name="b">
      <Navigation>
        <QueryPlanFunction result="collection" container="container_db_xbench.dbxml">
          <RQPlan>V(@id,=,'[to be calculated]')</RQPlan>
        </QueryPlanFunction>
        <Step axis="child" name="article" nodeType="element">
          <RQPlan>V(@id,=,'[to be calculated]')</RQPlan>
        </Step>
        <DbXmlFilter>
          <Navigation>
            <Step axis="attribute" name="id" nodeType="attribute"/>
            <DbXmlCompare name="equal">
              <Variable name="a"/>
            </DbXmlCompare>
          </Navigation>
        </DbXmlFilter>
      </Navigation>
    </ForBinding>
    <DOMConstructor type="element">
      <Name>
        <Sequence>
          <AnyAtomicTypeConstructor value="Output" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
        </Sequence>
      </Name>
      <Children>
        <Navigation>
          <Variable name="b"/>
          <Step axis="child" name="prolog" nodeType="element"/>
          <Step axis="child" name="title" nodeType="element">
            <OQPlan>P(node-element-equality-string,prefix,title)</OQPlan>
          </Step>
        </Navigation>
      </Children>
    </DOMConstructor>
  </FLWOR>
</XQuery>
```

ACERVO RESUMO DE ARTIGOS (SIGMOD)

TAB. 9.3 -Resultado UA múltiplas homogêneas (SIGMOD)

		à frio	à quente					resultado
		1 ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	136,034	117,969	117,708	118,398	118,702	118,489	118,2532
	c/índice	7,211	6,493	6,664	6,467	6,396	6,611	6,5262
C2	s/índice	159,014	141,914	141,389	141,23	141,291	141,979	141,5606
	c/índice	20,781	19,495	19,688	19,685	15,714	15,442	18,0048
C3	s/índice	121,162	104,139	102,892	102,709	103,035	102,832	103,1214
	c/índice	2,66	2,52	2,127	1,913	1,89	1,893	2,0686
C4	s/índice	145,751	150,561	125,832	144,181	126,503	127,561	134,9276
	c/índice	146,604	149,76	125,64	147,023	126,073	126,429	134,985
C5	s/índice	179,357	329,773	249,979	326,498	216,568	256,948	275,9532
	c/índice	142,168	245,262	266,497	165,798	258,987	252,772	237,8632
C6	s/índice	137,49	119,17	117,382	117,309	118,433	117,777	118,0142
	c/índice	10,844	10,325	10,265	6,54	6,582	6,651	8,0726



GRA. 9.2: Tempo de processamento das consultas com e sem índice UA múltiplas homogêneas (SIGMOD)

A escolha dos índices foi realizada com base nas consultas a serem realizadas, observando-se a estrutura dos documentos. A tabela a seguir, mostra os índices adicionados, em seguida são mostrados o plano de execução das consultas, após a inclusão dos índices.

TAB. 9.4 - Índices da UA múltipla homogênea (Currículo)

Index: node-element-substring-string for node {}:instituicao
Index: node-element-presence-none for node {}:doutorado
Index: node-element-equality-string for node {}:cidade
Index: node-element-substring-string for node {}:title
Index: node-element-equality-string edge-element-equality-string for node {}:nome
Index: unique-node-metadata-equality-string for node {http://www.sleepycat.com/2002/dbxml}:name

Para constatação dos índices utilizados, a seguir mostraremos os planos de execução associados a cada consulta.

C1: for $art in collection("container ")/IndexTermsPage/title[@id="00585"] return $art/title

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
      <OQPlan>V(node-attribute-equality-string,@id,=,'00585000')</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <Step axis="child" name="title" nodeType="element">
      <OQPlan>V(node-attribute-equality-string,@id,=,'00585000')</OQPlan>
    </Step>
    <DbXmlFilter>
      <Navigation>
        <Step axis="attribute" name="id" nodeType="attribute"/>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="00585000" typeuri="http://www.w3.org/2001/XMLSchema" typename="stri
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
    <Step axis="child" name="title" nodeType="element"/>
  </Navigation>
</XQuery>
```

C2: for $art in collection("container")/IndexTermsPage where $art/authors/author="Dick Tsur" return $art/author

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
      <OQPlan>V(node-element-equality-string,author,=,'Dick Tsur')</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <DbXmlFilter>
      <Navigation>
        <Step axis="child" name="authors" nodeType="element"/>
        <Step axis="child" name="author" nodeType="element">
          <OQPlan>V(node-element-equality-string,author,=,'Dick Tsur')</OQPlan>
        </Step>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="Dick Tsur" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
    <Step axis="child" name="author" nodeType="element">
      <OQPlan>P(node-element-equality-string,prefix,author)</OQPlan>
    </Step>
  </Navigation>
</XQuery>
```

C3: for $a in collection("container")/IndexTermsPage where $a/confyear="1993" order by $a/title return {$a/title}

```
<XQuery>
  <FLWOR>
    <ForBinding name="a">
      <Navigation>
        <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
          <OQPlan>V(node-element-equality-string,confyear,=,'1993')</OQPlan>
        </QueryPlanFunction>
        <Step axis="child" name="IndexTermsPage" nodeType="element"/>
        <DbXmlFilter>
          <Navigation>
            <Step axis="child" name="confyear" nodeType="element">
              <OQPlan>V(node-element-equality-string,confyear,=,'1993')</OQPlan>

            </Step>
            <DbXmlCompare name="equal">
              <Sequence>
                <AnyAtomicTypeConstructor value="1993" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
              </Sequence>
            </DbXmlCompare>
          </Navigation>
        </DbXmlFilter>
      </Navigation>
    </ForBinding>
    <Sort>
      <Specification modifier="ascending|empty_least">
        <Navigation>
          <Variable name="a"/>
          <Step axis="child" name="title" nodeType="element"/>
        </Navigation>
      </Specification>
    </Sort>
    <DOMConstructor type="element">
      <Name>
        <Sequence>
          <AnyAtomicTypeConstructor value="Output" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
        </Sequence>
      </Name>
      <Children>
        <Navigation>
          <Variable name="a"/>
          <Step axis="child" name="title" nodeType="element"/>
        </Navigation>
      </Children>
    </DOMConstructor>
  </FLWOR>
</XQuery>
```

C4:collection("container")/IndexTermsPage/categoryAndSubjectDescriptors [not(categoryAndSubjectDescriptorsTuple)]

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
      <OQPlan>P(node-element-presence-none,=,categoryAndSubjectDescriptorsTuple)</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <Step axis="child" name="categoryAndSubjectDescriptors" nodeType="element"/>

    <Step axis="child" name="categoryAndSubjectDescriptorsTuple" nodeType="element">
      <OQPlan>P(node-element-presence-none,=,categoryAndSubjectDescriptorsTuple)
</OQPlan>
    </Step>
    <Step axis="child" name="category" nodeType="element"/>
  </Navigation>
</XQuery>
```

C5: collection("container")/IndexTermsPage/categoryAndSubjectDescriptors
/categoryAndSubjectDescriptorsTuple/category

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
      <OQPlan>P(node-element-presence-none,=,categoryAndSubjectDescriptorsTuple)</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <Step axis="child" name="categoryAndSubjectDescriptors" nodeType="element"/>

    <Step axis="child" name="categoryAndSubjectDescriptorsTuple" nodeType="element">
      <OQPlan>P(node-element-presence-none,=,categoryAndSubjectDescriptorsTuple)
</OQPlan>
    </Step>
    <Step axis="child" name="category" nodeType="element"/>
  </Navigation>
</XQuery>
```

C6: for $a in collection("container")/IndexTermsPage/title[@id="00585"]/authors/author,
$b in collection("container")/IndexTermsPage/authors/author
where $a = $b return {$b/IndexTermsPage/title}

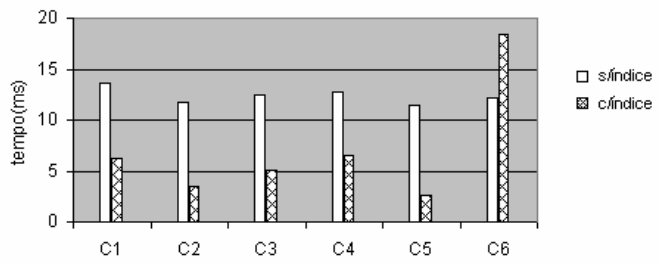
```
<XQuery>
 <FLWOR>
  <ForBinding name="a">
   <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
     <OQPlan>n(P(node-element-equality-string,prefix,author),V(node-attribute-equality-string,@id,=,'00585000'))</OQPlan>
    </QueryPlanFunction>
    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <Step axis="child" name="title" nodeType="element">
     <OQPlan>V(node-attribute-equality-string,@id,=,'00585000')</OQPlan>
    </Step>
    <DbXmlFilter>
     <Navigation>
      <Step axis="attribute" name="id" nodeType="attribute"/>
      <DbXmlCompare name="equal">
       <Sequence>
        <AnyAtomicTypeConstructor value="00585000" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
       </Sequence>
      </DbXmlCompare>
     </Navigation>
    </DbXmlFilter>
    <Step axis="child" name="authors" nodeType="element"/>
    <Step axis="child" name="author" nodeType="element">
     <OQPlan>P(node-element-equality-string,prefix,author)</OQPlan>
    </Step>
   </Navigation>
  </ForBinding>
  <ForBinding name="b">
   <Navigation>
    <QueryPlanFunction result="collection" container="container_db_sigmod.dbxml">
     <RQPlan>V(author,=,'[to be calculated]')</RQPlan>
    </QueryPlanFunction>

    <Step axis="child" name="IndexTermsPage" nodeType="element"/>
    <Step axis="child" name="authors" nodeType="element"/>
    <Step axis="child" name="author" nodeType="element">
     <RQPlan>V(author,=,'[to be calculated]')</RQPlan>
    </Step>
    <DbXmlCompare name="equal">
     <Variable name="a"/>
    </DbXmlCompare>
   </Navigation>
  </ForBinding>
  <DOMConstructor type="element">
   <Name>
    <Sequence>
     <AnyAtomicTypeConstructor value="Output" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
    </Sequence>
   </Name>
   <Children>
    <Navigation>
     <Variable name="b"/>
     <Step axis="child" name="IndexTermsPage" nodeType="element"/>
     <Step axis="child" name="title" nodeType="element"/>
    </Navigation>
   </Children>
  </DOMConstructor>
 </FLWOR>
</XQuery>
```

ACERVO CURRÍCULO

TAB. 9.5 - Resultado UA múltipla homogênea (currículo)

		à frio	à quente					resultado
		1 ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	13,881	13,967	14,091	13,727	13,67	12,704	13,6318
	c/índice	6,747	6,254	6,294	6,266	6,25	6,201	6,253
C2	s/índice	12,965	11,742	11,87	11,756	11,811	11,662	11,7682
	c/índice	3,865	3,686	3,662	3,636	3,214	3,195	3,4786
C3	s/índice	12,894	11,399	12,625	12,884	12,593	12,807	12,4616
	c/índice	5,734	5,535	5,104	4,823	4,792	4,845	5,0198
C4	s/índice	13,015	11,939	13,243	12,778	12,865	12,925	12,75
	c/índice	6,22	5,939	7,802	6,257	6,239	6,303	6,508
C5	s/índice	12,728	11,348	11,613	11,571	11,554	11,482	11,5136
	c/índice	2,189	2,287	2,427	2,449	2,449	3,13	2,5484
C6	s/índice	12,749	12,923	12,658	11,438	12,325	11,524	12,1736
	c/índice	18,839	20,042	19,387	17,653	17,614	17,687	18,4766



GRA. 9.3: Tempo de processamento das consultas com e sem índice UA múltiplas homogêneas (currículo)

A escolha dos índices foi realizada com base nas consultas a serem realizadas, observando-se a estrutura dos documentos. A tabela a seguir, mostra os índices adicionados, em seguida são mostrados o plano de execução das consultas, após a inclusão dos índices.

TAB. 9.6 -Índices da UA múltipla homogênea (currículo)

Index: node-element-substring-string for node {}:instituicao
Index: node-element-presence-none for node {}:doutorado
Index: node-element-equality-string for node {}:cidade
Index: node-element-substring-string for node {}:title
Index: edge-element-equality-string for node {}:nome
Index: unique-node-metadata-equality-string for node {http://www.sleepycat.com/2002/dbxml}:name

Para constatação dos índices utilizados, a seguir mostraremos os planos de

execução associados a cada consulta.

C1: for $cur in collection("container")
where contains($cur//instituicao,"IME") return $cur//dados_pessoais

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
      <OQPlan>V(node-element-substring-string,instituicao,=,'ime')</OQPlan>
    </QueryPlanFunction>
    <DbXmlFilter>
      <Navigation>
        <Step axis="descendant" name="instituicao" nodeType="element">
          <OQPlan>V(node-element-substring-string,instituicao,=,'ime')</OQPlan>
        </Step>
        <DbXmlContains>
          <Sequence>
            <AnyAtomicTypeConstructor value="IME" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlContains>
      </Navigation>
    </DbXmlFilter>
    <Step axis="descendant" name="dados_pessoais" nodeType="element"/>
  </Navigation>
</XQuery>
```

C2: collection("container ")[//doutorado]//nome

```
      <XQuery>
        <Navigation>
          <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
            <OQPlan>P(node-element-presence-none,=,doutorado)</OQPlan>
          </QueryPlanFunction>
          <DbXmlFilter>
            <Navigation gotoRoot="true">
              <Step axis="descendant" name="doutorado" nodeType="element">
                <OQPlan>P(node-element-presence-none,=,doutorado)</OQPlan>
              </Step>
            </Navigation>
          </DbXmlFilter>
          <Step axis="descendant" name="nome" nodeType="element"/>
        </Navigation>
      </XQuery>
```

C3: collection("container ")[//cidade="Rio de Janeiro"]//experiencia_profissional

```
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
      <OQPlan>V(node-element-equality-string,cidade,=,'Rio de Janeiro')</OQPlan>

    </QueryPlanFunction>
    <DbXmlFilter>
      <Navigation gotoRoot="true">
        <Step axis="descendant" name="cidade" nodeType="element">
          <OQPlan>V(node-element-equality-string,cidade,=,'Rio de Janeiro')</OQPlan>
        </Step>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="Rio de Janeiro" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
    <Step axis="descendant" name="experiencia_profissional" nodeType="element"/>

  </Navigation>
</XQuery>
```

C4: for $tese in collection("container")//tese where contains ($tese/title,"XML") return $tese

```
dbxml> queryPlan 'for $tese in collection("container_db_curriculum.dbxml")//tese
 where contains ($tese/title,"XML") return $tese'
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
      <OQPlan>V(node-element-substring-string,title,=,'xml')</OQPlan>
    </QueryPlanFunction>
    <Step axis="descendant" name="tese" nodeType="element"/>
    <DbXmlFilter>
      <Navigation>
        <Step axis="child" name="title" nodeType="element">
          <OQPlan>V(node-element-substring-string,title,=,'xml')</OQPlan>
        </Step>
        <DbXmlContains>
          <Sequence>
            <AnyAtomicTypeConstructor value="XML" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlContains>
      </Navigation>
    </DbXmlFilter>
  </Navigation>
</XQuery>
```

C5: collection("container")[//instituicao/nome="IME"]

```
dbxml> queryPlan 'collection("container_db_curriculum.dbxml")[//instituicao/nome ="IME"]'
<XQuery>
  <Navigation>
    <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
      <OQPlan>V(edge-element-equality-string,instituicao.nome,=,'IME')</OQPlan>
    </QueryPlanFunction>
    <DbXmlFilter>
      <Navigation gotoRoot="true">
        <Step axis="descendant" name="instituicao" nodeType="element"/>
        <Step axis="child" name="nome" nodeType="element">
          <OQPlan>V(edge-element-equality-string,instituicao.nome,=,'IME')</OQPlan>
        </Step>
        <DbXmlCompare name="equal">
          <Sequence>
            <AnyAtomicTypeConstructor value="IME" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
          </Sequence>
        </DbXmlCompare>
      </Navigation>
    </DbXmlFilter>
  </Navigation>
</XQuery>
```

C6: for $cur in collection("container") where contains ($cur//dados_pessoais/nome,”ana”)
order by $cur/dados_pessoais/nome return $cur

```
<XQuery>
  <FLWOR>
    <ForBinding name="cur">
      <Navigation>
        <QueryPlanFunction result="collection" container="container_db_curriculum.dbxml">
          <OQPlan>P(edge-element-equality-string,prefix,dados_pessoais.nome)</OQPlan>
        </QueryPlanFunction>
        <DbXmlFilter>
          <Navigation>
            <Step axis="descendant" name="dados_pessoais" nodeType="element"/>
            <Step axis="child" name="nome" nodeType="element">
              <OQPlan>P(edge-element-equality-string,prefix,dados_pessoais.nome)</OQPlan>
            </Step>
            <DbXmlContains>
              <OQPlan>P(node-element-equality-string,prefix,nome)</OQPlan>
              <Sequence>
                <AnyAtomicTypeConstructor value="maria" typeuri="http://www.w3.org/2001/XMLSchema" typename="string"/>
              </Sequence>
            </DbXmlContains>
          </Navigation>
        </DbXmlFilter>
      </Navigation>
    </ForBinding>
    <Sort>
      <Specification modifier="ascending|empty_least">
        <Navigation>
          <Variable name="cur"/>
          <Step axis="child" name="dados_pessoais" nodeType="element"/>
          <Step axis="child" name="nome" nodeType="element">
            <OQPlan>P(edge-element-equality-string,prefix,dados_pessoais.nome)</OQPlan>
          </Step>
        </Navigation>
      </Specification>
    </Sort>
    <Variable name="cur"/>
  </FLWOR>
</XQuery>
```

9.2 APÊNDICE 2: UNIDADE DE ARMAZENAMENTO ÚNICA HETEROGÊNEA

Contém os mesmos índices incluídos para as unidades de armazenamento única homogênea, sendo que armazenados em uma única unidade.

TAB. 9.7 - Índices da UA única heterogênea

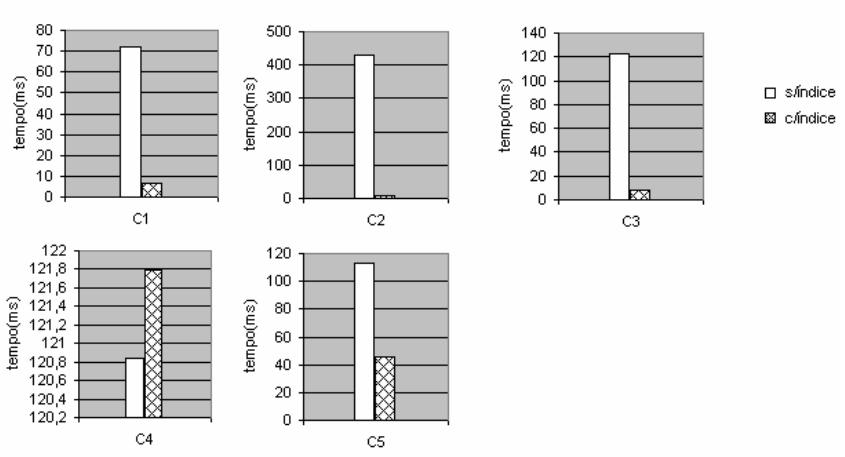
Index: node-element-equality-string for node {}:author
Index: node-element-presence-none for node {}:categoryAndSubjectDescriptorsTuple
Index: node-element-equality-string for node {}:cidade
Index: node-element-equality-string for node {}:confyear
Index: node-element-equality-string for node {}:country
Index: node-element-presence-none for node {}:doutorado
Index: node-element-presence-none for node {}:genre
Index: node-attribute-equality-string for node {}:id
Index: node-element-substring-string for node {}:instituicao
Index: node-element-equality-string for node {}:name
Index: unique-node-metadata-equality-string for node {http://www.sleepycat.com/2002/dbxml}:name
Index: edge-element-equality-string for node {}:nome
Index: node-element-substring-string for node {}:title

ACERVO ARTIGOS COMPLETOS (XBENCH)

TAB. 9.8 - Resultado ua única heterogênea (XBENCH)

		à frio	à quente					resultado
		1ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	121,032	72,958	73,533	71,358	71,614	71,496	72,1918
	c/índice	7,595	6,721	6,664	6,601	6,868	6,677	6,7062
C2	s/índice	567,223	430,246	427,706	430,479	427,425	431,107	429,3926
	c/índice	7,348	7,017	6,976	6,959	7,065	7,011	7,0056
C3	s/índice	269,459	126,74	123,75	122,746	121,029	120,914	123,0358
	c/índice	10,062	9,258	9,158	9,275	8,852	6,060	8,5206
C4	s/índice	261,195	120,694	120,473	120,509	121,096	121,423	120,839
	c/índice	265,466	122,156	121,169	121,774	121,620	122,222	121,7882
C5	s/índice	256,184	112,119	112,878	113,34	112,939	113,357	112,9266
	c/índice	123,,273	47,899	47,373	43,723	43,417	43,508	45,184
C6	s/índice	1.31E+11	1.30E+11	NF	NF	NF	NF	-------
	c/índice	42075,3	40471,2	40645	40733,8	40512,300	41578,700	40788,2

NF – Não finalizado

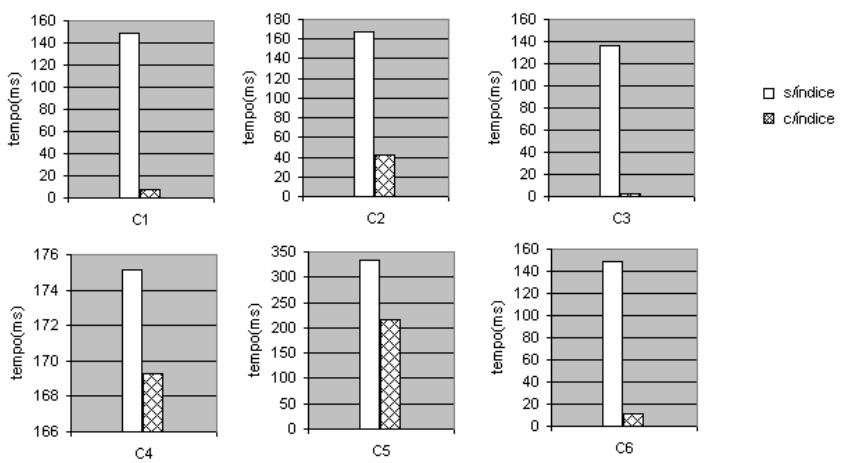


GRA. 9.4: Tempo de processamento das consultas com e sem índice UA única heterogêneas (XBENCH)

ACERVO RESUMO DE ARTIGOS (SIGMOD)

TAB. 9.9 - Resultado UA única heterogênea (SIGMOD)

		à frio	à quente					resultado
		1ª exec	2ª exec	3ª exec	4ª exec	5ª exec	6ª exec	média a quente
C1	s/índice	197,222	149,591	149,068	148,427	148,215	147,289	148,518
	c/índice	8,818	7,91	7,935	7,733	7,731	7,756	7,813
C2	s/índice	216,479	169,106	166,931	167,672	167,013	166,934	167,5312
	c/índice	47,281	44,726	45,2	41,959	38,128	40,185	42,0396
C3	s/índice	181,858	137,337	135,176	137,022	136,625	136,65	136,562
	c/índice	3,039	2,966	2,937	2,926	2,959	3,116	2,9808
C4	s/índice	204,179	205,02	155,948	182,288	157,09	175,302	175,1296
	c/índice	204,36	197,916	152,539	191,054	152,75	152,272	169,3062
C5	s/índice	236,19	468,172	456,302	238,256	253,621	244,962	332,2626
	c/índice	147,391	257,878	264,731	143,679	163,514	246,911	215,3426
C6	s/índice	194,895	150,8	148,448	148,448	146,704	148,036	148,4872
	c/índice	32,746	12,219	11,46	11,839	11,931	11,846	11,859

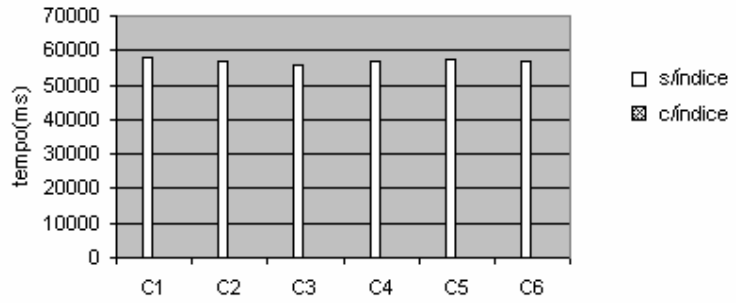


GRA. 9.5: Tempo de processamento das consultas com e sem índice UA única heterogênea (SIGMOD)

ACERVO CURRICULO

TAB. 9.10 Resultado UA única heterogênea (currículo)

		à frio	à quente					resultado
		1ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	57027,3	57835,8	59164,2	58409	58044,6	57357,2	58162,16
	c/índice	10,166	9,388	9,345	9,366	9,477	9,516	9,4184
C2	s/índice	57745,6	57130,9	56865	57071	57176,3	56175,2	56883,68
	c/índice	26,144	24,244	23,912	23,993	24,247	20,779	23,435
C3	s/índice	56356,5	56487,3	56055	55668,1	55696,7	55512	55883,82
	c/índice	6,152	5,36	5,125	5,234	5,014	3,785	4,9036
C4	s/índice	57069,4	57359,9	56376,8	57167,1	56930	56802,1	56927,18
	c/índice	10,188	9,317	9,079	9,481	9,328	9,136	9,2682
C5	s/índice	58268	57100,7	57158,7	56856,2	57157	59729,7	57600,46
	c/índice	4,234	3,953	3,999	3,925	3,922	3,896	3,939
C6	s/índice	57159	56844,2	56755,5	56544,7	56531,8	57156,5	56766,54
	c/índice	24,905	22,444	22,505	22,136	20,174	17,868	21,0254



GRA. 9.6: Tempo de processamento das consultas com e sem índice UA única heterogênea (currículo)

9.3 APÊNDICE 3: UNIDADES DE ARMAZENAMENTO MÚLTIPLAS HETEROGÊNEAS

Contém os mesmos índices incluídos para as unidades de armazenamento múltiplas homogêneas e única heterogênea, replicadas por todas as Unidades de armazenamento.

TAB. 9.11 - Índices das UA's múltiplas heterogêneas

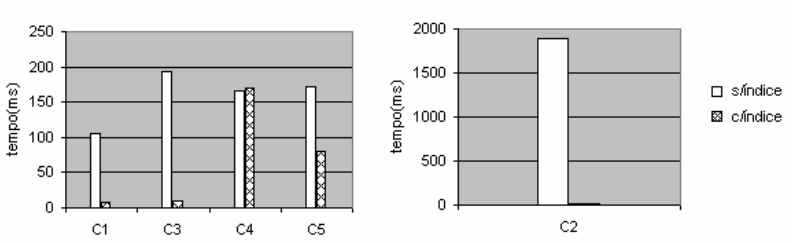
Index: node-element-equality-string for node {}:author
Index: node-element-presence-none for node {}:categoryAndSubjectDescriptorsTuple
Index: node-element-equality-string for node {}:cidade
Index: node-element-equality-string for node {}:confyear
Index: node-element-equality-string for node {}:country
Index: node-element-presence-none for node {}:doutorado
Index: node-element-presence-none for node {}:genre
Index: node-attribute-equality-string for node {}:id
Index: node-element-substring-string for node {}:instituicao
Index: node-element-equality-string for node {}:name
Index: unique-node-metadata-equality-string for node {http://www.sleepycat.com/2002/dbxml}:name
Index: edge-element-equality-string for node {}:nome
Index: node-element-substring-string for node {}:title

ACERVO ARTIGOS COMPLETOS (XBENCH)

TAB. 9.12 - Resultado UA múltipla heterogênea (XBENCH)

		à frio	à quente					resultado
		1ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	4970,43	104,176	103,726	102,936	106,883	105,54	104,6522
	c/índice	8,995	8,572	8,677	8,324	5,888	5,834	7,459
C2	s/índice	6104,26	1747,43	1822,68	1859,3	1924,92	2065,15	1883,896
	c/índice	21,394	20,394	12,931	12,583	12,52	12,687	14,223
C3	s/índice	350,514	192,063	192,524	196,814	196,229	192,666	194,0592
	c/índice	12,053	11,219	11,018	11,4	8,429	8,448	10,1028
C4	s/índice	315,666	156,82	170,728	171,324	167,055	167,871	166,7596
	c/índice	309,237	170,251	174,055	173,102	167,224	165,796	170,0856
C5	s/índice	319,183	173,232	175,333	172,858	163,366	172,11	171,3798
	c/índice	158,649	85,958	78,517	79,72	78,582	79,208	80,397
C6	s/índice	NF	NF	NF	NF	NF	NF	NF
	c/índice	98272,1	98657,9	109332,0	107334,0	109180,0	106190	19731,58

NF – Não finalizado

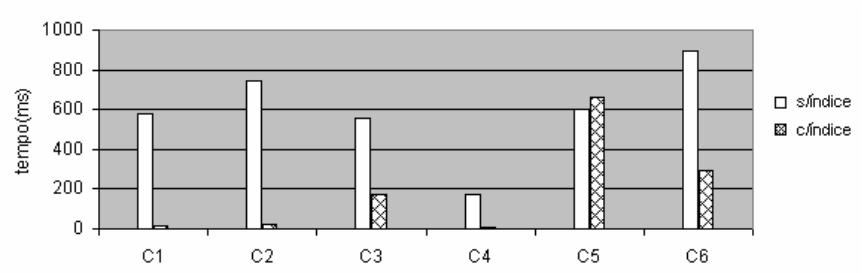


GRA. 9.7: Tempo de processamento das consultas com e sem índice UA múltipla heterogênea (XBENCH)

ACERVO RESUMO DE ARTIGOS (SIGMOD)

TAB. 9.13 - Resultado UA múltipla heterogênea (SIGMOD)

		à frio	à quente					resultado
		1ª exec	2ªexec	3ªexec	4ªexec	5ªexec	6ªexec	média a quente
C1	s/índice	249,246	516,89	521,788	520,139	664,303	687,532	582,1304
	c/índice	14,373	13,369	13,372	13,263	13,32	13,316	13,328
C2	s/índice	278,948	632,523	654,707	753,203	803,536	876,051	744,004
	c/índice	28,382	25,926	20,878	20,773	20,843	20,422	21,7684
C3	s/índice	235,172	478,956	485,885	559,828	608,965	651,614	557,0496
	c/índice	8,631	7,889	5,894	5,717	5,869	5,724	6,2186
C4	s/índice	232,954	568,751	625,818	692,099	553,08	579,027	603,755
	c/índice	238,373	582,441	622,082	626,467	730,363	757,464	663,7634
C5	s/índice	267,344	725,19	871,929	829,76	923,696	1118,89	893,893
	c/índice	172,71	268,65	293,232	236,738	328,99	345,6	294,642
C6	s/índice	250,29	511,023	520,854	600,052	627,954	665,121	585,0008
	c/índice	14,623	13,486	13,446	13,466	13,483	13,65	13,5062



GRA. 9.8: Tempo de processamento das consultas com e sem índice UA múltipla heterogênea (SIGMOD)

ACERVO CURRICULO

A execução sem a utilização de índices não foi finalizada e com a utilização de índices apresentou erros.

10 ANEXOS

10.1 ANEXO 1: FUNÇÃO PARA PROCESSAMENTO DO ALGORITMO FCM NO MATLAB

```
function FCM_GDXML(m)

% ------------------------------------------------------------------------------------------------------------------------
% Leitura da Matriz de dados
% cada coluna contem a ocorrência ou frequência de um termo no documento XML
% cada linha contem todos os atributos de um documento indicando a ocorrência ou freqüência deste.
% A Coluna 1 guarda os códigos dos documentos; esta coluna não será analisada
% Da segunda coluna em diante temos os dados a serem analisados
% ------------------------------------------------------------------------------------------------------------------------
Dados = load('matriz.txt');
COD_DOC = Dados(:, 1 );
Dados = Dados(:, [2:end]);

% ------------------------------------------------------------------------------------------------------------------------
% No caso de freqüência os dados devem ser normalizados
% Busca-se o maior valor da coluna; a normalização será feita dividindo-se a freqüência do atributo
% pelo maior valor existente para o atributo sendo lido ; assim teremos para todos os documentos
% um valor de atributo entre 0 e 1
% ------------------------------------------------------------------------------------------------------------------------
Maior_Freq_atrib = max(Dados);
[M N] = size(Dados);
for j = 1:N
   for i = 1:M
      Dados_N(i,j)=Dados(i,j)/Maior_Freq_atrib(1,j);
   end
end

% ------------------------------------------------------------------------------------------------------------------------
% Algoritmo
% ------------------------------------------------------------------------------------------------------------------------
[center,U,obj_fcn] = fcm(Dados_N,3,2);

maxU = max(U);


% ------------------------------------------------------------------------------------------------------------------------
% Processamento dos Cluster
% ------------------------------------------------------------------------------------------------------------------------

U(1,:)
U(2,:)
U(3,:)


% ------------------------------------------------------------------------------------------------------------------------
% cluster1 ( Calcula graus de Pertinencia)
% ------------------------------------------------------------------------------------------------------------------------
index1 = find(U(1,:) == maxU);
u1=zeros(1,size(index1));
for i = 1:size(index1)
   u1(1,i) = U(1,index1(i));
end
n1=zeros(1,size(index1));
for i = 1:size(index1)
  n1(1,i) = COD_DOC(index1(i));
end

% ------------------------------------------------------------------------------------------------------------------------
% cluster2 (Calcula  graus de Pertinencia)
% ------------------------------------------------------------------------------------------------------------------------
index2 = find(U(2,:) == maxU);
u2=zeros(1,size(index2));
for i = 1:size(index2)
   u2(1,i) = U(2,index2(i));
end
n2=zeros(1,size(index2));
for i = 1:size(index2)
   n2(1,i) = COD_DOC(index2(i));
end

% ------------------------------------------------------------------------------------------------------------------------
% cluster3(Calcula  graus de Pertinencia)
% ------------------------------------------------------------------------------------------------------------------------
index3 = find(U(3,:) == maxU);
u3=zeros(1,size(index3));
for i = 1:size(index3)
   u3(1,i) = U(3,index3(i));
end
n3=zeros(1,size(index3));
for i = 1:size(index3)
   n3(1,i) = COD_DOC(index3(i));
end
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo