We connect to remote information sources by storing the data items in the local database. More concretely, the recognizer identifies information sources related to common data with the local one, and \( W \) for a Web recognizer that identifies information sources related to remote databases which share common data structurally. URLs related to local data, a Web recognizer (\( W \)) is needed.

However, if some URL links (\( U \)) are available and the corresponding documents into the database. The system can manage to access those different databases. In this case, one has to switch over to remote database automati cally. In addition, to obtain the relationships between them and the relevant local data, a Web recognizer (\( W \)) is needed.

Abstractly, a WDDB can be defined as a triple \(<D, U, W>\), where \( D \) stands for a local document database to store XML documents structurally, \( U \) represents a set of URLs with each pointing to a remote database that shares some common data with the local one, and \( W \) is a Web recognizer that identifies information sources related to remote databases automatically. In addition, to obtain the relationships between them and the relevant local data, a Web recognizer (\( W \)) is needed.

In this paper, we discuss a WDDB system to provide a powerful mechanism to guide the access of information and to manage efficiently local documents and their semantic information source reachable over the network. Mainly, it contains:

1. A module for the schema management and the document database.
3. A module for web connection. In a local WDDB, a special functionality.

The system contains mainly three parts with each for a special functionality.

Recently, with the expansion of the Web, more and more comprehensive information repositories can be now visited easily through networks. A growing and challenging problem is how to quickly find information of interest to the user who is surfing the Web, one may get lost in the maze of hyperlinks. A great deal of work has been done to mitigate this problem to some extent, including search engines such as AltaVista, google, Lycos, and Yahoo! that may not be available locally. To decide where to send the query, an address book has to be maintained, which can be established manually or automatically using Web recognition techniques. A remote query has to be sent to another database, whereas a local query can be considered as tree-embedding problem. A great deal of work has been done to mitigate this problem to some extent, including search engines such as AltaVista, google, Lycos, and Yahoo! that may not be available locally. To decide where to send the query, an address book has to be maintained, which can be established manually or automatically using Web recognition techniques. A remote query has to be sent to another database, whereas a local query can be considered as tree-embedding problem. A local query is an instance of a tree embedding problem and can be sped-up using the so-called signature technique. A Web and Document Database (WDDB) is a system to manage efficiently local documents and their semantic information source reachable over the network. Mainly, it contains:

1. A module for the schema management and the document database.
3. A module for web connection. In a local WDDB, a special functionality.

The system contains mainly three parts with each for a special functionality.
To keep the tree structure of documents when loading them into a relational database, we propose the following storage representation.

In Fig. 2, we show a simple XML document, which contains element-tags, element-texts and attributes for elements. By means of the tags, the tree structure of the document is represented.

In a WDDB, documents are stored in a document database. The query evaluation is addressed in Section 4 in detail.

3. Storage of documents in a WDDB

In XML format. It may be connected to remote document databases distributed over the network. All the relevant document databases are considered to be semantically connected through URLs. In this section, we mainly discuss the storage of XML documents in a WDDB. The query against the local database while the remote query has to be sent to remote databases. In this section, we first give a general description of the WDDB’s queries in 4.1. Then, we discuss the evaluation of local queries and remote queries in 4.2 and 4.3, respectively.

A query may be composed of two parts: a local query and a remote query. The local query can be evaluated against the local database while the remote query has to be sent to remote databases. In this section, we first give a general description of the WDDB’s queries in 4.1. Then, we discuss the evaluation of local queries and remote queries in 4.2 and 4.3, respectively.

One should notice that a text always takes an element as its element, which is stored in the relation Element.

In the relation, the attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). In this relation, the attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). The attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). The attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). The attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). The attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute). The attribute parentID is used for the identifiers of the corresponding elements (stored in relation Attribute).
Several paths can be jointed together using $\land$ is a predicate, enquiring whether element "para" contains a where /letter//body is a path and [para $contains$'visited']

The following is a simple path-oriented query:

\[
\text{Example 1.} \quad \text{As an example, consider the trees:}
\]

\[
\begin{align*}
\text{Tree 1:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Y} \\
\text{Tree 2:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Z}
\end{align*}
\]

\[
\text{The above query can be represented in a compact form by integrating the common parts of multiple paths as shown below.}
\]

\[
\text{Assume that the local document database can answer the query , and how a local WDDB becomes aware of other } \text{issues in 4.3.}
\]

\[
\text{The problem is how to determine where to send a remote query , and how a local WDDB becomes aware of other document databases and knows what they have. We discuss the data on car-rental. The local database will issue a re-}
\]

\[
\text{A remote query can be of the form: <URL><query>. For example, the query to some remote document databases pointed to by some URLs, which contain the information on the car }
\]

\[
\text{by some URLs, which contain the information on the car }
\]

\[
\text{remote rental. If one of the remote document databases is able to provide the information on hotel room reservations, but fail to inform on car }
\]

\[
\text{rental. That is, it can provide the information on hotel room reservations, but fail to inform on car }
\]

\[
\text{The above query can be represented in a compact form by }
\]

\[
\text{Example 2.} \quad \text{As an example, consider the trees:}
\]

\[
\begin{align*}
\text{Tree 1:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Y} \\
\text{Car Type: Any} \\
\text{Tree 2:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Z} \\
\text{Car Type: Any}
\end{align*}
\]

\[
\text{The above query can be represented in a compact form by }
\]

\[
\begin{align*}
\text{Locate in Winnipeg and any car types that are available in }
\end{align*}
\]

\[
\text{rental company/}
\]

\[
\text{name=y}/
\]

\[
\text{car-rntal/}
\]

\[
\text{company/}
\]

\[
\text{name=y}/
\]

\[
\text{car-t ype=z].
\]

\[
\text{Example 3.} \quad \text{As an example, consider the trees:}
\]

\[
\begin{align*}
\text{Tree 1:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Y} \\
\text{Car Type: Any} \\
\text{Tree 2:} & \quad \text{Location: Winnipeg} \\
& \quad \text{Address: 510 Portage Ave.} \\
& \quad \text{Car Company Name: Z} \\
\text{Car Type: Any}
\end{align*}
\]

\[
\text{The above query can be represented in a compact form by }
\]

\[
\begin{align*}
\text{Define the so-call tree-embedding of the tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]

\[
\text{A tree is called a labeled tree if }
\]

\[
\text{An XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{Obviously, an XML document can be represented as a tree with the internal nodes labeled with tags and the leaves }
\]

\[
\text{show that for evaluating a query we will check whether a }
\]

\[
\text{tree representing the document as shown in Fig. 4.}
\]
\textbf{Definition 3.} Let \( T \) be a tree and \( r \) be the root of \( T \). The algorithm works top-down. First, it checks whether the subtree of \( T \) rooted at \( r \) matches the root \( r_1 \) of \( T_1 \), too. If such containment \( \mathbf{1} \) exists some \( r_2 \) of \( T_2 \), resp ecti vel y; \( T_1 \) contains \( r_1 \), too. If such containment \( \mathbf{1} \) exists some \( r_2 \) of \( T_2 \), resp ecti vel y; \( T_1 \) contains \( r_1 \), too. Then, the document signature: 110 110 11 1 1 10 can be obtained by superimposing the signatures of its child nodes. For example, for the local part of our exemplary query, the nodes.

\textbf{Definition 4.} Document signature is denoted \( \mathbf{1} \), the purpose of using a signature file is to screen out most of the nonqualifying documents. A signature failing to match the query signature guarantees that the corresponding subquery in \( \mathbf{1} \) is a tag, \( \mathbf{1} \) is a tag, \( \mathbf{1} \) is a tag.

The query signature tree and the query signature tree are shown in Fig. 5. Signature generation and comparison. Their signatures are shown as signatures with relevant documents in a database and the average number of documents. The goal of document signatures is to work as an inexact query object:

\begin{align*}
\text{John} & \lor \text{professor} \\
\text{hotel} & \lor \text{room} \\
\text{reservation} & \lor \text{street} \\
\text{515} & \text{Portage Ave.}
\end{align*}

The signatures of a non-leaf node in \( \mathbf{1} \) is of the form: \( \text{kw}_1 \lor \text{kw}_2 \lor \text{kw}_3 \lor \ldots \lor \text{kw}_n \) where each \( \text{kw}_i \) is of the form: \( \text{p}_1 \land \text{p}_2 \land \text{p}_3 \land \ldots \land \text{p}_m \). Then, all the paths appearing in \( \mathbf{1} \) are of the form: \( \text{kw}_1 \land \text{kw}_2 \land \text{kw}_3 \land \ldots \land \text{kw}_n \). Then, all the paths appearing in \( \mathbf{1} \) are of the form: \( \text{kw}_1 \land \text{kw}_2 \land \text{kw}_3 \land \ldots \land \text{kw}_n \). For example, for the local part of our exemplary query, the nodes.

\begin{align*}
\text{John} & \lor \text{professor} \\
\text{hotel} & \lor \text{room} \\
\text{reservation} & \lor \text{street} \\
\text{515} & \text{Portage Ave.}
\end{align*}
This can be done by maintaining a so-called association list, the system can switch over to the document associated with that query. As mentioned in the previous section, to evaluate a remote information source is recognized.

4.3.1 Then, in 4.3.2, we discuss how a remote information gathering is done. We notice that this optimization is recursively applied.

4.3.2 Web recognizer

The above algorithm is similar to the Algorithm following algorithm. Its inputs are a document tree and a signature set of URLs pointing to the web pages matching the pattern.

**Algorithm 6**

**Input:** two tree:

**Output:** if

**Let**

s be the subtrees of T1, and T2.

if

T1 does not match r

then

return T1

else if

T1 contains T2

then

return T2

else

let 2 does not match r

then

return T2

else

let

2 = T1 and T2

return T1 and T2.

Fig. 6. Query tree and query signature

\[ \text{pattern}_1; \text{pattern}_2; \text{pattern}_3 = \{ \text{let}, \ldots, \}, \text{tree-embedding} \text{pattern}_1; \text{tree-embedding} \text{pattern}_2; \text{tree-embedding} \text{pattern}_3 \}

\[ \text{tree-embedding} \text{pattern}_1 \text{tree-embedding} \text{pattern}_2 \text{tree-embedding} \text{pattern}_3 \]

\[ \text{name location} \text{?x} \text{date} \text{time} \text{day} \text{vehicle rental} \text{car} \text{reservation} \text{hotel} \text{room} \text{reservation} \text{hotel} \text{res} \]

\[ \text{tree-embedding} \text{pattern}_1 \text{tree-embedding} \text{pattern}_2 \text{tree-embedding} \text{pattern}_3 \]