74.783 Distributed Database System

Project

Query Evaluation and Web Recognition in Document Databases

Instructor: Dr. Yangjun Chen

Student: Kang Shi (6776229)

August 1, 2003
Abstract

A web and document databases (WDDB) is a system to manage efficiently local documents and semantic connection to remote databases. WDDB integrates search engine, query treatment and document management and provide a powerful mechanism to guide the access of information sources distributed all over the world. One WDDB system includes a local document database, a web connector, and a web recognizer. The query in WDDB system is a path-oriented query, and composed by local query and remote query. The query can be implemented by using tree-embedding algorithm and signature mechanism. WDDB system can send query to remote documents databases and get the result through web connector if the query cannot retrieval results in local database. In addition, WDDB system can collect internet information using the web recognizer efficiently.

1. Introduction

Internet develops in a very high speed in recent decades. People are accustomed to find useful information from Internet. However, a great amount of information always confuses people and makes them lose in the maze of hyperlinks. Therefore, how to find target information quickly and accurately is an important task. In addition, information collection is also an important task in modern internet. When people search information in Internet, they want to obtain not only the accurate result but also some related information of the searching target.

To solve these growing and challenging problems, some solutions have been developed. The first solution is search engines such as Google, AltaVista, Yahoo, and so on. Most search engines can analyze hyperlinks in web pages and put the keywords of web pages in search databases. User can get a group of corresponding hyperlinks of the web pages according to the search keywords. However, base on the keywords matching, the search engines have limited functions in numerous web pages and information. The second solution is web query languages such as W3QL [4]. The third solution is semi-structured data management systems. The last one is document databases. These solutions can not integrate search engine, query treatment and document management in one system.

Chen’s [1] research, web and document databases (WDDB), gives an integrated method to implement search engine, query treatment and document management. Based on XML
and distributed databases, WDDB system can be defined as a triple \(<D, U, W>\), where \(D\) represents a local XML documents database, \(U\) represents a set of URLs, and \(W\) represents a web recognizer.

2. System Architecture

The architecture of WDDB system includes 3 parts, document management, web connection, and web recognizer as described in figure 1.

![Figure 1  System Architecture of WDDB system](image)

The part of document management includes DTDs, schemas management, documents loading, query evaluation, documents database and interface. The module of schemas management and document loading establishes a data schema for a given XML DTD and converts the corresponding XML documents to the database relations according to the XML DTDs. The module of query evaluation can execute path-oriented queries on local and remote document database. The interface module supports a interaction between users and the WDDB system.

The part of web connection stores a set of URLs that point to remote document database related to the query concepts. Therefore, a query can be sent to the associated remote document database to get results when the query concepts are not available in the local documents database.

The part of web recognizer is used to recognize remote information sources for a query concept.
3. XML

The WDDB system is based on XML documents. XML [3] stands for extensible markup language. XML is a set of rules for defining semantic tags that break a document into parts and identify different parts of the documents. Figure 2 shows a simple XML document that contains information of a hotel.

```
<hotel-room-reservation filecode="1302">
    <name>Travel-lodge</name>
    <location>
        <city-or-district>Winnipeg</city-or-district>
        <state>Manitoba</state>
        <country>Canada</country>
        <address>
            <number>500</number>
            <street>Portage Ave.</street>
            <post-code>R3B 2E9</post-code>
        </address>
    </location>
    <type>
        <rooms>one-bed-room</rooms>
        <price>$119.00</price>
    </type>
    <reservation-time>
        <from>April 20, 2002</from>
        <to>April 28, 2002</to>
    </reservation-time>
</hotel-room-reservation>
```

An XML document is defined as having elements and attributes. Elements are always marked up with tags that are identified with “<””. One element may be associated with on or more attributes. In figure 2, “location”, “rooms”, “type” and so on are elements. Each element contains sub-elements or a text value. “filecode” is an attribute that associate with element “hotel-room-reservation”. An attribute has a value. These particular elements and attributes are declared by document type definitions (DTD). DTD provides a list of the elements and attributes contained in a document, as well as their relationships to one another. Also, DTDs specify a set of rules for the structure of a
4. Storage of Documents in a WDDB system

Documents in a WDDB system are stored as XML documents. According to the structure of XML documents and the description of DTDs, an XML document can be represented as a tree. The nodes in the tree are equivalent to the elements, attributes and text.

An elements node has element name as the label. Each node has no child node or has one more children nodes. One child node has one of the three types (element, attribute, and text). An attribute node has a label includes the attribute name and value. An attribute node has no child node. A text node has a string value as its label, and has no child.

To store the XML documents in relation databases correctly and efficiently, three policies must be applied. The first policy is DTD independent. That means the database schemas to store XML documents should not depend on DTDs or element types. Any XML document can be manipulated in predefined relations. The second policy is that the structures of a document stored in the database will not loss the structural information. The last one is that the maintenance cost should be kept in minimum.

Chen’s [1] research splits the whole XML document into three parts: elements, attributes, and text, and put the three parts into three relations individually. The elements relation
can be defined as:
{
    DocID:       <integer>,
    ID:          <integer>,
    Ename:       <string>,
    firstChildID: <integer>,
    siblingID:   <integer>,
    attributeID: <integer>.
}

<table>
<thead>
<tr>
<th>Doc ID</th>
<th>ID</th>
<th>Ename</th>
<th>firstChildID</th>
<th>SiblingID</th>
<th>AttributeID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>hotel-room-reservation</td>
<td>2</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>name%</td>
<td>1(T)</td>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>location</td>
<td>4</td>
<td>11</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>city-or-district%</td>
<td>2(T)</td>
<td>5</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>state%</td>
<td>3(T)</td>
<td>6</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>country%</td>
<td>4(T)</td>
<td>7</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>address</td>
<td>8</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>number%</td>
<td>5(T)</td>
<td>9</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>street%</td>
<td>6(T)</td>
<td>10</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>post-code%</td>
<td>7(T)</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>type</td>
<td>12</td>
<td>14</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>rooms%</td>
<td>8(T)</td>
<td>13</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>price%</td>
<td>9(T)</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>reservation-time</td>
<td>15</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>from%</td>
<td>10(T)</td>
<td>15</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>to%</td>
<td>11(T)</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 1  Relation elements

Table 1 shows the relation elements. In this relation, the value of “Ename” attribute suffixed with “%” indicates that the first child of this element is a text that stored in relation text. Therefore, the values of these elements in “firstChildID” point to the relation text. Furthermore, in the columns “siblingID”, the “*” represents the element has no sibling node.

The text relation can be defined as:
{
    DocID:       <integer>,
    textID:      <integer>,
}
The attribute relation can be defined as:

{ DocID: <integer>,
  att-ID: <integer>,
  parentID: <integer>,
  att-name: <string>,
  att-value: <string>.
}

In relation attribute, the attribute value of “parentID” points to a corresponding tuple in relation element.

## 5. Query in a WDDB system

Query is an important function in WDDB system. In WDDB system, a query includes two parts, a local query and a remote query. The local query can be evaluated against the
local database and the remote query was sent to remote databases.

### 5.1. Path-oriented queries

The query in WDDB system is a path-oriented query. Some notations are used to represent different signification.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>selects from immediate child nodes</td>
</tr>
<tr>
<td>//</td>
<td>selects from arbitrary descendant nodes</td>
</tr>
<tr>
<td>@</td>
<td>precedes attribute names</td>
</tr>
<tr>
<td>$</td>
<td>indicates the reserve words</td>
</tr>
<tr>
<td>?</td>
<td>Indicates the variables</td>
</tr>
<tr>
<td>^</td>
<td>“and” operator</td>
</tr>
<tr>
<td>[ ]</td>
<td>condition sentence in the same level</td>
</tr>
</tbody>
</table>

Table 4  Notations in query

In URL technology, http://localhost used to represent local web server. Therefore, different from Chen’s [1] research, both the local query and the remote query can be represented as: `<URL><query string>`. As a result, the format of the queries can be unified. The query can be represented in following form:

http://localhost/hotel-room-reservation/name?x^  
/hotel-room-reservation/location [city-or-district = Winnipeg’]^  
/hotel-room-reservation/location/address [street = ‘500 Portage Ave.’]^  
/car-rental/company/name?y^  
/car-rental/company/location [city-or-district = ‘Winnipeg’]^  
/car-rental/company/car-type?z

The query enquires the name of the hotel that located at 500 Portage Ave., Winnipeg. At the same time, the query enquires any car rental company located in Winnipeg and the available car type in that company. If the local database can just keep the hotel information, the first part of the query can be answered except the second one. In this case, the WDDB system will send the second part of the query to remote documents databases which contain the car rental information. The remote WDDB system will return the query result to local system.
5.2. Evaluation of local queries

5.2.1. Tree-embedding algorithm

In WDDB system, both the documents and queries can be considered as labeled trees and the evaluation of a local query can be thought of as a tree-embedding problem. Labeled tree is that each node in the tree has a label of some given alphabet. Obviously, an XML document can be represented as a labeled tree. Labeled tree one embeds labeled tree two if the nodes in tree two preserves labels and ancestor-ship as in tree one. In figure 4, T1 and T2 are both labeled trees, and T1 embeds T2 because the nodes in T2 keep the labels and ancestor-ship as in tree T1.

Chen describes an algorithm for check whether T2 is embedded in T1. In contrast from traditional algorithm, Chen’s algorithm is a bottom-up algorithm with better time complexity. The algorithm can be described as following:

```plaintext
tree-embedding (v)
input: the root node of T1.
output: the forests of T2 contained in T1.
S := ∅
if v is a leaf node
   then return S ⊕ {(u,.) | u ∈ leaves (T2) and label (u) = label (v)}
else {let v1, …, v k be all the children of v;
   for i from 1 to k do S := S ⊕ tree-embeeding(vi);
   S’ := {(u, .) | (u, -) ∈ S and label(u) = label(v);
   S” := {(u, .) | u ∈ leaves(T2) and label(u) = label(v);
   return S ⊃ S’ ⊃ S”;
}
In the algorithm,
   (v, i, j) represents an interval [i, j] under node v. That means the subtrees rooted
   at the nodes with the sibling node i, i+1, …, j.
```

Figure 4 labeled tree and tree embedding
(v, -) represents an interval includes all the children of v. 
(v, .) represents the subtree rooted at v. 
leaves (T) represents all the leaf nodes of a tree T.

5.2.2. Signature

To find all result of a query with the algorithm in 5.1.1, we have to check all documents in the databases. This process will take a long time, so another mechanism must be applied before running the algorithm to filter out some non-qualifying documents to reduce the cost of the query.

A signature [2] for a keyword is a hash-coded bit string of length k with m bit set to 1. For example, a hash function H ( ) is used to generate the signature of keyword “database”. If k = 12, m = 4, key word “database” can be decomposed into “dat”, “ata”, “tab”, “aba”, “bas” and “ase”. The hash function H ( ) is applied on each part like H(dat)=5, H(ata)=1, H(tab)=8, H(aba)=10. Therefore, the 1st, 5th, 8th, and 10th bits will be set to 1, and the signature of the “database” is “100 010 010 100”.

A document signature is a bit string that do bitwise inclusive OR operation with signature of each keywords in the document. For example, a document has three key words, “SGML”, “database”, and “information”. Their signatures are “010 000 100 110”, “100 010 010 100”, and “010 100 011 000”. Then the document signature is “110 110 111 110”.

A query signature is a bit string that generated by the signatures of the keywords in the query. In previous section, we have discussed that a query can be represented as a labeled tree. Using the labeled tree, a query signature can be generated. For example, the query “/hotel-room-reservation [/name?x ^ /locato in [/city-or-district = ‘Winnipeg’ ^ /address/street = ‘500 Portage Ave.’ ] ]” can be represented as a tree in figure 5.
In the query tree, the signature of each leaf node can be calculated using the hash function. The signature of “X” is “000 000 000 000” because “X” is a variable of the query. The signature of “Winnipeg” is “101 100 000 100” and the signature of “500 Portage Ave.” is “100 110 000 100”. As shown in figure 6, the signature of the parent node is generated by doing bitwise OR operation among their children nodes. For example, the signature of “location” is equal to the signature of “address” bitwise or the signature of “city-or-district”. The signature of the query is equal to the signature of the root node in the query tree.

When a query is executed, a signature for the query will be generated using the same
hash function as the hash function to generate the document signature. Before using the
tree-embedding algorithm to find corresponding results, a signature comparison is
executed between the signature of the query and the signature of the document. As a
result, many non-qualifying documents are discarded, and the query cost is decreased.
The signature comparison between the query and the document can bring three results. If
the query signature bitwise OR the document signature is equal to the query signature and
the document contains the query keyword, the document matches the query. If the query
signature bitwise OR the document signature is not equal to the query signature, the
document does not match the query. If the query signature bitwise OR the document
signature is equal to the signature of the query but the document does not contain the
query keywords, it is a false drop. In order to eliminate false drops, the document must be
examined by a tree-embedding check.

5.3. Web connection

When a query cannot find corresponding result in local documents databases, the WDDB
system will send the query to remote documents databases. Therefore, the WDDB system
must know the target to send the query. This process can be done by maintaining a
association list of concepts. In contrast from Chen’s research, each item of an association
list can be defined as <G, C, R, S>, where G represent an information unit. C represents a
set of URLs that connect to some remote databases containing the relevant information. R
represents a set of documents signatures of remote documents databases which the
corresponding URL points to. For example, before WDDB system sends a query to
remote database through the URL, the signature of the query and the signature of the
remote documents database are compared first. The query can be sent through the
matching databases’ URLs to reduce the communication cost. S is a descriptor of the
relationship between G and C.

In an association list, a same concept may appear several times and some concepts are
possibly closely related. To handle these issues, a mediator [5] is composed of two parts:
ontology and a set of articulations. In WDDB system, a mediator can be defined to be a
tree structure and a set of URLs. In the tree structure, each node is a pattern used to
identify relevant information sources. The URLs point to the web pages matching the
pattern.

5.4. Web recognizer

To collect remote information, recognizing remote web pages is important. The HTML
and XML web pages consist of a hierarchical structure. Consequently, the
tree-embedding algorithm can be used as a recognizing mechanism.
6. Conclusion

WDDB system is an integrated system to implement search engine, query treatment and document management. One WDDB system is consisted of three parts: a local document database, a web connector, and a web recognizer. The WDDB system is based on XML and converts the XML documents structures to relations and trees. The query in WDDB system is a path-oriented query, and the query can be implemented by using tree-embedding algorithm and signature mechanism. Furthermore, WDDB system can send the local query to remote documents databases and get the result through web connector when the local query cannot retrieval results in local database. In addition, WDDB system can collect internet information using the web recognizer.

Reference