MULTI-DIMENSIONAL INDEXING FOR XML DATA

by

DO YOUN KIM

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2005

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervising professor, Dr. Ramez Elmasri. He always gave me motivation and new ideas for my thesis. I am really thankful for his help and constant support during the course of my thesis.

I wish to thank my committee members, Dr. Leonidas Fegaras and Dr. Gautam Das, for their spending time with me and their suggestions.

I would also like to thank my family for supporting me and always showing me love. I would also like to thank all my friends in Korea and the United States who provided helpful suggestions and powerful encouragement.

November 2, 2005

ABSTRACT

MULTI-DIMENSIONAL INDEXING FOR XML DATA

Publication No. ______

Do Youn Kim, M.S.

The University of Texas at Arlington, 2005

Supervising Professor: Ramez Elmasri

The Extensible Markup Language (XML) is becoming a dominant standard for exchanging and retrieving data over the internet. As XML in data exchanging grows, various XML indexing techniques have been proposed for fast and efficient query processing. A survey of theses techniques [9] categorized them into sequence-based indexes, structural indexes, dimension-based indexes and keyword-based indexes based on their properties. In this thesis, we focus on the multi-dimensional XML indexing; more precisely, it can be categorized as a 2-dimension based indexing, which is used in this paper. We studied the properties of multi-dimensional XML indexing and created an implementation to evaluate its performance. It performs well with arbitrary path expressions and retrieves results from a relational database, which is stable and safe. We

also compare the performance of the 2-dimensional indexing with the structure based indexing, which stores summary information of XML documents' tree nodes in main memory.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Introduction

Extensible Markup Language (XML) is quickly becoming the de facto standard for exchanging data over the internet. For XML data, the structure of data sources can be diverse and the schemas are not predefined. One reason why XML has been successful over the internet may be the data type underlying the XML paradigm, namely the tree [33]. Being represented by hierarchical tree, efficient and elegant algorithms, especially recursive algorithms, can be applied to XML. XML has been widely adopted as a universal data exchange format over the World Wide Web.

With the increasing popularity of XML for data representation, a lot of research is going on concerning efficient accessing and querying of XML data. Several query languages for XML [1, 5, 6, 15, 19, 28, 38] have been proposed recently. XML Query languages such as XQuery [5], XPath [38], Lorel [28], and Quilt [6] use path expressions to traverse XML data graph. The target nodes will be extracted if the path among the tree nodes matches the sequence of labels of the path expression. However, the target nodes can be scattered at different locations in the XML document and the physical disk storage. If queries for partial matching were executed, the naïve search takes superfluous navigation and is exhaustive. The performance of query processing will be degraded and inefficient. Thus, optimizing cost of the navigation on XML data graph is important when processing XML queries; indexing can be use to improve the performance of XML query processing.

XML indexing can speed up the performance of query processing. Depending on the structure and size of XML files, the expected reliability and performance vary, and different indexing techniques may be applied to get satisfactory results. Hence, extra flexibility in indexing is needed to query XML data.

1.2 Goals of the Thesis

A lot of research is going on concerning XML indexing and several techniques have been proposed to improve it. In [23], a survey of XML indexing techniques is provided and the techniques are categorized into:

- Sequence based indexes [21, 32]
- Structure based indexes [9, 10, 17, 22, 26, 27]
- Dimension based indexes [24, 33]
- Keyword based indexes [7, 14, 35]

Formerly, [32] focused on the sequence based indexing and compared three different sequence based indexing techniques on ViST [12]. In [17], N. Manandhar focused on the structure based indexing techniques. The paper proposed to use offset and size to store node information and evaluated query performance on different sizes of XML files. In this paper, the focus will be on multi-dimensional XML indexing techniques.

The contributions of this thesis are:

• Implementation of multi-dimensional indexing, based on 2 dimensions

• Proposal of techniques of XML document reconstruction

• Design and implementation of query processing algorithm

• Comparison with structured based indexing performance

This thesis is a part of a project to evaluate and compare XML indexing techniques. Thus, the goal of this paper is to contribute to our collection of XML indexing techniques by adding on multi-dimension indexing techniques for XML data. Subsequent work will focus on comparing the various techniques.

1.3 Thesis Outline

The rest of this paper is organized as follows. In Chapter 2, we present an overview of XML, XPath and XQuery. We also survey XML indexing techniques. In Chapter 3, we discuss multi-dimensional XML indexing. Chapter 4 describes the implementation of 2-dimensional XML index, document loading, query processing, and reconstruction of XML for presenting query results. In Chapter 5, we present experimental results, and compare 2-dimension indexing with structure based XML indexing. We overview related work in Chapter 6. Finally, Chapter 7 concludes the contribution of this paper.

CHAPTER II

OVERVIEW OF XML

In this chapter we will discuss about XML, XPath, and XQuery. XML data is a semi-structured in nature. To processing queries on XML data, several languages have been proposed. We will overview on XPath and XQuery. Also we discuss about various XML indexing techniques in section 2.4.

2.1 XML

XML is an abbreviation of Extensible Markup Language. Exactly speaking, it is a language to define markup language rather than a simple markup language. That is, contrary to HTML, XML gives names to elements to describe data. The structure of XML data is not predefined.

There are three types of XML documents, which are data-centric, document-centric, and mixed content. Data-centric XML documents are usually highly structured and marked up with XML tags. They are used in sales orders, patient records, and scientific data. Document-centric XML documents are used in user manuals, static Web Pages, and Marketing flyers with its SGML-like capabilities. Mixed-Content XML Documents are hybrid of two types of XML documents. Sample XML file is shown in Figure 2.1

```xml
<?xml version="1.0"?>
<department>
	<departname>Computer Science</departname>
	<gradstudent>
		<name>
			<lastname>Kim</lastname>
			<firstname>Do Youn</firstname>
		</name>
		<phone>4054104960</phone>
		<email>dkim@cse.uta.edu</email>
	</gradstudent>
	<faculty>
		<name>
			<lastname>Elmasri</lastname>
			<firstname>Ramez</firstname>
		</name>
		<email>elmasri@cse.uta.edu</email>
		<office>GACB</office>
	</faculty>
</department>
```

Figure 2.1 An example of XML document

To describe data in XML document, the validation should matter. Each element tag should be matched with a closed tag and nested properly. Also optional DTD (Document Type Definition) can be used to define the structure of document. Once XML document references DTD, it can not work correctly without the external schema. Thus, XML documents should be well-formed and validated always.

```
<!ELEMENT department (departname,gradstudent*,faculty*)>
<!ELEMENT gradstudent (name,phone,email)>
<!ELEMENT name (lastname,firstname)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT faculty (name,email,office)>
<!ELEMENT name (lastname,firstname)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT office (#PCDATA)>
```

Figure 2.2 An example of XML DTD

2.2 XPath

XPath expressions are well adapted for XML query language. XPath is used to extract interesting nodes out of the XML tree. It uses a set of syntax rules to identify nodes in an XML document.

XPath query consists of location paths and optional predicates. A location path can be absolute or relative. That means context node (starting point of node) should not be a root node. A sequence of location steps, syntactically separated by '/', is used to identify a set of nodes in an XML documents. Location steps consist of two parts, which are an Axis and a Node Test. There are thirteen XPath axes available:

Table 2.1 XPath Axes

Axis	Node-set
ancestor	contains the ancestors of the context node
ancestor-or-self	contains the context node and the ancestors of the context node
attribute	contains the attributes of the context node
child	contains the children of the context node
descendant	contains the descendants of the context node
descendant-or-self	contains the context node and the descendants of the context node
following	contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes
following-sibling	contains all the following siblings of the context node
namespace	contains the namespace nodes of the context node
parent	contains the parent of the context node, if there is one
preceding	contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes
preceding-sibling	contains all the preceding siblings of the context node
self	contains just the context node itself

An Axis defines a node-set relative to the current node and tells which direction to travel from the context node to next nodes. On the other hand, a Node Test identifies a node within an axis. XPath also has optional part of zero or more predicates to modify the set of selected nodes. XPath axes will be more discussed in the Chapter 3.

The syntax for XPath expression is: **Axisname::nodetest [predicate]**

/department [departname='cs']/gradstudent/name
→ gives names of all gradstudents who are under department named 'cs'

/department/*/phone
→ gives any phone number under department

//faculty/email
→ gives all email addresses of faculties in any place of the document

Figure 2.3 Examples of XPath expressions

// is short for /descendant-or-self::node( )/. For example, //email is short for /descendant-or-self::node( )/child::email and so will select any email element in the document.

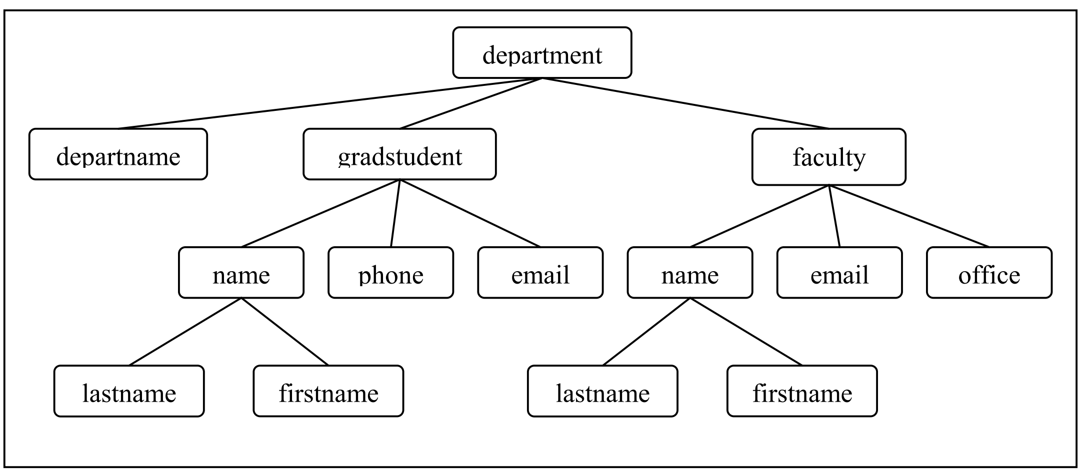

Figure 2.4 Hierarchical tree structures

XML documents can be represented in hierarchical tree structure. The tree representation of the XML document in Figure 2.1 is as shown in Figure 2.4

To traverse a XML tree, there are two parameters in necessity: a context node, which is the starting point in the tree, and a sequence of location steps evaluated from left to right at a time. Results of XPath query may be empty, a single node or contain several nodes.

2.3 XQuery

XQuery is influenced by ODMG and OQL and evolved from Quilt [6]. XQuery is purely a functional language for querying XML, processing XML data and it evaluates to a value. It is a superset of XPath 1.0 but does not support all its axes. It only supports self, parent, child, descendant-or-self, descendant and attribute nodal relationships. It supports the same data models, operators and functions as XPath.

XQuery may access elements from XML documents and construct new values (elements). Query nesting is allowed at any place and on any level. XQuery has formal semantics based on the XML abstract data model.

A simple example of XQuery is as follows:

- Doc(“cseuta.xml”)/department/departname

The difference between XPath and XQuery is that XPath expressions may return a node set whereas the same XQuery expression will return node sequence.

There are 7 types of expressions in XQuery [36]: path expression, Element constructors, FLWOR expressions, operators and functions expressions, conditional expressions, quantified expressions, and modifying data type expressions. The FLWOR expression is similar to the select-from-where statement in SQL query. The expression forms the skeleton of the XQuery expression.

XQuery supports data types required for XML documents using node values in addition to the primitive data types. XQuery uses FLWOR expressions which consists of For (loops over the elements of a sequence), Let (binds variable to entire list expression), Where (optional – applies predicate), Order by (imposes an order) and Return (gives the result) [23]. The example of FLWOR expressions is in Figure 2.5.

```
<gradstudents>{
   for $b in document('cseuta.xml')//gradstudent
   where $b/name/firstname = 'Do Youn'
     and $b/name/lastname = 'Kim'
   return <gradstudent>
            $b/phone,
            $b/email
         </gradstudent>
}</gradstudents>
```

Figure 2.5 FLWOR expressions in XQuery

A query in XQuery is an expression that reads a sequence of atomic values and returns a sequence. The expression in XQuery can be path expression, element constructors, function calls, arithmetic and logical expressions, conditional expressions, quantified expressions, expressions on sequences, expressions on type. The various expressions can be used together both sequentially and nested. The simplest query is XPath expression.

XQuery also supports user-defined functions whose definitions appear in the query prolog as well as the built-in functions. The function parameters and results could be sequences, nodes or primitive values.

XQuery has several comparison operators. It supports sort by and conditional statements like if-then-else. The queried written in XQuery are successfully adopted for the type of data that XML documents have.

2.4 Survey of XML Indexing Techniques

In recent years, a lot of interest in the indexing of XML data has resulted in many papers that try to improve the performance querying XML data. In [23], a survey of XML indexing techniques is provided and the techniques are categorized into sequence based indexes, structure based indexes, dimension based indexes, and keyword based indexes.

XML indexing techniques can be categorized into two; one is indexing techniques using a relational database [7, 24, 33] and the other is those using byte offset and size [9, 10, 14, 21, 22, 26, 27, 35]. In the former techniques, XML document is preprocessed and stored in a relational database, and query results are reconstructed from the relational database. In the latter techniques, XML document elements are indexed using byte <offset, size> to retrieve the element test from the document.

In sequence based indexing, the XML data is transferred into sequences using either depth-first traversal or breadth first traversal [21]. Then the sequence will be correspondent with the related nodes in XML tree [32].

In structure based indexes, structural summary will construct a summary graph based on the notion of bisimilarity. Structural summary of graph has been proposed by authors in [9, 10, 22, 26, 27]. Two nodes are bisimilar if the path labels into them are identical. Path expressions can be directly evaluated in the summary graph and they can retrieve label matching nodes without referring to the original data graph [17].

The authors in [7, 14, 35] have suggested keyword based indexing. The keyword based indexing focuses on keyword search in which users do not have to learn any schema or query language. They use distance between keywords; and distance between keyword and the resulting XML element to find the proximity among keywords.

In dimension based indexing, the pre-order rank (if necessary, with post-order rank) will be used to locate interesting nodes on the XML tree. XISS [24], a new system for indexing and storing XML data based on a linear numbering scheme for elements, quickly determines the ancestor-descendant relationship between elements in the hierarchy of XML data. The author proposed three new algorithms to calculate a regular path expression: EE-Join for scanning sorted element to another, EA-Join for scanning sorted elements and attributes to find element-attribute pairs, and KC-Join for finding Kleene-Closure on repeated paths or elements, highly effective particularly for searching paths that are very long or whose lengths are unknown. Torsten Grust has proposed a new index structure, XPath Accelerator [33], mapping every element node onto the two-dimension plane. It supports the multi-dimensional indexing by assigning each node with the pre-order and the post-order on the x-axis and the y-axis respectively. The two-dimension plane can be divided into the four major axes of XPath steps: descendant, ancestor, following, and preceding. It can be implemented using a relational database system and performed especially well if the database system supports spatial indexing techniques such as R-tree.

Although XML query processing in the main memory is attractive due to its fast loading the source, it might be performed expensively by the limitation of the main memory to process larger XML files such as 10MB, 100MB, and so on. In that respect, the advantage of storing XML data into the database can be told as the alternative solution of it. If a XML file like Shakespeare's plays is big and less-frequently update, the overall cost of the multi-dimensional XML indexing storing in a relation database is quite reasonable. We implement multi-dimensional XML indexing to store node information in a relational database and propose a technique of the XML file reconstruction using JAVA vector class. By transforming the original query into a flat n-ary self-join SQL query, one time access to the relational database is enough to get the results directly from the database. Database indices, such as B-trees, are optimized to support four major axes responding to rectangular regions queries in the pre/post plane.

CHAPTER III

MULTI-DIMENSIONAL INDEXING FOR XML DATA

As defined in [23], one of the categories in XML indexing techniques is dimension-based indexing. Two remarkable researches, interval-based indexing [24] and multi-dimensional indexing [33], are going on concerning dimension-based XML indexes. The common point of two papers is that they use Preorder and Postorder to represent XML tree. In this thesis, multi-dimensional indexing is focused. The author of [33] proposed a new index structure, XPath Accelerator, to support the multi-dimension approach. In section 3.1, we review the Pre/Postorder and encoding document regions. XPath Accelerator proposed the two callback procedures, which is based on SAX [29] parser and stack S, to load an XML document instance into the database. We will look at the B-tree supported XPath Accelerator and this chapter is based on materials from [2, 3, 4, 11, 13, 20, 21, 29, 33, 34]

3.1 XPath Document Regions

In this section, we will discuss more about the XPath Axes, which is mentioned in the section 2.2, and review the encoding XPath document regions.

3.1.1 XPath Axes

XPath language is based on the path expression to traverse tree-shaped XML data. Each XML tree node can be one of several node kinds that are element, attribute, text, comment, processing instruction. A well-formed XML document always has the proper nesting of start and end tags.

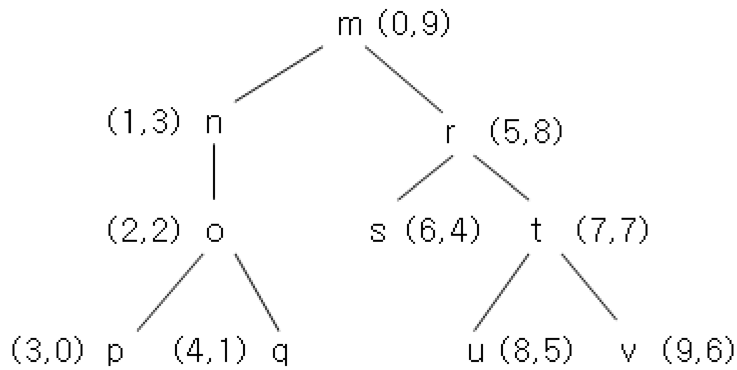

Figure 3.1 Preorder/Postorder rank assignment

XPath path expressions specify a tree traversal by two parameters:

(1) a sequence of context nodes which notifies the starting point of the traversal

(2) a list of steps which is syntactically separated by / and evaluated from left to right.

For each context node, a step's axis establishes a subset of document nodes. In [33], this subset of document has been called a document region. The subsets are combined together and sorted by document order to give the starting point of the traversal to the next XPath path step.

There are four axes which will be focused: descendant, ancestor, following and preceding. To make an easy identification, these four axes will be called as major axes from now on.

3.1.2 XML Document Partitions

The four major axes (descendant, ancestor, following and preceding) of the any context node are partitioned by the given node. Regardless of choice of x, the node set contains every single document node exactly once.

x/descendant ∪ x/ancestor ∪ x/following ∪ x/preceding ∪ {x}

If the choice of node x is r in Figure 3.1, then,

(r/descendant ∪ r/ancestor ∪ r/following ∪ r/preceding ∪ {r}) = {m,⋯,v}.

This document partitioning is to create an index structure such that, for any given context node, the set of nodes will be determined in the four document partitions. Those partitions are associated with four XPath axes. The rest of the XPath axes (parent, child, descendant-or-self, ancestor-or-self, following-sibling, and preceding-sibling) determine specific supersets or subsets of these node sets which are easy to characterize [34].

Due to this partitioning property of the four major axes, the multi-dimensional indexing is designed. That is, the multi-dimensional indexing can be represented inside the database because of the property that each document node will be contained exactly once.

These discrete partitions in XML document tree motivate to call them with document regions, which will be continually discussed on the next stage.

3.1.3 Encoding XML Document Regions

To support tree-shaped node hierarchy, the encoding has to map the input XML document into a domain in which a node's region. To encode the XML tree nodes, an XML document needs to:

(1) establish the region notion induced by the four major XPath axes, and

(2) be efficiently supported by relational databases host.

To accommodate easy encoding the hierarchical tree nodes, efficient number scheme should be used. Pre/post order numbering scheme is proposed in [20] and applied into multi-dimensional based indexes [24, 33]. The document order of the nodes is not easy to be determined because a sequential read of the textual XML representation of the instance informally. However, preorder traversal of the document tree has much more useful characterization of document order. A tree node x is visited and given its preorder rank pre(x) before its children are recursively traversed from left to right.

In Figure 3.1, left and right ranks in the parentheses are the preorder and the postorder respectively. In preorder traversal, the document order is $m < n < o < p < q < r < s < t < u < v$, and thus pre(m) = 0, …, pre(v) = 9. In postorder traversal, a node x is assigned its postorder rank post(x) after all its children have been traversed from left to right. Thus the postorder is post(p) = 0, post(q) = 1, …, post(m) = 9 (see Figure 3.1 for the complete pre/postorder rank assignment).

In [4, 20, 24], one can use pre(x) and post(x) to efficiently characterize the descendants x' of the node x.

x' is a descendant of x

$pre(x) < pre(x') \wedge post(x') < post(x)$

During a sequential read of the XML document instances, we have seen the start tag <x> before <x'> and the end tag </x> after </x'>. Thus, the element corresponding to x' is a subset of the element corresponding to x.

Figure 3.2 shows that a given context node r partitions the other nodes on the plane into four groups, AC, FL, PR, and DC. This characterizes the descendant axis of context node x, but we can use pre(x) and post(x) to characterize all four major axes in an equally simple manner.

Node r induces a partition of the plane into four disjoint regions:

(1) the lower-right partition DC contains all descendants of r,

(2) the upper-left partition AC traces the ancestors of r,

(3) the lower-left partition PR hosts the nodes preceding r, and

(4) the upper-right partition FL represents the nodes following r

The rest of the XPath axes (parent, child, descendant-or-self, ancestor-or-self, following-sibling, and preceding-sibling) determine specific supersets or subsets of these node sets

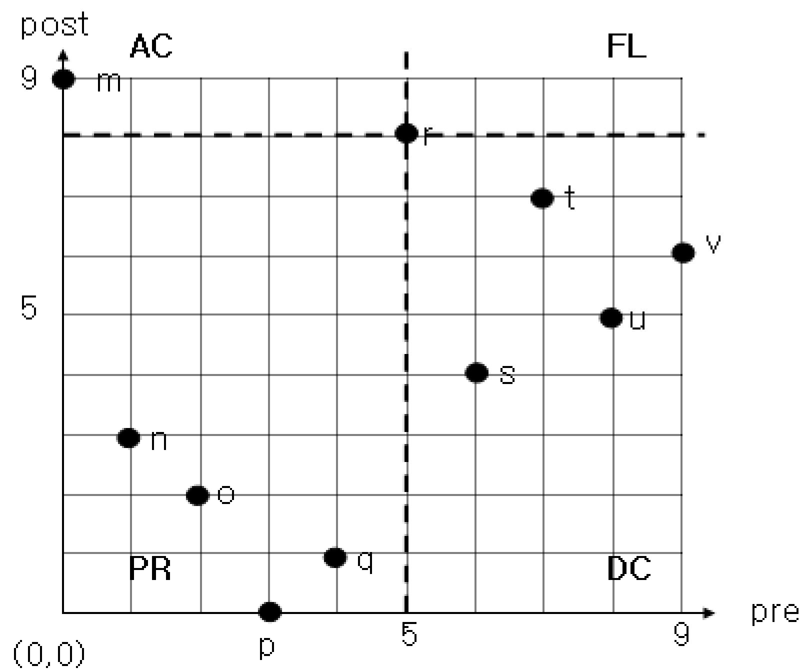

Figure 3.2 Node distributions in the pre/post plane

For any node in XML document tree, this characterization is applied. On this pre/post plane, any node x can be the start point of the XPath traversal. This is the important feature especially for XQuery because XQuery is a fully compositional query language [5]. Arbitrary expressions yield arbitrary context node sequences where XPath traversal may resume (e.g. iteration construction with for).

Evaluating a step along a major axis is querying a rectangular region in the pre/post plane. Database indexes are highly optimized to support this kind of query.

To complete node tests, extra node information should be kept in the databases:

- name(x) which stores the element tag name or attribute name of node x if x is not empty

• kind(x) which is one of {node, elem, attr, text, comment, processing-instruction}

These two factors completes the encoding represented by its 5-dimesional descriptor desc(x) = <pre(x), post(x), par(x), kind(x), name(x)>.

Table 3.1 XPath axes α and corresponding query windows *window*(α,x) with context node x

Axis α	Query window *window*(α,x)				
	pre	post	par	kind	name
child	<(pre(x),∞)	(0,post(x))	pre(x)	elem	* >
descendant	<(pre(x),∞)	(0,post(x))	*	elem	* >
descendant-or-self	<(pre(x),∞)	(0,post(x))	*	elem	* >
parent	<(par(x), par(x))	(post(x), ∞)	*	elem	* >
ancestor	<(0,pre(x))	(post(x), ∞)	*	elem	* >
ancestor-or-self	<(0,pre(x))	(post(x), ∞)	*	elem	* >
following	<(pre(x),∞)	(post(x), ∞)	*	elem	* >
preceding	<(0,pre(x))	(0,post(x))	*	elem	* >
following-sibling	<(pre(x),∞)	(post(x), ∞)	par(x)	elem	* >
preceding-sibling	<(0,pre(x))	(0,post(x))	par(x)	elem	* >
attribute	<(pre(x),∞)	(0,post(x))	pre(x)	attr	* >

An XPath axis corresponds to a specific query window in the space of node descriptors. Table 3.1 summarizes the window together with the corresponding axes. A * entry indicates a don't care match which always succeeds. The example of application to a kind test τ is

window(preceding::text(),x) = <(0,pre(x)),(0,post(x)), *, text, *>.

The encoding shows as if there were only one document, whereas a system may store many. The multiple documents can be stored into one global document by using a global root node that has various documents as its children. The important thing is query windows should be executed within document boundaries by storing minimum and maximum preorder and postorder rank.

3.2 XML Instance Loading

To load an XML document instance into the database, the nodes can be mapped into the 5-dimensional descriptor space. Each XML document node should be stored exactly once so that the size of the loaded index will be linear in the size of the input instance.

All five components of the node descriptors can be computed during a single sequential parsing by using SAX [29]. SAX is the Simple API for XML, originally a Java-only API. SAX was the first widely adopted API for XML in Java, and is a "de facto" standard.

During loading the size of temporary memory is bounded by the XML document heights, not by the size of it. Two callback procedures are proposed by the author of [33]. A stack S is maintained to keep track of elements whose opening tag has been read but whose closing tag is still to come. When the closing tad comes to be read, its postorder is specified and the 5-descriptor node is ready to be inserted into the database table accel.

No additional temporary memory space is needed because sequential but yet partial node information is computed within an application. The stack operations, push, pop, top, and empty are used to practice the two callback procedure shown in Figure 3.3.

```
startElement(t, a, atts)
x ← <pre = gpre, post = null, par = (S.top()).pre, att = a, tag = t>;
S.push(x);
Gpre ← gpre + 1;
For x' IN atts DO
        startElement(x', true, nil);
        endElement(x');

endElement(t)
x ← S.pop();
x.post ← gpost + 1;
insert x into table accel;
```

Figure 3.3 Two callback procedure with stack S

Loading is initiated as follows:

gpre = 0; gpost = 0;

S.empty();

S.push(<pre = -1, post = null, par = null, att = null, tag = null>);

SAXparseFile();

S.pop();

In the procedures, t is a tag name, a is a boolean value against attribute and atts is a list of attribute names (parameters of these will be discussed in SAX of the next chapter). All the null values in the loading initiation are undefined values.

All the attributes x' in an attributes list is associated with element and treated like XML elements. The attribute nodes will be distinguished from element nodes by boolean values of att in 5-dimensional descriptors.

The actual element content (CDATA) of an XML document can be treated like an additional child of its containing element by storing right next to the containing node or maintained in a separate table cdata <pre, cdata> by establishing pre as a foreign key referencing the accel table.

If the DTD of the input XML document is used, a simple translation table can be set up to map element names to numerical values before loading starts.

When updating the accel table, renumbering the preorder and postorder should be performed. To delete a node, it is sufficient to remove its' descriptor entry from table accel.

3.3 XPath Evaluation

The evaluation is based on the relational SQL (Structured Query Language). Assume that the node descriptors of XML document are loaded into 5-column table accel. If e denotes an XPath path expression and α denotes an axis, SQL query scheme is defined

```
query(e/α) = SELECT x'.*
             FROM query(e) x, accel x'
             WHERE x' INSIDE window(α, x).
```

INSIDE, which mimics a SQL keyword, symbolizes the window test. This recursive translation scheme is provided by any subset of node descriptors in table accel (if e is an absolute path expression) or by the document root (the only node t is with pre(x) = 0).

The translation scheme generates an SQL query of nesting depth n for a path expression of n steps. The example for the XPath expression /descendant::n1/follwoing -sibling::n2 is

```
SELECT x2.*
FROM accel x1, accel x2
WHERE 0 < x1.pre
AND x1.tag = n1
AND x2.pre > x1.pre AND x2.post > x1.post
AND x2.par = x1.par
AND x2.tag = n2.
```

3.4 R-tree Supported XPath Acceleration

XPath Accelerator has been originally designed to be supported by a multi-dimensional access method. Although XPath Accelerator with two B-tree indices is implemented as 2-dimensional XML indexing, XPath performance is better if R-tree based backend would be available. Performances between two setups are compared each other in [33].

A typical node distribution in the pre/post plane for an XML instance of 100 nodes is shown in Figure 3.4 (left). Many nodes are packed on the diagonal of the plane, while upper left is sparsely populated and lower right is completely empty.

In Figure 3.4 (right), coverage and overlapping of the R-tree leaves are minimized (leaf capacity is 6 nodes). With R-tree packing techniques [13], node descriptors are inserted in preorder value at the cost of using temporary storage for sorting. This leads to 100% storage utilization in the R-tree leaves and improves query performance considerably [33].

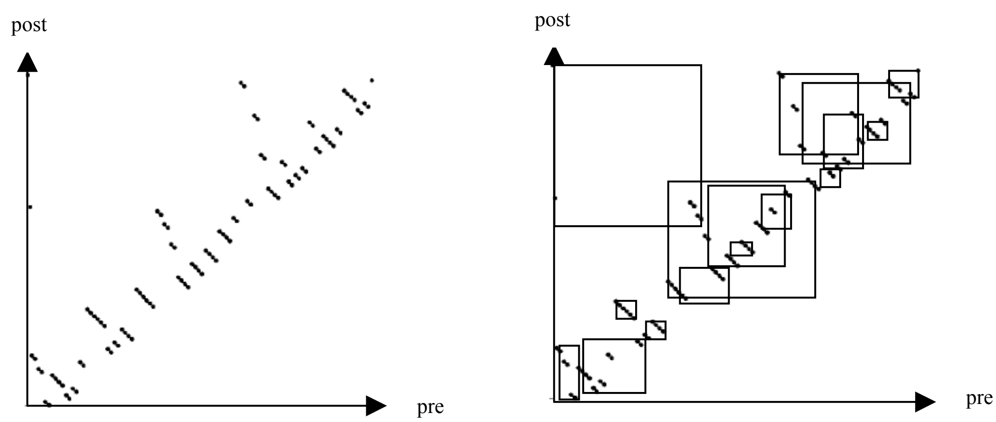

Figure 3.4 Example of a pre/post rank distribution(left) and leaf level of a preorder packed R-tree(right)

R-tree packing in preorder has other benefits that the R-tree window queries returned the result nodes in ascending preorder (in document order). For XQuery specification demands document order on forests resulting from path expressions, this saves the implementation of sorting. For XPath, preorder packing with R-tree facilitates the implementation of context positions [38], which are used in XPath predicates.

CHAPTER IV

IMPLEMENTATION OF 2-DIMENSIONAL XML INDEX

The XPath accelerator [33] has been implemented in two ways; R-tree implementation and B-tree implementation. For the R-tree implementation, R-tree should be supported by database host. In [33], R-tree supported XPath acceleration is optimized by the XML bulk-loading process and R-tree packing techniques [13]. B-tree implementation is associated with B-tree indices. Two ascending B-tree indices on the pre and post columns of the accel table are created to sort query result in document order. B-tree implementation using 2-dimensional plane motivated us to call 2-dimensional XML indexing, whereas R-tree XPath accelerator is supported by a multi-dimensional access method. In this thesis B-tree supported XPath accelerator is implemented. For the both systems use the same base of storage structure and evaluation scheme, 2-dimensional XML indexing is part of multi-dimensional indexing.

4.1 Storage Structure

For 2-dimensional indexing, the same storage structure has been used as described in section 3.1.3. A XML tree node x is converted into node descriptor desc(x) and stored into a 5-column table accel with <pre, post par, kind, tag>. The information of a descriptor will complete a record in accel table of 2-dimensional implementation.

CDATA is not maintained a separate table as mentioned in section 3.3. If 'cseuta.xml' file in Figure 2.1 is stored into the table accel then it should be like the Table 4.1.

Table 4.1 Relational table accel records with 5-dimensional descriptors

pre	post	par	kind	tag
2	0	1	c	Computer Science
1	1	0	e	departname
6	2	5	c	Kim
5	3	4	e	lastname
8	4	7	c	Do Youn
7	5	4	e	firstname
4	6	3	e	name
10	7	9	c	4054104960
9	8	3	e	phone
12	9	11	c	dkim@cse.uta.edu
11	10	3	e	email
3	11	0	e	gradstudent
16	12	15	c	Elmasri
15	13	14	e	lastname
18	14	17	c	Ramez
17	15	14	e	firstname
14	16	13	e	name
20	17	19	c	elmasri@cse.uta.edu
19	18	13	e	email
22	19	21	c	GACB
21	20	13	e	office
13	21	0	e	faculty
0	22	-1	e	department

The table accel has 5 columns to accommodate 5-dimensional descriptors. Column pre contains preorder of instance node, column post contains postorder of instance node, and column par contain the parent node of the context node. Those tree columns get an input as an integer value. Column kind contains one single character data to represent the record is either element node (e) or character data (c). In the column tag, either a real tag name or CDATA can be stored that is tokenized from XML parser (attribute instance are excluded to test for this implementation). Note that each preorder and postorder are unique among their categories, whereas the value of par can be repeated. The very last record of the accel table is a root node and the value of the par record is '-1' for initiation.

DDL (Data Definition Language) for the table accel associated with "cseuta.xml" file is,

```
create table accel_cseuta (
    pre integer, post integer, par integer,
    kind char(1), tag text);
```

The table name is altered by 'Accelerator.java' file to identify easily among the multiple tables accel in RDBMS. 'Accelerator.java' is main java file to load XML document to database. There are no primary keys in this table but two B-tree indices are created on pre and post columns right after.

4.2 Loading Implementation

Node descriptors are stored in the relation database for XML querying. 2-dimensional indexing uses relational technology to accelerate XPath query processing. For loading implementation of 2-dimensional index, MySQL and SAX have been used. Also JDBC is mentioned in this section.

4.2.1 Overview of MySQL

MySQL is a popular Open Source SQL database management system, which is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility. The SQL part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases and is defined by the ANSI/ISO SQL Standard.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

The InnoDB transactional storage engine has been stable since version 3.23.49. InnoDB is being used in large, heavy-load production systems. MySQL 3.22 had a 4GB (4 gigabyte) limit on table size. With the MyISAM storage engine in MySQL 3.23, the maximum table size was increased to 65536 terabytes ($256^7 - 1$ bytes). With this larger allowed table size, the maximum effective table size for MySQL databases is usually determined by operating system constraints on file sizes, not by MySQL internal limits [16].

For this implementation, MySQL 4.1.14 is used. This will be reviewed in section 5.1 Introduction and Experimental Setup.

4.2.2 Overview of SAX

XML records are loaded in XML parsers, such as a DOM [8] and SAX [29], to be stored in a file. The DOM parses the XML input document, and constructs the complete XML document tree in memory. XML application then issues DOM library calls to explore, manipulate, or generate XML documents.

Unlike DOM parser, SAX (Simple API for XML) is not a W3C standard. It is "de facto" standard API which has been developed jointly by members of the XML-DEV in about 1998. SAX is originally a Java-only API, but there are versions for several programming language environments other than Java [29].

SAX parser is not limited by memory because its processor does not maintain the whole tree data structure. Instead, the SAX parser sends events to the application whenever a certain chunk of XML data has been recognized. For XML data simply consist of Character and ASCII data, SAX parser forms the mass of textual data into chunk, usually 2048 characters.

```
<?xml version="1.0">
<speech>
  <!– Enter OTHELLO and IAGO -->
  <line from="IAGO" to="OTHELLO">
     Will you think so?
  </line>
</speech>
```

Figure 4.1 Sample XML document for SAX

To find out about the start and end of the document, SAX provides methods called 'startDocument' and 'end Document'. The SAX parser calls procedure 'startElement' whenever it receives notification of the beginning of an element. The 'startElement' method has name and atts as parameters. The parameter name contains the element type name and the parameter atts contains the attributes attached to the element, if any. The SAX parser calls procedure 'endElement' whenever it receives notification of the end of an element. The procedure 'endElement' has parameters name, which contains the element type name. The parameter procedure 'characters' whenever it receives notification of character data. This procedure contains parameters ch, start and length. The ch contains the characters from the XML document, start contains the start position in the array, and length contains the number of characters to read from the array. The SAX events are shown in Figure 4.1.

Table 4.2 Sample SAX Event with parameters

Event	Parameters sent
startDocument	
startElement	t = "speech"
comment	c = " Enter OTHELLO and IAGO "
startElement	t = "line", ("from", "IAGO"), ("to", "OTHELLO")
characters	buf = "Will you think so?", len = 18
endElement	t = "line"
endElement	t = "speech"
endDocument	

4.2.3 JDBC

JDBC is short for "Java DataBase Connectivity". JDBC technology is an API that provides cross-DMBS connectivity to a wide range of SQL databases. It makes it possible to do three things:

- establish a connection with a database or access any tabular data source such as spreadsheets or flat files
- send SQL statements, and
- process the results.

MySQL provides connectivity for Java applications via a JDBC driver. This driver allows directly communicating between applications developed in Java programming language by using MySQL protocol. The driver is downloadable from [16] and a simple tutorial can be found there.

4.3 Indexing using Relational Techniques

A combination of B-tree indices leads to a good performance. The two ascending B-tree indices on the pre and post columns of the accel table are created to query results that were sorted in document order.

XPath axis query window is searched using two independent B-tree range scans on both the pre and post indices. In [33], the author introduces Shrink-Wrap for // axis to reduce query window.

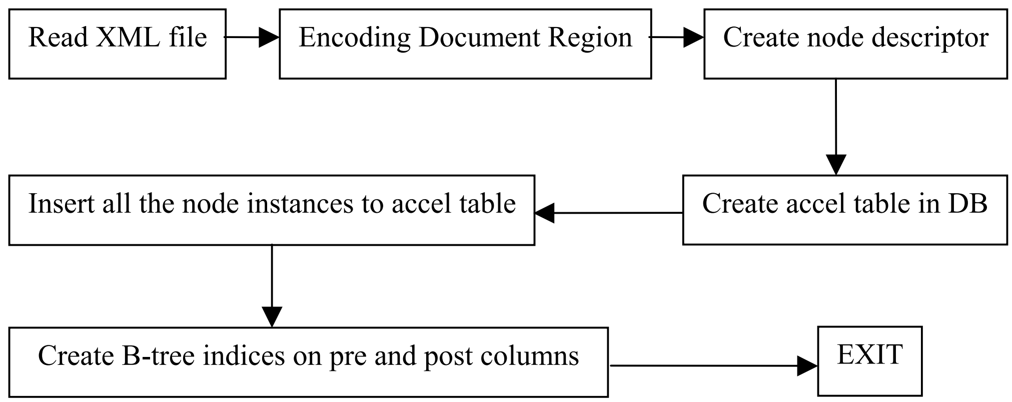

Figure 4.2 Construction of 2-dimensional index

Figure 4.2 shows the steps to create a relational techniques indexing, 2-dimensional index. An XML documents is parsed by SAX parser and the each node is recorded to DB table as soon as the postorder is determined. The step to create B-tree indices on pre and post columns can be done right after an accel table is created in database.

4.4 Querying Implementation

The main part of querying implementation is to create a SQL statement that represents XPath query accordingly. If a query is '/department/gradstudent/name', then the query is parsed and tokenized the string to 'department', 'gradstudent', and 'name', separated by /. Since the SQL statement is informed three steps in the query, declaration alternative relation names v2, v1, and v0 is needed to create a self join SQL statement. For different relations, v2, v1, and v0, the tuples that satisfy the v2.tag="department" and v2.pre=v1.par and v1.tag="gradstudent" and v1.pre=v0.par and v0.tag="name" and v0.kind="e" will be retrieved. The complete SQL statement for the query is

```
SELECT DISTINCT v0.*
 FROM accel v2,accel v1,accel v0
 WHERE
 v2.tag="department" and v2.pre=v1.par and
 v1.tag="gradstudent" and v1.pre=v0.par and
 v0.tag="name" and v0.kind="e";
```

In the case of an absolute location '//', the translation to the SQL statement is much easier. As the SQL query is simple, the result is extracted very quickly. For example the query of '//name' is translated to

```
SELECT DISTINCT v0.*
 FROM accel v0
 WHERE
 v0.tag="name" and v0.kind="e";
```

For the arbitrary path like '/department/*/name', this '*' means the any node that is a descendant node of the department. Wild card '*' yields a where condition v0.pre>v1.pre and v0.post<v1.post in SQL statement. The condition, 'v0.pre>v1.pre and v0.post<v1.post' gives the result node set of 'descendant' in major document regions.

2-dimensional index is implemented with the automated SQL translator. The SQL statement for '/department/*/name/' is as follows.

```
SELECT DISTINCT v0.*
 FROM accel v1,accel v0
 WHERE
 v1.tag="department" and v1.pre!=v0.par and
 v0.pre>v1.pre and v0.post<v1.post and
 v0.tag="name" and v0.kind="e";
```

A query with predicate like '/department[deptname=Computer Sciences]/gradstudent' is not much different as other SQL statement. Since it has four substring variables, 'department', 'deptname', 'Computer Sciences', and gradstudent', four different aliases [25] (tuple variables) are needed. The statement of SQL query is

```
SELECT DISTINCT v0.*
 FROM accel v3,accel p3,accel c3,accel v0
 WHERE
 v3.tag="department" and v3.pre=v0.par and
 p3.tag="deptname" and v3.pre=p3.par and
 c3.pre=p3.pre+1 and c3.tag="Computer Sciences" and
 v0.tag="gradstudent" and v0.kind="e";
```

The aliases (tuple variables) are needed as many as the number of tokenized substrings that have to be evaluated. The query processing steps are shown in Figure 4.3.

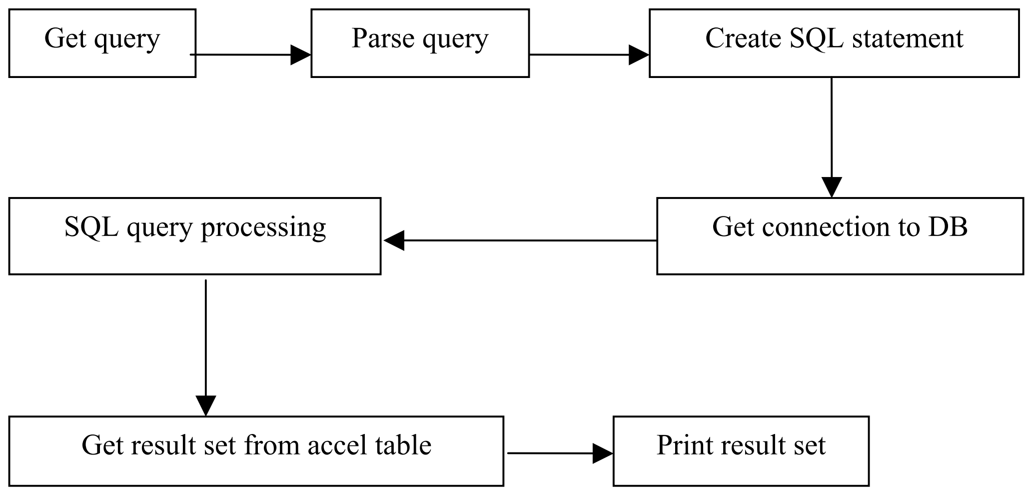

Figure 4.3 Query processing steps

First thing to query is to get a valid query from users (there are some restricted XPath expressions for this implementation). Then parsing query with implemented Java coding by StringTokenizer. The created SQL statement is sent and executed in DB using JDBC. Then the result set is able to be collected from DB. The result set is printed in the last step of Figure 4.3, yet the reconstruction is needed to finalize the querying process.

4.5 Reconstruction

Although the resulting nodes are queried, it is not easy to see the contents of the nodes. Each node of the result node set gives pre and post order. When one node is chosen to see the contents, some procedures are required to see the all descendants node of the selected node. Since unnecessary accesses to database occur overhead, a vector class has been used to store all the descendant nodes from database. The simplified function DescNodesToVector is shown in 4.2.

```
// Storing Descendant Nodes to Vector
DescNodesToVector (int preorder, int postorder){
Get connection with MySQL;
s = "SELECT v.*
        FROM accel_cseuta v
        WHERE v.pre>preorder and v.post<=postorder;
Send SQL statement(s);
Get results from DB;
Store records to Vector NV;}
```

Figure 4.4 Storing Descendant Nodes to Vector

By using the function above, a selected node will save descendants or self nodes, which contains the context node and the descendants of the context node, to the vector NV in ascending postorder. In this way, the processing is executed in main memory rather than calculated between main memory and database, back and forth.

DescNodesToVector has two parameters; preorder and postorder as integer values. The SQL statement is constructed to get all the descendant nodes and the node itself from the Document Region.

After storing descendant nodes information to Vector NV, a procedure is needed to print descendant nodes and self node in appropriate order. A function PrintVectorToXML in Figure 4.3 is presented to show the procedure to reconstruct XML document from a context node.

```
// PrintVectorToXML Fuction(Reconstruction)
PrintVectorToXML(Vector t){
  1: Print the node with open tags whose parent value is the minimum;

  2: Find a node whose parent node is node of 1;
    (while doing 4: node from 3)
  3: Print the node of 2 with open tags;
  4: do 2-3 until it gets to print Character data;

  5: Find a node whose preorder value is the parent value of node 3;
   (while doing 7: node from 6)
  6: Print the node of 5 with close tags;
  7: do 5-6 until it gets to find a node with same parent node of node 6;

  8: If the node from 7 is not printed out yet, Go to 2;
  9: Else then Ends
}
```

Figure 4.5 Reconstruction of XML document

For the XML document is structured that one node may contain other nodes, open tag should wait for close tag until those contained nodes are completely printed out.

Line 1 gets to print root node of the result document. Then a procedure of line 2-4 prints open tags and character data and a procedure of line 5-7 prints closing tag. Line 8 will check sibling nodes to print.

CHAPTER V

EXPERIMENTAL RESULTS

In this section, we describe the experimental setup and XML data that are used in our experiment. Querying performance will be evaluated and compared with the performance of structured-base indexing, which is a previous work of our project.

5.1 Experimental Setup

A machine with a Pentium 4 processor, 2.8GHz, 512 MB of RAM and 60 GB hard drive has been used. For the relational database, MySQL 4.1.14 has been used on the computer. For XML parser, SAX 2.0 has been used. Coding has been done in a Java programming language communicating that MySQL via JDBC driver that the MySQL's site provides [16].

5.2 Data

Two different XML files are used to evaluate query performances. The 2-dimentional indexing does not support the instance of the attributes of XML documents and DTD. Therefore some modification may be done on the original data sets.

The first XML data set is downloaded from the website of Niagara Experimental Data, The University of Wisconsin Madison [18]. A XML data 'medicine.xml' has 2840 nodes and the file size is 83 KB. It contains information about students, staff and faculty within a department.

The second XML data set is Shakespeare's plays, 'othello.xml'. It is distributed by Jon Bosak [31] and contains 6194 nodes in a 251 KB file. The XML document contains a lot of various acts and scenes of a play. Both two data sets are widely used in various experiments.

Table 5.1 XML Data sets

Data Set	Nodes	Labels	Height	Size on disk
medicine.xml	2840	18	4	83 KB
othello.xml	6194	14	5	251KB

5.3 Queries and Results

XML documents are evaluated with XPath queries given in Table 5.2. Wild card '*', '/', '//', and predicate [x = value] have been used for queries.

Table 5.2 Sample Queries

Datasets	Query	Path Expressions
Medicine	Q1	//gradstudent/email
	Q2	//phone
	Q3	//gradstudent/address/city
	Q4	/department/*/phone
	Q5	/department/gradstudent/address/city
Othello	Q6	//stagedir
	Q7	/play/act/title
	Q8	/play/*/title
	Q9	/play/act[title=ACT I]/scene
	Q10	/play/act[title=ACT I]/scene/speech
	Q11	/play/act[title=ACT I]/scene/speech/line
	Q12	//act[title=ACT I]/scene/speech/line
	Q13	//act[title=ACT I]/scene/stagedir

In [17], the results of the experiment vary depending on the value k of A(k)-index. The research to obtain query results with A(2)-index for medicine was performed the best as well as accurately. So, the result set of A(2)-index for medicine is chosen to be compared with the query results of 2-dimensional indexing for the same data set, medicine. The comparisons of two indexes are shown in Table 5.4.

To make a fair comparison of query performance between two indexes, the reconstruction time is added to the data retrieval time in Table 5.4 and 5.6. The denomination of data retrieval time for Table 5.3 and 5.5 is millisecond. Regardless of the number of resulting nodes, the reconstruction time varies from 20ms to 70ms.

Table 5.3 Data Retrieval time on Medicine

Query	Data Nodes	2-dimensional	Nodes in Result set
Q1	2840	30	74
Q2	2840	32	256
Q3	2840	29	72
Q4	2840	40	256
Q5	2840	29	74

Table 5.4 Query Performance on Medicine

Query	Structure based	2-dimensional	Nodes in Result set
Q1	260	60	74
Q2	614	52	256
Q3	338	74	72
Q4	645	62	256
Q5	177	49	74

Table 5.5 Data Retrieval time on Othello

Query	Data Nodes	2-dimensional	Nodes in Result set
Q6	6194	44	208
Q7	6194	57	5
Q8	6194	55	26
Q9	6194	54	3
Q10	6194	68	163
Q11	6194	121	737
Q12	6194	112	737
Q13	6194	36	129

Table 5.6 Query Performance on Othello

Query	Structure based	2-dimensional	Nodes in Result set
Q6	625	74	208
Q7	208	82	5
Q8	1822	75	26
Q9	266	124	3
Q10	1053	108	163
Q11	412	151	737
Q12	377	142	737
Q13	344	71	129

The query results are significantly improved by performed with 2-dimensional indexing. For the result of Q2, 2-dimensional indexing is nineteen times faster than structured based indexing (the denomination of the time elapsed is millisecond).

The query results of the different A(k)-indexes for Othello are presented in multiple tables of [17]. Among the query results of the different A(k)-indexes for Othello, the best correct results are chosen to be compared with the query results of 2-

dimensional indexing for Othello. The comparisons of two indexes are shown in Table 5.6.

For the elapsed time for Q8, the Structure based indexing takes 1822 milliseconds to execute, whereas 2-dimensional indexing takes 55 milliseconds. 2-dimensional indexing outperforms the Structured based indexing by 33 times for the query (Manandhar of [17] mentioned that the actual total time should have been received by deducting the printing node time from the elapsed time of the query results. However, only the 5 nodes are printed for Q7. So, the time recorded on Q7 is close to the time taken to access it.)

As the result, Q7 shows that the elapsed time is not exactly proportional to the nodes of the result set for 2-dimensional indexing. Q7 takes more time than Q6 for the 2-dimensional indexing, even though the resulting nodes of Q7 are fewer than those of Q6. This is because SQL with the JOIN operation for the database still has to check the records in the table to extract interesting nodes. Two factors determine the query processing time; the number of resulting nodes and the complexity of the JOIN operation in SQL.

Considering the loading time for 2-dimensional indexing into a database, the improved results in the query performance are inevitable. However, there are many XML documents less-frequently updated such as SIGMOD records [30]. The XML data may be updated in every quarter at most. Shakespeare's Plays are immutable and the size of the XML documents can be multiple MBs.

2-dimensional indexing is well suited for the huge documents that are not frequently changeable to provide efficient and reliable performance for query processing.

CHAPTER VI

RELATED WORK

This chapter will discuss about some other works which are closely related to dimension-based indexing.

In [24], a new system has been proposed for indexing and storing XML data based on a numbering scheme [20] for elements. This numbering scheme is extended Preorder Numbering Scheme. This numbering scheme quickly determines the ancestor-descendant relationship between elements in the hierarchy of XML data.

Also new path-join algorithms have been proposed in [24] to efficiently process regular path expression queries for XML data. The EE-Join algorithm is supposed to be highly effective particularly for searching paths that are very long or whose lengths are unknown. The prototype system implementation can be test in [37].

In [2], an index over the prefix-encoding of the paths in an XML document tree is presented. In prefix-encoding, each leaf *l* of the document tree is prefixed by the sequence of element tags that one encounters during a path traversal form the document root to *l*. The authors of [2] proposed so-called refined paths to remedy the drawback that arbitrary paths need multiple index lookups or post-processing phase.

In [9], T-index structure is proposed by Milo and Suciu to maintain equivalence classes of document nodes that are indistinguishable with respect to a given path template. T-index represents only parts of documents relevant to a specific path template rather than the whole document tree. T-index allows to trade space for generality.

In [11], relational interval tree (RI-tree) is tailored to efficiently respond to interval queries. The RI-tree could be a promising candidate to index the pre/post plane because B-trees supported by database host suffice to query the RI-tree efficiently.

CHAPTER VII

CONCLUSION

In this chapter, we will discuss the conclusion and contribution that this thesis made. Also the future work is mentioned.

7.1 Conclusion and Contribution

This thesis has been motivated by the requests of efficient information retrieval from XML documents. A lot of researches are going on indexing techniques to improve performance of query processing.

XISS [37] project and XPath Accelerator [33] using Pre/Post plane motivate us to call dimension-based indexing. We used SAX parser and MySQL database to implement the B-tree supported XPath Accelerator.

In [17], N. Manandhar focused on the structure based indexing techniques. The paper proposed to use offset and size to store node information and evaluated query performance on different sizes of XML files. The implementation of this thesis used the same data set, 'medicine.xml' and 'othello.xml', to be compared easily with the queries performance of structured based indexing. The result is that the initial loading time into database takes time tremendously but it outperforms the structure based indexing. The other beneficial effect of dimension based indexing is to overcome main memory's limitation that the implementation of structured based indexing suffers.

This thesis is a part of a project to evaluate and compare XML indexing techniques. Thus, the goal of this paper is to contribute to our collection of XML indexing techniques by adding on multi-dimension indexing techniques for XML data. Thus, the main contributions of this thesis are to implement the multi-dimensional indexing, based on 2 dimensions and to use relational techniques for XML indexing.

7.2 Future Work

The query processor of 2-dimensional indexing does not support attribute nodes. However, 'Accelerator.java' includes code for loading and storing attributes. An attribute will be treated as a document node and stored into a relational database table before its parent's child nodes or leaf node (if, any) are stored. To execute XPath query processor for predicates, additional coding is needed within 'Query.java'.

For the implementation of 2-dimensional index, DTD (Document Type Definition) on XML documents should be supported to recognize the various document schemes. Study of SQL query optimization may improve performance of query process. Also, more statistical data sets should be used to evaluate both implementations (e.g. data sets that are generated from XML generators).

As part of our on going project on survey of XML indexing techniques [23], the remaining indexing techniques and their behavior should be compared by the statistical data sets.

APPENDIX A

PARTIAL PREPROCESSED XML DATA FILE

Results of "medicine.xml"

SELECT DISTINCT v0.*
FROM accel_medicine v1,accel_medicine v0
WHERE
v1.tag="gradstudent" and v1.pre=v0.par and
v0.tag="email" and v0.kind="e";
Query1=//gradstudent/email
11 10 3 e email
34 33 26 e email
55 54 47 e email
78 77 70 e email
99 98 91 e email
120 119 112 e email
141 140 133 e email
164 163 156 e email
185 184 177 e email
206 205 198 e email
227 226 219 e email
250 249 242 e email
273 272 265 e email
294 293 286 e email
315 314 307 e email
338 337 330 e email
359 358 351 e email
380 379 372 e email
401 400 393 e email
422 421 414 e email
443 442 435 e email
464 463 456 e email
485 484 477 e email
509 508 501 e email
530 529 522 e email
551 550 543 e email
572 571 564 e email
595 594 587 e email
616 615 608 e email
637 636 629 e email
658 657 650 e email
679 678 671 e email
700 699 692 e email
723 722 715 e email
744 743 736 e email
765 764 757 e email
788 787 780 e email
809 808 801 e email
832 831 824 e email
853 852 845 e email
876 875 868 e email
897 896 889 e email
920 919 912 e email
941 940 933 e email
962 961 954 e email
983 982 975 e email
1004 1003 996 e email
1025 1024 1017 e email
1046 1045 1038 e email
1069 1068 1061 e email
1090 1089 1082 e email
1111 1110 1103 e email
1134 1133 1126 e email
1155 1154 1147 e email
1178 1177 1170 e email
1201 1200 1193 e email
1222 1221 1214 e email
1245 1244 1237 e email
1266 1265 1258 e email
1287 1286 1279 e email
1308 1307 1300 e email
1331 1330 1323 e email
1354 1353 1346 e email
1377 1376 1369 e email
1400 1399 1392 e email
1421 1420 1413 e email
1442 1441 1434 e email
1463 1462 1455 e email
1484 1483 1476 e email
1505 1504 1497 e email
1528 1527 1520 e email
1549 1548 1541 e email
1572 1571 1564 e email
1595 1594 1587 e email

Total Query Time: 0.50(sec) Number of Result Nodes: 74

SELECT DISTINCT v0.*
FROM accel_medicine v0
WHERE
v0.tag="phone" and v0.kind="e";
Query2=//phone
9 8 3 e phone
32 31 26 e phone
53 52 47 e phone
76 75 70 e phone
97 96 91 e phone
118 117 112 e phone
139 138 133 e phone
162 161 156 e phone
183 182 177 e phone
204 203 198 e phone
225 224 219 e phone
248 247 242 e phone
271 270 265 e phone
292 291 286 e phone
313 312 307 e phone
336 335 330 e phone
357 356 351 e phone
378 377 372 e phone
399 398 393 e phone
420 419 414 e phone
441 440 435 e phone
462 461 456 e phone
483 482 477 e phone
507 506 501 e phone
528 527 522 e phone
549 548 543 e phone
570 569 564 e phone
593 592 587 e phone
614 613 608 e phone
635 634 629 e phone

656 655 650 e phone
677 676 671 e phone
698 697 692 e phone
721 720 715 e phone
742 741 736 e phone
763 762 757 e phone
786 785 780 e phone
807 806 801 e phone
830 829 824 e phone
851 850 845 e phone
874 873 868 e phone
895 894 889 e phone
918 917 912 e phone
939 938 933 e phone
960 959 954 e phone
981 980 975 e phone
1002 1001 996 e phone
1023 1022 1017 e phone
1044 1043 1038 e phone
1067 1066 1061 e phone
1088 1087 1082 e phone
1109 1108 1103 e phone
1132 1131 1126 e phone
1153 1152 1147 e phone
1176 1175 1170 e phone
1199 1198 1193 e phone
1220 1219 1214 e phone
1243 1242 1237 e phone
1264 1263 1258 e phone
1285 1284 1279 e phone
1306 1305 1300 e phone
1329 1328 1323 e phone
1352 1351 1346 e phone
1375 1374 1369 e phone
1398 1397 1392 e phone
1419 1418 1413 e phone
1440 1439 1434 e phone
1461 1460 1455 e phone
1482 1481 1476 e phone
1503 1502 1497 e phone
1526 1525 1520 e phone
1547 1546 1541 e phone
1570 1569 1564 e phone
1593 1592 1587 e phone
1616 1615 1610 e phone
1635 1634 1629 e phone
1654 1653 1648 e phone
1673 1672 1667 e phone
1692 1691 1686 e phone
1711 1710 1705 e phone
1730 1729 1724 e phone
1749 1748 1743 e phone
1768 1767 1762 e phone
1787 1786 1781 e phone
1806 1805 1800 e phone
1825 1824 1819 e phone
1844 1843 1838 e phone
1863 1862 1857 e phone
1882 1881 1876 e phone
1901 1900 1895 e phone
1920 1919 1914 e phone
1939 1938 1933 e phone
1958 1957 1952 e phone
1977 1976 1971 e phone
1996 1995 1990 e phone
2015 2014 2009 e phone
2034 2033 2028 e phone
2053 2052 2047 e phone
2072 2071 2066 e phone
2091 2090 2085 e phone
2110 2109 2104 e phone
2129 2128 2123 e phone
2148 2147 2142 e phone
2167 2166 2161 e phone
2186 2185 2180 e phone
2205 2204 2199 e phone
2224 2223 2218 e phone
2243 2242 2237 e phone
2262 2261 2256 e phone
2281 2280 2275 e phone
2300 2299 2294 e phone
2319 2318 2313 e phone
2338 2337 2332 e phone
2357 2356 2351 e phone
2376 2375 2370 e phone
2395 2394 2389 e phone
2414 2413 2408 e phone
2433 2432 2427 e phone
2452 2451 2446 e phone
2471 2470 2465 e phone
2490 2489 2484 e phone
2509 2508 2503 e phone
2528 2527 2522 e phone
2547 2546 2541 e phone
2566 2565 2560 e phone
2585 2584 2579 e phone
2604 2603 2598 e phone
2623 2622 2617 e phone
2642 2641 2636 e phone
2661 2660 2655 e phone
2680 2679 2674 e phone
2699 2698 2693 e phone
2718 2717 2712 e phone
2737 2736 2731 e phone
2756 2755 2750 e phone
2775 2774 2769 e phone
2794 2793 2788 e phone
2813 2812 2807 e phone
2832 2831 2826 e phone
2851 2850 2845 e phone
2870 2869 2864 e phone
2889 2888 2883 e phone
2908 2907 2902 e phone
2928 2927 2922 e phone
2947 2946 2941 e phone
2966 2965 2960 e phone
2985 2984 2979 e phone
3004 3003 2998 e phone
3023 3022 3017 e phone
3042 3041 3036 e phone
3061 3060 3055 e phone
3080 3079 3074 e phone

3099 3098 3093 e phone
3118 3117 3112 e phone
3137 3136 3131 e phone
3156 3155 3150 e phone
3175 3174 3169 e phone
3194 3193 3188 e phone
3213 3212 3207 e phone
3232 3231 3226 e phone
3251 3250 3245 e phone
3270 3269 3264 e phone
3289 3288 3283 e phone
3308 3307 3302 e phone
3327 3326 3321 e phone
3346 3345 3340 e phone
3365 3364 3359 e phone
3384 3383 3378 e phone
3403 3402 3397 e phone
3422 3421 3416 e phone
3441 3440 3435 e phone
3460 3459 3454 e phone
3479 3478 3473 e phone
3498 3497 3492 e phone
3517 3516 3511 e phone
3536 3535 3530 e phone
3555 3554 3549 e phone
3574 3573 3568 e phone
3593 3592 3587 e phone
3612 3611 3606 e phone
3631 3630 3625 e phone
3650 3649 3644 e phone
3669 3668 3663 e phone
3688 3687 3682 e phone
3707 3706 3701 e phone
3726 3725 3720 e phone
3745 3744 3739 e phone
3764 3763 3758 e phone
3783 3782 3777 e phone
3802 3801 3796 e phone
3821 3820 3815 e phone
3840 3839 3834 e phone
3859 3858 3853 e phone
3879 3878 3873 e phone
3898 3897 3892 e phone
3917 3916 3911 e phone
3936 3935 3930 e phone
3955 3954 3949 e phone
3974 3973 3968 e phone
3993 3992 3987 e phone
4012 4011 4006 e phone
4031 4030 4025 e phone
4050 4049 4044 e phone
4069 4068 4063 e phone
4088 4087 4082 e phone
4107 4106 4101 e phone
4126 4125 4120 e phone
4145 4144 4139 e phone
4164 4163 4158 e phone
4183 4182 4177 e phone
4202 4201 4196 e phone
4221 4220 4215 e phone
4240 4239 4234 e phone
4259 4258 4253 e phone
4278 4277 4272 e phone
4297 4296 4291 e phone
4316 4315 4310 e phone
4335 4334 4329 e phone
4355 4354 4349 e phone
4374 4373 4368 e phone
4393 4392 4387 e phone
4412 4411 4406 e phone
4431 4430 4425 e phone
4450 4449 4444 e phone
4469 4468 4463 e phone
4488 4487 4482 e phone
4507 4506 4501 e phone
4526 4525 4520 e phone
4545 4544 4539 e phone
4564 4563 4558 e phone
4583 4582 4577 e phone
4602 4601 4596 e phone
4621 4620 4615 e phone
4640 4639 4634 e phone
4659 4658 4653 e phone
4678 4677 4672 e phone
4697 4696 4691 e phone
4716 4715 4710 e phone
4735 4734 4729 e phone
4747 4746 4741 e phone
4759 4758 4753 e phone
4771 4770 4765 e phone
4783 4782 4777 e phone
4795 4794 4789 e phone
4807 4806 4801 e phone
4819 4818 4813 e phone
4831 4830 4825 e phone
4843 4842 4837 e phone
4855 4854 4849 e phone
4867 4866 4861 e phone
4877 4876 4871 e phone
4889 4888 4883 e phone
4899 4898 4893 e phone
4909 4908 4903 e phone
4919 4918 4913 e phone
4929 4928 4923 e phone

Total Query Time: 0.30(sec) Number of Result Nodes: 256

```
SELECT DISTINCT v0.*
 FROM accel_medicine v2,accel_medicine
v1,accel_medicine v0
 WHERE
 v2.tag="gradstudent" and v2.pre=v1.par and
 v1.tag="address" and v1.pre=v0.par and
 v0.tag="city" and v0.kind="e";
```

Query3=//gradstudent/address/city
16 14 15 e city
37 35 36 e city
60 58 59 e city
81 79 80 e city
102 100 101 e city
123 121 122 e city
146 144 145 e city

167 165 166 e city
188 186 187 e city
209 207 208 e city
232 230 231 e city
255 253 254 e city
276 274 275 e city
297 295 296 e city
320 318 319 e city
341 339 340 e city
362 360 361 e city
383 381 382 e city
404 402 403 e city
425 423 424 e city
446 444 445 e city
467 465 466 e city
490 488 489 e city
512 510 511 e city
533 531 532 e city
554 552 553 e city
577 575 576 e city
598 596 597 e city
619 617 618 e city
640 638 639 e city
661 659 660 e city
682 680 681 e city
705 703 704 e city
726 724 725 e city
747 745 746 e city
770 768 769 e city
791 789 790 e city
814 812 813 e city
835 833 834 e city
858 856 857 e city
879 877 878 e city
902 900 901 e city
923 921 922 e city
944 942 943 e city
965 963 964 e city
986 984 985 e city
1007 1005 1006 e city
1028 1026 1027 e city
1051 1049 1050 e city
1072 1070 1071 e city
1093 1091 1092 e city
1116 1114 1115 e city
1137 1135 1136 e city
1160 1158 1159 e city
1183 1181 1182 e city
1204 1202 1203 e city
1227 1225 1226 e city
1248 1246 1247 e city
1269 1267 1268 e city
1290 1288 1289 e city
1313 1311 1312 e city
1336 1334 1335 e city
1359 1357 1358 e city
1382 1380 1381 e city
1403 1401 1402 e city
1424 1422 1423 e city
1445 1443 1444 e city
1466 1464 1465 e city
1487 1485 1486 e city
1510 1508 1509 e city
1531 1529 1530 e city
1554 1552 1553 e city
1577 1575 1576 e city
1600 1598 1599 e city

Total Query Time: 0.30(sec) Number of Result Nodes: 74

```
SELECT DISTINCT v0.*
 FROM accel_medicine v1,accel_medicine v0
 WHERE
 v1.tag="department" and v1.pre!=v0.par and
 v0.pre>v1.pre and v0.post<v1.post and
 v0.tag="phone" and v0.kind="e";
```

Query4=/department/*/phone
9 8 3 e phone
32 31 26 e phone
53 52 47 e phone
76 75 70 e phone
97 96 91 e phone
118 117 112 e phone
139 138 133 e phone
162 161 156 e phone
183 182 177 e phone
204 203 198 e phone
225 224 219 e phone
248 247 242 e phone
271 270 265 e phone
292 291 286 e phone
313 312 307 e phone
336 335 330 e phone
357 356 351 e phone
378 377 372 e phone
399 398 393 e phone
420 419 414 e phone
441 440 435 e phone
462 461 456 e phone
483 482 477 e phone
507 506 501 e phone
528 527 522 e phone
549 548 543 e phone
570 569 564 e phone
593 592 587 e phone
614 613 608 e phone
635 634 629 e phone
656 655 650 e phone
677 676 671 e phone
698 697 692 e phone
721 720 715 e phone
742 741 736 e phone
763 762 757 e phone
786 785 780 e phone
807 806 801 e phone
830 829 824 e phone
851 850 845 e phone
874 873 868 e phone
895 894 889 e phone
918 917 912 e phone
939 938 933 e phone
960 959 954 e phone

981 980 975 e phone
1002 1001 996 e phone
1023 1022 1017 e phone
1044 1043 1038 e phone
1067 1066 1061 e phone
1088 1087 1082 e phone
1109 1108 1103 e phone
1132 1131 1126 e phone
1153 1152 1147 e phone
1176 1175 1170 e phone
1199 1198 1193 e phone
1220 1219 1214 e phone
1243 1242 1237 e phone
1264 1263 1258 e phone
1285 1284 1279 e phone
1306 1305 1300 e phone
1329 1328 1323 e phone
1352 1351 1346 e phone
1375 1374 1369 e phone
1398 1397 1392 e phone
1419 1418 1413 e phone
1440 1439 1434 e phone
1461 1460 1455 e phone
1482 1481 1476 e phone
1503 1502 1497 e phone
1526 1525 1520 e phone
1547 1546 1541 e phone
1570 1569 1564 e phone
1593 1592 1587 e phone
1616 1615 1610 e phone
1635 1634 1629 e phone
1654 1653 1648 e phone
1673 1672 1667 e phone
1692 1691 1686 e phone
1711 1710 1705 e phone
1730 1729 1724 e phone
1749 1748 1743 e phone
1768 1767 1762 e phone
1787 1786 1781 e phone
1806 1805 1800 e phone
1825 1824 1819 e phone
1844 1843 1838 e phone
1863 1862 1857 e phone
1882 1881 1876 e phone
1901 1900 1895 e phone
1920 1919 1914 e phone
1939 1938 1933 e phone
1958 1957 1952 e phone
1977 1976 1971 e phone
1996 1995 1990 e phone
2015 2014 2009 e phone
2034 2033 2028 e phone
2053 2052 2047 e phone
2072 2071 2066 e phone
2091 2090 2085 e phone
2110 2109 2104 e phone
2129 2128 2123 e phone
2148 2147 2142 e phone
2167 2166 2161 e phone
2186 2185 2180 e phone
2205 2204 2199 e phone
2224 2223 2218 e phone
2243 2242 2237 e phone
2262 2261 2256 e phone
2281 2280 2275 e phone
2300 2299 2294 e phone
2319 2318 2313 e phone
2338 2337 2332 e phone
2357 2356 2351 e phone
2376 2375 2370 e phone
2395 2394 2389 e phone
2414 2413 2408 e phone
2433 2432 2427 e phone
2452 2451 2446 e phone
2471 2470 2465 e phone
2490 2489 2484 e phone
2509 2508 2503 e phone
2528 2527 2522 e phone
2547 2546 2541 e phone
2566 2565 2560 e phone
2585 2584 2579 e phone
2604 2603 2598 e phone
2623 2622 2617 e phone
2642 2641 2636 e phone
2661 2660 2655 e phone
2680 2679 2674 e phone
2699 2698 2693 e phone
2718 2717 2712 e phone
2737 2736 2731 e phone
2756 2755 2750 e phone
2775 2774 2769 e phone
2794 2793 2788 e phone
2813 2812 2807 e phone
2832 2831 2826 e phone
2851 2850 2845 e phone
2870 2869 2864 e phone
2889 2888 2883 e phone
2908 2907 2902 e phone
2928 2927 2922 e phone
2947 2946 2941 e phone
2966 2965 2960 e phone
2985 2984 2979 e phone
3004 3003 2998 e phone
3023 3022 3017 e phone
3042 3041 3036 e phone
3061 3060 3055 e phone
3080 3079 3074 e phone
3099 3098 3093 e phone
3118 3117 3112 e phone
3137 3136 3131 e phone
3156 3155 3150 e phone
3175 3174 3169 e phone
3194 3193 3188 e phone
3213 3212 3207 e phone
3232 3231 3226 e phone
3251 3250 3245 e phone
3270 3269 3264 e phone
3289 3288 3283 e phone
3308 3307 3302 e phone
3327 3326 3321 e phone
3346 3345 3340 e phone
3365 3364 3359 e phone

3384 3383 3378 e phone
3403 3402 3397 e phone
3422 3421 3416 e phone
3441 3440 3435 e phone
3460 3459 3454 e phone
3479 3478 3473 e phone
3498 3497 3492 e phone
3517 3516 3511 e phone
3536 3535 3530 e phone
3555 3554 3549 e phone
3574 3573 3568 e phone
3593 3592 3587 e phone
3612 3611 3606 e phone
3631 3630 3625 e phone
3650 3649 3644 e phone
3669 3668 3663 e phone
3688 3687 3682 e phone
3707 3706 3701 e phone
3726 3725 3720 e phone
3745 3744 3739 e phone
3764 3763 3758 e phone
3783 3782 3777 e phone
3802 3801 3796 e phone
3821 3820 3815 e phone
3840 3839 3834 e phone
3859 3858 3853 e phone
3879 3878 3873 e phone
3898 3897 3892 e phone
3917 3916 3911 e phone
3936 3935 3930 e phone
3955 3954 3949 e phone
3974 3973 3968 e phone
3993 3992 3987 e phone
4012 4011 4006 e phone
4031 4030 4025 e phone
4050 4049 4044 e phone
4069 4068 4063 e phone
4088 4087 4082 e phone
4107 4106 4101 e phone
4126 4125 4120 e phone
4145 4144 4139 e phone
4164 4163 4158 e phone
4183 4182 4177 e phone
4202 4201 4196 e phone
4221 4220 4215 e phone
4240 4239 4234 e phone
4259 4258 4253 e phone
4278 4277 4272 e phone
4297 4296 4291 e phone
4316 4315 4310 e phone
4335 4334 4329 e phone
4355 4354 4349 e phone
4374 4373 4368 e phone
4393 4392 4387 e phone
4412 4411 4406 e phone
4431 4430 4425 e phone
4450 4449 4444 e phone
4469 4468 4463 e phone
4488 4487 4482 e phone
4507 4506 4501 e phone
4526 4525 4520 e phone
4545 4544 4539 e phone
4564 4563 4558 e phone
4583 4582 4577 e phone
4602 4601 4596 e phone
4621 4620 4615 e phone
4640 4639 4634 e phone
4659 4658 4653 e phone
4678 4677 4672 e phone
4697 4696 4691 e phone
4716 4715 4710 e phone
4735 4734 4729 e phone
4747 4746 4741 e phone
4759 4758 4753 e phone
4771 4770 4765 e phone
4783 4782 4777 e phone
4795 4794 4789 e phone
4807 4806 4801 e phone
4819 4818 4813 e phone
4831 4830 4825 e phone
4843 4842 4837 e phone
4855 4854 4849 e phone
4867 4866 4861 e phone
4877 4876 4871 e phone
4889 4888 4883 e phone
4899 4898 4893 e phone
4909 4908 4903 e phone
4919 4918 4913 e phone
4929 4928 4923 e phone

Total Query Time: 0.190(sec) Number of Result Nodes: 256

```
SELECT DISTINCT v0.*
 FROM accel_medicine v3,accel_medicine
v2,accel_medicine v1,accel_medicine v0
 WHERE
 v3.tag="department" and v3.pre=v2.par and
 v2.tag="gradstudent" and v2.pre=v1.par and
 v1.tag="address" and v1.pre=v0.par and
 v0.tag="city" and v0.kind="e";
```

Query5=/department/gradstudent/address/city
16 14 15 e city
37 35 36 e city
60 58 59 e city
81 79 80 e city
102 100 101 e city
123 121 122 e city
146 144 145 e city
167 165 166 e city
188 186 187 e city
209 207 208 e city
232 230 231 e city
255 253 254 e city
276 274 275 e city
297 295 296 e city
320 318 319 e city
341 339 340 e city
362 360 361 e city
383 381 382 e city
404 402 403 e city
425 423 424 e city

446 444 445 e city
467 465 466 e city
490 488 489 e city
512 510 511 e city
533 531 532 e city
554 552 553 e city
577 575 576 e city
598 596 597 e city
619 617 618 e city
640 638 639 e city
661 659 660 e city
682 680 681 e city
705 703 704 e city
726 724 725 e city
747 745 746 e city
770 768 769 e city
791 789 790 e city
814 812 813 e city
835 833 834 e city
858 856 857 e city
879 877 878 e city
902 900 901 e city
923 921 922 e city
944 942 943 e city
965 963 964 e city
986 984 985 e city
1007 1005 1006 e city
1028 1026 1027 e city
1051 1049 1050 e city
1072 1070 1071 e city
1093 1091 1092 e city
1116 1114 1115 e city
1137 1135 1136 e city
1160 1158 1159 e city
1183 1181 1182 e city
1204 1202 1203 e city
1227 1225 1226 e city
1248 1246 1247 e city
1269 1267 1268 e city
1290 1288 1289 e city
1313 1311 1312 e city
1336 1334 1335 e city
1359 1357 1358 e city
1382 1380 1381 e city
1403 1401 1402 e city
1424 1422 1423 e city
1445 1443 1444 e city
1466 1464 1465 e city
1487 1485 1486 e city
1510 1508 1509 e city
1531 1529 1530 e city
1554 1552 1553 e city
1577 1575 1576 e city
1600 1598 1599 e city

Total Query Time: 0.30(sec) Number of Result Nodes: 74

SELECT DISTINCT v0.*
FROM accel_medicine v1,accel_medicine v0
WHERE
v1.tag="gradstudent" and v1.pre=v0.par and
v0.tag="email" and v0.kind="e";
Query1=//gradstudent/email
11 10 3 e email
34 33 26 e email
55 54 47 e email
78 77 70 e email
99 98 91 e email
120 119 112 e email
141 140 133 e email
164 163 156 e email
185 184 177 e email
206 205 198 e email
227 226 219 e email
250 249 242 e email
273 272 265 e email
294 293 286 e email
315 314 307 e email
338 337 330 e email
359 358 351 e email
380 379 372 e email
401 400 393 e email
422 421 414 e email
443 442 435 e email
464 463 456 e email
485 484 477 e email
509 508 501 e email
530 529 522 e email
551 550 543 e email
572 571 564 e email
595 594 587 e email
616 615 608 e email
637 636 629 e email
658 657 650 e email
679 678 671 e email
700 699 692 e email
723 722 715 e email
744 743 736 e email
765 764 757 e email
788 787 780 e email
809 808 801 e email
832 831 824 e email
853 852 845 e email
876 875 868 e email
897 896 889 e email
920 919 912 e email
941 940 933 e email
962 961 954 e email
983 982 975 e email
1004 1003 996 e email
1025 1024 1017 e email
1046 1045 1038 e email
1069 1068 1061 e email
1090 1089 1082 e email
1111 1110 1103 e email
1134 1133 1126 e email
1155 1154 1147 e email
1178 1177 1170 e email
1201 1200 1193 e email
1222 1221 1214 e email
1245 1244 1237 e email
1266 1265 1258 e email

1287 1286 1279 e email
1308 1307 1300 e email
1331 1330 1323 e email
1354 1353 1346 e email
1377 1376 1369 e email
1400 1399 1392 e email
1421 1420 1413 e email
1442 1441 1434 e email
1463 1462 1455 e email
1484 1483 1476 e email
1505 1504 1497 e email
1528 1527 1520 e email
1549 1548 1541 e email
1572 1571 1564 e email
1595 1594 1587 e email

Total Query Time: 40(msec) Number of Result Nodes: 74

SELECT DISTINCT v0.*
FROM accel_medicine v0
WHERE
v0.tag="phone" and v0.kind="e";
Query2=//phone
9 8 3 e phone
32 31 26 e phone
53 52 47 e phone
76 75 70 e phone
97 96 91 e phone
118 117 112 e phone
139 138 133 e phone
162 161 156 e phone
183 182 177 e phone
204 203 198 e phone
225 224 219 e phone
248 247 242 e phone
271 270 265 e phone
292 291 286 e phone
313 312 307 e phone
336 335 330 e phone
357 356 351 e phone
378 377 372 e phone
399 398 393 e phone
420 419 414 e phone
441 440 435 e phone
462 461 456 e phone
483 482 477 e phone
507 506 501 e phone
528 527 522 e phone
549 548 543 e phone
570 569 564 e phone
593 592 587 e phone
614 613 608 e phone
635 634 629 e phone
656 655 650 e phone
677 676 671 e phone
698 697 692 e phone
721 720 715 e phone
742 741 736 e phone
763 762 757 e phone
786 785 780 e phone
807 806 801 e phone
830 829 824 e phone
851 850 845 e phone
874 873 868 e phone
895 894 889 e phone
918 917 912 e phone
939 938 933 e phone
960 959 954 e phone
981 980 975 e phone
1002 1001 996 e phone
1023 1022 1017 e phone
1044 1043 1038 e phone
1067 1066 1061 e phone
1088 1087 1082 e phone
1109 1108 1103 e phone
1132 1131 1126 e phone
1153 1152 1147 e phone
1176 1175 1170 e phone
1199 1198 1193 e phone
1220 1219 1214 e phone
1243 1242 1237 e phone
1264 1263 1258 e phone
1285 1284 1279 e phone
1306 1305 1300 e phone
1329 1328 1323 e phone
1352 1351 1346 e phone
1375 1374 1369 e phone
1398 1397 1392 e phone
1419 1418 1413 e phone
1440 1439 1434 e phone
1461 1460 1455 e phone
1482 1481 1476 e phone
1503 1502 1497 e phone
1526 1525 1520 e phone
1547 1546 1541 e phone
1570 1569 1564 e phone
1593 1592 1587 e phone
1616 1615 1610 e phone
1635 1634 1629 e phone
1654 1653 1648 e phone
1673 1672 1667 e phone
1692 1691 1686 e phone
1711 1710 1705 e phone
1730 1729 1724 e phone
1749 1748 1743 e phone
1768 1767 1762 e phone
1787 1786 1781 e phone
1806 1805 1800 e phone
1825 1824 1819 e phone
1844 1843 1838 e phone
1863 1862 1857 e phone
1882 1881 1876 e phone
1901 1900 1895 e phone
1920 1919 1914 e phone
1939 1938 1933 e phone
1958 1957 1952 e phone
1977 1976 1971 e phone
1996 1995 1990 e phone
2015 2014 2009 e phone
2034 2033 2028 e phone
2053 2052 2047 e phone
2072 2071 2066 e phone

2091	2090	2085	e	phone
2110	2109	2104	e	phone
2129	2128	2123	e	phone
2148	2147	2142	e	phone
2167	2166	2161	e	phone
2186	2185	2180	e	phone
2205	2204	2199	e	phone
2224	2223	2218	e	phone
2243	2242	2237	e	phone
2262	2261	2256	e	phone
2281	2280	2275	e	phone
2300	2299	2294	e	phone
2319	2318	2313	e	phone
2338	2337	2332	e	phone
2357	2356	2351	e	phone
2376	2375	2370	e	phone
2395	2394	2389	e	phone
2414	2413	2408	e	phone
2433	2432	2427	e	phone
2452	2451	2446	e	phone
2471	2470	2465	e	phone
2490	2489	2484	e	phone
2509	2508	2503	e	phone
2528	2527	2522	e	phone
2547	2546	2541	e	phone
2566	2565	2560	e	phone
2585	2584	2579	e	phone
2604	2603	2598	e	phone
2623	2622	2617	e	phone
2642	2641	2636	e	phone
2661	2660	2655	e	phone
2680	2679	2674	e	phone
2699	2698	2693	e	phone
2718	2717	2712	e	phone
2737	2736	2731	e	phone
2756	2755	2750	e	phone
2775	2774	2769	e	phone
2794	2793	2788	e	phone
2813	2812	2807	e	phone
2832	2831	2826	e	phone
2851	2850	2845	e	phone
2870	2869	2864	e	phone
2889	2888	2883	e	phone
2908	2907	2902	e	phone
2928	2927	2922	e	phone
2947	2946	2941	e	phone
2966	2965	2960	e	phone
2985	2984	2979	e	phone
3004	3003	2998	e	phone
3023	3022	3017	e	phone
3042	3041	3036	e	phone
3061	3060	3055	e	phone
3080	3079	3074	e	phone
3099	3098	3093	e	phone
3118	3117	3112	e	phone
3137	3136	3131	e	phone
3156	3155	3150	e	phone
3175	3174	3169	e	phone
3194	3193	3188	e	phone
3213	3212	3207	e	phone
3232	3231	3226	e	phone
3251	3250	3245	e	phone
3270	3269	3264	e	phone
3289	3288	3283	e	phone
3308	3307	3302	e	phone
3327	3326	3321	e	phone
3346	3345	3340	e	phone
3365	3364	3359	e	phone
3384	3383	3378	e	phone
3403	3402	3397	e	phone
3422	3421	3416	e	phone
3441	3440	3435	e	phone
3460	3459	3454	e	phone
3479	3478	3473	e	phone
3498	3497	3492	e	phone
3517	3516	3511	e	phone
3536	3535	3530	e	phone
3555	3554	3549	e	phone
3574	3573	3568	e	phone
3593	3592	3587	e	phone
3612	3611	3606	e	phone
3631	3630	3625	e	phone
3650	3649	3644	e	phone
3669	3668	3663	e	phone
3688	3687	3682	e	phone
3707	3706	3701	e	phone
3726	3725	3720	e	phone
3745	3744	3739	e	phone
3764	3763	3758	e	phone
3783	3782	3777	e	phone
3802	3801	3796	e	phone
3821	3820	3815	e	phone
3840	3839	3834	e	phone
3859	3858	3853	e	phone
3879	3878	3873	e	phone
3898	3897	3892	e	phone
3917	3916	3911	e	phone
3936	3935	3930	e	phone
3955	3954	3949	e	phone
3974	3973	3968	e	phone
3993	3992	3987	e	phone
4012	4011	4006	e	phone
4031	4030	4025	e	phone
4050	4049	4044	e	phone
4069	4068	4063	e	phone
4088	4087	4082	e	phone
4107	4106	4101	e	phone
4126	4125	4120	e	phone
4145	4144	4139	e	phone
4164	4163	4158	e	phone
4183	4182	4177	e	phone
4202	4201	4196	e	phone
4221	4220	4215	e	phone
4240	4239	4234	e	phone
4259	4258	4253	e	phone
4278	4277	4272	e	phone
4297	4296	4291	e	phone
4316	4315	4310	e	phone
4335	4334	4329	e	phone
4355	4354	4349	e	phone
4374	4373	4368	e	phone
4393	4392	4387	e	phone

4412 4411 4406 e phone
4431 4430 4425 e phone
4450 4449 4444 e phone
4469 4468 4463 e phone
4488 4487 4482 e phone
4507 4506 4501 e phone
4526 4525 4520 e phone
4545 4544 4539 e phone
4564 4563 4558 e phone
4583 4582 4577 e phone
4602 4601 4596 e phone
4621 4620 4615 e phone
4640 4639 4634 e phone
4659 4658 4653 e phone
4678 4677 4672 e phone
4697 4696 4691 e phone
4716 4715 4710 e phone
4735 4734 4729 e phone
4747 4746 4741 e phone
4759 4758 4753 e phone
4771 4770 4765 e phone
4783 4782 4777 e phone
4795 4794 4789 e phone
4807 4806 4801 e phone
4819 4818 4813 e phone
4831 4830 4825 e phone
4843 4842 4837 e phone
4855 4854 4849 e phone
4867 4866 4861 e phone
4877 4876 4871 e phone
4889 4888 4883 e phone
4899 4898 4893 e phone
4909 4908 4903 e phone
4919 4918 4913 e phone
4929 4928 4923 e phone

Total Query Time: 20(msec) Number of Result Nodes: 256

SELECT DISTINCT v0.*
FROM accel_medicine v2,accel_medicine v1,accel_medicine v0
WHERE
v2.tag="gradstudent" and v2.pre=v1.par and
v1.tag="address" and v1.pre=v0.par and
v0.tag="city" and v0.kind="e";
Query3=//gradstudent/address/city
16 14 15 e city
37 35 36 e city
60 58 59 e city
81 79 80 e city
102 100 101 e city
123 121 122 e city
146 144 145 e city
167 165 166 e city
188 186 187 e city
209 207 208 e city
232 230 231 e city
255 253 254 e city
276 274 275 e city
297 295 296 e city
320 318 319 e city
341 339 340 e city
362 360 361 e city
383 381 382 e city
404 402 403 e city
425 423 424 e city
446 444 445 e city
467 465 466 e city
490 488 489 e city
512 510 511 e city
533 531 532 e city
554 552 553 e city
577 575 576 e city
598 596 597 e city
619 617 618 e city
640 638 639 e city
661 659 660 e city
682 680 681 e city
705 703 704 e city
726 724 725 e city
747 745 746 e city
770 768 769 e city
791 789 790 e city
814 812 813 e city
835 833 834 e city
858 856 857 e city
879 877 878 e city
902 900 901 e city
923 921 922 e city
944 942 943 e city
965 963 964 e city
986 984 985 e city
1007 1005 1006 e city
1028 1026 1027 e city
1051 1049 1050 e city
1072 1070 1071 e city
1093 1091 1092 e city
1116 1114 1115 e city
1137 1135 1136 e city
1160 1158 1159 e city
1183 1181 1182 e city
1204 1202 1203 e city
1227 1225 1226 e city
1248 1246 1247 e city
1269 1267 1268 e city
1290 1288 1289 e city
1313 1311 1312 e city
1336 1334 1335 e city
1359 1357 1358 e city
1382 1380 1381 e city
1403 1401 1402 e city
1424 1422 1423 e city
1445 1443 1444 e city
1466 1464 1465 e city
1487 1485 1486 e city
1510 1508 1509 e city
1531 1529 1530 e city
1554 1552 1553 e city
1577 1575 1576 e city
1600 1598 1599 e city

Total Query Time: 30(msec) Number of Result Nodes: 74

```
----------------------------
SELECT DISTINCT v0.*
 FROM accel_medicine v1,accel_medicine v0
 WHERE
 v1.tag="department" and v1.pre!=v0.par and
 v0.pre>v1.pre and v0.post<v1.post and
 v0.tag="phone" and v0.kind="e";
Query4=/department/*/phone
9  8  3  e  phone
32  31  26  e  phone
53  52  47  e  phone
76  75  70  e  phone
97  96  91  e  phone
118  117  112  e  phone
139  138  133  e  phone
162  161  156  e  phone
183  182  177  e  phone
204  203  198  e  phone
225  224  219  e  phone
248  247  242  e  phone
271  270  265  e  phone
292  291  286  e  phone
313  312  307  e  phone
336  335  330  e  phone
357  356  351  e  phone
378  377  372  e  phone
399  398  393  e  phone
420  419  414  e  phone
441  440  435  e  phone
462  461  456  e  phone
483  482  477  e  phone
507  506  501  e  phone
528  527  522  e  phone
549  548  543  e  phone
570  569  564  e  phone
593  592  587  e  phone
614  613  608  e  phone
635  634  629  e  phone
656  655  650  e  phone
677  676  671  e  phone
698  697  692  e  phone
721  720  715  e  phone
742  741  736  e  phone
763  762  757  e  phone
786  785  780  e  phone
807  806  801  e  phone
830  829  824  e  phone
851  850  845  e  phone
874  873  868  e  phone
895  894  889  e  phone
918  917  912  e  phone
939  938  933  e  phone
960  959  954  e  phone
981  980  975  e  phone
1002  1001  996  e  phone
1023  1022  1017  e  phone
1044  1043  1038  e  phone
1067  1066  1061  e  phone
1088  1087  1082  e  phone
1109  1108  1103  e  phone
1132  1131  1126  e  phone
1153  1152  1147  e  phone
1176  1175  1170  e  phone
1199  1198  1193  e  phone
1220  1219  1214  e  phone
1243  1242  1237  e  phone
1264  1263  1258  e  phone
1285  1284  1279  e  phone
1306  1305  1300  e  phone
1329  1328  1323  e  phone
1352  1351  1346  e  phone
1375  1374  1369  e  phone
1398  1397  1392  e  phone
1419  1418  1413  e  phone
1440  1439  1434  e  phone
1461  1460  1455  e  phone
1482  1481  1476  e  phone
1503  1502  1497  e  phone
1526  1525  1520  e  phone
1547  1546  1541  e  phone
1570  1569  1564  e  phone
1593  1592  1587  e  phone
1616  1615  1610  e  phone
1635  1634  1629  e  phone
1654  1653  1648  e  phone
1673  1672  1667  e  phone
1692  1691  1686  e  phone
1711  1710  1705  e  phone
1730  1729  1724  e  phone
1749  1748  1743  e  phone
1768  1767  1762  e  phone
1787  1786  1781  e  phone
1806  1805  1800  e  phone
1825  1824  1819  e  phone
1844  1843  1838  e  phone
1863  1862  1857  e  phone
1882  1881  1876  e  phone
1901  1900  1895  e  phone
1920  1919  1914  e  phone
1939  1938  1933  e  phone
1958  1957  1952  e  phone
1977  1976  1971  e  phone
1996  1995  1990  e  phone
2015  2014  2009  e  phone
2034  2033  2028  e  phone
2053  2052  2047  e  phone
2072  2071  2066  e  phone
2091  2090  2085  e  phone
2110  2109  2104  e  phone
2129  2128  2123  e  phone
2148  2147  2142  e  phone
2167  2166  2161  e  phone
2186  2185  2180  e  phone
2205  2204  2199  e  phone
2224  2223  2218  e  phone
2243  2242  2237  e  phone
2262  2261  2256  e  phone
2281  2280  2275  e  phone
2300  2299  2294  e  phone
2319  2318  2313  e  phone
2338  2337  2332  e  phone
2357  2356  2351  e  phone
```

2376	2375	2370	e	phone
2395	2394	2389	e	phone
2414	2413	2408	e	phone
2433	2432	2427	e	phone
2452	2451	2446	e	phone
2471	2470	2465	e	phone
2490	2489	2484	e	phone
2509	2508	2503	e	phone
2528	2527	2522	e	phone
2547	2546	2541	e	phone
2566	2565	2560	e	phone
2585	2584	2579	e	phone
2604	2603	2598	e	phone
2623	2622	2617	e	phone
2642	2641	2636	e	phone
2661	2660	2655	e	phone
2680	2679	2674	e	phone
2699	2698	2693	e	phone
2718	2717	2712	e	phone
2737	2736	2731	e	phone
2756	2755	2750	e	phone
2775	2774	2769	e	phone
2794	2793	2788	e	phone
2813	2812	2807	e	phone
2832	2831	2826	e	phone
2851	2850	2845	e	phone
2870	2869	2864	e	phone
2889	2888	2883	e	phone
2908	2907	2902	e	phone
2928	2927	2922	e	phone
2947	2946	2941	e	phone
2966	2965	2960	e	phone
2985	2984	2979	e	phone
3004	3003	2998	e	phone
3023	3022	3017	e	phone
3042	3041	3036	e	phone
3061	3060	3055	e	phone
3080	3079	3074	e	phone
3099	3098	3093	e	phone
3118	3117	3112	e	phone
3137	3136	3131	e	phone
3156	3155	3150	e	phone
3175	3174	3169	e	phone
3194	3193	3188	e	phone
3213	3212	3207	e	phone
3232	3231	3226	e	phone
3251	3250	3245	e	phone
3270	3269	3264	e	phone
3289	3288	3283	e	phone
3308	3307	3302	e	phone
3327	3326	3321	e	phone
3346	3345	3340	e	phone
3365	3364	3359	e	phone
3384	3383	3378	e	phone
3403	3402	3397	e	phone
3422	3421	3416	e	phone
3441	3440	3435	e	phone
3460	3459	3454	e	phone
3479	3478	3473	e	phone
3498	3497	3492	e	phone
3517	3516	3511	e	phone
3536	3535	3530	e	phone
3555	3554	3549	e	phone
3574	3573	3568	e	phone
3593	3592	3587	e	phone
3612	3611	3606	e	phone
3631	3630	3625	e	phone
3650	3649	3644	e	phone
3669	3668	3663	e	phone
3688	3687	3682	e	phone
3707	3706	3701	e	phone
3726	3725	3720	e	phone
3745	3744	3739	e	phone
3764	3763	3758	e	phone
3783	3782	3777	e	phone
3802	3801	3796	e	phone
3821	3820	3815	e	phone
3840	3839	3834	e	phone
3859	3858	3853	e	phone
3879	3878	3873	e	phone
3898	3897	3892	e	phone
3917	3916	3911	e	phone
3936	3935	3930	e	phone
3955	3954	3949	e	phone
3974	3973	3968	e	phone
3993	3992	3987	e	phone
4012	4011	4006	e	phone
4031	4030	4025	e	phone
4050	4049	4044	e	phone
4069	4068	4063	e	phone
4088	4087	4082	e	phone
4107	4106	4101	e	phone
4126	4125	4120	e	phone
4145	4144	4139	e	phone
4164	4163	4158	e	phone
4183	4182	4177	e	phone
4202	4201	4196	e	phone
4221	4220	4215	e	phone
4240	4239	4234	e	phone
4259	4258	4253	e	phone
4278	4277	4272	e	phone
4297	4296	4291	e	phone
4316	4315	4310	e	phone
4335	4334	4329	e	phone
4355	4354	4349	e	phone
4374	4373	4368	e	phone
4393	4392	4387	e	phone
4412	4411	4406	e	phone
4431	4430	4425	e	phone
4450	4449	4444	e	phone
4469	4468	4463	e	phone
4488	4487	4482	e	phone
4507	4506	4501	e	phone
4526	4525	4520	e	phone
4545	4544	4539	e	phone
4564	4563	4558	e	phone
4583	4582	4577	e	phone
4602	4601	4596	e	phone
4621	4620	4615	e	phone
4640	4639	4634	e	phone
4659	4658	4653	e	phone
4678	4677	4672	e	phone

4697 4696 4691 e phone
4716 4715 4710 e phone
4735 4734 4729 e phone
4747 4746 4741 e phone
4759 4758 4753 e phone
4771 4770 4765 e phone
4783 4782 4777 e phone
4795 4794 4789 e phone
4807 4806 4801 e phone
4819 4818 4813 e phone
4831 4830 4825 e phone
4843 4842 4837 e phone
4855 4854 4849 e phone
4867 4866 4861 e phone
4877 4876 4871 e phone
4889 4888 4883 e phone
4899 4898 4893 e phone
4909 4908 4903 e phone
4919 4918 4913 e phone
4929 4928 4923 e phone

Total Query Time: 50(msec) Number of Result Nodes: 256

SELECT DISTINCT v0.*
FROM accel_medicine v3,accel_medicine
v2,accel_medicine v1,accel_medicine v0
WHERE
v3.tag="department" and v3.pre=v2.par and
v2.tag="gradstudent" and v2.pre=v1.par and
v1.tag="address" and v1.pre=v0.par and
v0.tag="city" and v0.kind="e";
Query5=/department/gradstudent/address/city
16 14 15 e city
37 35 36 e city
60 58 59 e city
81 79 80 e city
102 100 101 e city
123 121 122 e city
146 144 145 e city
167 165 166 e city
188 186 187 e city
209 207 208 e city
232 230 231 e city
255 253 254 e city
276 274 275 e city
297 295 296 e city
320 318 319 e city
341 339 340 e city
362 360 361 e city
383 381 382 e city
404 402 403 e city
425 423 424 e city
446 444 445 e city
467 465 466 e city
490 488 489 e city
512 510 511 e city
533 531 532 e city
554 552 553 e city
577 575 576 e city
598 596 597 e city
619 617 618 e city
640 638 639 e city
661 659 660 e city
682 680 681 e city
705 703 704 e city
726 724 725 e city
747 745 746 e city
770 768 769 e city
791 789 790 e city
814 812 813 e city
835 833 834 e city
858 856 857 e city
879 877 878 e city
902 900 901 e city
923 921 922 e city
944 942 943 e city
965 963 964 e city
986 984 985 e city
1007 1005 1006 e city
1028 1026 1027 e city
1051 1049 1050 e city
1072 1070 1071 e city
1093 1091 1092 e city
1116 1114 1115 e city
1137 1135 1136 e city
1160 1158 1159 e city
1183 1181 1182 e city
1204 1202 1203 e city
1227 1225 1226 e city
1248 1246 1247 e city
1269 1267 1268 e city
1290 1288 1289 e city
1313 1311 1312 e city
1336 1334 1335 e city
1359 1357 1358 e city
1382 1380 1381 e city
1403 1401 1402 e city
1424 1422 1423 e city
1445 1443 1444 e city
1466 1464 1465 e city
1487 1485 1486 e city
1510 1508 1509 e city
1531 1529 1530 e city
1554 1552 1553 e city
1577 1575 1576 e city
1600 1598 1599 e city

Total Query Time: 30(msec) Number of Result Nodes: 74

Results of "Othello.xml"

```
SELECT DISTINCT v0.*
 FROM accel_othello v0
 WHERE
 v0.tag="stagedir" and v0.kind="e";
```
Query6=//stagedir

55 53 52 e STAGEDIR
265 263 52 e STAGEDIR
478 476 52 e STAGEDIR
515 513 52 e STAGEDIR
517 515 52 e STAGEDIR
590 588 52 e STAGEDIR
595 593 592 e STAGEDIR
690 688 592 e STAGEDIR
747 745 592 e STAGEDIR
776 774 592 e STAGEDIR
800 799 592 e STAGEDIR
818 816 592 e STAGEDIR
938 936 592 e STAGEDIR
943 942 940 e STAGEDIR
993 989 992 e STAGEDIR
1001 999 940 e STAGEDIR
1063 1061 940 e STAGEDIR
1116 1114 940 e STAGEDIR
1125 1122 1118 e STAGEDIR
1217 1213 1216 e STAGEDIR
1346 1343 1341 e STAGEDIR
1453 1451 940 e STAGEDIR
1773 1770 1766 e STAGEDIR
1793 1793 940 e STAGEDIR
1814 1812 940 e STAGEDIR
2038 2036 940 e STAGEDIR
2087 2085 940 e STAGEDIR
2095 2093 2092 e STAGEDIR
2152 2150 2092 e STAGEDIR
2227 2225 2092 e STAGEDIR
2256 2254 2092 e STAGEDIR
2258 2256 2092 e STAGEDIR
2277 2275 2092 e STAGEDIR
2298 2296 2092 e STAGEDIR
2318 2315 2305 e STAGEDIR
2371 2369 2349 e STAGEDIR
2412 2410 2092 e STAGEDIR
2426 2423 2421 e STAGEDIR
2430 2427 2421 e STAGEDIR
2440 2438 2092 e STAGEDIR
2668 2664 2667 e STAGEDIR
2691 2688 2664 e STAGEDIR
2710 2708 2092 e STAGEDIR
2767 2764 2756 e STAGEDIR
2775 2771 2774 e STAGEDIR
2811 2809 2092 e STAGEDIR
2999 2997 2092 e STAGEDIR
3059 3057 2092 e STAGEDIR
3064 3063 3061 e STAGEDIR
3094 3092 3061 e STAGEDIR
3099 3097 3096 e STAGEDIR
3128 3125 3119 e STAGEDIR
3138 3136 3096 e STAGEDIR
3140 3138 3096 e STAGEDIR
3259 3257 3096 e STAGEDIR
3296 3295 3096 e STAGEDIR
3316 3313 3311 e STAGEDIR
3460 3458 3096 e STAGEDIR
3513 3511 3096 e STAGEDIR
3519 3515 3518 e STAGEDIR
3524 3522 3096 e STAGEDIR
3548 3546 3096 e STAGEDIR
3550 3548 3096 e STAGEDIR
3579 3577 3096 e STAGEDIR
3586 3583 3581 e STAGEDIR
3609 3607 3096 e STAGEDIR
3615 3611 3614 e STAGEDIR
3618 3615 3611 e STAGEDIR
3626 3623 3611 e STAGEDIR
3634 3632 3096 e STAGEDIR
3646 3644 3096 e STAGEDIR
3863 3860 3852 e STAGEDIR
3883 3880 3874 e STAGEDIR
3893 3891 3096 e STAGEDIR
4110 4108 3096 e STAGEDIR
4169 4166 4112 e STAGEDIR
4217 4214 4188 e STAGEDIR
4233 4231 3096 e STAGEDIR
4241 4239 4238 e STAGEDIR
4250 4248 4238 e STAGEDIR
4252 4250 4238 e STAGEDIR
4323 4321 4238 e STAGEDIR
4360 4357 4355 e STAGEDIR
4362 4359 4355 e STAGEDIR
4400 4397 4395 e STAGEDIR
4406 4404 4238 e STAGEDIR
4457 4455 4238 e STAGEDIR
4462 4460 4459 e STAGEDIR
4490 4488 4459 e STAGEDIR
4495 4493 4492 e STAGEDIR
4605 4603 4492 e STAGEDIR
4607 4605 4492 e STAGEDIR
4811 4809 4492 e STAGEDIR
5321 5317 5320 e STAGEDIR
5335 5331 5334 e STAGEDIR
5372 5370 4492 e STAGEDIR
5417 5414 5374 e STAGEDIR
5463 5460 5458 e STAGEDIR
5472 5470 4492 e STAGEDIR
5497 5495 4492 e STAGEDIR
5577 5573 5576 e STAGEDIR
5596 5593 5589 e STAGEDIR
5618 5615 5589 e STAGEDIR
6022 6019 6003 e STAGEDIR
6033 6030 6028 e STAGEDIR
6049 6047 4492 e STAGEDIR
6087 6085 4492 e STAGEDIR
6092 6090 6089 e STAGEDIR
6166 6164 6089 e STAGEDIR
6215 6212 6208 e STAGEDIR
6225 6222 6220 e STAGEDIR
6503 6501 6089 e STAGEDIR
6530 6528 6089 e STAGEDIR
6629 6626 6624 e STAGEDIR
6710 6708 6089 e STAGEDIR

```
6712  6710  6089  e  STAGEDIR
6752  6749  6741  e  STAGEDIR
6841  6839  6089  e  STAGEDIR
6849  6847  6846  e  STAGEDIR
7027  7025  6846  e  STAGEDIR
7042  7039  7029  e  STAGEDIR
7080  7077  7063  e  STAGEDIR
7190  7187  7183  e  STAGEDIR
7206  7203  7183  e  STAGEDIR
7228  7225  7223  e  STAGEDIR
7386  7383  7381  e  STAGEDIR
7426  7424  6846  e  STAGEDIR
7465  7463  6846  e  STAGEDIR
7471  7467  7470  e  STAGEDIR
7630  7627  7625  e  STAGEDIR
7641  7639  6846  e  STAGEDIR
7658  7656  6846  e  STAGEDIR
7665  7663  6846  e  STAGEDIR
7710  7706  7709  e  STAGEDIR
7785  7781  7784  e  STAGEDIR
7818  7816  6846  e  STAGEDIR
7868  7865  7847  e  STAGEDIR
7876  7874  6846  e  STAGEDIR
7951  7949  6846  e  STAGEDIR
7956  7954  7953  e  STAGEDIR
8039  8036  8034  e  STAGEDIR
8049  8047  7953  e  STAGEDIR
8082  8078  8081  e  STAGEDIR
8091  8089  7953  e  STAGEDIR
8297  8294  8288  e  STAGEDIR
8305  8302  8288  e  STAGEDIR
8313  8311  7953  e  STAGEDIR
8370  8368  7953  e  STAGEDIR
8381  8379  7953  e  STAGEDIR
8595  8592  8590  e  STAGEDIR
8603  8600  8590  e  STAGEDIR
8605  8602  8590  e  STAGEDIR
8829  8827  7953  e  STAGEDIR
8834  8833  8831  e  STAGEDIR
8881  8879  8831  e  STAGEDIR
8996  8992  8995  e  STAGEDIR
9013  9010  8992  e  STAGEDIR
9019  9016  8992  e  STAGEDIR
9036  9032  9035  e  STAGEDIR
9198  9196  8831  e  STAGEDIR
9206  9204  9203  e  STAGEDIR
9231  9229  9203  e  STAGEDIR
9269  9267  9203  e  STAGEDIR
9276  9274  9203  e  STAGEDIR
9287  9285  9203  e  STAGEDIR
9294  9292  9203  e  STAGEDIR
9301  9299  9203  e  STAGEDIR
9303  9301  9203  e  STAGEDIR
9340  9338  9203  e  STAGEDIR
9342  9340  9203  e  STAGEDIR
9388  9386  9203  e  STAGEDIR
9461  9459  9203  e  STAGEDIR
9478  9476  9203  e  STAGEDIR
9536  9534  9203  e  STAGEDIR
9643  9640  9638  e  STAGEDIR
9654  9651  9647  e  STAGEDIR
9673  9669  9672  e  STAGEDIR
9678  9675  9669  e  STAGEDIR
9692  9690  9203  e  STAGEDIR
9720  9717  9713  e  STAGEDIR
9764  9761  9751  e  STAGEDIR
9770  9768  9203  e  STAGEDIR
9775  9773  9772  e  STAGEDIR
9810  9807  9777  e  STAGEDIR
10135  10133  9772  e  STAGEDIR
10141  10137  10140  e  STAGEDIR
10157  10153  10156  e  STAGEDIR
10169  10165  10168  e  STAGEDIR
10201  10197  10200  e  STAGEDIR
10215  10212  10206  e  STAGEDIR
10217  10214  10206  e  STAGEDIR
10312  10310  9772  e  STAGEDIR
10491  10489  9772  e  STAGEDIR
10614  10612  9772  e  STAGEDIR
10629  10625  10628  e  STAGEDIR
10703  10701  9772  e  STAGEDIR
10759  10757  9772  e  STAGEDIR
10789  10787  9772  e  STAGEDIR
10811  10808  10802  e  STAGEDIR
10821  10819  9772  e  STAGEDIR
10836  10832  10835  e  STAGEDIR
10848  10846  9772  e  STAGEDIR
10906  10905  9772  e  STAGEDIR
10931  10929  9772  e  STAGEDIR
11155  11153  9772  e  STAGEDIR
11174  11172  9772  e  STAGEDIR
11187  11183  11186  e  STAGEDIR
11210  11208  9772  e  STAGEDIR

Total Query Time: 50(msec)  Number of Result Nodes: 208
----------------------------
SELECT DISTINCT v0.*
 FROM accel_othello v2,accel_othello v1,accel_othello v0
 WHERE
 v2.tag="play" and v2.pre=v1.par and
 v1.tag="act" and v1.pre=v0.par and
 v0.tag="title" and v0.kind="e";
Query7=/play/act/title
50  49  49  e  TITLE
2090  2089  2089  e  TITLE
4236  4235  4235  e  TITLE
6844  6843  6843  e  TITLE
9201  9200  9200  e  TITLE

Total Query Time: 50(msec)  Number of Result Nodes: 5
----------------------------
SELECT DISTINCT v0.*
 FROM accel_othello v1,accel_othello v0
 WHERE
 v1.tag="play" and v1.pre!=v0.par and
 v0.pre>v1.pre and v0.post<v1.post and
 v0.tag="title" and v0.kind="e";
Query8=/play/*/title
13  12  12  e  TITLE
50  49  49  e  TITLE
53  51  52  e  TITLE
593  591  592  e  TITLE
```

941 939 940 e TITLE
2090 2089 2089 e TITLE
2093 2091 2092 e TITLE
3062 3060 3061 e TITLE
3097 3095 3096 e TITLE
4236 4235 4235 e TITLE
4239 4237 4238 e TITLE
4460 4458 4459 e TITLE
4493 4491 4492 e TITLE
6090 6088 6089 e TITLE
6844 6843 6843 e TITLE
6847 6845 6846 e TITLE
7954 7952 7953 e TITLE
8832 8830 8831 e TITLE
9201 9200 9200 e TITLE
9204 9202 9203 e TITLE
9773 9771 9772 e TITLE

Total Query Time: 60(msec) Number of Result Nodes: 21

```
SELECT DISTINCT v0.*
 FROM accel_othello v4,accel_othello v3,accel_othello
p3,accel_othello c3,accel_othello v0
 WHERE
 v4.tag="play" and v4.pre=v3.par and
 v3.tag="act" and v3.pre=v0.par and
 p3.tag="title" and v3.pre=p3.par and
 c3.pre=p3.pre+1 and c3.tag="ACT I" and
 v0.tag="scene" and v0.kind="e";
```
Query9=/play/act[title=ACT I]/scene
52 589 49 e SCENE
592 937 49 e SCENE
940 2086 49 e SCENE

Total Query Time: 60(msec) Number of Result Nodes: 3

```
SELECT DISTINCT v0.*
 FROM accel_othello v5,accel_othello v4,accel_othello
p4,accel_othello c4,accel_othello v1,accel_othello v0
 WHERE
 v5.tag="play" and v5.pre=v4.par and
 v4.tag="act" and v4.pre=v1.par and
 p4.tag="title" and v4.pre=p4.par and
 c4.pre=p4.pre+1 and c4.tag="ACT I" and
 v1.tag="scene" and v1.pre=v0.par and
 v0.tag="speech" and v0.kind="e";
```
Query10=/play/act[title=ACT I]/scene/speech
57 62 52 e SPEECH
66 69 52 e SPEECH
73 74 52 e SPEECH
78 129 52 e SPEECH
133 134 52 e SPEECH
138 149 52 e SPEECH
153 154 52 e SPEECH
158 209 52 e SPEECH
213 216 52 e SPEECH
220 233 52 e SPEECH
237 238 52 e SPEECH
242 247 52 e SPEECH
251 252 52 e SPEECH
256 261 52 e SPEECH
267 270 52 e SPEECH
274 275 52 e SPEECH
279 280 52 e SPEECH
284 285 52 e SPEECH
289 304 52 e SPEECH
308 309 52 e SPEECH
313 314 52 e SPEECH
318 319 52 e SPEECH
323 324 52 e SPEECH
328 341 52 e SPEECH
345 346 52 e SPEECH
350 355 52 e SPEECH
359 360 52 e SPEECH
364 367 52 e SPEECH
371 374 52 e SPEECH
378 389 52 e SPEECH
393 394 52 e SPEECH
398 401 52 e SPEECH
405 406 52 e SPEECH
410 411 52 e SPEECH
415 416 52 e SPEECH
420 461 52 e SPEECH
465 474 52 e SPEECH
480 511 52 e SPEECH
519 534 52 e SPEECH
538 539 52 e SPEECH
543 554 52 e SPEECH
558 559 52 e SPEECH
563 568 52 e SPEECH
572 575 52 e SPEECH
579 586 52 e SPEECH
597 606 592 e SPEECH
610 611 592 e SPEECH
615 638 592 e SPEECH
642 665 592 e SPEECH
669 672 592 e SPEECH
676 681 592 e SPEECH
685 686 592 e SPEECH
692 697 592 e SPEECH
701 706 592 e SPEECH
710 711 592 e SPEECH
715 734 592 e SPEECH
738 743 592 e SPEECH
749 750 592 e SPEECH
754 757 592 e SPEECH
761 762 592 e SPEECH
766 767 592 e SPEECH
771 772 592 e SPEECH
778 779 592 e SPEECH
783 784 592 e SPEECH
788 789 592 e SPEECH
793 796 592 e SPEECH
803 804 592 e SPEECH
808 809 592 e SPEECH
813 814 592 e SPEECH
820 821 592 e SPEECH
825 830 592 e SPEECH
834 873 592 e SPEECH
877 886 592 e SPEECH
890 895 592 e SPEECH
899 908 592 e SPEECH

912 917 592 e SPEECH
921 934 592 e SPEECH
946 949 940 e SPEECH
953 956 940 e SPEECH
960 961 940 e SPEECH
965 974 940 e SPEECH
978 985 940 e SPEECH
989 992 940 e SPEECH
996 997 940 e SPEECH
1003 1004 940 e SPEECH
1008 1013 940 e SPEECH
1017 1018 940 e SPEECH
1022 1049 940 e SPEECH
1053 1054 940 e SPEECH
1058 1059 940 e SPEECH
1065 1070 940 e SPEECH
1074 1075 940 e SPEECH
1079 1090 940 e SPEECH
1094 1097 940 e SPEECH
1101 1102 940 e SPEECH
1106 1107 940 e SPEECH
1111 1112 940 e SPEECH
1118 1127 940 e SPEECH
1131 1144 940 e SPEECH
1148 1149 940 e SPEECH
1153 1154 940 e SPEECH
1158 1161 940 e SPEECH
1165 1176 940 e SPEECH
1180 1191 940 e SPEECH
1195 1202 940 e SPEECH
1206 1209 940 e SPEECH
1213 1216 940 e SPEECH
1220 1221 940 e SPEECH
1225 1262 940 e SPEECH
1266 1291 940 e SPEECH
1295 1302 940 e SPEECH
1306 1315 940 e SPEECH
1319 1332 940 e SPEECH
1336 1337 940 e SPEECH
1341 1354 940 e SPEECH
1358 1359 940 e SPEECH
1363 1449 940 e SPEECH
1455 1464 940 e SPEECH
1468 1479 940 e SPEECH
1483 1502 940 e SPEECH
1506 1525 940 e SPEECH
1529 1550 940 e SPEECH
1554 1575 940 e SPEECH
1579 1594 940 e SPEECH
1598 1619 940 e SPEECH
1623 1626 940 e SPEECH
1630 1631 940 e SPEECH
1635 1636 940 e SPEECH
1640 1651 940 e SPEECH
1655 1656 940 e SPEECH
1660 1684 940 e SPEECH
1688 1717 940 e SPEECH
1721 1726 940 e SPEECH
1730 1731 940 e SPEECH
1735 1736 940 e SPEECH
1740 1749 940 e SPEECH
1753 1762 940 e SPEECH
1766 1777 940 e SPEECH
1781 1782 940 e SPEECH
1786 1789 940 e SPEECH
1797 1810 940 e SPEECH
1816 1817 940 e SPEECH
1821 1822 940 e SPEECH
1826 1827 940 e SPEECH
1831 1832 940 e SPEECH
1836 1837 940 e SPEECH
1841 1844 940 e SPEECH
1848 1851 940 e SPEECH
1855 1866 940 e SPEECH
1870 1873 940 e SPEECH
1877 1906 940 e SPEECH
1910 1911 940 e SPEECH
1915 1970 940 e SPEECH
1974 1977 940 e SPEECH
1981 1998 940 e SPEECH
2002 2004 940 e SPEECH
2008 2009 940 e SPEECH
2013 2014 940 e SPEECH
2018 2019 940 e SPEECH
2023 2024 940 e SPEECH
2028 2029 940 e SPEECH
2033 2034 940 e SPEECH
2040 2083 940 e SPEECH

Total Query Time: 70(msec) Number of Result Nodes: 163

SELECT DISTINCT v0.*
FROM accel_othello v6,accel_othello v5,accel_othello p5,accel_othello c5,accel_othello v2,accel_othello v1,accel_othello v0
WHERE
v6.tag="play" and v6.pre=v5.par and
v5.tag="act" and v5.pre=v2.par and
p5.tag="title" and v5.pre=p5.par and
c5.pre=p5.pre+1 and c5.tag="ACT I" and
v2.tag="scene" and v2.pre=v1.par and
v1.tag="speech" and v1.pre=v0.par and
v0.tag="line" and v0.kind="e";
Query11=/play/act[title=ACT I]/scene/speech/line
60 57 57 e LINE
62 59 57 e LINE
64 61 57 e LINE
69 66 66 e LINE
71 68 66 e LINE
76 73 73 e LINE
81 78 78 e LINE
83 80 78 e LINE
85 82 78 e LINE
87 84 78 e LINE
89 86 78 e LINE
91 88 78 e LINE
93 90 78 e LINE
95 92 78 e LINE
97 94 78 e LINE
99 96 78 e LINE
101 98 78 e LINE
103 100 78 e LINE

105 102 78 e LINE
107 104 78 e LINE
109 106 78 e LINE
111 108 78 e LINE
113 110 78 e LINE
115 112 78 e LINE
117 114 78 e LINE
119 116 78 e LINE
121 118 78 e LINE
123 120 78 e LINE
125 122 78 e LINE
127 124 78 e LINE
129 126 78 e LINE
131 128 78 e LINE
136 133 133 e LINE
141 138 138 e LINE
143 140 138 e LINE
145 142 138 e LINE
147 144 138 e LINE
149 146 138 e LINE
151 148 138 e LINE
156 153 153 e LINE
161 158 158 e LINE
163 160 158 e LINE
165 162 158 e LINE
167 164 158 e LINE
169 166 158 e LINE
171 168 158 e LINE
173 170 158 e LINE
175 172 158 e LINE
177 174 158 e LINE
179 176 158 e LINE
181 178 158 e LINE
183 180 158 e LINE
185 182 158 e LINE
187 184 158 e LINE
189 186 158 e LINE
191 188 158 e LINE
193 190 158 e LINE
195 192 158 e LINE
197 194 158 e LINE
199 196 158 e LINE
201 198 158 e LINE
203 200 158 e LINE
205 202 158 e LINE
207 204 158 e LINE
209 206 158 e LINE
211 208 158 e LINE
216 213 213 e LINE
218 215 213 e LINE
223 220 220 e LINE
225 222 220 e LINE
227 224 220 e LINE
229 226 220 e LINE
231 228 220 e LINE
233 230 220 e LINE
235 232 220 e LINE
240 237 237 e LINE
245 242 242 e LINE
247 244 242 e LINE
249 246 242 e LINE
254 251 251 e LINE
259 256 256 e LINE
261 258 256 e LINE
263 260 256 e LINE
270 267 267 e LINE
272 269 267 e LINE
277 274 274 e LINE
282 279 279 e LINE
287 284 284 e LINE
292 289 289 e LINE
294 291 289 e LINE
296 293 289 e LINE
298 295 289 e LINE
300 297 289 e LINE
302 299 289 e LINE
304 301 289 e LINE
306 303 289 e LINE
311 308 308 e LINE
316 313 313 e LINE
321 318 318 e LINE
326 323 323 e LINE
331 328 328 e LINE
333 330 328 e LINE
335 332 328 e LINE
337 334 328 e LINE
339 336 328 e LINE
341 338 328 e LINE
343 340 328 e LINE
348 345 345 e LINE
353 350 350 e LINE
355 352 350 e LINE
357 354 350 e LINE
362 359 359 e LINE
367 364 364 e LINE
369 366 364 e LINE
374 371 371 e LINE
376 373 371 e LINE
381 378 378 e LINE
383 380 378 e LINE
385 382 378 e LINE
387 384 378 e LINE
389 386 378 e LINE
391 388 378 e LINE
396 393 393 e LINE
401 398 398 e LINE
403 400 398 e LINE
408 405 405 e LINE
413 410 410 e LINE
418 415 415 e LINE
423 420 420 e LINE
425 422 420 e LINE
427 424 420 e LINE
429 426 420 e LINE
431 428 420 e LINE
433 430 420 e LINE
435 432 420 e LINE
437 434 420 e LINE
439 436 420 e LINE
441 438 420 e LINE
443 440 420 e LINE
445 442 420 e LINE

447 444 420 e LINE
449 446 420 e LINE
451 448 420 e LINE
453 450 420 e LINE
455 452 420 e LINE
457 454 420 e LINE
459 456 420 e LINE
461 458 420 e LINE
463 460 420 e LINE
468 465 465 e LINE
470 467 465 e LINE
472 469 465 e LINE
474 471 465 e LINE
476 473 465 e LINE
483 480 480 e LINE
485 482 480 e LINE
487 484 480 e LINE
489 486 480 e LINE
491 488 480 e LINE
493 490 480 e LINE
495 492 480 e LINE
497 494 480 e LINE
499 496 480 e LINE
501 498 480 e LINE
503 500 480 e LINE
505 502 480 e LINE
507 504 480 e LINE
509 506 480 e LINE
511 508 480 e LINE
513 510 480 e LINE
522 519 519 e LINE
524 521 519 e LINE
526 523 519 e LINE
528 525 519 e LINE
530 527 519 e LINE
532 529 519 e LINE
534 531 519 e LINE
536 533 519 e LINE
541 538 538 e LINE
546 543 543 e LINE
548 545 543 e LINE
550 547 543 e LINE
552 549 543 e LINE
554 551 543 e LINE
556 553 543 e LINE
561 558 558 e LINE
566 563 563 e LINE
568 565 563 e LINE
570 567 563 e LINE
575 572 572 e LINE
577 574 572 e LINE
582 579 579 e LINE
584 581 579 e LINE
586 583 579 e LINE
588 585 579 e LINE
600 597 597 e LINE
602 599 597 e LINE
604 601 597 e LINE
606 603 597 e LINE
608 605 597 e LINE
613 610 610 e LINE
618 615 615 e LINE
620 617 615 e LINE
622 619 615 e LINE
624 621 615 e LINE
626 623 615 e LINE
628 625 615 e LINE
630 627 615 e LINE
632 629 615 e LINE
634 631 615 e LINE
636 633 615 e LINE
638 635 615 e LINE
640 637 615 e LINE
645 642 642 e LINE
647 644 642 e LINE
649 646 642 e LINE
651 648 642 e LINE
653 650 642 e LINE
655 652 642 e LINE
657 654 642 e LINE
659 656 642 e LINE
661 658 642 e LINE
663 660 642 e LINE
665 662 642 e LINE
667 664 642 e LINE
672 669 669 e LINE
674 671 669 e LINE
679 676 676 e LINE
681 678 676 e LINE
683 680 676 e LINE
688 685 685 e LINE
695 692 692 e LINE
697 694 692 e LINE
699 696 692 e LINE
704 701 701 e LINE
706 703 701 e LINE
708 705 701 e LINE
713 710 710 e LINE
718 715 715 e LINE
720 717 715 e LINE
722 719 715 e LINE
724 721 715 e LINE
726 723 715 e LINE
728 725 715 e LINE
730 727 715 e LINE
732 729 715 e LINE
734 731 715 e LINE
736 733 715 e LINE
741 738 738 e LINE
743 740 738 e LINE
745 742 738 e LINE
752 749 749 e LINE
757 754 754 e LINE
759 756 754 e LINE
764 761 761 e LINE
769 766 766 e LINE
774 771 771 e LINE
781 778 778 e LINE
786 783 783 e LINE
791 788 788 e LINE
796 793 793 e LINE
798 795 793 e LINE

806 803 803 e LINE
811 808 808 e LINE
816 813 813 e LINE
823 820 820 e LINE
828 825 825 e LINE
830 827 825 e LINE
832 829 825 e LINE
837 834 834 e LINE
839 836 834 e LINE
841 838 834 e LINE
843 840 834 e LINE
845 842 834 e LINE
847 844 834 e LINE
849 846 834 e LINE
851 848 834 e LINE
853 850 834 e LINE
855 852 834 e LINE
857 854 834 e LINE
859 856 834 e LINE
861 858 834 e LINE
863 860 834 e LINE
865 862 834 e LINE
867 864 834 e LINE
869 866 834 e LINE
871 868 834 e LINE
873 870 834 e LINE
875 872 834 e LINE
880 877 877 e LINE
882 879 877 e LINE
884 881 877 e LINE
886 883 877 e LINE
888 885 877 e LINE
893 890 890 e LINE
895 892 890 e LINE
897 894 890 e LINE
902 899 899 e LINE
904 901 899 e LINE
906 903 899 e LINE
908 905 899 e LINE
910 907 899 e LINE
915 912 912 e LINE
917 914 912 e LINE
919 916 912 e LINE
924 921 921 e LINE
926 923 921 e LINE
928 925 921 e LINE
930 927 921 e LINE
932 929 921 e LINE
934 931 921 e LINE
936 933 921 e LINE
949 946 946 e LINE
951 948 946 e LINE
956 953 953 e LINE
958 955 953 e LINE
963 960 960 e LINE
968 965 965 e LINE
970 967 965 e LINE
972 969 965 e LINE
974 971 965 e LINE
976 973 965 e LINE
981 978 978 e LINE
983 980 978 e LINE
985 982 978 e LINE
987 984 978 e LINE
992 991 989 e LINE
999 996 996 e LINE
1006 1003 1003 e LINE
1011 1008 1008 e LINE
1013 1010 1008 e LINE
1015 1012 1008 e LINE
1020 1017 1017 e LINE
1025 1022 1022 e LINE
1027 1024 1022 e LINE
1029 1026 1022 e LINE
1031 1028 1022 e LINE
1033 1030 1022 e LINE
1035 1032 1022 e LINE
1037 1034 1022 e LINE
1039 1036 1022 e LINE
1041 1038 1022 e LINE
1043 1040 1022 e LINE
1045 1042 1022 e LINE
1047 1044 1022 e LINE
1049 1046 1022 e LINE
1051 1048 1022 e LINE
1056 1053 1053 e LINE
1061 1058 1058 e LINE
1068 1065 1065 e LINE
1070 1067 1065 e LINE
1072 1069 1065 e LINE
1077 1074 1074 e LINE
1082 1079 1079 e LINE
1084 1081 1079 e LINE
1086 1083 1079 e LINE
1088 1085 1079 e LINE
1090 1087 1079 e LINE
1092 1089 1079 e LINE
1097 1094 1094 e LINE
1099 1096 1094 e LINE
1104 1101 1101 e LINE
1109 1106 1106 e LINE
1114 1111 1111 e LINE
1121 1118 1118 e LINE
1123 1120 1118 e LINE
1127 1124 1118 e LINE
1129 1126 1118 e LINE
1134 1131 1131 e LINE
1136 1133 1131 e LINE
1138 1135 1131 e LINE
1140 1137 1131 e LINE
1142 1139 1131 e LINE
1144 1141 1131 e LINE
1146 1143 1131 e LINE
1151 1148 1148 e LINE
1156 1153 1153 e LINE
1163 1160 1158 e LINE
1168 1165 1165 e LINE
1170 1167 1165 e LINE
1172 1169 1165 e LINE
1174 1171 1165 e LINE
1176 1173 1165 e LINE
1178 1175 1165 e LINE

1183	1180	1180	e	LINE
1185	1182	1180	e	LINE
1187	1184	1180	e	LINE
1189	1186	1180	e	LINE
1191	1188	1180	e	LINE
1193	1190	1180	e	LINE
1198	1195	1195	e	LINE
1200	1197	1195	e	LINE
1202	1199	1195	e	LINE
1204	1201	1195	e	LINE
1211	1208	1206	e	LINE
1216	1215	1213	e	LINE
1223	1220	1220	e	LINE
1228	1225	1225	e	LINE
1230	1227	1225	e	LINE
1232	1229	1225	e	LINE
1234	1231	1225	e	LINE
1236	1233	1225	e	LINE
1238	1235	1225	e	LINE
1240	1237	1225	e	LINE
1242	1239	1225	e	LINE
1244	1241	1225	e	LINE
1246	1243	1225	e	LINE
1248	1245	1225	e	LINE
1250	1247	1225	e	LINE
1252	1249	1225	e	LINE
1254	1251	1225	e	LINE
1256	1253	1225	e	LINE
1258	1255	1225	e	LINE
1260	1257	1225	e	LINE
1262	1259	1225	e	LINE
1264	1261	1225	e	LINE
1269	1266	1266	e	LINE
1271	1268	1266	e	LINE
1273	1270	1266	e	LINE
1275	1272	1266	e	LINE
1277	1274	1266	e	LINE
1279	1276	1266	e	LINE
1281	1278	1266	e	LINE
1283	1280	1266	e	LINE
1285	1282	1266	e	LINE
1287	1284	1266	e	LINE
1289	1286	1266	e	LINE
1291	1288	1266	e	LINE
1293	1290	1266	e	LINE
1298	1295	1295	e	LINE
1300	1297	1295	e	LINE
1302	1299	1295	e	LINE
1304	1301	1295	e	LINE
1309	1306	1306	e	LINE
1311	1308	1306	e	LINE
1313	1310	1306	e	LINE
1315	1312	1306	e	LINE
1317	1314	1306	e	LINE
1322	1319	1319	e	LINE
1324	1321	1319	e	LINE
1326	1323	1319	e	LINE
1328	1325	1319	e	LINE
1330	1327	1319	e	LINE
1332	1329	1319	e	LINE
1334	1331	1319	e	LINE
1339	1336	1336	e	LINE
1344	1341	1341	e	LINE
1348	1345	1341	e	LINE
1350	1347	1341	e	LINE
1352	1349	1341	e	LINE
1354	1351	1341	e	LINE
1356	1353	1341	e	LINE
1361	1358	1358	e	LINE
1366	1363	1363	e	LINE
1368	1366	1363	e	LINE
1371	1368	1363	e	LINE
1373	1370	1363	e	LINE
1375	1372	1363	e	LINE
1377	1374	1363	e	LINE
1379	1376	1363	e	LINE
1381	1378	1363	e	LINE
1383	1380	1363	e	LINE
1385	1382	1363	e	LINE
1387	1384	1363	e	LINE
1389	1386	1363	e	LINE
1391	1388	1363	e	LINE
1393	1390	1363	e	LINE
1395	1392	1363	e	LINE
1397	1394	1363	e	LINE
1399	1396	1363	e	LINE
1401	1398	1363	e	LINE
1403	1400	1363	e	LINE
1405	1402	1363	e	LINE
1407	1404	1363	e	LINE
1409	1406	1363	e	LINE
1411	1408	1363	e	LINE
1413	1410	1363	e	LINE
1415	1412	1363	e	LINE
1417	1414	1363	e	LINE
1419	1416	1363	e	LINE
1421	1418	1363	e	LINE
1423	1420	1363	e	LINE
1425	1422	1363	e	LINE
1427	1424	1363	e	LINE
1429	1426	1363	e	LINE
1431	1428	1363	e	LINE
1433	1430	1363	e	LINE
1435	1432	1363	e	LINE
1437	1434	1363	e	LINE
1439	1436	1363	e	LINE
1441	1438	1363	e	LINE
1443	1440	1363	e	LINE
1445	1442	1363	e	LINE
1447	1444	1363	e	LINE
1449	1446	1363	e	LINE
1451	1448	1363	e	LINE
1458	1455	1455	e	LINE
1460	1457	1455	e	LINE
1462	1459	1455	e	LINE
1464	1461	1455	e	LINE
1466	1463	1455	e	LINE
1471	1468	1468	e	LINE
1473	1470	1468	e	LINE
1475	1472	1468	e	LINE
1477	1474	1468	e	LINE
1479	1476	1468	e	LINE

1481	1478	1468	e	LINE
1486	1483	1483	e	LINE
1488	1485	1483	e	LINE
1490	1487	1483	e	LINE
1492	1489	1483	e	LINE
1494	1491	1483	e	LINE
1496	1493	1483	e	LINE
1498	1495	1483	e	LINE
1500	1497	1483	e	LINE
1502	1499	1483	e	LINE
1504	1501	1483	e	LINE
1509	1506	1506	e	LINE
1511	1508	1506	e	LINE
1513	1510	1506	e	LINE
1515	1512	1506	e	LINE
1517	1514	1506	e	LINE
1519	1516	1506	e	LINE
1521	1518	1506	e	LINE
1523	1520	1506	e	LINE
1525	1522	1506	e	LINE
1527	1524	1506	e	LINE
1532	1529	1529	e	LINE
1534	1531	1529	e	LINE
1536	1533	1529	e	LINE
1538	1535	1529	e	LINE
1540	1537	1529	e	LINE
1542	1539	1529	e	LINE
1544	1541	1529	e	LINE
1546	1543	1529	e	LINE
1548	1545	1529	e	LINE
1550	1547	1529	e	LINE
1552	1549	1529	e	LINE
1557	1554	1554	e	LINE
1559	1556	1554	e	LINE
1561	1558	1554	e	LINE
1563	1560	1554	e	LINE
1565	1562	1554	e	LINE
1567	1564	1554	e	LINE
1569	1566	1554	e	LINE
1571	1568	1554	e	LINE
1573	1570	1554	e	LINE
1575	1572	1554	e	LINE
1577	1574	1554	e	LINE
1582	1579	1579	e	LINE
1584	1581	1579	e	LINE
1586	1583	1579	e	LINE
1588	1585	1579	e	LINE
1590	1587	1579	e	LINE
1592	1589	1579	e	LINE
1594	1591	1579	e	LINE
1596	1593	1579	e	LINE
1601	1598	1598	e	LINE
1603	1600	1598	e	LINE
1605	1602	1598	e	LINE
1607	1604	1598	e	LINE
1609	1606	1598	e	LINE
1611	1608	1598	e	LINE
1613	1610	1598	e	LINE
1615	1612	1598	e	LINE
1617	1614	1598	e	LINE
1619	1616	1598	e	LINE
1621	1618	1598	e	LINE
1626	1623	1623	e	LINE
1628	1625	1623	e	LINE
1633	1630	1630	e	LINE
1638	1635	1635	e	LINE
1643	1640	1640	e	LINE
1645	1642	1640	e	LINE
1647	1644	1640	e	LINE
1649	1646	1640	e	LINE
1651	1648	1640	e	LINE
1653	1650	1640	e	LINE
1658	1655	1655	e	LINE
1663	1660	1660	e	LINE
1665	1662	1660	e	LINE
1667	1664	1660	e	LINE
1669	1666	1660	e	LINE
1671	1668	1660	e	LINE
1673	1670	1660	e	LINE
1675	1672	1660	e	LINE
1677	1674	1660	e	LINE
1679	1676	1660	e	LINE
1681	1678	1660	e	LINE
1683	1681	1660	e	LINE
1686	1683	1660	e	LINE
1691	1688	1688	e	LINE
1693	1690	1688	e	LINE
1695	1692	1688	e	LINE
1697	1694	1688	e	LINE
1699	1696	1688	e	LINE
1701	1698	1688	e	LINE
1703	1700	1688	e	LINE
1705	1702	1688	e	LINE
1707	1704	1688	e	LINE
1709	1706	1688	e	LINE
1711	1708	1688	e	LINE
1713	1710	1688	e	LINE
1715	1712	1688	e	LINE
1717	1714	1688	e	LINE
1719	1716	1688	e	LINE
1724	1721	1721	e	LINE
1726	1723	1721	e	LINE
1728	1725	1721	e	LINE
1733	1730	1730	e	LINE
1738	1735	1735	e	LINE
1743	1740	1740	e	LINE
1745	1742	1740	e	LINE
1747	1744	1740	e	LINE
1749	1746	1740	e	LINE
1751	1748	1740	e	LINE
1756	1753	1753	e	LINE
1758	1755	1753	e	LINE
1760	1757	1753	e	LINE
1762	1759	1753	e	LINE
1764	1761	1753	e	LINE
1769	1766	1766	e	LINE
1771	1768	1766	e	LINE
1775	1772	1766	e	LINE
1777	1774	1766	e	LINE
1779	1776	1766	e	LINE
1784	1781	1781	e	LINE
1789	1786	1786	e	LINE

1791 1788 1786 e LINE
1800 1797 1797 e LINE
1802 1799 1797 e LINE
1804 1801 1797 e LINE
1806 1803 1797 e LINE
1808 1805 1797 e LINE
1810 1807 1797 e LINE
1812 1809 1797 e LINE
1819 1816 1816 e LINE
1824 1821 1821 e LINE
1829 1826 1826 e LINE
1834 1831 1831 e LINE
1839 1836 1836 e LINE
1844 1841 1841 e LINE
1846 1843 1841 e LINE
1851 1848 1848 e LINE
1853 1850 1848 e LINE
1858 1855 1855 e LINE
1860 1857 1855 e LINE
1862 1859 1855 e LINE
1864 1861 1855 e LINE
1866 1863 1855 e LINE
1868 1865 1855 e LINE
1873 1870 1870 e LINE
1875 1872 1870 e LINE
1880 1877 1877 e LINE
1882 1879 1877 e LINE
1884 1881 1877 e LINE
1886 1883 1877 e LINE
1888 1885 1877 e LINE
1890 1887 1877 e LINE
1892 1889 1877 e LINE
1894 1891 1877 e LINE
1896 1893 1877 e LINE
1898 1895 1877 e LINE
1900 1897 1877 e LINE
1902 1899 1877 e LINE
1904 1901 1877 e LINE
1906 1903 1877 e LINE
1908 1905 1877 e LINE
1913 1910 1910 e LINE
1918 1915 1915 e LINE
1920 1917 1915 e LINE
1922 1919 1915 e LINE
1924 1921 1915 e LINE
1926 1923 1915 e LINE
1928 1925 1915 e LINE
1930 1927 1915 e LINE
1932 1929 1915 e LINE
1934 1931 1915 e LINE
1936 1933 1915 e LINE
1938 1935 1915 e LINE
1940 1937 1915 e LINE
1942 1939 1915 e LINE
1944 1941 1915 e LINE
1946 1943 1915 e LINE
1948 1945 1915 e LINE
1950 1947 1915 e LINE
1952 1949 1915 e LINE
1954 1951 1915 e LINE
1956 1953 1915 e LINE
1958 1955 1915 e LINE
1960 1957 1915 e LINE
1962 1959 1915 e LINE
1964 1961 1915 e LINE
1966 1963 1915 e LINE
1968 1965 1915 e LINE
1970 1967 1915 e LINE
1972 1969 1915 e LINE
1977 1974 1974 e LINE
1979 1976 1974 e LINE
1984 1981 1981 e LINE
1986 1983 1981 e LINE
1988 1985 1981 e LINE
1990 1987 1981 e LINE
1992 1989 1981 e LINE
1994 1991 1981 e LINE
1996 1993 1981 e LINE
1998 1995 1981 e LINE
2000 1997 1981 e LINE
2005 2003 2002 e LINE
2011 2008 2008 e LINE
2016 2013 2013 e LINE
2021 2018 2018 e LINE
2026 2023 2023 e LINE
2031 2028 2028 e LINE
2036 2033 2033 e LINE
2043 2040 2040 e LINE
2045 2042 2040 e LINE
2047 2044 2040 e LINE
2049 2046 2040 e LINE
2051 2048 2040 e LINE
2053 2050 2040 e LINE
2055 2052 2040 e LINE
2057 2054 2040 e LINE
2059 2056 2040 e LINE
2061 2058 2040 e LINE
2063 2060 2040 e LINE
2065 2062 2040 e LINE
2067 2064 2040 e LINE
2069 2066 2040 e LINE
2071 2068 2040 e LINE
2073 2070 2040 e LINE
2075 2072 2040 e LINE
2077 2074 2040 e LINE
2079 2076 2040 e LINE
2081 2078 2040 e LINE
2083 2080 2040 e LINE
2085 2082 2040 e LINE

Total Query Time: 110(msec) Number of Result Nodes: 737

SELECT DISTINCT v0.*
FROM accel_othello v5,accel_othello p5,accel_othello c5,accel_othello v2,accel_othello v1,accel_othello v0
WHERE
v5.tag="act" and v5.pre=v2.par and
p5.tag="title" and v5.pre=p5.par and
c5.pre=p5.pre+1 and c5.tag="ACT I" and
v2.tag="scene" and v2.pre=v1.par and
v1.tag="speech" and v1.pre=v0.par and

v0.tag="line" and v0.kind="e";
Query12=//act[title=ACT I]/scene/speech/line
60 57 57 e LINE
62 59 57 e LINE
64 61 57 e LINE
69 66 66 e LINE
71 68 66 e LINE
76 73 73 e LINE
81 78 78 e LINE
83 80 78 e LINE
85 82 78 e LINE
87 84 78 e LINE
89 86 78 e LINE
91 88 78 e LINE
93 90 78 e LINE
95 92 78 e LINE
97 94 78 e LINE
99 96 78 e LINE
101 98 78 e LINE
103 100 78 e LINE
105 102 78 e LINE
107 104 78 e LINE
109 106 78 e LINE
111 108 78 e LINE
113 110 78 e LINE
115 112 78 e LINE
117 114 78 e LINE
119 116 78 e LINE
121 118 78 e LINE
123 120 78 e LINE
125 122 78 e LINE
127 124 78 e LINE
129 126 78 e LINE
131 128 78 e LINE
136 133 133 e LINE
141 138 138 e LINE
143 140 138 e LINE
145 142 138 e LINE
147 144 138 e LINE
149 146 138 e LINE
151 148 138 e LINE
156 153 153 e LINE
161 158 158 e LINE
163 160 158 e LINE
165 162 158 e LINE
167 164 158 e LINE
169 166 158 e LINE
171 168 158 e LINE
173 170 158 e LINE
175 172 158 e LINE
177 174 158 e LINE
179 176 158 e LINE
181 178 158 e LINE
183 180 158 e LINE
185 182 158 e LINE
187 184 158 e LINE
189 186 158 e LINE
191 188 158 e LINE
193 190 158 e LINE
195 192 158 e LINE
197 194 158 e LINE
199 196 158 e LINE
201 198 158 e LINE
203 200 158 e LINE
205 202 158 e LINE
207 204 158 e LINE
209 206 158 e LINE
211 208 158 e LINE
216 213 213 e LINE
218 215 213 e LINE
223 220 220 e LINE
225 222 220 e LINE
227 224 220 e LINE
229 226 220 e LINE
231 228 220 e LINE
233 230 220 e LINE
235 232 220 e LINE
240 237 237 e LINE
245 242 242 e LINE
247 244 242 e LINE
249 246 242 e LINE
254 251 251 e LINE
259 256 256 e LINE
261 258 256 e LINE
263 260 256 e LINE
270 267 267 e LINE
272 269 267 e LINE
277 274 274 e LINE
282 279 279 e LINE
287 284 284 e LINE
292 289 289 e LINE
294 291 289 e LINE
296 293 289 e LINE
298 295 289 e LINE
300 297 289 e LINE
302 299 289 e LINE
304 301 289 e LINE
306 303 289 e LINE
311 308 308 e LINE
316 313 313 e LINE
321 318 318 e LINE
326 323 323 e LINE
331 328 328 e LINE
333 330 328 e LINE
335 332 328 e LINE
337 334 328 e LINE
339 336 328 e LINE
341 338 328 e LINE
343 340 328 e LINE
348 345 345 e LINE
353 350 350 e LINE
355 352 350 e LINE
357 354 350 e LINE
362 359 359 e LINE
367 364 364 e LINE
369 366 364 e LINE
374 371 371 e LINE
376 373 371 e LINE
381 378 378 e LINE
383 380 378 e LINE
385 382 378 e LINE
387 384 378 e LINE

389 386 378 e LINE
391 388 378 e LINE
396 393 393 e LINE
401 398 398 e LINE
403 400 398 e LINE
408 405 405 e LINE
413 410 410 e LINE
418 415 415 e LINE
423 420 420 e LINE
425 422 420 e LINE
427 424 420 e LINE
429 426 420 e LINE
431 428 420 e LINE
433 430 420 e LINE
435 432 420 e LINE
437 434 420 e LINE
439 436 420 e LINE
441 438 420 e LINE
443 440 420 e LINE
445 442 420 e LINE
447 444 420 e LINE
449 446 420 e LINE
451 448 420 e LINE
453 450 420 e LINE
455 452 420 e LINE
457 454 420 e LINE
459 456 420 e LINE
461 458 420 e LINE
463 460 420 e LINE
468 465 465 e LINE
470 467 465 e LINE
472 469 465 e LINE
474 471 465 e LINE
476 473 465 e LINE
483 480 480 e LINE
485 482 480 e LINE
487 484 480 e LINE
489 486 480 e LINE
491 488 480 e LINE
493 490 480 e LINE
495 492 480 e LINE
497 494 480 e LINE
499 496 480 e LINE
501 498 480 e LINE
503 500 480 e LINE
505 502 480 e LINE
507 504 480 e LINE
509 506 480 e LINE
511 508 480 e LINE
513 510 480 e LINE
522 519 519 e LINE
524 521 519 e LINE
526 523 519 e LINE
528 525 519 e LINE
530 527 519 e LINE
532 529 519 e LINE
534 531 519 e LINE
536 533 519 e LINE
541 538 538 e LINE
546 543 543 e LINE
548 545 543 e LINE
550 547 543 e LINE
552 549 543 e LINE
554 551 543 e LINE
556 553 543 e LINE
561 558 558 e LINE
566 563 563 e LINE
568 565 563 e LINE
570 567 563 e LINE
575 572 572 e LINE
577 574 572 e LINE
582 579 579 e LINE
584 581 579 e LINE
586 583 579 e LINE
588 585 579 e LINE
600 597 597 e LINE
602 599 597 e LINE
604 601 597 e LINE
606 603 597 e LINE
608 605 597 e LINE
613 610 610 e LINE
618 615 615 e LINE
620 617 615 e LINE
622 619 615 e LINE
624 621 615 e LINE
626 623 615 e LINE
628 625 615 e LINE
630 627 615 e LINE
632 629 615 e LINE
634 631 615 e LINE
636 633 615 e LINE
638 635 615 e LINE
640 637 615 e LINE
645 642 642 e LINE
647 644 642 e LINE
649 646 642 e LINE
651 648 642 e LINE
653 650 642 e LINE
655 652 642 e LINE
657 654 642 e LINE
659 656 642 e LINE
661 658 642 e LINE
663 660 642 e LINE
665 662 642 e LINE
667 664 642 e LINE
672 669 669 e LINE
674 671 669 e LINE
679 676 676 e LINE
681 678 676 e LINE
683 680 676 e LINE
688 685 685 e LINE
695 692 692 e LINE
697 694 692 e LINE
699 696 692 e LINE
704 701 701 e LINE
706 703 701 e LINE
708 705 701 e LINE
713 710 710 e LINE
718 715 715 e LINE
720 717 715 e LINE
722 719 715 e LINE
724 721 715 e LINE

726 723 715 e LINE
728 725 715 e LINE
730 727 715 e LINE
732 729 715 e LINE
734 731 715 e LINE
736 733 715 e LINE
741 738 738 e LINE
743 740 738 e LINE
745 742 738 e LINE
752 749 749 e LINE
757 754 754 e LINE
759 756 754 e LINE
764 761 761 e LINE
769 766 766 e LINE
774 771 771 e LINE
781 778 778 e LINE
786 783 783 e LINE
791 788 788 e LINE
796 793 793 e LINE
798 795 793 e LINE
806 803 803 e LINE
811 808 808 e LINE
816 813 813 e LINE
823 820 820 e LINE
828 825 825 e LINE
830 827 825 e LINE
832 829 825 e LINE
837 834 834 e LINE
839 836 834 e LINE
841 838 834 e LINE
843 840 834 e LINE
845 842 834 e LINE
847 844 834 e LINE
849 846 834 e LINE
851 848 834 e LINE
853 850 834 e LINE
855 852 834 e LINE
857 854 834 e LINE
859 856 834 e LINE
861 858 834 e LINE
863 860 834 e LINE
865 862 834 e LINE
867 864 834 e LINE
869 866 834 e LINE
871 868 834 e LINE
873 870 834 e LINE
875 872 834 e LINE
880 877 877 e LINE
882 879 877 e LINE
884 881 877 e LINE
886 883 877 e LINE
888 885 877 e LINE
893 890 890 e LINE
895 892 890 e LINE
897 894 890 e LINE
902 899 899 e LINE
904 901 899 e LINE
906 903 899 e LINE
908 905 899 e LINE
910 907 899 e LINE
915 912 912 e LINE
917 914 912 e LINE
919 916 912 e LINE
924 921 921 e LINE
926 923 921 e LINE
928 925 921 e LINE
930 927 921 e LINE
932 929 921 e LINE
934 931 921 e LINE
936 933 921 e LINE
949 946 946 e LINE
951 948 946 e LINE
956 953 953 e LINE
958 955 953 e LINE
963 960 960 e LINE
968 965 965 e LINE
970 967 965 e LINE
972 969 965 e LINE
974 971 965 e LINE
976 973 965 e LINE
981 978 978 e LINE
983 980 978 e LINE
985 982 978 e LINE
987 984 978 e LINE
992 991 989 e LINE
999 996 996 e LINE
1006 1003 1003 e LINE
1011 1008 1008 e LINE
1013 1010 1008 e LINE
1015 1012 1008 e LINE
1020 1017 1017 e LINE
1025 1022 1022 e LINE
1027 1024 1022 e LINE
1029 1026 1022 e LINE
1031 1028 1022 e LINE
1033 1030 1022 e LINE
1035 1032 1022 e LINE
1037 1034 1022 e LINE
1039 1036 1022 e LINE
1041 1038 1022 e LINE
1043 1040 1022 e LINE
1045 1042 1022 e LINE
1047 1044 1022 e LINE
1049 1046 1022 e LINE
1051 1048 1022 e LINE
1056 1053 1053 e LINE
1061 1058 1058 e LINE
1068 1065 1065 e LINE
1070 1067 1065 e LINE
1072 1069 1065 e LINE
1077 1074 1074 e LINE
1082 1079 1079 e LINE
1084 1081 1079 e LINE
1086 1083 1079 e LINE
1088 1085 1079 e LINE
1090 1087 1079 e LINE
1092 1089 1079 e LINE
1097 1094 1094 e LINE
1099 1096 1094 e LINE
1104 1101 1101 e LINE
1109 1106 1106 e LINE
1114 1111 1111 e LINE

1121	1118	1118	e	LINE
1123	1120	1118	e	LINE
1127	1124	1118	e	LINE
1129	1126	1118	e	LINE
1134	1131	1131	e	LINE
1136	1133	1131	e	LINE
1138	1135	1131	e	LINE
1140	1137	1131	e	LINE
1142	1139	1131	e	LINE
1144	1141	1131	e	LINE
1146	1143	1131	e	LINE
1151	1148	1148	e	LINE
1156	1153	1153	e	LINE
1163	1160	1158	e	LINE
1168	1165	1165	e	LINE
1170	1167	1165	e	LINE
1172	1169	1165	e	LINE
1174	1171	1165	e	LINE
1176	1173	1165	e	LINE
1178	1175	1165	e	LINE
1183	1180	1180	e	LINE
1185	1182	1180	e	LINE
1187	1184	1180	e	LINE
1189	1186	1180	e	LINE
1191	1188	1180	e	LINE
1193	1190	1180	e	LINE
1198	1195	1195	e	LINE
1200	1197	1195	e	LINE
1202	1199	1195	e	LINE
1204	1201	1195	e	LINE
1211	1208	1206	e	LINE
1216	1215	1213	e	LINE
1223	1220	1220	e	LINE
1228	1225	1225	e	LINE
1230	1227	1225	e	LINE
1232	1229	1225	e	LINE
1234	1231	1225	e	LINE
1236	1233	1225	e	LINE
1238	1235	1225	e	LINE
1240	1237	1225	e	LINE
1242	1239	1225	e	LINE
1244	1241	1225	e	LINE
1246	1243	1225	e	LINE
1248	1245	1225	e	LINE
1250	1247	1225	e	LINE
1252	1249	1225	e	LINE
1254	1251	1225	e	LINE
1256	1253	1225	e	LINE
1258	1255	1225	e	LINE
1260	1257	1225	e	LINE
1262	1259	1225	e	LINE
1264	1261	1225	e	LINE
1269	1266	1266	e	LINE
1271	1268	1266	e	LINE
1273	1270	1266	e	LINE
1275	1272	1266	e	LINE
1277	1274	1266	e	LINE
1279	1276	1266	e	LINE
1281	1278	1266	e	LINE
1283	1280	1266	e	LINE
1285	1282	1266	e	LINE
1287	1284	1266	e	LINE
1289	1286	1266	e	LINE
1291	1288	1266	e	LINE
1293	1290	1266	e	LINE
1298	1295	1295	e	LINE
1300	1297	1295	e	LINE
1302	1299	1295	e	LINE
1304	1301	1295	e	LINE
1309	1306	1306	e	LINE
1311	1308	1306	e	LINE
1313	1310	1306	e	LINE
1315	1312	1306	e	LINE
1317	1314	1306	e	LINE
1322	1319	1319	e	LINE
1324	1321	1319	e	LINE
1326	1323	1319	e	LINE
1328	1325	1319	e	LINE
1330	1327	1319	e	LINE
1332	1329	1319	e	LINE
1334	1331	1319	e	LINE
1339	1336	1336	e	LINE
1344	1341	1341	e	LINE
1348	1345	1341	e	LINE
1350	1347	1341	e	LINE
1352	1349	1341	e	LINE
1354	1351	1341	e	LINE
1356	1353	1341	e	LINE
1361	1358	1358	e	LINE
1366	1363	1363	e	LINE
1368	1366	1363	e	LINE
1371	1368	1363	e	LINE
1373	1370	1363	e	LINE
1375	1372	1363	e	LINE
1377	1374	1363	e	LINE
1379	1376	1363	e	LINE
1381	1378	1363	e	LINE
1383	1380	1363	e	LINE
1385	1382	1363	e	LINE
1387	1384	1363	e	LINE
1389	1386	1363	e	LINE
1391	1388	1363	e	LINE
1393	1390	1363	e	LINE
1395	1392	1363	e	LINE
1397	1394	1363	e	LINE
1399	1396	1363	e	LINE
1401	1398	1363	e	LINE
1403	1400	1363	e	LINE
1405	1402	1363	e	LINE
1407	1404	1363	e	LINE
1409	1406	1363	e	LINE
1411	1408	1363	e	LINE
1413	1410	1363	e	LINE
1415	1412	1363	e	LINE
1417	1414	1363	e	LINE
1419	1416	1363	e	LINE
1421	1418	1363	e	LINE
1423	1420	1363	e	LINE
1425	1422	1363	e	LINE
1427	1424	1363	e	LINE
1429	1426	1363	e	LINE
1431	1428	1363	e	LINE

1433	1430	1363	e	LINE
1435	1432	1363	e	LINE
1437	1434	1363	e	LINE
1439	1436	1363	e	LINE
1441	1438	1363	e	LINE
1443	1440	1363	e	LINE
1445	1442	1363	e	LINE
1447	1444	1363	e	LINE
1449	1446	1363	e	LINE
1451	1448	1363	e	LINE
1458	1455	1455	e	LINE
1460	1457	1455	e	LINE
1462	1459	1455	e	LINE
1464	1461	1455	e	LINE
1466	1463	1455	e	LINE
1471	1468	1468	e	LINE
1473	1470	1468	e	LINE
1475	1472	1468	e	LINE
1477	1474	1468	e	LINE
1479	1476	1468	e	LINE
1481	1478	1468	e	LINE
1486	1483	1483	e	LINE
1488	1485	1483	e	LINE
1490	1487	1483	e	LINE
1492	1489	1483	e	LINE
1494	1491	1483	e	LINE
1496	1493	1483	e	LINE
1498	1495	1483	e	LINE
1500	1497	1483	e	LINE
1502	1499	1483	e	LINE
1504	1501	1483	e	LINE
1509	1506	1506	e	LINE
1511	1508	1506	e	LINE
1513	1510	1506	e	LINE
1515	1512	1506	e	LINE
1517	1514	1506	e	LINE
1519	1516	1506	e	LINE
1521	1518	1506	e	LINE
1523	1520	1506	e	LINE
1525	1522	1506	e	LINE
1527	1524	1506	e	LINE
1532	1529	1529	e	LINE
1534	1531	1529	e	LINE
1536	1533	1529	e	LINE
1538	1535	1529	e	LINE
1540	1537	1529	e	LINE
1542	1539	1529	e	LINE
1544	1541	1529	e	LINE
1546	1543	1529	e	LINE
1548	1545	1529	e	LINE
1550	1547	1529	e	LINE
1552	1549	1529	e	LINE
1557	1554	1554	e	LINE
1559	1556	1554	e	LINE
1561	1558	1554	e	LINE
1563	1560	1554	e	LINE
1565	1562	1554	e	LINE
1567	1564	1554	e	LINE
1569	1566	1554	e	LINE
1571	1568	1554	e	LINE
1573	1570	1554	e	LINE
1575	1572	1554	e	LINE
1577	1574	1554	e	LINE
1582	1579	1579	e	LINE
1584	1581	1579	e	LINE
1586	1583	1579	e	LINE
1588	1585	1579	e	LINE
1590	1587	1579	e	LINE
1592	1589	1579	e	LINE
1594	1591	1579	e	LINE
1596	1593	1579	e	LINE
1601	1598	1598	e	LINE
1603	1600	1598	e	LINE
1605	1602	1598	e	LINE
1607	1604	1598	e	LINE
1609	1606	1598	e	LINE
1611	1608	1598	e	LINE
1613	1610	1598	e	LINE
1615	1612	1598	e	LINE
1617	1614	1598	e	LINE
1619	1616	1598	e	LINE
1621	1618	1598	e	LINE
1626	1623	1623	e	LINE
1628	1625	1623	e	LINE
1633	1630	1630	e	LINE
1638	1635	1635	e	LINE
1643	1640	1640	e	LINE
1645	1642	1640	e	LINE
1647	1644	1640	e	LINE
1649	1646	1640	e	LINE
1651	1648	1640	e	LINE
1653	1650	1640	e	LINE
1658	1655	1655	e	LINE
1663	1660	1660	e	LINE
1665	1662	1660	e	LINE
1667	1664	1660	e	LINE
1669	1666	1660	e	LINE
1671	1668	1660	e	LINE
1673	1670	1660	e	LINE
1675	1672	1660	e	LINE
1677	1674	1660	e	LINE
1679	1676	1660	e	LINE
1681	1678	1660	e	LINE
1683	1681	1660	e	LINE
1686	1683	1660	e	LINE
1691	1688	1688	e	LINE
1693	1690	1688	e	LINE
1695	1692	1688	e	LINE
1697	1694	1688	e	LINE
1699	1696	1688	e	LINE
1701	1698	1688	e	LINE
1703	1700	1688	e	LINE
1705	1702	1688	e	LINE
1707	1704	1688	e	LINE
1709	1706	1688	e	LINE
1711	1708	1688	e	LINE
1713	1710	1688	e	LINE
1715	1712	1688	e	LINE
1717	1714	1688	e	LINE
1719	1716	1688	e	LINE
1724	1721	1721	e	LINE
1726	1723	1721	e	LINE

1728	1725	1721	e	LINE
1733	1730	1730	e	LINE
1738	1735	1735	e	LINE
1743	1740	1740	e	LINE
1745	1742	1740	e	LINE
1747	1744	1740	e	LINE
1749	1746	1740	e	LINE
1751	1748	1740	e	LINE
1756	1753	1753	e	LINE
1758	1755	1753	e	LINE
1760	1757	1753	e	LINE
1762	1759	1753	e	LINE
1764	1761	1753	e	LINE
1769	1766	1766	e	LINE
1771	1768	1766	e	LINE
1775	1772	1766	e	LINE
1777	1774	1766	e	LINE
1779	1776	1766	e	LINE
1784	1781	1781	e	LINE
1789	1786	1786	e	LINE
1791	1788	1786	e	LINE
1800	1797	1797	e	LINE
1802	1799	1797	e	LINE
1804	1801	1797	e	LINE
1806	1803	1797	e	LINE
1808	1805	1797	e	LINE
1810	1807	1797	e	LINE
1812	1809	1797	e	LINE
1819	1816	1816	e	LINE
1824	1821	1821	e	LINE
1829	1826	1826	e	LINE
1834	1831	1831	e	LINE
1839	1836	1836	e	LINE
1844	1841	1841	e	LINE
1846	1843	1841	e	LINE
1851	1848	1848	e	LINE
1853	1850	1848	e	LINE
1858	1855	1855	e	LINE
1860	1857	1855	e	LINE
1862	1859	1855	e	LINE
1864	1861	1855	e	LINE
1866	1863	1855	e	LINE
1868	1865	1855	e	LINE
1873	1870	1870	e	LINE
1875	1872	1870	e	LINE
1880	1877	1877	e	LINE
1882	1879	1877	e	LINE
1884	1881	1877	e	LINE
1886	1883	1877	e	LINE
1888	1885	1877	e	LINE
1890	1887	1877	e	LINE
1892	1889	1877	e	LINE
1894	1891	1877	e	LINE
1896	1893	1877	e	LINE
1898	1895	1877	e	LINE
1900	1897	1877	e	LINE
1902	1899	1877	e	LINE
1904	1901	1877	e	LINE
1906	1903	1877	e	LINE
1908	1905	1877	e	LINE
1913	1910	1910	e	LINE
1918	1915	1915	e	LINE
1920	1917	1915	e	LINE
1922	1919	1915	e	LINE
1924	1921	1915	e	LINE
1926	1923	1915	e	LINE
1928	1925	1915	e	LINE
1930	1927	1915	e	LINE
1932	1929	1915	e	LINE
1934	1931	1915	e	LINE
1936	1933	1915	e	LINE
1938	1935	1915	e	LINE
1940	1937	1915	e	LINE
1942	1939	1915	e	LINE
1944	1941	1915	e	LINE
1946	1943	1915	e	LINE
1948	1945	1915	e	LINE
1950	1947	1915	e	LINE
1952	1949	1915	e	LINE
1954	1951	1915	e	LINE
1956	1953	1915	e	LINE
1958	1955	1915	e	LINE
1960	1957	1915	e	LINE
1962	1959	1915	e	LINE
1964	1961	1915	e	LINE
1966	1963	1915	e	LINE
1968	1965	1915	e	LINE
1970	1967	1915	e	LINE
1972	1969	1915	e	LINE
1977	1974	1974	e	LINE
1979	1976	1974	e	LINE
1984	1981	1981	e	LINE
1986	1983	1981	e	LINE
1988	1985	1981	e	LINE
1990	1987	1981	e	LINE
1992	1989	1981	e	LINE
1994	1991	1981	e	LINE
1996	1993	1981	e	LINE
1998	1995	1981	e	LINE
2000	1997	1981	e	LINE
2005	2003	2002	e	LINE
2011	2008	2008	e	LINE
2016	2013	2013	e	LINE
2021	2018	2018	e	LINE
2026	2023	2023	e	LINE
2031	2028	2028	e	LINE
2036	2033	2033	e	LINE
2043	2040	2040	e	LINE
2045	2042	2040	e	LINE
2047	2044	2040	e	LINE
2049	2046	2040	e	LINE
2051	2048	2040	e	LINE
2053	2050	2040	e	LINE
2055	2052	2040	e	LINE
2057	2054	2040	e	LINE
2059	2056	2040	e	LINE
2061	2058	2040	e	LINE
2063	2060	2040	e	LINE
2065	2062	2040	e	LINE
2067	2064	2040	e	LINE
2069	2066	2040	e	LINE
2071	2068	2040	e	LINE

2073 2070 2040 e LINE
2075 2072 2040 e LINE
2077 2074 2040 e LINE
2079 2076 2040 e LINE
2081 2078 2040 e LINE
2083 2080 2040 e LINE
2085 2082 2040 e LINE

Total Query Time: 130(msec) Number of Result Nodes: 737

```
SELECT DISTINCT v0.*
 FROM accel_othello v4,accel_othello p4,accel_othello c4,accel_othello v1,accel_othello v0
 WHERE
 v4.tag="act" and v4.pre=v1.par and
 p4.tag="title" and v4.pre=p4.par and
 c4.pre=p4.pre+1 and c4.tag="ACT I" and
 v1.tag="scene" and v1.pre=v0.par and
 v0.tag="stagedir" and v0.kind="e";
```

Query13=//act[title=ACT I]/scene/stagedir

55 53 52 e STAGEDIR
265 263 52 e STAGEDIR
478 476 52 e STAGEDIR
515 513 52 e STAGEDIR
517 515 52 e STAGEDIR
590 588 52 e STAGEDIR
595 593 592 e STAGEDIR
690 688 592 e STAGEDIR
747 745 592 e STAGEDIR
776 774 592 e STAGEDIR
800 799 592 e STAGEDIR
818 816 592 e STAGEDIR
938 936 592 e STAGEDIR
943 942 940 e STAGEDIR
1001 999 940 e STAGEDIR
1063 1061 940 e STAGEDIR
1116 1114 940 e STAGEDIR
1453 1451 940 e STAGEDIR
1793 1793 940 e STAGEDIR
1814 1812 940 e STAGEDIR
2038 2036 940 e STAGEDIR
2087 2085 940 e STAGEDIR

Total Query Time: 40(msec) Number of Result Nodes: 22

REFERENCES

[1] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu, “A Query Language for XML”, In proceeding of the Eight World-Wide Web Conference, 1999.

[2] B. Cooper, N. Sample, M. Franklin, G. Hjaltason, and M. Shadmon, “A Fast Index for Semistructured Data”, The 27th international Conference on Very Large Data Bases, September, 2001.

[3] C. Bohm, S. Berchtold, H. Kriegel, and U. Michel, “Multidimensional Index Structures in Relational Databases”, Journal of Intelligent Information Systems, 2000.

[4] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, “On Supporting Containment Queries in Relational Database Management Systems”, ACM SIGMOD International Conference on Management of Data, May, 2001.

[5] D. Chamberlin, “XQuery: An XML query language”, IBM systems Journal, Vol 41, No. 4, 2002.

[6] D. Chamberlin, J. Robie and D. Florescu, “Quilt: An XML Query Language for Heterogeneous Data Sources”, Proceedings of WebDB, 2000.

[7] D. Florescu, D. Kossmann and I. Manolescu, “Integrating Keyword Search into XML Query Processing”, The International Journal of Computer and Telecommunications Networking, Vol 33, Issue 1-6(119-135), 2000.

[8] DOM (Document Object Model), http://www.w3.org/DOM/

[9] D. Suciu and T. Milo, “Index structures for path expressions”, In ICDT: 7th International Conference on Database Theory, 1999.

[10] H. He and J. Yang, "Multiresolution Indexing of XML for Frequent Queries", ICDE, 2004.

[11] H. Kriegel, M. Potke, and T. Seidl, "Managing Intervals Efficiently in Object-Relational Databases", In the 26th International Conference on Very Large Databases, September, 2000.

[12] H. Wang, S. Park, W. Fan, and P. S. Yu, "ViST: A dynamic index method for Querying XML Data by Tree Structures", SIGMOD, 2003.

[13] I. Kamel and C. Faloutsos, "On Packing R-Trees", International Conference on Information and Knowledge Management, November, 1993.

[14] L. Guo, F. Shao, C. Botev, and J. Shanmugasudaram, "XRANK: Ranked Keyword Search over XML Documents", SIGMOD, June, 2003.

[15] M. F. Fernandez, D. Florescu, A. Y. Levy and D. Suciu, "A query language for a Web-site management system", ACM SIGMOD Record, 26(3):4-11, 1997.

[16] MySQL, http://dev.mysql.com/doc/refman/4.1/

[17] N. Manandhar, "Structure Based XML Indexing", MS Thesis, CSE Department, University of Texas at Arlington, December, 2005.

[18] Niagara data, http://www.cs.wisc.edu/niagara/data.html

[19] P. Buneman, S. Davidson, G. Hillebrand and D. Suciu, "A Query Language and Optimization Techniques for Unstructured Data", In Proceeding of the 1996 ACM SIGMOD International Conference on Management of Data, pages 505-516, June, 1996.

[20] P. Dietz and D. Sleator, "Two Algorithms for Maintaining Order in a List", ACM Symposium on Theory of Computing, 1987.

[21] P. Rao, and B. Moon, “PRIX: Indexing and Querying XML Using Prufer Sequences”, ICDE, 2004.

[22] Q. Chen, A. Lim, and K.W. Ong, “D(k)-Index: An Adaptive Structural Summary for Graph-Structured Data”, SIGMOD, June, 2003.

[23] Qing Li, R. Elmasri, S. Prabhakar, N. Manandhar, Do Y. Kim, “A Survey of XML Indexing Techniques”.

[24] Quanzhong Li and B. Moon, “Indexing and Querying XML Data for Regular Path Expressions”, Proceedings of the 27th VLDB Conference, 2001.

[25] R. Elmasri and S. Navathe, “Fundamentals of Database Systems: Forth Edition”, Addison Wesley, 2004.

[26] R. Goldman and J. Widom, “Dataguides: Enabling query formulation and optimization in semistructured databases”, In Twenty-Third International Conference on Very Large DataBases, pages 436.445, 1997.

[27] R. Kaushik, P. Shenoy, P. Bohannon and Ehud Gudes, “Exploiting Local Similarity for Efficient Indexing of Paths in Graph-Structured Data”, ICDE, 2002.

[28] S. Abiteboul, D. Quass, J. McHugh, J. Widom and J. L. Wiener, “The Lorel Query Languages for Semistructured Data”, International Journal on Digital Libraries, 1:68-88, 1997.

[29] SAX (Simple API for XML), http://www.saxproject.org/.

[30] SIGMOD Record in XML, http://www.sigmod.org/record/xml/.

[31] Shakespeare’s plays, http://metalab.unc.edu/bosak/xml/eg/shaks200.zip.

[32] S. Prabhakar, "Indexing Techniques for XML Data", MS Thesis, CSE Department, University of Texas at Arlington, December, 2004.

[33] T. Grust, "Accelerating XPath Location Steps", ACM SIGMOD 2002.

[34] T.Grust, M, Keulen, and J. Teubner, "Accelerating XPath Evaluation in Any RDBMS", ACM Transactions on Database Systems, March, 2004.

[35] V. Hristidis, Y. Papakonstantinou, and A. Balmin, "Keyword Proximity Search on XML Graphs", ICDE, 2003.

[36] World Wide Web Consortium, http://www.w3.org/.

[37] XISS, http://www.cs.arizona.edu/xiss/XISS.htm.

[38] XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath.

BIOGRAPHICAL INFORMATION

Do Youn Kim was born in Seoul, Korea on February 19, 1977. He earned B. S. in Computer Science from Western Illinois University in May, 2003. He came to UTA in August, 2003 as a Masters student in Computer Science. His major interest area was Database and XML. He earned his Masters degree in Computer Science in December 2005. He was going to work for the mobile communication company after graduation.