8-1998

Join processing in web databases

Sourav S. BHOWMICK

Wee-Keong NG

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Citation

BHOWMICK, Sourav S.; NG, Wee-Keong; and LIM, Ee Peng. Join processing in web databases. (1998). *9th International Conference and Workshop on Database and Expert Systems Applications (DEXA'98)*. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/988

Join Processing in Web Databases

S. S. Bhowmick W.-K. Ng E.-P. Lim S. K. Madria

Center for Advanced Information Systems, School of Applied Science,
Nanyang Technological University, Singapore 639798, SINGAPORE
{sourav,wkn,aseplim,skm}@cais.ntu.edu.sg

Abstract. Recently, there has been increasing interests in data models and query languages for unstructured data in the World Wide Web. When web data is harnessed in a *web warehouse*, new and useful information can be derived through appropriate information manipulation. In our web warehousing project, we introduce a new operator called the *web join*. Like its relational counterpart, web join combines information from two *web tables* to yield a new web table. This paper discusses various issues in web join such as join semantics, joinability, and join evaluation.

1 Introduction

Given the high rate of growth of the volume of data available on the WWW, locating information of interest in such an anarchic setting becomes a more difficult process everyday. Thus, there is the recognition of the undeferring need for effective and efficient tools for information consumers, who must be able to easily locate and manipulate information in the Web. We aim to build a web warehouse containing information extracted from the Web that may also interoperate with conventional data warehouses. To meet the warehousing objective, we materialize web information as a *web table* and define a set of web operators so as to equip the warehouse with the basic capability to manipulate web tables and correlate additional, useful, related web information residing in the web tables [7].

In this paper we focus on the web join operator. It serves two important purposes: First, like its relational counterpart, web join combines information from two web tables based on *some criteria* to yield a new web table. Second, this information can be stored in a separate web table and can be used for future queries. One of the objectives of web join is to capitalize on the reuse of retrieved data from the WWW in order to reduce execution time of queries.

Even though the notion of a web join is similar to that of a relational join, there are number of new challenges due to the richer nature of the web data model. For example, what exactly is a web join operation? How do we perform a

* This work was supported in part by the Nanyang Technological University, Ministry of Education (Singapore) under Academic Research Fund #4-12034-5060, #4-12034-3012, #4-12034-6022. Any opinions, findings, and recommendations in this paper are those of the authors and do not reflect the views of the funding agencies.

web join? What useful information can we gather if we perform a web join? Due to the graphical nature of the Web, the web join process is a much complicated and challenging problem than for relational join. In this paper, we address some of these challenges. In particular, our contributions are:

1. We formally define the notion of web join and discuss the necessity of join operation in a web database.
2. We determine the conditions required to perform a web join between two *web tables*.

2 Related Work

There has been considerable work in data model and query languages for the World Wide Web. But we are not aware of any work which deals with join operation in web databases. For example, Mendelzon, Mihaila and Milo [6] proposed a WebSQL query language based on a formal calculus for querying the WWW. The result of WebSQL query is a set web tuples flattened immediately to linear tuples. This limits the expressiveness of queries to a certain extent as complex queries involving operators such as web join are not possible.

Konopnicki and Shmueli proposed a high level querying system called the W3QS [4] for the WWW whereby users may specify content and structure queries on the WWW and maintain the results of queries as database views of the WWW. In W3QL queries are always made to the WWW. Past query result are not manipulated for the evaluation of future queries. This limit the usage of web operators like web join to derive additional information from the past queries.

Fiebig, Weiss and Moerkotte extended relational algebra to the World Wide Web by augmenting the algebra with new domains (data types) [3], and functions that apply to the domains. The extended model is known as RAW (Relational Algebra for the Web). Only two low level operators on relations, *scan* and *index-scan*, have been proposed to expand an URL address attribute in a relation and to rank results returned by web search engine(s) respectively. RAW provides minor improvement on the existing relational model to accommodate and manipulate web data and there is no notion of a join operation similar to ours.

Inspired by concepts in declarative logic, Lakshmanan, Sadri and Subramanian designed WebLog [5] to be a language for querying and restructuring web information. But there is no formal definition of web operations such as join.

Other proposals, namely Lorel [1] and UnQL [2], aim at querying heterogeneous and semistructured information. These languages adopt a lightweight data model to represent data, based on labeled graphs, and concentrate on the development of powerful query languages for these structures. Moreover, in both these proposals there is no notion of join operation similar to ours .

3 Background

In this section we describe our basic data model and web algebra and then provides the motivation.

3.1 Web Data Model(WDM)

We proposed a data model for a web warehouse in [7] and [8]. Our data model consist of hierarchy of web objects. The fundamental objects are *Nodes* and *Links*. Nodes correspond to HTML or plain text documents and links correspond to hyper-links interconnecting the documents in the World Wide Web. We define a Node type and a Link type to refer to these two sets of distinct objects. These objects consists of a set of attributes as shown below:

```
Node = [url, title, format, size, date, text]
Link = [source-url, target-url, label, link-type]
```

The WDM supports structured or topological querying; different sets of keywords may be specified on the nodes and additional criteria may be defined for the hyperlinks among the nodes. Thus, the query is a graph-like structure and it is used to match the portions of the WWW satisfying the conditions. In this way, the query result is a set of directed graphs consisting of nodes and links (called *web tuples*) reflecting the query graph. A collection of these web tuples is called a *web table*. If the web table is materialized, we associate a *name* with the table. A web table is associated with a schema and is formally defined as 4-tuple $M = \langle X_n, X_\ell, C, P \rangle$ where X_n is a set of node variables, X_ℓ is a set of link variables, C is a set of connectivities in CNF, P is a set of predicates in CNF. The web schema of a web table is the query graph that is used to derive the table. A set of web tables and a set of web schemas is called a *web database*.

We also proposed a *Web Algebra* for retrieving information from the Web and manipulating these information to derive additional information. The algebra provides a formal foundation for data representation and manipulation. We have defined a set of web operators with web semantics so as to equip the web database with the basic capability to manipulate web tables. These operators include *web select*, *web join*, *web union*, *web intersection* and so on. These operators build the foundation for the web algebra. For more details on WDM and web algebra and its motivation see[7]. We illustrate our Web Data Model with an example in the next subsection.

3.2 Motivating Example

Suppose, an user Bill, wishes to find all one bedroom and two bedroom apartments in Singapore whose monthly rent is between \$600—\$1,000 and \$1,000—\$1,200 respectively, and all areas in Singapore from which travel time to Raffles Place is around 20 minutes by bus.

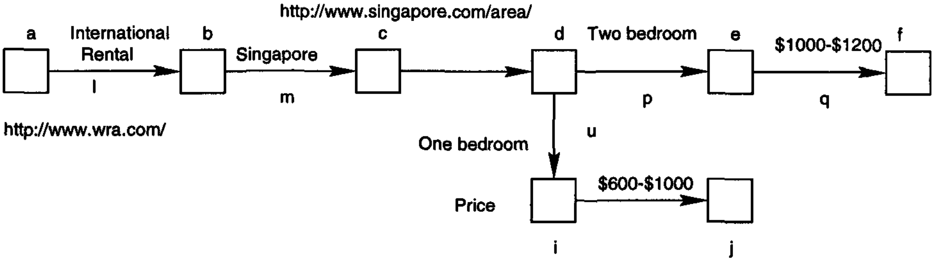

Fig. 1. Web schema(query graph) of Accommodation web table

Suppose, there are two web sites *World Rental Agency* (WRA) (`http://www.wra.com/`) and *World Transportation Agency* (WTA) (`http://www.wta.com/`) which provides apartment rental details and transportation details respectively for different countries in the world. From there, Bill figured there would be links with anchor labels 'International Rental', 'Singapore', 'Two bedroom', in WRA web site and 'International Transport', 'Singapore', 'Raffles Place', 'Bus' in WTA web site that might be used. For instance, in the WRA web site, the 'International Rental' link would point to a list of countries in the world. From this list the link 'Singapore' point to a web page containing list of residential areas in Singapore. From the hyperlinks associated with each area, he can probe further to find the list of one bedroom and two bedroom apartments with monthly rent between $600—$1,000 and $1,000—$1,200 respectively. With these web sites, Bill constructed the query graphs as shown in Figures 1 and 2.

When Bill's query is evaluated, he receives a set of web tuples each satisfying the query graphs. The result of each query is stored in the web tables `Accommodation` and `Transport` respectively.

The web schema of the web tables `Accommodation` and `Transport` is shown below in Examples 1 and 2 respectively. Note that the variables in the figures denote arbitrary instances of Node and Link. Observe that some of the nodes and links have keywords imposed on them. These keywords express the content of the web document or the label of the hyperlink between the web documents. Two special symbols `#` and `-` are used for those node and link variables which are not *bound* by the predicates of the schema.

Example 1. Find the list of all one bedroom and two bedroom apartments in Singapore whose monthly rent is between $600—$1,000 and $1,000—$1,200 respectively, starting from the home page of WRA.

Let the schema of the above query be $M_i = \langle X_{i,n}, X_{i,\ell}, C_i, P_i \rangle$ where $X_{i,n} = \{a, b, c, d, e, f, i, j\}$, $X_{i,\ell} = \{l, m, p, q, u, k\text{-}\}$, $C_i \equiv k_{i_1} \wedge k_{i_2} \wedge k_{i_3} \wedge k_{i_4} \wedge k_{i_5} \wedge k_{i_6} \wedge k_{i_7}$ such that $k_{i_1} = a\langle \ell \rangle b$, $k_{i_2} = b\langle m \rangle c$, $k_{i_3} = c\langle\text{-}\rangle d$, $k_{i_4} = d\langle p \rangle e$, $k_{i_5} = e\langle q \rangle f$, $k_{i_6} = d\langle u \rangle i$, $k_{i_7} = i\langle k \rangle j$ and $P_i \equiv p_{i_1} \wedge p_{i_2} \wedge p_{i_3} \wedge p_{i_4} \wedge p_{i_5} \wedge p_{i_6} \wedge p_{i_7} \wedge p_{i_8} \wedge p_{i_9} \wedge p_{i_{10}} \wedge p_{i_{11}} \wedge p_{i_{12}} \wedge p_{i_{13}} \wedge p_{i_{14}}$ such that $p_{i_1}(a) \equiv [a.$`url EQUALS "http://www.wra.com/"`$]$, $p_{i_2}(b) \equiv [b.$`title EQUALS "Country List"`$]$, $p_{i_3}(\ell) \equiv [\ell.$`label EQUALS "Internat-`

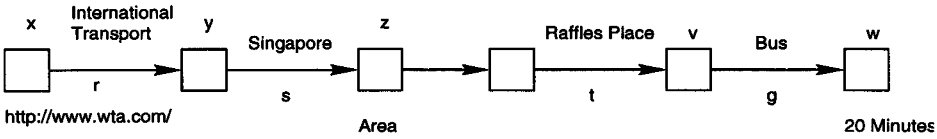

Fig. 2. Web schema(query graph) of Transport web table

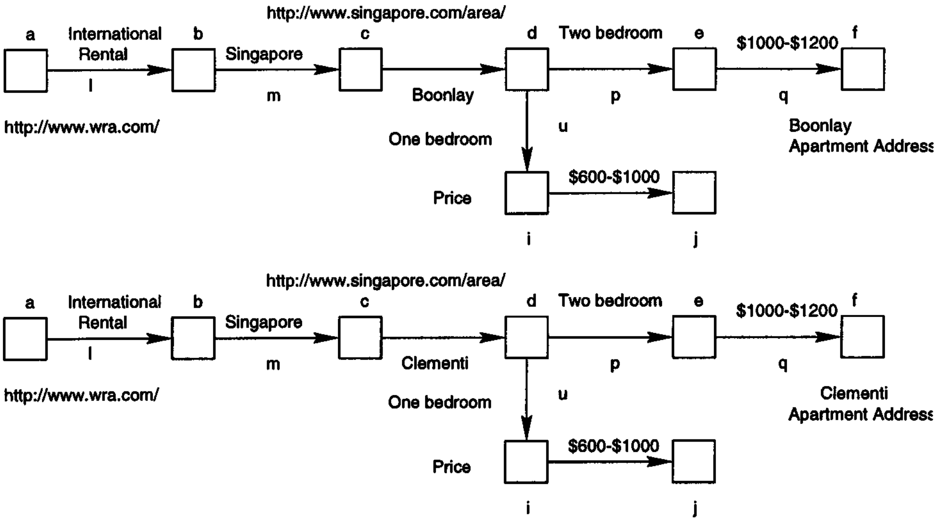

Fig. 3. Accommodation web table

-ional Rental"], $p_{i_4}(c) \equiv$ [c.url EQUALS "http://www.singapore.com/area/"], $p_{i_5}(m) \equiv$ [m.label EQUALS "Singapore"], $p_{i_6}(d) \equiv$ [d.title CONTAINS "Apartment - Type"], $p_{i_7}(e) \equiv$ [e.title CONTAINS "Price"], $p_{i_8}(p) \equiv$ [p.label CONTAINS "Two bedroom"], $p_{i_9}(q) \equiv$ [q.label CONTAINS "$1000 - $1200"], $p_{i_{10}}(f) \equiv$ [f.text CONTAINS "Apartment Address"], $p_{i_{11}}(i) \equiv$ [i.title CONTAINS "Price"], $p_{i_{12}}(k) \equiv$ [k.label CONTAINS "$600 - $1000"], $p_{i_{13}}(j) \equiv$ [j.text CONTAINS "Apartment Address"]., $p_{i_{14}}(u) \equiv$ [u.label CONTAINS "One bedroom"]. ▮

Example 2. Find the list of all areas in Singapore from where travel time to Raffles Place is around 20 minutes by bus starting from the home page of WTA.

Let the schema of the above query be $M_j = \langle X_{j,n}, X_{j,\ell}, C_j, P_j \rangle$ where $X_{j,n} = \{x, y, z, v, w, \#\}$, $X_{j,\ell} = \{r, s, t, g, -\}$, $C_j \equiv k_{j_1} \wedge k_{j_2} \wedge k_{j_3} \wedge k_{j_4} \wedge k_{j_5}$ such that $k_{j_1} = x\langle r \rangle y$, $k_{j_2} = y\langle s \rangle z$, $k_{j_3} = z\langle - \rangle \#$, $k_{j_4} = \#\langle t \rangle v$, $k_{j_5} = v\langle g \rangle w$ and $P_j \equiv p_{j_1} \wedge p_{j_2} \wedge p_{j_3} \wedge p_{j_4} \wedge p_{j_5} \wedge p_{j_6} \wedge p_{j_7} \wedge p_{j_8} \wedge p_{j_9}$ such that $p_{j_1}(x) \equiv$ [x.url EQUALS "http://www.wta.com/"], $p_{j_2}(y) \equiv$ [y.title EQUALS "Country List"], $p_{j_3}(r) \equiv$ [r.label EQUALS "International Transport"], $p_{j_4}(z) \equiv$ [z.url EQUALS "http://www.singapore.com /area/"], $p_{j_5}(s) \equiv$

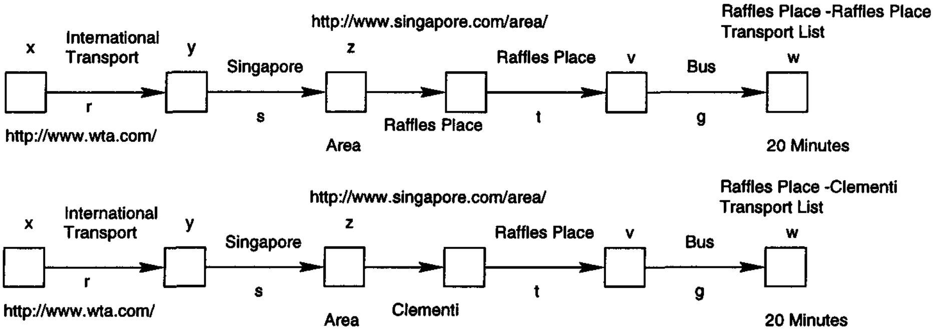

Fig. 4. Transport web table

[s.label EQUALS "Singapore"], $p_{j_6}(v) \equiv$ [v.title CONTAINS "Transport"], $p_{j_7}(w) \equiv$ [w.text CONTAINS "20 Minutes"], $p_{j_8}(g) \equiv$ [g.label CONTAINS "Bus"], $p_{j_9}(t) \equiv$ [t.label CONTAINS "Raffles Place"]. ∎

Figures 3 and 4 show a small portion of the web tables Accommodation and Transport respectively. Each connected, directed subgraph in the figures represents a web tuple of that web table. Each web tuple matches a portion of the WWW satisfying the conditions described in the schema. For example, the first web tuple in Figure 3 stores information about one bedroom (monthly rent $600—$1,000) and two bedroom apartment list (monthly rent $1,000—$1,200) in Boonlay. Note that each web tuple is structurally identical to the web schema.

In particular, Bill wishes to find out a list of one bedroom (monthly rent $600—$1,000) and two bedroom apartments (monthly rent is between 1,000-1,200 dollars), and from where the time taken to travel to Raffles Place by bus is around 20 minutes. In order to get this information Bill compares the Accommodation and Transport tables. For a reasonably small size of the web tables, browsing each web tuples and comparing them with that of other web table for information of interest is a feasible option. But just as browsing relational databases is often an ineffective way to retrieve information, the same holds for browsing web databases having web tables of significant size to gather composite information. Thus, what we need is a operator that allows to gather composite, relevant information from the two web tables Accommodation and Transport. In the next section we introduce the web join operator used to integrate information from two web tables.

4 Concept of Web Join

We now define a web join. We have two input web tables, Accommodation and Transport in the web database which we shall be using in the rest of the paper to illustrate web join. In the next section, we shall discuss the construction of joined schema and the joined web table.

We first introduce some terms we shall be using to explain web join in this paper.

- *Joinable nodes*
  We define joinable nodes as node variables participating in the join.
- *Join connectivity*
  Let $x\langle\rho\rangle y$ be the connectivity of a schema where x and y are node variables as defined in the schema. This connectivity is called a join connectivity if any one of the node variables participate in the web join. That is, $x\langle\rho\rangle y$ is a join connectivity if the join node of the schema is x or y.

The web join operator is used to combine two web tables by *concatenating* a web tuple of one table with a web tuple of other table whenever there exists joinable nodes. Let W_i and W_j be two web tables with schemas $M_i = \langle X_{i,n}, X_{i,\ell}, C_{i,p}, P_i\rangle$ and $M_j = \langle X_{j,n}, X_{j,\ell}, C_j, P_j\rangle$ respectively. Then W_i and W_j are *joinable* if there exist at least one node variable in M_i and in M_j which refers to identical (having the same URL) node or web document. Let us elaborate further. Consider the predicates of the node variables c and z as defined in the web schemas of `Accommodation` and `Transport` respectively in Examples 1 and 2:

$$p_{i_4}(c) \equiv [c.\texttt{url EQUALS "http://www.singapore.com/area/"}],$$
$$p_{j_4}(z) \equiv [z.\texttt{url EQUALS "http://www.singapore.com/area/"}]$$

Since the node variables c and z of M_i and M_j respectively refers to the same web document at url '`http://www.singapore.com/area/`', the web tables `Accommodation` and `Transport` are joinable. The joinable nodes are c and z. We store the joined web tuples in a separate web table. As one of the joinable nodes with identical URLs is superfluous, we remove any one of them in the joined schema. Thus, in the resulting web table, we keep only one joinable node variable (i.e, c or z). Formally, we express natural web join between two web tables as follows:

$$W = W_i \bowtie W_j$$

where W_i and W_j are the two web tables participating in the join and W is the resultant web table created by the join and satisfying schema $M = \langle X_n, X_\ell, C, P\rangle$. Let us illustrate with an example:

Example 3. Consider the `Accommodation` and `Transport` tables as described in Examples 1 and 2. Performing a web join on these web tables creates a joined web table (Figure 6). Due to space limitations, we only show a partial view of the joined web table. The schema of this joined web table is shown below (Figure 5). The construction details of the joined schema and the joined web table is explained in [9]. Let the joined schema be $M = \langle X_n, X_\ell, C, P\rangle$ where $X_n = \{a, b, c, d, e, f, i, j, x, y, v, w, \#\}$, $X_\ell = \{l, m, p, q, u, k, r, s, t, g, \text{-}\}$, $C \equiv k_1 \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5 \wedge k_6 \wedge k_7 \wedge k_8 \wedge k_9 \wedge k_{10} \wedge k_{11} \wedge k_{12}$ such that $k_1 = a\langle\ell\rangle b$, $k_2 = b\langle m\rangle c$, $k_3 = c\langle\text{-}\rangle d$, $k_4 = d\langle p\rangle e$, $k_5 = e\langle q\rangle f$, $k_6 = d\langle u\rangle i$, $k_7 = i\langle k\rangle j$, $k_8 = x\langle r\rangle y$, $k_9 = y\langle s\rangle c$, $k_{10} = c\langle\text{-}\rangle\#$, $k_{11} = \#\langle t\rangle v$, $k_{12} = v\langle g\rangle w$ and $P \equiv p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5 \wedge p_6 \wedge p_7 \wedge p_8 \wedge p_9 \wedge p_{10} \wedge p_{11} \wedge p_{12} \wedge p_{13} \wedge p_{14} \wedge p_{15} \wedge p_{16} \wedge p_{17} \wedge p_{18} \wedge p_{19} \wedge$

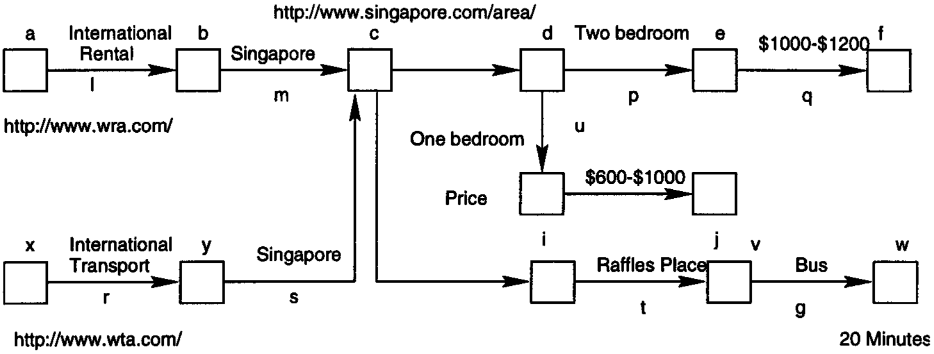

Fig. 5. Schema of the joined web table

$p_{20} \wedge p_{21} \wedge p_{22}$ such that $p_1(a) \equiv$ [a.url EQUALS "http://www.wra.com/"], $p_2(b) \equiv$ [b.title EQUALS "Country List"], $p_3(\ell) \equiv$ [ℓ.label EQUALS "International Rental"], $p_4(c) \equiv$ [c.url EQUALS "http://www.singapore.com/area/"], $p_5(m) \equiv$ [m.label EQUALS "Singapore"], $p_6(d) \equiv$ [d.title CONTAINS "Apartment Type"], $p_7(e) \equiv$ [e.title CONTAINS "Price"], $p_8(p) \equiv$ [p.label CONTAINS "Two bedroom"], $p_9(q) \equiv$ [q.label CONTAINS "$1000 - $1200"], $p_{10}(f) \equiv$ [f.text CONTAINS "Apartment Add ress"], $p_{11}(x) \equiv$ [x.url EQUALS "http:// www.wta.com/"], $p_{12}(y) \equiv$ [y.title EQUALS "Country List"], $p_{13}(r) \equiv$ [r.label EQUALS "International Transport"], $p_{14}(s) \equiv$ [s.label EQUALS "Singapore"], $p_{15}(v) \equiv$ [v.title CONTAINS "Transport"], $p_{16}(w) \equiv$ [w.text CONTAINS "20 Minutes"], $p_{17}(g) \equiv$ [g.label CONTAINS "Bus"], $p_{18}(t) \equiv$ [t.label CONTAINS "Raffles Place"], $p_{19}(i) \equiv$ [i.title CONTAINS "Price"], $p_{20}(k) \equiv$ [k.label CONTAINS "$600 - $1000"], $p_{21}(j) \equiv$ [j.text CONTAINS "Apartment Address"]., $p_{22}(u) \equiv$ [u.label CONTAINS "One bedroom"]. ∎

5 Join Existence

Given two web tables, first we determine if these two web tables are joinable. Two web tables are joinable if there exist at least one web tuple in each web table which has a joinable node. We can identify these set of joinable node(s) from the schemas of the two web tables if the predicates in the two input schemas satisfy some *joinability conditions*. In this section, we determine what these conditions are. Given two web tables, we first inspect their schemas to find out whether they satisfy any of these conditions. If the schemas satisfy these conditions then a web join is possible between the two input web tables and the output of the join existence process is a set of joinable node variable(s). However, if we cannot identify a set of joinable node variable(s) from the schemas then we cannot perform web join.

Let us define some terms to facilitate our exposition. Given a web table W and its schema $M = \langle X_n, X_\ell, C, P \rangle$, let p be a predicate of P such that:

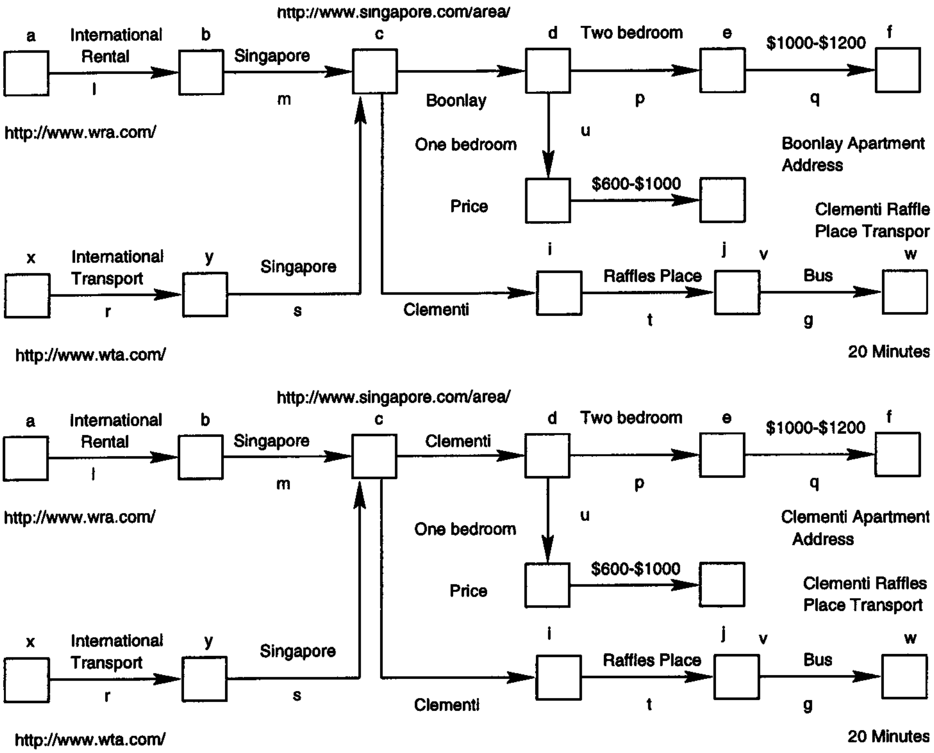

Fig. 6. Web join

- $\arg(p) \in X_n \cup X_\ell$ is the argument of the predicate;
- $\mathrm{attr}(p) \in \{\texttt{url}, \texttt{text}, \texttt{title}, \texttt{format}, \texttt{date}, \texttt{size}\} \cup \{\texttt{source\_url}, \texttt{target\_url}, \texttt{label}, \texttt{link\_type}\}$ is an attribute;
- $\mathrm{op}(p) \in \{\texttt{EQUALS}, \texttt{CONTAINS}\}$ is the operator in the predicate; and
- $\mathrm{val}(p)$ is the operand of the $\mathrm{op}(p)$.

For example, for the predicate $p(x) \equiv [x.\texttt{url EQUALS "http://www.wta.com/"}]$, we have $\arg(p) = x$, $\mathrm{attr}(p) = \texttt{url}$, $\mathrm{op}(p) = \texttt{EQUALS}$ and $\mathrm{val}(p) = \texttt{http : //www.wta.com/}$. Note that $\arg(p)$ represents a node or link of the schema. Hereafter, we use $\arg(p)$ to denote a node or link.

We now highlight the joinability conditions. The joinability conditions may be based on node predicates and/or link predicates of the input web schemas. The conditions for a web join based on node predicates is:

Condition 5.1 Two web tables are joinable if there exist at least a pair of predicates p_i and p_j belonging to schemas M_i and M_j respectively such that the node represented by the predicates have identical URLs. That is, $\mathrm{attr}(p_i) = \mathrm{attr}(p_j) = \texttt{url}$, $\mathrm{op}(p_i) = \mathrm{op}(p_j) = \texttt{EQUALS}$, and $\mathrm{val}(p_i) = \mathrm{val}(p_j)$. The joinable nodes are $\arg(p_i)$ and $\arg(p_j)$.

The conditions based on only link predicates are as follows:

Let c be a connectivity of the form $x\langle\rho\rangle y$. We refer to node variables x, y as lnode(c) and rnode(c) respectively, and use connlink(c) to refer to the set of link variables appearing in ρ. Thus, lnode(c) $\in X_n$, rnode(c) $\in X_n$ and connlink(c) $\subseteq X_\ell$. Clearly, c describes a path or a set of possible paths between two nodes lnode(c) and rnode(c). For instance, the expression $x\langle(ef|g)\rangle y$ says that there exist either a simple link $g \in X_\ell$, or two links $e \in X_\ell$ followed by $f \in X_\ell$, between lnode(c) and rnode(c).

Let c_i and c_j be two connectivities in schemas M_i and M_j respectively. Then the web tables corresponding to M_i and M_j are joinable if one or more of the following conditions hold:

Condition 5.2 The `source_url` of the leftmost link variables e and f of c_i and c_j respectively are identical, i.e., arg(p_i) $= e$, arg(p_j) $= f$, attr(p_i) = attr(p_j) = `source_url`, op(p_i) = op(p_j) = `EQUALS`, val(p_i) = val(p_j). Thus, lnode(c_i) and lnode(c_j) are joinable.

Condition 5.3 The `target_url` of the rightmost link variables g and h of c_i and c_j respectively are identical, i.e., arg(p_i) $= g$, arg(p_j) $= h$, attr(p_i) = attr(p_j) = `target_url`, op(p_i) = op(p_j) = `EQUALS`, val(p_i) = val(p_j). Thus, rnode(c_i) and rnode(c_j) are joinable. ∎

Condition 5.4 The `source_url` of the leftmost link variables e and the `target_url` of the rightmost link variable g of c_i and c_j respectively are identical, i.e., arg(p_i) $= e$, arg(p_j) $= g$, attr(p_i) = `source_url`, attr(p_j) = `target_url`, op(p_i) = op(p_j) = `EQUALS` and val(p_i) = val(p_j). Thus, lnode(c_i) and rnode(c_j) are joinable. ∎

The conditions based on node and link predicates are as follows:

Let p_i be a node predicate and p_j a link predicate in schemas M_i and M_j respectively. Then the web tables corresponding to M_i and M_j are joinable if any one of the following conditions hold:

Condition 5.5 The `url` of the node variable x and the `target_url` of the link variable g of p_i and p_j respectively are identical. Formally, arg(p_i) $= x$, arg(p_j) $= g$, attr(p_i) = `url`, attr(p_j) = `target_url`, op(p_i) = op(p_j) = `EQUALS` and val(p_i) = val(p_j). The joinable nodes are x and rnode(c_j). ∎

Condition 5.6 The `url` of the node variable x and the `source_url` of the link variable f of p_i and p_j respectively are identical. Formally, arg(p_i) $= x$, arg(p_j) $= f$, attr(p_i) = `url`, attr(p_j) = `source_url`, op(p_i) = op(p_j) = `EQUALS` and val(p_i) = val(p_j). The joinable nodes are x and lnode(c_j). ∎

6 Summary and Future Work

In this paper, we have motivated the need for a join operation for the web data model and we have introduced the notion of web join that enable us to combine web tuples from two web tables whenever joinable nodes exist. Please refer to [9] for detailed version of this paper.

Presently, we have implemented the first version of the web join operator and have interfaced it with other web operators. To explore different possible join conditions, we have created a synthetic database and have implemented the join operation for all joinability conditions using the synthetic database. The current web join operator can be used efficiently for simple web queries. Currently, we are in the process of extending the prototype of web join operator to support more complex web queries.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Weiner. The Lorel Query Language for Semistructured Data. *Journal of Digital Libraries*, 1(1):68-88, April 1997.
2. P. Buneman, S. Davidson, G. Hillebrand, D. Suciu. A query language and optimization techniques for unstructured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Canada, June 1996.
3. T. Fiebig, J. Weiss, G. Moerkotte. RAW: A Relational Algebra for the Web. *Workshop on Management of Semistructured Data (PODS/SIGMOD'97)*, Tucson, Arizona, May 16, 1997.
4. D. Konopnicki, O. Shmueli. W3QS: A Query System for the World Wide Web. *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995.
5. L.V.S. Lakshmanan, F. Sadri., I.N. Subramanian A Declarative Language for Querying and Restructuring the Web *Proceedings of the Sixth International Workshop on Research Issues in Data Engineering*, February, 1996.
6. A. O. Mendelzon, G. A. Mihaila, T. Milo. Querying the World Wide Web. *Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida,
7. W. K. Ng, E.-P. Lim, C. T. Huang, S. Bhowmick, F. Q. Qin. Web Warehousing: An Algebra for Web Information. *Proceedings of IEEE International Conference on Advances in Digital Libraries (ADL'98)*, Santa Barbara, California, April 22–24, 1998.
8. S. Bhowmick, W. K. Ng, E.-P. Lim. Web Information Coupling in Web Databases. *Submitted for publication*
9. S. Bhowmick, W. K. Ng, E.-P. Lim. Join Processing of Web Databases. *Technical Report - CAIS-TR-98-12, Centre for Advanced Information Systems, Nanyang Technological University, Singapore.*1998.