01 Jan 2000

Reducing Cognitive Overheads in a Web Warehouse using Reverse-Osmosis

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Wee Keong Ng

Ee-Peng Lim

Sourav S. Bhowmick

Recommended Citation

S. K. Madria et al., "Reducing Cognitive Overheads in a Web Warehouse using Reverse-Osmosis," *Proceedings of the 7th International Conference on Parallel and Distributed Systems, 2000*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2000.
The definitive version is available at https://doi.org/10.1109/ICPADS.2000.857696

Reducing Cognitive Overheads in a Web Warehouse using Reverse-Osmosis

SOURAV S BHOWMICK[1] SANJAY MADRIA[2] WEE KEONG NG[1] EE-PENG LIM[1]

Centre for Advanced Information Systems[1], School of Applied Science, Nanyang Technological University, Singapore 639798
{p517026,awkng,aseplim}@ntu.edu.sg

Department of Computer Science[2], Purdue University, West Lafayette, IN 47907
skm@cs.purdue.edu

Abstract

This paper provides a quantitative analysis of reducing cognitive overheads in a web warehouse *using an important class of operation called* reverse osmosis. *The analysis is used to examine two different cognitive overheads of locating relevant* nodes *or information and display time of a* web table . *A reverse-osmosis operation enables us to eliminate irrelevant information from a collection of web documents stored in the form of a web table. We call such operation reverse-osmosis because it is analogous to the* reverse osmosis process *in the field of water purification. We discuss formal algorithm of reverse-osmosis operation.*

1 Introduction

With an enormous amount of data stored in the World Wide Web, it is increasingly important to design and develop powerful web warehousing tools for querying and analysis of such data. The key objective of our web warehousing project, called WHOWEDA (*WareHouse of WEb DAta*) [3] is to design and implement a web warehouse that materializes and manages useful information from the Web. Informally, our web warehouse can be conceived of as a collection of *web tables*. A set of *web tuples* is called a web table. A web tuple is a directed graph consisting of a set of *nodes* and *links* and satisfies a *web schema*. Nodes and links contain content and metadata information associated with web documents and hyperlinks among the web documents. The web schema contains meta-information that binds a set of web tuples in a web table. To facilitate manipulation of Web data stored in web tables, we have defined a set of web algebraic operators (i.e., *global web coupling*, *web join*, *web select* etc.) [4, 11]. These web operators enable us to build new web tables by extracting relevant data from the Web and to generate new web tables from existing ones.

In the Web context, although there is an increasing research effort on querying the Web, on building and maintaining Web sites and on materializing *views* of the Web [9], to the best of our knowledge very little work has been carried out on minimizing cognitive overheads generated by Web data. In this paper, we introduce an important class of operation in a web warehouse that we call as *reverse-osmosis* to reduce cognitive overheads associated with warehouse data. A reverse-osmosis operation enables us to eliminate irrelevant information from a collection of web documents stored in the form of a web table. We call this operation reverse-osmosis because it is analogous to the reverse osmosis process in water purification. *Osmosis* [1] is the natural tendency for water to spontaneously pass through a semi-permeable membrane separating two solutions of different concentrations (strengths). The water will naturally pass from the weaker (less concentrated) solution containing fewer particles of dissolved substance to the stronger (more concentrated) solution containing more particles of a dissolved substance. On the other hand, reverse osmosis [1] is the process by which water or other medium tends to flow across a membrane in the direction opposite to that for natural osmosis when subjected to a *hydrostatic pressure* greater than the *osmotic pressure*. The process of reverse osmosis forces water with a greater concentration of contaminants (the source water) into a tank containing water with extreme low concentration of contaminants (the processed water). High water pressure on the source side is used to "reverse" the natural osmotic process, with the semi-permeable membrane still permitting the passage of water while rejecting most of the other contaminants. In WHOWEDA reverse-osmosis is used to "purify contaminated" web tables. That is, to eliminate irrelevant information from a web table. In our web warehouse, reverse-osmosis is triggered off by the *web project* operator. A query involving web project operation is called a reverse-osmosis query. Given a web table W, a web project operator eliminates a portions of a web tuple in a web table satisfying certain *project conditions*. The web project operation reduces the number of nodes and links (some of them may be irrelevant) in the original web table. If we assume that the input web table is a "contaminated" web table with the existence of irrelevant nodes (contaminants), then similar to the

0-7695-0568-6/00 $10.00 © 2000 IEEE

concept that a reverse osmosis process is triggered off when the hydrostatic pressure is greater than the osmotic pressure, a reverse-osmosis operation in WHOWEDA is initiated by a query involving web project operation. The project conditions are analogous to the semi-permeable membrane that filters out contaminants in the reverse osmosis operation. The output web table (*RO-web table*) is analogous to the processed water (free from contaminants).

In this paper, we develop an algorithm to create web table generated as a result of reverse-osmosis operation. Furthermore, we analyze quantitatively the following issues: (1) Reduction in cognitive overhead that results from decreasing the number of irrelevant nodes by reverse-osmosis operation. (2) Decrease in cognitive overhead from the decrease in time to generate the display of the web table due to the elimination of "contaminated" nodes.

2 Background

In this section, we begin by briefly discussing the global web coupling operator which we use to extract related information from the Web. Next, we discuss the web operators which we will be using in the subsequent sections to explain reverse-osmosis operations (i.e., web join [5], local web coupling [4]). These operators provide the flexibility of combining two or more web tables based on *different conditions*. Note that, in this paper we discuss these operators only to the extent it is necessary to understand the concept of reverse-osmosis operation.

2.1 Global Web Coupling

Global web coupling [1] retrieves a set of inter-linked documents satisfying a user's query. It is the first step in populating WHOWEDA. To initiate global coupling, the user specifies a web query in the form of a *coupling query*. A web tuple matches a portion of the WWW and satisfies the conditions described in the coupling query.

A coupling query is a 5-tuple $G = \langle X_n, X_\ell, C, P, Q\rangle$ where X_n is a set of *node variables*, X_ℓ is a set of *link variables*, C is a set of *connectivities*, P is a set of *predicates* over the node and link variables and Q is a set of *predicates* on the complete coupling query. A node or link variable denotes a set of documents or hyperlinks respectively satisfying similar characteristics defined by the predicates specified on these variables. These variables are either *bound* or *free*. Bound node or link variables have conditions imposed on them in the form of predicates. On the other hand, a *free* variable do not have any predicate defined over it. The connectivities between the node variables express hyperlink structure. The last component Q of the coupling query imposes additional constraints over the coupling query in the form of predicates. We illustrate global web coupling with the following example.

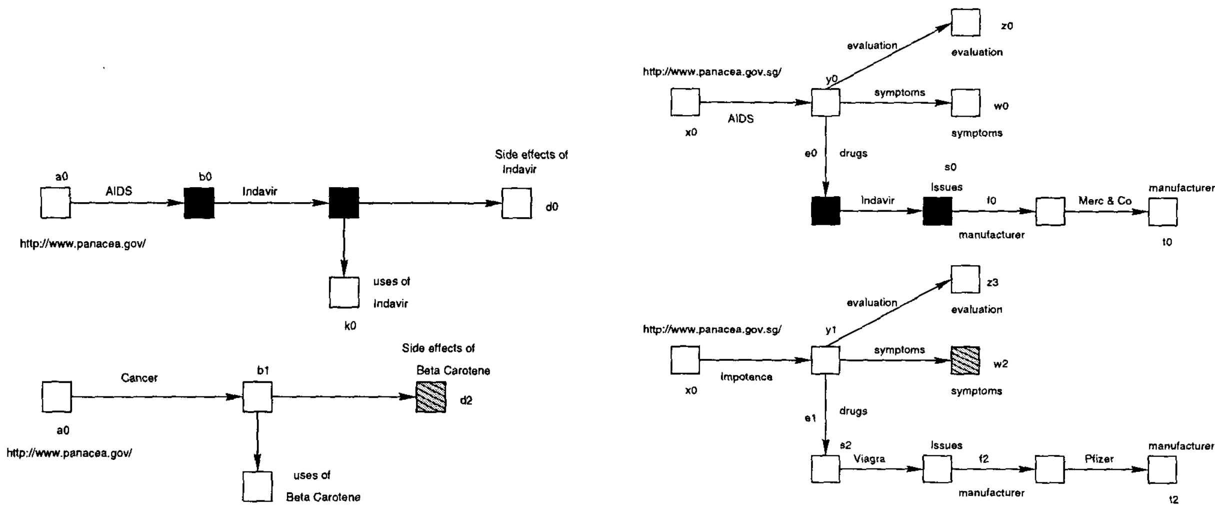
Example 1 Assume that the web sites at www.panacea.gov and www.panacea.gov.sg are geographically-separated web sites of *Panacea Inc.* providing information on various issues related to diseases and drugs. In particular, the web site at www.panacea.gov provides drugs related information (i.e., usage, dosage, manufacturer, description, side-effects etc.) for various diseases. The web site at www.panacea.gov.sg supplies disease related information (i.e., symptoms, evaluation strategies, drugs for diseases etc.) Suppose a user wishes to find the following information: (1) A list of drugs for various diseases, their side effects and uses starting from the web site at www.panacea.gov. (2) A list of diseases and their symptoms, evaluation strategies, drugs used for these diseases and manufacturer details of these drugs.

These queries are specified as two coupling queries as given below. These queries are evaluated by the global web coupling operator and a set of results in the form of web tuples satisfying each coupling query is stored in web tables Drugs and Diseases respectively (Figure 1). The *web schemas* of Drugs and Diseases are then created from the coupling queries [7]. The formal representation of the queries are given below:

Let the query to create Drugs be $G_i = \langle X_{n_i}, X_{\ell_i}, C_i, P_i, Q_i\rangle$ where $X_{n_i} = \{a, b, k, d\}$, $X_{\ell_i} = \{-\}$, $C_i \equiv k_1 \wedge k_2 \wedge k_3$ such that $k_1 = a\langle -\rangle b$, $k_2 = b\langle -\{1,5\}\rangle d$, $k_3 = b\langle -\{1,6\}\rangle k$ and $P_i = \{p_1, p_2, p_3, p_4\}$ such that $p_1(a) \equiv$ [a.url EQUALS "http://www.panacea.gov/"], $p_2(b) \equiv$ [b.title CONTAINS "Drug List"], $p_3(k) \equiv$ [k.title CONTAINS "uses"], $p_4(d) \equiv$ [d.title CONTAINS "side effects"] and $Q_i = \emptyset$. Observe that the expression '{1, 5}' specifies that at least one and at the most 5 successive hyperlinks must be traversed from an instance of b to reach an instance of d. Similarly, the connectivity k_3 specifies that at least 1 and at the most 6 successive hyperlinks connect an instance of b with an instance of k. Also note that the symbols '#' and '-' denotes a free node and link variables respectively.

Let the query to create Diseases be $G_j = \langle X_{n_j}, X_{\ell_j}, C_j, P_j, Q_j\rangle$ where $X_{n_j} = \{x, y, z, w, s, t, \#\}$, $X_{\ell_j} = \{e, f, -\}$, $C_j \equiv k_4 \wedge k_5 \wedge k_6 \wedge k_7 \wedge k_8 \wedge k_9$ such that $k_4 = x\langle -\rangle y$, $k_5 = y\langle -\{1,4\}\rangle z$, $k_6 = y\langle -\{1,4\}\rangle w$, $k_7 = y\langle e\rangle \#$, $k_8 = \#\langle -\rangle s$, $k_9 = s\langle f\text{-}\{1,6\}\rangle t$ and $P_j = \{p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}\}$ such that $p_4(x) \equiv$ [x.url EQUALS "http://www.panacea.gov.sg/"], $p_5(y) \equiv$ [y.title CONTAINS "Issues"], $p_6(z) \equiv$ [z.title CONTAINS "evaluation"], $p_7(w) \equiv$ [w.title CONTAINS "symptoms"],

[1]In an earlier paper [11], the operator to retrieve a set of inter-linked web documents was called *web derive*. Later, we switched the terminology to *global web coupling* in [4] as we believe that this is essentially a mechanism to *couple* related information from different portion of the Web.



(a) Partial View of web table "Drugs". (b) Partial view of web table "Diseases"

Figure 1. Web tables.

$p_8(s) \equiv$ [s.title CONTAINS "issues"], $p_9(t) \equiv$ [t.title CONTAINS "manufacturer"], $p_{10}(e) \equiv$ [e.label EQUALS "drugs"], $p_{11}(f) \equiv$ [f.label EQUALS "manufacturers"]

Note that each web tuple in Drugs and Diseases contains information about a particular drug and disease respectively. Observe that global web coupling not only couples related documents, it also retrieves the topological structure of the inter-related Web documents. ∎

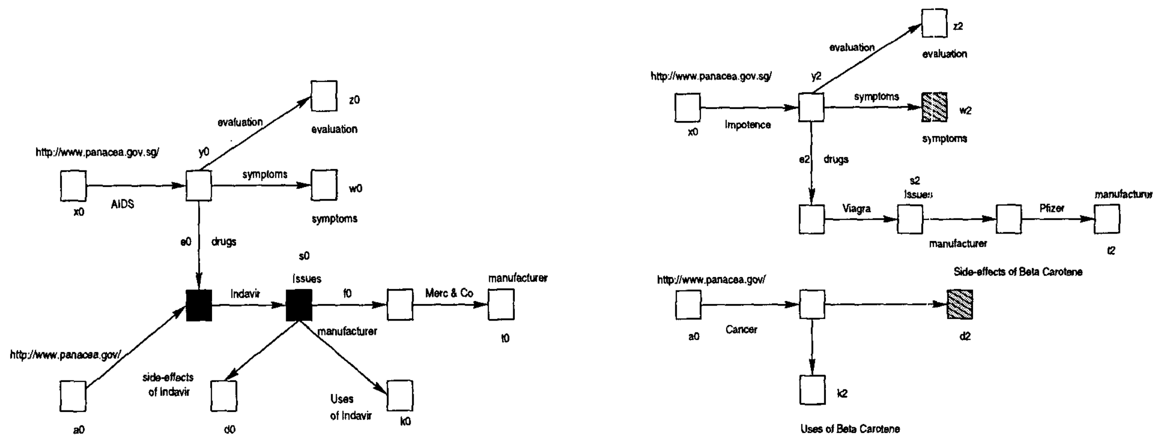
2.2 Local Web Coupling

Local web coupling operator [4] is used to combine two web tables by *integrating* web tuples of one table with the tuples of the other web table whenever there exists *content-related nodes*. Local web coupling or local coupling (for brevity) enables us to correlate Web documents containing similar information (but not necessarily identical) across multiple Web sites. Given two web tables, local coupling is initiated explicitly by specifying a pair(s) of node variables (*coupling node variables*) and a set of keyword(s) to relate them. The result of local web coupling is a web table consisting of a set of collections of inter-related Web documents from the two input tables.

We elaborate local web coupling with an example. Consider the web tables Drugs and Diseases as depicted in Figures 1(a) and 1(b). Suppose we want correlate those web tuples in Drugs and Diseases in which side-effects of drugs and symptoms of diseases contains "nausea". This can be achieved with the help of local web coupling operation where we can couple the web tuples together by specifying keyword constraint "nausea" on the node variables w and d (called coupling node variables), in Diseases and Drugs respectively. Observe that the instances of w and d contains information about symptoms and side-effects of various diseases and drugs respectively. Figure 2(b) depicts a web tuple in the web table generated from local web coupling operation. The patterned nodes are the instances of coupling node variables over which the local web coupling is performed.

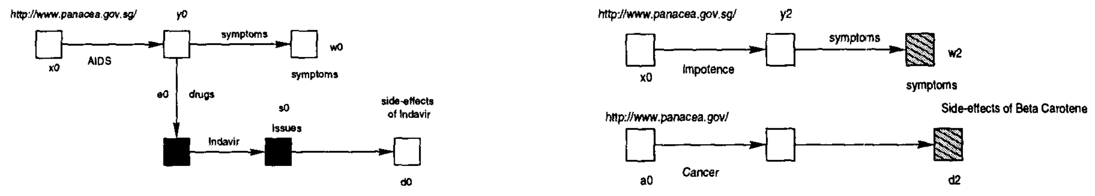
2.3 Web Join

The web join operator [5] is used to combine two web tables by *joining* a web tuple of one table with a web tuple of other table whenever there exists *joinable* nodes. We define *joinable* nodes as nodes participating in web join. In WHOWEDA, if two nodes are *identical* then they are joinable. We consider two nodes or Web documents identical when they have the same URL and last modification date. Let $w_a \in W_1$ and $w_b \in W_2$ be two web tuples. Then these tuples are joinable if there exist at-least one node in w_a which is identical (joinable) to a node in w_b. The joined web tuple contains the nodes from both the input web tuples. We materialize the joined web tuple in a separate web table. As one of the joinable node in each joinable node pair is superfluous, we remove one of them from the joined web tuple. For example, consider the first web tuples of Drugs and Diseases in Figures 1(a) and 1(b). Since the nodes represented by black boxes are joinable nodes (we assume they are identical and belongs to the web site at `www.panacea.gov.sg`), the tuples are joinable. The joined web tuple is shown in Figure 2(a). Observe that we remove one of the nodes from each pair of joinable nodes



(a) A joined web tuple. (b) A locally coupled web tuple.

Figure 2. Web join and local web coupling.



(a) (b)

Figure 3. Partial view of projected web tables.

in the joined web tuple.

3 Reverse-Osmosis Operation

In this section, we first discuss reverse-osmosis operation. Within this context we introduce the web project operator which is used in the reverse-osmosis operation. Then, we discuss the algorithm for generating *RO-web table.*

3.1 Reverse-Osmosis and Web Project

Naively, it seems that web join and local web coupling are useful mechanisms to combine related information from two web tables. However, these mechanisms are apt to have its virtues along one dimension compensated by deficiencies along another. On the one hand, the integrated web tuples are more *informative* and complete. By informative we mean that all the nodes in the two web tuples and their inter-link structure are preserved after the web join or local web coupling operation. This is useful when the user has doubts or lacks understanding of how the inter-linked web pages in two different web tables are related to each other. On the other hand, the more informative the joined or locally coupled web tuple, the higher are the cognitive overheads associated with these tuples. That is, longer it takes to be displayed and the more time (search time) is required to find relevant nodes. For instance, finding symptoms of diseases and side-effects of drugs can be frustrating from a pool of large number of nodes in Figure 2. Thus, a mechanism is required to minimize the cognitive overheads such as the search time for relevant nodes and display time of the web table.

The reverse-osmosis operation is used to eliminate these irrelevant data from the web table. It extracts portions of a web tuple in a web table satisfying certain conditions. Reverse-osmosis is triggered off by the *web project* operator. The web project operator reduces the number of nodes and links in a web table based on the *project conditions* by eliminating these from the web table. Recall that the project conditions are analogous to the semi-permeable membrane in the reverse osmosis operation in the context of water purification which filters out contaminants.

Given a web table W with schema $S = \langle X_n, X_\ell, C, P \rangle$, a web project on W computes a new web table W_p or a *web bag* W_b with schema $S_p = \langle X_{n_p}, X_{\ell_p}, C_p, P_p \rangle$. Observe that the output web table W_p (*RO-web table*) is analogous to the processed water (free from contaminants) in reverse osmosis. The components of S_p depends on the project condition(s). Note that, unlike relational project, the web project operation does not remove duplicate web tuples automatically. The projected collection of web tuples may contain identical web tuples. In this case, it is called a web bag [2]. Furthermore in web project, we specify the node variables to be eliminated in the project conditions, as opposed to relational project, where we specify the attributes to be projected from a relation.

Formally, we define web project as $W_b = \pi_{\langle project\_condition(s)\rangle}(W)$ where π is the web project operator. The duplicate elimination process is then initiated explicitly by the user and is performed by the following operation: W_p = **Distinct**(W_b) where W_b is a web bag and W_p is the projected web table with distinct web tuples. Note that if the projected web table contains distinct web tuples then $W_p = \pi_{\langle project\_condition(s)\rangle}(W)$. A user may explicitly specify any one of the conditions or any combination of the following three conditions to initiate a web project operation: (1) A user may specify a set of node variables which he may wish to eliminate from the web table. (2) A user may wish to eliminate all the instances of node variables between two node variables. (3) A user may restrict the set of nodes to be eliminated within a limited number of links starting from the specified node variable.

Example 2 Suppose a user wishes to extract information related to symptoms of various diseases and side effects of drugs used for these diseases. Clearly, these information are already stored in tables Diseases and Drugs. The web join operator enables us to compute these related information from the two web tables (Figure 2(a)). However, it also couple irrelevant information i.e., information related to evaluation strategies of diseases (instances of node variable z), manufacturers and uses of various drugs (instances of t and k). The reverse-osmosis operation enables us to filter out this irrelevant information. For example, we may specify the project condition $P_c = \{z, t, s, a, d, (s, t)\}$ on the joined web table and initiate the reverse-osmosis operation using the web project operator. A projected web tuple is shown in Figure 3(a). Similarly, we can eliminate irrelevant nodes from the locally coupled web tuple using reverse-osmosis. A projected web tuple is shown in Figure 3(b). ∎

3.2 Algorithm of Reverse-Osmosis

In this section, we describe the algorithm for creating a "less contaminant" web bag or web table called RO-web table or RO-web bag using reverse-osmosis.

The formal representation of the algorithm of reverse-osmosis operation is shown in Figure 4. It takes as input a "contaminated" web table W and a set of project conditions P_c and creates a "cleaner" web table or web bag W_b called RO-web table or RO-web bag. The algorithm first identifies the nodes and links to be eliminated from the web table based on the project condition P_c and all the instances of all the node and link variables in the web table. The method of identifying these nodes and links are encapsulated in the operations **GetProjectNodeIds**(P_c), **GetProjectLinkIds**(P_c), **GetAllNodeIds**(W) and **GetAllLinkIds**(W) respectively (steps (1) to (4)). The set of identifiers of all distinct nodes and links in the RO-web table is given as $N_{ro} = N - N_p$ and $L_{ro} = L - L_p$ respectively (steps (5) and (6)). Then, the algorithm executes the steps (8) to (17) and removes from each web tuple the nodes and links to be eliminated. Next, steps (18) to (21) retrieves the node objects in the RO-web table from W. Finally, the algorithm determines if the collection of resultant web tuples is a web bag or distinct web tuples (steps (22) to (25)). If identical tuples are located, then the collection of web tuples is a RO-web bag. Otherwise, the set of tuples after web project operation results in a RO-web table with distinct web tuples.

4 Minimizing Cognitive Overheads

In this section, first we discuss the reduction in cognitive overhead of locating relevant nodes in a RO-web tuple compared to a joined or locally coupled web tuple. Then, we compute the difference in display time of a RO-web table compared to a joined or locally coupled web table. Within this context, we examine the parameters which affect the time to locate relevant nodes and display time.

4.1 Locating Relevant Nodes

Recall that a web tuple is a directed graph containing nodes (web documents) and links between the nodes. However, not all nodes in a web tuple may be relevant to the user. The predicates on the link and node variables as defined in the web schema may expedite the process of locating relevant nodes in a web tuple since it may enable us to speculate with reasonable accuracy the contents of nodes and links. Although it is possible to identify relevant nodes efficiently in a web tuple containing only few nodes, in general locating relevant nodes in a web tuple containing large number of nodes may be a frustrating and tedious process. This situation may occur in a concatenated web tuple in a joined or locally coupled web table where there is a high possibility of existence of large number of nodes (since it combines nodes from two web tuples into a single joined or coupled web tuple). Moreover, large number of free nodes and links may exist in a web tuple, making the task of locating relevant nodes significantly harder. Note that, it may be difficult to surmise the contents of an free node or link as they are not defined by predicates in the web schema. Thus, the exis-

```
Input: Project conditions set $P_c$, Web table $W$ with schema $S = \langle X_n, X_\ell, C, P\rangle$.
Output: $RO$-web bag or $RO$-web table $W_b$.

(1)   $N_p$ = GetProjectNodeIds($P_c$);
          /* $N_p$ is a set of identifiers of nodes which are to be eliminated based on the project condition $P_c$ */
(2)   $L_p$ = GetProjectLinkIds($P_c$);
          /* $L_p$ is a set of identifiers of links which are to be eliminated based on the project condition $P_c$ */
(3)   $N$ = GetAllNodeIds($W$);
          /* returns a set of identifiers of the nodes in the web table $W$. */
(4)   $L$ = GetAllLinkIds($W$);
          /* returns a set of identifiers of the links in the web table $W$. */
(5)   $N_{ro} = N - N_p$;
(6)   $L_{ro} = L - L_p$;
(7)   /* Eliminate irrelevant nodes */
(8)   for ($i = 1$ to $|W|$) {
(9)       $T_n(i)$ = GetTupleNodeIds($t_i$);
          /* returns a set of identifiers of the nodes in a web tuple $t \in W$ */
(10)      $T_l(i)$ = GetTupleLinkIds($t_i$);
          /* returns a set of identifiers of the links in a web tuple $t \in W$ */
(11)      $N_p(i) = N_p \cap T_n(i)$;
(12)      $L_p(i) = L_p \cap T_l(i)$;
(13)      $T_n(i) = T_n(i) - N_p(i)$;
(14)      $T_l(i) = T_l(i) - L_p(i)$;
(15)      Store $T_n(i)$ and $T_l(i)$ in web tuple pool of $W_b$;
(16)      Store $N_t(i)$ and $T_l(i)$ in $S_d$;
(17)  }
(18)  /* Store nodes in $RO$-web table file */
(19)  for ($k = 1$ to $|N_{ro}|$) {
(20)      $N_{ro}(k)$ = GetNode($N_w(k), L_w$); /* Retrieves the node for resultant web table from $W$ */
(15)      Store $N_{ro}(k)$ in web table file of $W_b$;
(21)  }
(22)  if ($S_d$ is a multi-set)
(23)      The resultant web table is bag;
(24)  else
(25)  The resultant collection of web tuples is distinct;
(26)  return $W_b$;
```

Figure 4. Algorithm for Reverse-Osmosis Operation.

tence of free nodes and links aggravate the problem of identifying relevant nodes in a joined or locally coupled web tuple. For instance, re-consider the query in Example 2. Finding symptoms of diseases and side-effects of drugs used for these diseases is frustrating from a pool of large number of nodes in Figure 2(a). Similarly, identifying relevant nodes from a locally coupled web tuple (Figure 2(b)) is tiresome. However, it is relatively easier to locate relevant nodes from the projected web tuples in Figure 3.

For a given query, a node in a web tuple may be *relevant* or *not relevant*. For instance, instances of node variables t, s are not relevant to the query in Example 2. Let p_r be the probability that a particular node in a web tuple is relevant for the user's query. The time taken by the user to decide whether a node in a web tuple is relevant for a query is influenced by the type of node (i.e., bound or free) and the probability values assigned to that node. For each node n_i in a web tuple the decision time can be computed using Hick's Law [8]: $I_c\left[(p_{r_i}\log_2(1/p_{r_i}+1)+p_{nr_i}\log_2(1/p_{nr_i}+1)\right]$ where I_c is the *information-theoretic entropy* of the decision and p_{nr} is the probability value of a node being not relevant to the query.

The time taken by a user to make a decision whether a node is relevant based on the meta-data information of the node varies with the set of predicates defined on its node variable (in case the node variable is defined by the schema). For example, in Figure 2(a), the predicate on the node w_0 indicate the information contained in w_0 deals with symptoms of a disease. If it is relevant to the user's query then p_{r_i} for the node w_0 is high. However, in the case of free nodes the decision making time may increase since there are no predicates attached to these nodes. In order to decide if the node is relevant to the query we have to click on the node to inspect the meta-data information associated with the node. If the free nodes are not relevant to the query then it is assigned a very high p_{nr}.

Proposition 1 *The difference in total decision time over all the nodes in the joined or locally coupled web table and the RO-web table (denoted by ΔT_u) is given by*

$$\Delta T_u = \sum_{i=1}^{|W_{in}|} T_{u_{in}}(i) - \sum_{i=1}^{|W_{ro}|} T_{u_{ro}}(i) \qquad (1)$$

where

$$T_{u_{in}}(i) = I_c\left[\sum_{i=1}^{N_{w_{in}}(t_i)} p_{r_i}\left(\log_2\frac{(1/p_{r_i}+1)}{\left(\frac{1}{1-p_{r_i}}+1\right)}\right)\right] + I_c\left[\sum_{i=1}^{N_{w_{in}}(t_i)} \log_2\left(\frac{1}{1-p_{r_i}}+1\right)\right]$$

$$T_{u_{ro}}(i) = I_c\left[\sum_{i=1}^{N_{w_{ro}}(t_i)} p_{r_i}\left(\log_2\frac{(1/p_{r_i}+1)}{\left(\frac{1}{1-p_{r_i}}+1\right)}\right)\right] + I_c\left[\sum_{i=1}^{N_{w_{ro}}(t_i)} \log_2\left(\frac{1}{1-p_{r_i}}+1\right)\right]$$

∎

The proof of the above proposition is given in [6].

Observation 1 The total decision time depends on number of web tuples in the coupled or *RO*-web table and number of nodes in each tuple and p_r of the nodes. Increase in number of identical web tuples in the *RO*-web bag reduces the number of web tuples in the *RO*-web table after duplicate elimination. Thus, the total decision time for *RO*-web table decreases compared to the joined or coupled web table. Moreover, the number of nodes in each web tuple in a joined or locally coupled tuple is larger than that in a *RO*-web tuple. Also, the joined or locally coupled web table may have more free nodes (p_r is less for free nodes) compared to *RO*-web table. ∎

An important element in the measure of cognitive overheads associated with joined or locally coupled web table compared to a *RO*-web table is the comparison between the positioning time (T_p) of the mouse on the nodes of a joined or locally coupled web tuple and *RO* web tuple. This is especially important for a user when he wishes to inspect the meta-data information of the nodes. Note that, in WHOWEDA the meta-data information of the nodes (web documents) of a web tuple can be displayed by clicking on these nodes. There are two scenarios when an user may need to click on the nodes in a web tuple to inspect meta-data information: Existence of free node and link variables and the predicates defined on the bound node variables are not crisp enough to pinpoint the actual content of the node.

The positioning time of the mouse on an object can be measured by using Fitt's Law [8, 12]. According to Fitt's Law, the time T_p to move the hand to a target of size S which lies a distance D away is given by $T_p = I_M \log_2(D/S + 0.5)$ where $I_M = 100[70 \sim 120]$ msec/bit. We use Fitt's Law to compute the positioning time of the mouse on a node in a joined or locally coupled tuple and *RO*-web tuple.

Proposition 2 *Let S be the size of each node in the web table, D_i be the distance moved by the mouse in order to reach a node from a fixed reference. The difference in total positioning time on all the nodes in the joined or locally coupled web table and the RO-web table (denoted by ΔT_p) is given by*

$$\Delta T_p = \sum_{i=1}^{|W_{in}|} T_{p_{in}}(i) - \sum_{i=1}^{|W_{ro}|} T_{p_{ro}}(i) \qquad (2)$$

where

$$T_{p_{in}}(i) = 0.1\sum_{k=1}^{N_{w_{in}}(t_i)} \log_2\left(\frac{2D_k}{S}+0.5\right),$$
$$T_{p_{ro}}(i) = 0.1\sum_{s=1}^{N_{w_{ro}}(t_i)} \log_2\left(\frac{2D_s}{S}+0.5\right)$$

∎

The proof of the above proposition is given in [6].

Observation 2 Note that from Equation 2 we can infer that lesser is the total positioning time for *RO*-web table the higher is ΔT_p. Moreover the total positioning time depends on number of web tuples in the web table and number of nodes in each tuple and how they are oriented with respect to a fixed reference. Note that increase in number of identical web tuples in the *RO*-web bag reduces the number of web tuples in the *RO*-web table after duplicate elimination. Thus, the total positioning time for *RO*-web table decreases compared to the joined or locally coupled web table. Moreover, the number of nodes in each joined or locally coupled web tuple is larger than that in a *RO*-web tuple. Thus, $\left[\sum_{k=1}^{N_{w_{in}}(t_i)} \log_2\left(\frac{2D_k}{S}+0.5\right) - \sum_{s=1}^{N_{w_{ro}}(t_i)} \log_2\left(\frac{2D_s}{S}+0.5\right)\right]$ for each web tuple increases as there is an increase in number of nodes to be eliminated. ∎

4.2 Display Time

We now compute the difference between the display time of a *RO*-web table compared to that of a joined or locally coupled web table. In general, time to display a web tuple (denoted as T_d) is a linear function of size of the web tuple (denoted as S_t): $T_d = \sigma + S_t/\beta$, [2] where S_t is measured in

[2]Letting $\sigma = 0$ may simplify the equation, but this is not a necessary assumption. If system response time is slow, then σ should be reinstated.

number of nodes and links in a web tuple, and β is the display rate in nodes per second and σ is the system response time.

Proposition 3 *Let β be the display rate of number of nodes and their out-bound links per second and σ is system response time. Then, the difference between the display time (ΔT_d) of a joined or locally coupled web table W_{in} and RO-web table is given by*

$$\Delta T_d = (|W_{in}| - |W_{ro}|)\,\sigma + \frac{\sum_{i=1}^{|W_{in}|} N_{w_{in}}(t_i) - \sum_{i=1}^{|W_{ro}|} N_{w_{ro}}(t_i)}{\beta} \quad (3)$$

∎

The proof of the above proposition is given in [6].

Observation 3 Consider Equation 3. The display time of RO-web table is minimized if ΔT_d increases. That is, if $(|W_{in}| - |W_{ro}|)$ increases or $\left(\sum_{i=1}^{|W_{in}|} N_{w_{in}}(t_i) - \sum_{i=1}^{|W_{ro}|} N_{w_{ro}}(t_i)\right)$ increases. This implies that the difference in display time of RO-web table and joined or locally coupled web table increases under the following circumstances. First, if the size of the coupled web table is large compared to the RO-web table. Second, the difference between the total number of nodes in the joined or locally coupled web table and that in the RO-web table is large. However, this is possible if number of nodes to be eliminated from each coupled web tuple is large.

Note that when $\sigma = 0$, Equation 3 reduces to

$$\Delta T_d = \frac{\sum_{i=1}^{|W_{in}|} N_{w_{in}}(t_i) - \sum_{i=1}^{|W_{ro}|} N_{w_{ro}}(t_i)}{\beta}$$

This signifies that if we ignore system response time, then the optimization of display time of RO-web table is independent of difference between the size of joined or locally coupled web table and the RO-web table. ∎

5 Conclusions

In this paper, we have introduced the concept of reverse-osmosis operation in a web warehouse. Our paper focus on how reverse-osmosis can be used to minimize the cognitive overheads such as search time for locating relevant nodes and display time of a web table. We also provide formal algorithm to generate RO-web table. As part of future work, we wish to conduct an experiment to determine forms and parameters that influence the minimization of cognitive overheads.

The primary motivation of this work originates from the WHOWEDA project. In this project, we explore various aspects of web warehousing. Besides the importance of reverse-osmosis operation , some other issues that we are looking into include web information access and manipulation in a web warehouse, indexing and storage of web data in the web warehouse, view maintenance of web warehouse and mining useful information in the web warehouse context. We believe that web warehousing is an important new application area for databases, combining commercial interest with intriguing research questions.

References

[1] Z. AMJAD. Reverse Osmosis - Membrane Technology, Water Chemistry and Industrial. *Chapman & Hall*, 1993.

[2] S. BHOWMICK, S. K. MADRIA, W.-K. NG, E.-P. LIM. Web Bags: Are They Useful in A Web Warehouse? *Proceedings of 5th International Conference of Foundation of Data Organization (FODO'98)*, Kobe, Japan, November 1998.

[3] S. BHOWMICK, S. K. MADRIA, W.-K. NG, E.-P. LIM. Web Warehousing: Design and Issues. *Proceedings of International Workshop on Data Warehousing and Data Mining (DWDM'98) (in conjunction with ER'98)*, Singapore, 1998.

[4] S. BHOWMICK, W.-K. NG, E.-P. LIM. Information Coupling in Web Databases. *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, Singapore, 1998.

[5] S. BHOWMICK, S. K. MADRIA, W.-K. NG, E.-P. LIM. Detecting and Representing Relevant Web Deltas Using Web Join. *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS'00)* , Taiwan, 2000.

[6] S. S. BHOWMICK, W.-K. NG. Reducing Cognitive Overheads in a Web Warehouse Using Reverse-Osmosis. *Technical Report*, CAIS-TR-28-99, School of Applied Science, Nanyang Technological University, Singapore, 1999.

[7] S. S. BHOWMICK, W.-K. NG, S. MADRIA. Generating Schemas in WHOWEDA. *Technical Report*, CAIS-TR-7-00, School of Applied Science, Nanyang Technological University, Singapore, 2000. Submitted for publication.

[8] S. K. CARD. Human Limits and the VDT Computer Interface. *Readings in Human-Computer Interaction: A Multidisciplinary approach*, 1987.

[9] D. FLORESCU, A. LEVY, A. MENDELZON. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Records*, Vol 27, No. 3, Sept 1998.

[10] D. A. NORMAN. Design Principles for Human-Computer Interfaces. *Readings in Human-Computer Interaction: A Multidisciplinary approach*, 1987.

[11] 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.

[12] A. T. WELFORD. *Fundamentals of Skills*, 1968.