A Rule-based Query Language for HTML

Mengchi Liu
Department of Computer Science
University of Regina
Regina, Saskatchewan
Canada S4S 0A2
mliu@cs.uregina.ca

Tok Wang Ling
School of Computing
National University of Singapore
Lower Kent Ridge Road
Singapore 119260
lingtw@comp.nus.edu.sg

Abstract

With the recent popularity of the web, enormous amount of information is now available on line. Most web documents available over the web are in HTML format and are hierarchically structured in nature. How to query such web documents based on their internal hierarchical structure becomes more and more important. In this paper, we present a rule-based language called WebQL to support effective and flexible web queries. Unlike other web query languages, WebQL is a high level declarative query language with a logical semantics. It allows us to query web documents based on their internal hierarchical structures. It supports not only negation and recursion, but also query result restructuring in a natural way. We also describe the implementation of the system that supports the WebQL query language.

1 Introduction

With the recent popularity of the web, enormous amount of information is now available on line. Most web documents available over the web conform to the HTML specification. They are intended to be human readable through a browser and thus are constructed following some common conventions and often exhibit some hierarchical structure. How to query such web documents based on its internal structure becomes more and more important.

In the past few years, a number of query languages and systems have been developed in the database community to retrieve data from the web, such as W3QS [6], WebSQL [15], WebLog [10], UnQL [5], Lorel [2], WebQOL [3], Strudel [7] and Florid [9]. For surveys, see [1, 8, 16]. However, most proposals use relational, graph-based or tree-based data models to represent the web data. They focus on inter-document structures, with little attention to intra-document structure and thus can only represent the web data at a very rough level. For example, none of the existing data models and query languages can capture the internal structure of the latest DBLP bibliography server web page of Michael Ley at *http://www.informatik.uni-trier.de/∼ley/db* shown in Figure 1 in a simple and natural way and use such structure to express practical queries.

In [13], we presented a conceptual model for HTML. It has only a few simple constructs but is able to represent the complex hierarchical structure in the web documents at a high level that is close to human conceptualization/visualization of the documents. Also, a set of rules were presented to convert HTML documents into this conceptual model.

In this paper, we present a rule-based language called WebQL based on this conceptual model. Unlike other web query languages, WebQL is a declarative query language with a logical semantics. It allows us to query web documents based on their internal hierarchical structures. It supports not only negation and recursion, but also query result restructuring in a natural way. It can be used in two different ways: one is for the user to query the internal structure of the web documents; the other is for the user to query parts of the web documents when they know part of the internal structure of the web documents. As the conceptual model on which the rule-based language is based is high level, user queries in the rule-based language is also quite high level.

The rest of the paper is organized as follows. Section 2 presents the syntax of the language. Section 3 gives several query examples. Section 4 provides the logical semantics for the language. Section 5 describes the implementation of our web query and inference system that supports WebQL. Section 6 summarizes and points out further research issues.

2 Syntax of WebQL

We assume the existence of two kinds of disjoint symbols: a set $\mathcal{C}$ of constants containing the set $\mathcal{U}$ of URLs, and a set $\mathcal{V}$ of variables started with '\$' followed by a string and '\$' itself is an anonymous variable.

Definition 1 The *terms* are defined recursively as follows:

(1) A constant is a *lexical* term.
(2) If X is a constant or a variable, and Y is a URL or a variable, then $X\langle Y\rangle$ is a *linking* term, and X is called the *label* and Y is called the *anchor* of the linking term. When X or Y is the anonymous variable, we can simply use $\langle Y\rangle$ or $X\langle\rangle$ respectively.
(3) If X and Y are terms, then $X \Rightarrow Y$ is an *attributed* term.
(4) If $X_1, ..., X_n$ are terms, then $\{X_1, ..., X_n\}$ is a *bag* term.
(5) A variable is either an *atomic* term, a *linking* term, a *label* term, a *URL* term, an *attributed* term, or a *bag* term depending on the context.

Example 1 The following are several examples of terms:

Lexical terms:	*CS Dept, John Smith, \$Name*
Linking terms:	*Faculty⟨fac.html⟩, Faculty⟨\$U⟩, ⟨\$U⟩, Faculty⟨⟩, \$F⟨fac.html⟩, ⟨fac.html⟩*
Attributed terms:	*Title ⇒CS Dept, Program ⇒{\$D}, \$A ⇒\$V, \$A ⇒\$L⟨\$U⟩, \$A⟨\$U⟩⇒\$V*
Bag terms	*{\$X}, {\$X, John}, {Author⟨\$U⟩}*

A term is *ground* if it has no variables. An *object* is a ground term. Corresponding to terms, four kinds of objects are distinguished in WebQL: lexical, linking, attributed, and bag objects.

Definition 2 The *expressions* are defined as follows:

(1) Let U be a URL or a variable and T a term. Then $U : T$ is a *positive expression*.
(2) If P is a positive expression, then $\neg P$ is a *negative expression*.
(3) Arithmetic, string and bag operation expressions are defined using terms in the usual way.

Example 2 The following are several examples of expressions where *a* stands for some constant URL:

Positive expressions:
a : \$X, a : {\$X}, a : {Answer ⇒\$X}, \$U : \$V
Negative expressions:
¬ a : \$X, ¬ a : {Faculty⇒{John}}, ¬ a : {Faculty⟨⟩}
Arithmetic expressions:
*\$A = \$B * 2, \$Age = 2001 - \$Birthyear*
String expressions:
\$FName = John + \$LName, John ∈ \$FName
Bag expressions:
John ∈ \$Faculty, \$S = \$S1 ∪ \$S2

An expression is *ground* if it contains no variables.

Definition 3 A *rule* has the form A :– $L_1, ..., L_n$, where A is a positive expression $u : T$, each L_i is a positive expression, a negative expression, or an arithmetic, string or bag operation expression defined using terms. A rule is *safe* if all variables in the head are covered or limited as defined in [4, 11, 17].

For a negative expression with a bag term in the body of a rule, we can move the negation sign into the bag for convenience. For example, we can use *a : {¬Faculty⇒{John}}* to stand for *¬ a : {Faculty⇒{John}}*. We can also combine positive and negative expressions with the same URL for convenience. For example, we can use the following expression

a : {Faculty⇒{John}, ¬Faculty⇒{Mary}}

to stand for *a:{Faculty⇒{John}, a:{¬ Faculty⇒{Mary}}* in the body of a rule.

Note that the anonymous variable *\$* may appear several times in a rule and their different appearances in general stand for different variables. Thus, it cannot appear in the head of a safe rule.

Definition 4 A *web document* is a safe rule with empty body.

In other words, a web document is ground positive expression.

Example 3 The following is an example of web object:

http://www.cs.uregina.ca/csdept.html : {
  Title ⇒CSDept,
  People⟨people.html⟩⇒{
    Faculty⟨fac.html⟩,
    Staff⟨staff.html⟩,
    Students⟨students.html ⟩},
  Programs ⇒{
    Ph.D Program⟨phd.html⟩,
    M.Sc Program⟨msc.html⟩,
    B.Sc Program⟨bsc.html⟩},
  Research⟨research.html⟩
}

Using the methods presented in [13], we can convert most HTML documents into web documents of WebQL.

Example 4 Consider part of the latest DBLP bibliography server web page of Michael Ley at *http://www.informatik.uni-trier.de/∼ley/db* shown in Figure 1. We can convert it into a web document as shown in Figure 2 with simplified URLs such as a_1, b_1, etc. to fit in the paper.

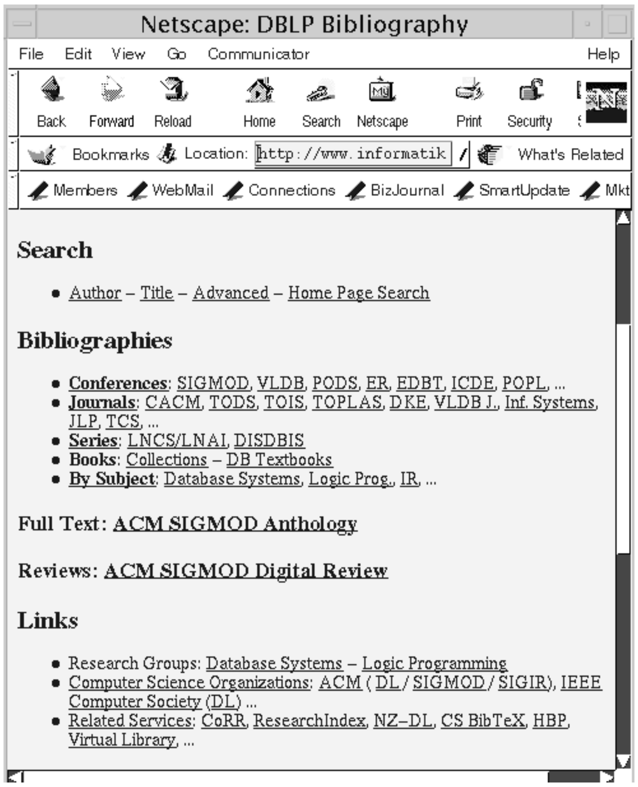

Figure 1. DBLP Bibliography

http://www.informatik.uni-trier.de/∼ley/db : {
Title ⇒*DBLP Bibliography*,
Body ⇒{
Search ⇒{*Author*$\langle a_1\rangle$, *Title*$\langle a_2\rangle$, *Advanced*$\langle a_3\rangle$,
Home Page Search$\langle a_4\rangle$},
Bibliographies ⇒ {
Conferences$\langle b_1\rangle$ ⇒{*SIGMOD*$\langle b_{11}\rangle$, *VLDB*$\langle b_{12}\rangle$,
PODS$\langle b_{13}\rangle$, *ER*$\langle b_{14}\rangle$, ...},
Journals$\langle b_2\rangle$ ⇒{*CACM*$\langle b_{21}\rangle$, *TODS*$\langle b_{22}\rangle$,
TOIS$\langle b_{23}\rangle$, *TOPLAS*$\langle b_{24}\rangle$, ...},
Series$\langle b_3\rangle$ ⇒{*LNCS/LNAI*$\langle b_{41}\rangle$, *DISDBIS*$\langle b_{42}\rangle$},
Books ⇒{*Collections*$\langle b_{51}\rangle$,*DB Textbook*$\langle b_{52}\rangle$},
By Subjects$\langle b_4\rangle$ ⇒{*Database Systems*$\langle b_{61}\rangle$,
Logic Prog$\langle b_{62}\rangle$, *IR*$\langle b_{63}\rangle$}},
Full Text ⇒*ACM SIGMOD Anthology*$\langle c_1\rangle$,
Reviews ⇒*ACM SIGMOD Digital Review*$\langle c_2\rangle$,
Links ⇒{
Research Groups ⇒{*Database Systems*⟨*d1*⟩,
Logic Programming⟨*d2*⟩},
Computer Science Organization$\langle e_1\rangle$ ⇒{
ACM$\langle e_{11}\rangle$ (*DL*$\langle e_{12}\rangle$, *SIGMOD*$\langle e_{13}\rangle$, *SIGIR* $\langle e_{14}\rangle$),
IEEE Computer Society$\langle e_{15}\rangle$(*DL*$\langle e_{16}\rangle$)}
Related Services $\langle f_1\rangle$ ⇒{
CoRR$\langle f_{11}\rangle$, *ResearchIndex*$\langle f_{12}\rangle$, *NZ-DL*$\langle f_{13}\rangle$,
CS BibTex$\langle f_{14}\rangle$, *HBP*$\langle f_{15}\rangle$, *Virtual Library* $\langle f_{16}\rangle$}}
}

Figure 2. DBLP Web Document

In deductive database languages, a query is normally defined as a rule with empty head. If the query contains no variables, the query result is either true or false. If the query contains variables, the query result is a set of bindings that make each ground query true. However, for the web queries, we want not only the set of bindings that make the query true but also proper structuring of the query results. The head of the rule can be used for this purpose. Also, complex queries over the web documents may need more than one rule to express. Thus, we introduce our notion of query as follows.

Definition 5 A *query* is a set of safe rules whose heads have the same URL.

In order to make queries easier, we introduce the following shorthands for rules, terms and expressions appearing in rules:

(1) $X.$ stands for $X \Rightarrow \$$
(2) $X_1.X_2...X_n$ stands for $X_1 \Rightarrow \{X_2 \Rightarrow ...\{X_n\}...\}$
(3) A :– ... $*^n$ X... stands for the following $n+1$ rules:

- A :– ...X...
- A :– ...$\$.X$...

...

- A :– ... $\underbrace{\$...\$}_{n}.X$...

If there are several such dot notations in a rule, then it stands for their various combinations as outlined above.

(4) A :– ... $*$ ... stands for A :– ... $*^c$ X... for some fixed number c.
(5) A :– ...$\langle X\rangle : Y$ stands for A :– ...$\langle X\rangle, X : Y$
(6) A :– ...$\langle X\rangle.Y$ stands for A :– ...$\langle X\rangle, X : \{Y\}$

In other words, $*^n$ stands for 0 to n anonymous variables in the path.

3 Query Examples

The following queries are based on the DBLP web document shown in Figure 2. To make them simple, we use as u_l to stand for *http://www.informatik.uni-trier.de/∼ley/db* and u_o for the URL of the query result.

(Q_1) Copy the contents of the document at the given URL u_l into a local file given by the URL u_o:

u_o: *$X* :– u_l : *$X*

Note that no matter what document pointed by u_l is, such as HTML, postscript, image, executable, etc., it is copied to the destination u_o. However, if we know that it is an HTML document which can be converted into a bag, then we can use the following query instead:

u_o: {*$X*} :– u_l : {*$X*}

which says that every element denoted by *$X* in the bag is also an element in the result bag. The notion {*$X*} in the body of the rule means that *$X* is an element in the corresponding bag whereas the notion {$X} in the head of the rule is used to group the result into a bag. It corresponds to a partial set term in Relationlog [12].

(Q_2). List the objects under the attribute *Search*:

u_o: {*Answer* ⇒*$X*} :– u_l : {**Search* ⇒*$X*}

The result to this query based on the web document in Figure 2 is as follows:

{*Answer*⇒{*Author*⟨a_1⟩, *Title*⟨a_2⟩, *Advanced*⟨a_3⟩, ...}}

(Q_3). List the anchors (URLs) under the attribute *Search*:

u_o: {*Answer*⇒{*$X*}} :– u_l: {**Search* ⇒{⟨*$X* ⟩}}

The result is {*Answer* ⇒{a_1, a_2, a_3}}

(Q_4). List the labels under the attribute *Search*:

u_o: {*Answer* ⇒{*$X*}} :– u_l: {**Search* ⇒{*$X*⟨⟩}}

The result is {*Answer* ⇒{*Author, Title, Advanced, ...*}}. Note that this query can be represented equivalently using the dot notation in WebQL as follows:

u_o: {*Answer* ⇒{*$X*}} :– u_l: {**Search.$X*⟨⟩}

(Q_5). List all the attributes at the first and second levels:

u_o: {*Answer* ⇒{*$X*}} :– u_l: {*$X* ⇒*$Y*}
u_o: {*Answer* ⇒{*$Y*}} :– u_l: {*$X* ⇒*$Y* ⇒*$Z*}

The result is {*Answer* ⇒{*Title, Body, Search, ...*}}. This query can also be represented using equivalently using the dot notation:

u_o: {*Answer* ⇒{*$X*}} :– u_l: {*$X.*}
u_o: {*Answer* ⇒{*$Y*}} :– u_l: {*$X.$Y.*}

(Q_6). Obtain the URL of TODS:

u_o: {*Answer* ⇒*$X*} :– u_l: {**TODS*⟨*$X*⟩}

The result is {*Answer* ⇒b_{22}}

(Q_7). Obtain all the URLs in the page.

u_o: {*Answer*⇒{*$U*}} :– u_l: {***⟨*$U*⟩}

(Q_8). Obtain all the URLs together with their labels.

u_o: {*Answer* ⇒{*$L*⟨*$U*⟩}} :– u_l: {**$L*⟨*$U*⟩}

(Q_9). Get all the URLs reachable from the page.

u_r: {*$X*} :– u_l: {***⟨*$X*⟩}
u_r: {*$X*} :– u_r: {*$Y*}, *$Y*: {***⟨*$X*⟩}

Note that this query involves multiple web documents and are recursive since the web document defined is used in the second rule.

(Q_{10}). Get all the URLs together with their labels reachable from the page.

u_r: {*$L*⟨*$U*⟩} :– u_l: {**$L*⟨*$U*⟩}
u_r: {*$L*⟨*$U*⟩} :– u_r: {**$L*⟨*$U*⟩}, *$U*: {**$L*⟨*$U*⟩}

```
CanadaURL: {
   Title ⇒Canada,
   Body ⇒{
      Geographic ⇒{
         Land boundaries ⇒{
            border countries ⇒{US}
         ...
}

USURL: {
   Title ⇒US,
   Body ⇒{
      Geographic ⇒{
         Land boundaries ⇒{
            border countries ⇒{Canada, Mexico}
         ...
}

MexicoURL: {
   Title ⇒Mexico,
   Body ⇒{
      Geographic ⇒{
         Land boundaries ⇒{
            border countries ⇒{US, ...}
         ...
}

GermanyURL: { ... }

FranceURL: { ... }

...
```

Figure 3. CIA World Factbook

In order to demonstrate the expressive power of WebQL, let us consider the CIA world factbook 2000 at *http://www.odci.gov/cia/publications/factbook*. This web server contains detailed information about each country in the world in HTML format, such as its location, geographic coordinates, area, land boundaries population, etc. We can view the web server as a set of web documents and therefore we can query them and inference useful information. Figure 3 shows part of the web documents in a simplified form.

(Q_{11}) Find countries that border both Germany and France.

$u_o : \{Answer \Rightarrow \{\$N\}$:–
  \$U: {*Title* ⇒ *\$N*,
    **border country* ⇒ {*Germany, France*},

(Q_{12}) Find countries that border Germany but not France.

$u_o : \{Answer \Rightarrow \{\$N\}$:–
  \$U: {*Title* ⇒ *\$N*,
    **border country* ⇒ {*Germany*},
    ¬ **border country* ⇒ {*France*}}

Note that this query involves negation.

(Q_{13}) Find pairs of countries that border the same countries.

u_o:{*Answer*⇒{*Country1*⇒*\$N1*, *Country2*⇒*\$N2*}} :–
  \$U1: {*Title* ⇒*\$N1*, **border countries* ⇒*\$Cs*}
  \$U2: {*Title* ⇒*\$N2*, **border countries* ⇒*\$Cs*}
  N1 ≠ *N2*

(Q_{14}) Find all the countries that can be reached from Canada by land transportation means.

u_r: {*Answer*⇒{*\$C*}} :–
  \$C: {*Title* ⇒*Canada*, **border countries* ⇒{*\$C*}}
u_r : {*Answer*⇒{*\$C*}}:–
  u_r: {*Answer* ⇒{*\$X*}}
  \$U: {*Title* ⇒*\$X*, **border countries* ⇒{*\$C*}}

Note that this is another recursive query.

4 Semantics of WebQL

In this section, we define the Herbrand-like logical semantics for WebQL queries.

Definition 6 The *Herbrand universe* U_H of WebQL is the set of all ground terms that can be formed.

In other words, U_H the domain of all possible objects.

Definition 7 The *Herbrand base* B_H of WebQL is the set of all ground web documents that can be formed using terms in U_H.

That is, B_H is the set of all possible web documents that can be formed.

Definition 8 A *web database* W_D is a subset of B_H.

In other words, a web database is a set of web documents. For example, the CIA world factbook shown in Figure 3 is a web database. The whole world-wide web is also a web database.

Definition 9 A *ground substitution* θ is a mapping from the set of web variables $\mathcal{V} - \{\$\}$ to U_H.

The use of anonymous variables allows us to simply our query rules as demonstrated in the examples above. However, when we deal with semantics, we disallow anonymous variables. We assume that each appearance of anonymous variable is replaced by a non-anonymous variable that never occur in the query rules. This is why we do not map anonymous variable $\$$ to any object in the above definition.

In order to define the semantics, we now introduce the following auxiliary notions.

Definition 10 An object o is *part-of* of an object o', denoted by $o \preceq o'$, if and only if one of the following hold:

(1) both are constants and $o = o'$;
(2) both are linking objects such that one of the following holds:
  - $o \equiv l\langle\rangle$ and $o' \equiv l'\langle u\rangle$ such that $l \preceq l'$;
  - $o \equiv l\langle u\rangle$ and $o' \equiv l'\langle u\rangle$ such that $l \preceq l'$;
  - $o \equiv \langle u\rangle$ and $o' \equiv l'\langle u\rangle$.
(3) both are attributed objects: $o \equiv a \Rightarrow v$ and $o' \equiv a' \Rightarrow v'$ such that $a \preceq a'$ and $v \preceq v'$;
(4) both are bag objects such that for each $o_i \in o - o'$, there exists $o'_i \in o' - o$ such that $o_i \preceq o'_i$.

The part-of relationship between objects o and o' captures the fact that o is part of o'.

Example 5 The following are several examples:

Faculty ⪯ *Faculty*
Faculty ⪯ *Faculty*⟨*fac.html*⟩
⟨*fac.html*⟩⪯ *Faculty*⟨*fac.html*⟩
Programs ⇒{*M.Sc Program*} ⪯
Programs ⇒{*Ph.D Program, M.Sc Program*}
{*Title* ⇒*CSDept*} ⪯ {*Title* ⇒*CSDept, Faculty*⟨*fac.html*⟩}

We need this notion because ground positive expressions in the body of a query should always be part of some web documents. Thus, we extend the part-of relationship to web documents and web databases as follows.

Definition 11 Let $W \equiv u : t$, $W' \equiv u' : t'$ be two web documents. Then W is *part-of* W', denoted by $W \preceq W'$, if and only if $u = u'$ and $t \preceq t'$.

Definition 12 Let DB and DB' be two web databases. Then DB is *part-of* DB', denoted by $DB \preceq DB'$, if and only if for each $W \in DB - DB'$, there exists $W' \in DB' - DB$ such that $W \preceq W'$.

Definition 13 Let DB be a web database. The notion of satisfaction (denoted by $\models$) and its negation (denoted by $\not\models$) based on DB are defined as follows.

(1) For a ground positive expression $u : t$, $DB \models u : t$ if and only if there exists $u : t' \in DB$ such that $t \preceq t'$.
(2) For a ground negative expression $\neg u : t$, $DB \models \neg u : t$ if and only if $DB \not\models u : t$
(3) For each ground arithmetic, string, or bag operation expression ψ, $DB \models \psi$ if and only if ψ is satisfied in the usual sense.
(4) For a rule r of the form $A \text{ :- } L_1, ..., L_n$, $DB \models r$ if and only if for every ground substitution θ, $DB \models \theta L_1, ..., DB \models \theta L_n$ implies $DB \models \theta A$

In other words, a ground positive expression is satisfied if and only if it is part of a web document in the web database; a ground negative expression is satisfied if and only if it is not part of a web document; and a rule is satisfied if there is a web document in the database that satisfies the head of the rule for each ground substitution that makes the body of the rule satisfied.

Example 6 Let DB denote the web database containing DBLP web document in Example 4. Then we have

$$\begin{array}{l}
DB \models u_l : \{*\mathit{Search} \Rightarrow \{\mathit{Author}\langle\rangle\}\} \\
DB \models u_l : \{*\mathit{Search} \Rightarrow \{\mathit{Title}\langle\rangle\}\} \\
DB \models u_l : \{*\mathit{Search} \Rightarrow \{\mathit{Advanced}\langle\rangle\}\} \\
DB \models \neg u_l : \{*\mathit{Search} \Rightarrow \{\mathit{SIGMOD}\langle\rangle\}\} \\
DB \models \neg u_l : \{*\mathit{Journals} \Rightarrow \{\mathit{Author}\langle\rangle\}\} \\
DB \models 6 = 3 * 2 \\
DB \models \mathit{John\ Smith} = \mathit{John} + \mathit{Smith} \\
DB \models \mathit{John} \in \{\mathit{John, Mary, Tony}\}
\end{array}$$

Note that given a query that is a set of safe rules, an existing web database cannot satisfy the head of these rules. We have to generate a new web document using the rules so that the new web document and the existing web database together can satisfy the query.

Definition 14 Let Q be a query. A *model* M of Q is a web database that satisfies Q. A model M of Q is *minimal* if and only if for each model N of Q, $M \preceq N$.

As in deductive databases, we are interested in a minimal model of the query that can be computed bottom-up. We first introduce several auxiliary notions.

Definition 15 Let DB be a web database and Q a set of rules. The *immediate logical consequence* operator T_Q over DB is defined as follows:

$$\begin{array}{rl}
T_Q(DB) = \{\theta A \mid & A \text{ :- } L_1, ..., L_n \in Q \text{ and there exists} \\
& \text{a ground substitution } \theta \text{ such that} \\
& DB \models \theta L_1, ..., DB \models \theta L_n\}
\end{array}$$

Example 7 Consider query Q_4 in the last subsection and the database DB above, we have

$$\begin{array}{rl}
T_{Q_4}(DB) = \{ & u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Author}\}\}, \\
& u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Title}\}\}, \\
& u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Advanced}\}\}, \\
& u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Home\ Page\ Search}\}\}\}
\end{array}$$

Note that the operator T_Q does not perform grouping. Therefore, we introduce the following notions.

Definition 16 Two objects o and o' are *compatible* if and only if one of the following holds:

(1) both are constants and are equal;
(2) $o \equiv a \Rightarrow v$ and $o' \equiv a \Rightarrow v'$ such that v and v' are compatible;
(3) both are bag objects.

A set of objects are *compatible* if and only each pair of them is compatible.

Example 8 The following pairs are compatible:

Author and *Author*
$\{\mathit{Author}\}$ and $\{\mathit{Title}\}$
$\mathit{Answer} \Rightarrow \{\mathit{Author}\}$ and $\mathit{Answer} \Rightarrow \{\mathit{Title}\}$

Definition 17 Two web document $u : t$ and $u' : t'$ are *compatible* if and only if $u = u'$ and t and t' are compatible. A set of web documents are *compatible* if and only if each pair of them is compatible.

Example 9 The following set of web objects are compatible.

$$\begin{array}{l}
u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Author}\}\}, \\
u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Title}\}\}, \\
u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Advanced}\}\}, \\
u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Home\ Page\ Search}\}\}
\end{array}$$

Definition 18 Let S be a set of (web) objects and S' a compatible subset of S. Then S' is a *maximal* compatible set in S if there does not exist a (web) object $o \in S - S'$ that is compatible with each object in S'.

Definition 19 Let S be a set of objects. The *grouping* operator G is defined recursively on S as follows:

(1) If S is a singleton set $S = \{o\}$, then $G(S) = o$
(2) If S is a compatible set of attributed objects $S = \{a \Rightarrow v_1, ..., a \Rightarrow v_n\}$, then $G(S) = a \Rightarrow G(\{v_1, ..., v_n)$
(3) S is a set of bag objects, then
$G(S) = \cup \{G(S') \mid S' = \{o \mid o \in s, \text{ and } s \in S\}$ is a maximal compatible bag of objects$\}$

It is extended to a set of web objects as follows:

(1) If S is a compatible set of web objects of the form $u : o_1, ..., u : o_n$, then $G(S) = u : G(\{o_1, ..., o_n\})$
(2) If S is divided into maximal compatible subsets $S_1, ..., S_n$ such that $S = S_1 \cup ... \cup S_n$, then $G(S) = G(S_1) \cup ... \cup G(S_n)$

Definition 20 The *powers* of the operation T_Q over the web database DB are defined as follows:

$$
\begin{aligned}
&T_Q \uparrow 0(DB) = DB \\
&T_Q \uparrow n(DB) = T_Q(G(T_Q \uparrow n-1(DB))) \cup \\
&\qquad\qquad\qquad T_Q \uparrow n-1(DB) \\
&T_Q \uparrow \omega(DB) = \cup_{n=0}^{\infty} T_Q \uparrow n(DB)
\end{aligned}
$$

Example 10 Continuing with the Example 7, we have

$$
\begin{aligned}
&G(T_{Q_4} \uparrow \omega(DB)) = DB \cup G(\{ \\
&\quad u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Author}\}\}, \\
&\quad u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Title}\}\}, \\
&\quad u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Advanced}\}\}, \\
&\quad u_o : \{\mathit{Answer} \Rightarrow \{\mathit{Home\ Page\ Search}\}\} \\
&\quad \}) \\
&= DB \cup u_0 : G(\{ \\
&\quad \{\mathit{Answer} \Rightarrow \{\mathit{Author}\}\}, \\
&\quad \{\mathit{Answer} \Rightarrow \{\mathit{Title}\}\}, \\
&\quad \{\mathit{Answer} \Rightarrow \{\mathit{Advanced}\}\}, \\
&\quad \{\mathit{Answer} \Rightarrow \{\mathit{Home\ Page\ Search}\}\} \\
&\quad \}) \\
&= DB \cup u_0\text{: } \{ \\
&\quad \mathit{Answer} \Rightarrow \\
&\quad G(\{\{\mathit{Author}\},\{\mathit{Title}\},\{\mathit{Advanced}\}, \\
&\qquad \{\mathit{Home\ Page\ Search}\} \\
&\quad \}) \\
&= DB \cup u_0\text{: } \{ \\
&\quad \mathit{Answer} \Rightarrow \\
&\qquad \{\mathit{Author, Title, Advanced, Home\ Page\ Search}\} \\
&\quad \}
\end{aligned}
$$

which contains the original database DB plus a new web document that satisfies the head of the rule for query Q_4.

Theorem 1 Let DB be a web database and Q a set of query rules. Then $G(T_Q \uparrow \omega(DB))$ is a minimal model of Q.

The semantics of rules in a rule-based language are usually given by the minimal model of the rules, since non-minimal models may contain things that cannot be derived. We do the same for WebQL.

Definition 21 Let DB be a web database and Q a set of query rules. Then the *semantics* of Q under DB is given by $G(T_Q \uparrow \omega(DB))$.

Therefore, given a recursive query, we just need to compute its fixpoint bottom-up and construct the web document that satisfies the rules in the query.

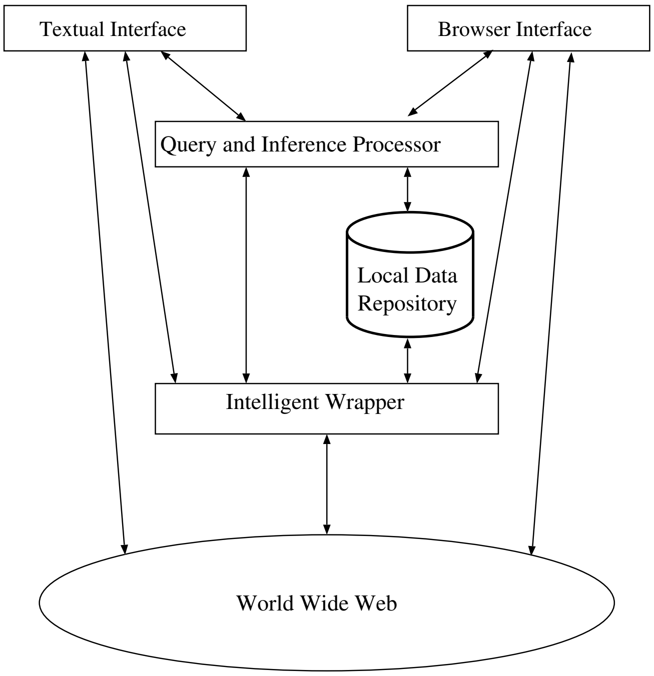

Figure 4. System Architecture

5 Implementation

The WebQL language presented in this paper is part of our web search and inference system project that is currently under implementation at the University of Regina. The architecture of the system being implemented is shown in Figure 4.

The system is organized into four layers. The first layer is the entire world-wide web.

The second layer is the intelligent wrapper. It accesses the world-wide web through the Internet and extra structure and data stored in the local data repository with proper indexing supports for efficient query processing. It converts between web documents in HTML/XML and web documents in WebQL, allows the user to adjust web documents by adding or removing attributes and maintain such adjustment information in the local data repository, and cashes web documents in the local data repository to speed up query answering.

The third layer is the query and inference processor, which is mainly in charge of query processing. It communicates with the user interface layer and uses the data in the local data repository to process the user queries. For recursive queries, it uses semi-naive bottom-up fixpoint computation to generate the result. Simple keyword-based search is also supported by the query and inference processor.

The fourth layer is the user interface. Two kinds of user interfaces are provided: textual user interface and browser user interface. They provide different kinds of environment for the user to express queries and view the results. They accept user commands and queries, display web documents like lynx and netscape respectively, display web documents converted by intelligent wrapper, and invoke the query and inference processor to process queries. It also provides various templates to generate web documents in HTML/XML for query results.

6 Conclusion

In this paper, we have presented WebQL, a rule-based language for querying the HTML documents over the web based on the conceptual model proposed in [13]. Unlike other web query languages, WebQL provides a simple but very powerful way to query both the structure and contents of the HTML documents and to restructure the results. As already shown in the implementation section, this query language can indeed be used to query XML documents since XML documents can be converted into our web documents much easier than HTML documents. We have also defined a fixpoint bottomup semantics for WebQL. The system that supports WebQL is currently under implementation and will soon be available from the web page at *http://www.cs.uregina.ca/∼mliu/WebQL/*.

We would like to extend the functionality of WebQL by adding other useful features to make it a really useful tool for web query and inference and investigate the computability and complexity issues of WebQL queries. Using WebQL, we would also like to develop data extraction tools and data integration tools based on the method proposed in [14]. Our objective is to build an intelligent web search engine on top of the query and inference system.

Acknowledgments The research was partially supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors are also grateful to Yibin Su for implementing the system.

References

[1] S. Abiteboul. Querying Semistructured Data. In *Proceedings of the International Conference on Data Base Theory*, pages 1–18. Springer-Verlag LNCS 1186, 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel Query Language for Semistructured Data. *Intl. Journal of Digital Libraries*, 1(1):68–88, 1997.

[3] G. Arocena and A. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. In *Proceedings of the International Conference on Data Engineering*, pages 24–33. IEEE Computer Society, 1998.

[4] C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur. Set Construction in a Logic Database Language. *Journal of Logic Programming*, 10(3,4):181–232, 1991.

[5] P. Buneman, S. Davidson, G. Hilebrand, and D. Suciu. A Query Language and Optimization Techniques for Unstructured Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 505–516, 1996.

[6] O. S. D. Konopnicki. W3QS: A Query System for the World-Wide Web. In *Proceedings of the International Conference on Very Large Data Bases*, pages 54–65, Zurich,Switzerland, 1995. Morgan Kaufmann Publishers, Inc.

[7] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for a Web-Site Management System. *SIGMOD Record*, pages 4–11, 1997.

[8] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 26(3), 1997.

[9] R. Himmeroder, G. Lausen, B. Ludascher, and C. Schlepphorst. On a declarative semantics for web queries. In *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, pages 386–398, Switzerland, 1997. Springer-Verlag LNCS.

[10] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. A Declarative Language for Querying and Restructuring the Web. In *Proceedings of the 6th International Workshop on Research Issues in Data Engineering*, 1996.

[11] M. Liu. ROL: A Deductive Object Base Language. *Information Systems*, 21(5):431 – 457, 1996.

[12] M. Liu. Relationlog: A Typed Extension to Datalog with Sets and Tuples. *Journal of Logic Programming*, 36(3):271–299, 1998.

[13] M. Liu and T. W. Ling. A Conceptual Model for the Web. In *Proceedings of the International Conference on Conceptual Modeling (ER 2000)*, Salt Lake City, October 9-12 2000. Springer-Verlag LNCS.

[14] M. Liu and T. W. Ling. A Data Model for Semistructured Data with Partial and Inconsistent Information. In *Proceedings of the International Conference on Advances in Database Technology (EDBT 2000)*, pages 317–331, Konstanz, Germany, March 27-31 2000. Springer-Verlag LNCS 1777.

[15] A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web. In *Proceedings of the First International Conference on Parellel and Distributed Information System*, pages 80–91, 1996.

[16] A. O. Mendelzon and T. Milo. Formal Models of Web Queries. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 1997.

[17] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.