01 Jan 2000

Detecting and Representing Relevant Web Deltas Using Web Join

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

Wee Keong

Ee-Peng Lim

Sourav S. Bhowmick

Recommended Citation

S. K. Madria et al., "Detecting and Representing Relevant Web Deltas Using Web Join," *Proceedings of the 20th International Conference on Distributed Computing Systems, 2000*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2000.
The definitive version is available at https://doi.org/10.1109/ICDCS.2000.840935

Detecting and Representing Relevant Web Deltas using Web Join

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

In this paper, we show how to detect and represent web deltas*, i.e., changes in Web information, that are relevant to a user's query in the context of our* web warehousing *system called* WHOWEDA *(Warehouse of Web Data). In* WHOWEDA*, Web information are materialized views stored in* web tables *and can be manipulated and analyzed using a set of* web algebraic operators*. In this paper, we present a mechanism to detect* relevant *web deltas using* web join *and* outer web join*. We show how to represent these changes using* delta web tables*.*

Keywords: web deltas, web warehousing, web tables, web join, outer web join.

1 Introduction

Detecting changes to Web data is a challenging problem because the information sources in the Web are autonomous and typical database approaches to detect changes based on triggering mechanisms are not usable [4]. Consider the following scenario.

Example 1 Assume that there is a Web site at `http://www.panacea.gov/` which provides information related to drugs used for various diseases. The Web page at `www.panacea.gov` (denoted by a_0) contains a list of diseases. From this list each link of a particular disease points to a web page (denoted by b_0, b_1, b_2 etc. for various drugs) containing a list of drugs used for prevention of the disease. From the hyperlinks associated with each drug, one can probe further to find document (denoted by u_0, u_1 etc.) containing a list of various issues related to a particular drug, i.e., "description", "uses", "side-effects" etc.. From the hyperlinks associated with each issue, one can retrieve details of these issues for a particular drug.

Let us consider some modification to this Web site as shown in the Figures 1 and 2 respectively. These figures depicts the structure of this Web site as on 15th January, 2000 and February, 2000 respectively. Note that the black boxes, patterned boxes and grey boxes in these figures depict addition of new documents, modification of existing documents and deletion of existing documents respectively. Furthermore, the dashed dotted arrows indicates addition, deletion or modification of hyperlinks.

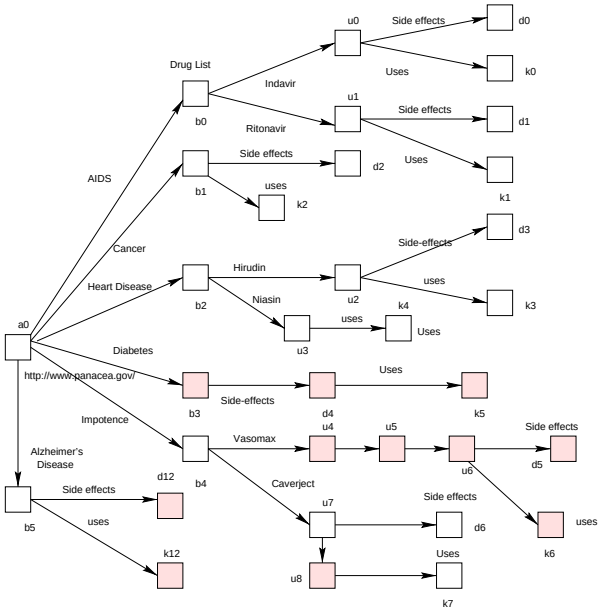

Figure 1. Web site on 15th January, 2000.

Suppose on 15th January, 2000, a user wish to find out periodically (say every 30 days), information related to side effects and uses of drugs for various diseases and also changes to these information compared to its previous version. This query requires access to previous states of the Web site and a mechanism to detect these changes automatically, features that are not supported by the Web or the existing search engines. Thus, we need a mechanism to compute and represent changes in the context of Web data. ∎

Although there is an increasing research effort on querying the Web [6], there is very little work on change detection and representation of Web data. The AT & T Internet Differ-

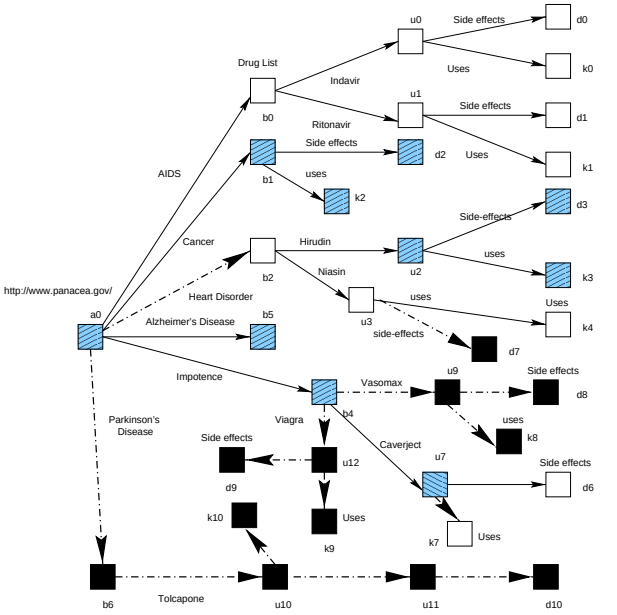

Figure 2. Web site on 15th February, 2000.

ence Engine (AIDE) [5] is a system that finds and displays changes to pages on the World Wide Web. The system consists of several components, including a web crawler that detect changes, an archive of past versions of pages, a tool called *HtmlDiff* to highlight changes between versions of a page, and a graphical interface to view the relationship between pages over time. In [4], the *snapshot-delta* approach has been used for representing changes in semistructured data. The authors present a simple and general model, *DOEM*, for representing changes and also present a language, *Chorel*, for querying over data and changes represented in *DOEM*. This model is founded on the OEM data model and the Lorel language [1]. It uses *annotations* on the nodes and arcs of an OEM graph to represent changes.

In this paper, we show how to detect and represent *web deltas*, i.e., changes in Web information, that are relevant to a user's query in the context of our *web warehousing* system called WHOWEDA (*W*are*h*ouse *o*f *We*b *Da*ta) [3]. Our work on change detection has three key characteristics: First we focus on detecting *relevant* web deltas. That is, our goal is to detect and represent only those web deltas that are relevant to a user's query, not any arbitrary web deltas. Secondly, our focus is on detecting and representing relevant changes between old and new versions of a set of inter-linked Web documents. In particular, we are interested in detecting those Web documents in a Web site which have been added to or deleted from the site, or those documents which are no longer considered relevant to a user's query. We also want to identify a set of documents which has undergone content modification compared to their antecedent. Furthermore, we wish to determine how these modified Web documents are related to one another and with other Web documents in the context of a user's query. Finally, we present a mechanism for detecting and representing relevant web deltas using a set of *web algebraic operators*.

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, metadata and structural 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.) [3]. 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. Note that we do not elaborate on the generation of *web schemas* of the web tables resulted from *global web coupling*, *web join* or *outer web join* operations. As we shall see, the web schemas of these web tables do not play a pivotal role in the web delta generation.

As Web data in our web warehouse are materialized views stored in the form of web tables, any changes to the relevant Web data are also reflected in the corresponding web tables. To describe a web delta between two versions of Web data, we use the notion of *delta web tables.* Delta web tables encapsulate the changes that have occurred in the Web, such as addition, modification or deletion of a set of web documents, in the context of a user's query. Intuitively, in order to detect web deltas, we materialize the old and new versions of data in two web tables. Next, we create a set of web tables by manipulating these input web tables using the *web join* and *outer web join* operators. Finally, we create a set of delta web tables by further manipulating the joined and outer joined web tables.

2 Change Detection Problem

In this section, we briefly introduce a mechanism called *global web coupling* for retrieving relevant data from the Web and then describe the change detection problem informally using Example 1. Due to space constraints, the formal definition of the change detection problem is given in [2].

Global web coupling [3] retrieves a set of web tuples 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 vari-*

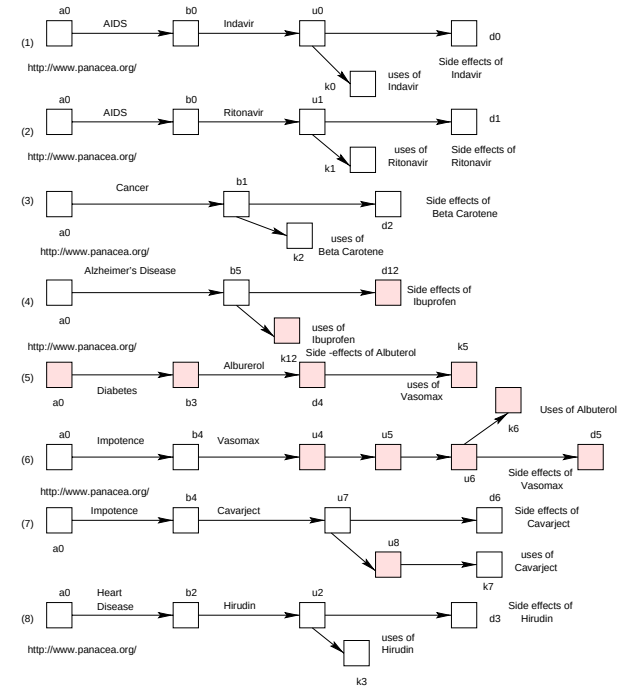

Figure 3. Partial View of web table "Drugs".

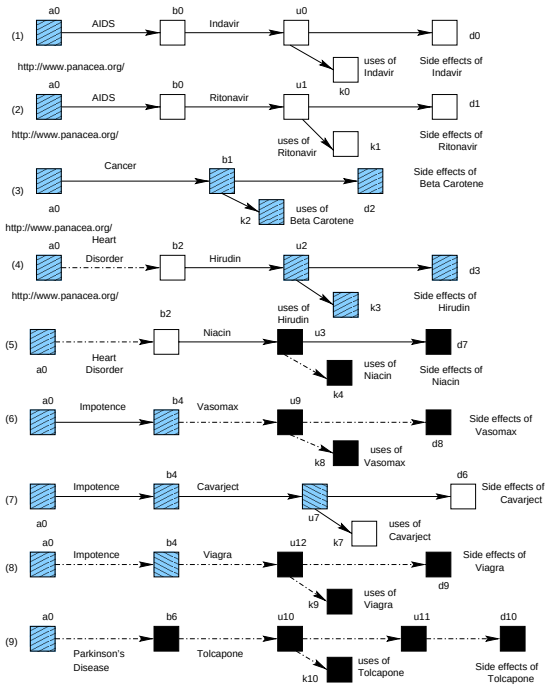

Figure 4. Partial View of "New Drugs".

ables, 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 hyperlinked structure. The last component Q of the coupling query imposes additional constraints over the coupling query in the form of predicates.

A *polling global coupling* is a global web coupling operation that repeatedly scans the Web for results based on some given criteria. The attribute *polling frequency* is used in a coupling query predicate to enforce the global web coupling operation to be executed periodically. An example of polling global coupling is given below.

Example 2 Consider Example 1. To initiate global web coupling to couple information related to side-effects and uses of various drugs from the Web site at `www.panacea.gov`, a user constructs a coupling query on 15th January, 2000 as follows: $G = \langle X_n, X_\ell, C, P, Q\rangle$ where $X_n = \{a, b, k, d\}$, $X_\ell = \{\text{-}\}$, $C \equiv k_1 \wedge k_2 \wedge k_3$ such that $k_1 = a\langle\text{-}\rangle b$, $k_2 = b\langle\text{-}\{1, 6\}\rangle d$, $k_3 = b\langle\text{-}\{1, 3\}\rangle k$ and $P = \{p_1, p_2, p_3, p_4\}$ such that $p_1(a) \equiv [a.\texttt{url EQUALS "http://www.panacea.gov/"}]$, $p_2(b) \equiv [b.\texttt{title CONTAINS "Drug List"}]$, $p_3(k) \equiv [k.\texttt{title CONTAINS "uses"}]$, $p_4(d) \equiv [d.\texttt{title CONTAINS "side effects"}]$ and $Q \equiv q_1$ where $q_1 \equiv [\texttt{polling frequency EQUALS "30 days"}]$. Note that that the expression '$\{1, 6\}$' in the connectivity k_2 specifies that at least one and at the most 6 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 3 successive hyperlinks connect an instance of b with an instance of k. Also note that the symbol '-' denotes a free link variable.

The above query will be polled every 30 days (i.e., 15th January, 2000, 15th February, 2000 etc. all at 10:00 PM) and all web tuples satisfying the connectivities and predicates are retrieved from the Web. The sets of interlinked documents retrieved by this operation are materialized in web tables Drugs and New Drugs respectively as shown in Figures 3 and 4. ∎

Given two such web tables, i.e., Drugs and New Drugs, containing the snapshots of two versions of relevant Web data, the problem of change detection is to find the set of tuples containing nodes which are inserted into or deleted from Drugs or those tuples containing nodes which are modified in Drugs to transform it into New Drugs. Note that these web tuples will reflect changes to the Web site that are relevant to the user.

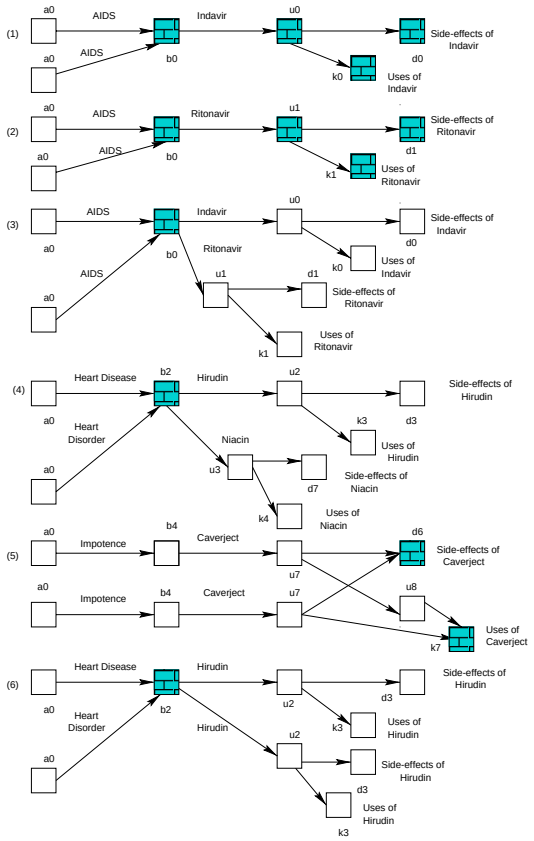

Figure 5. Partial view of joined web table.

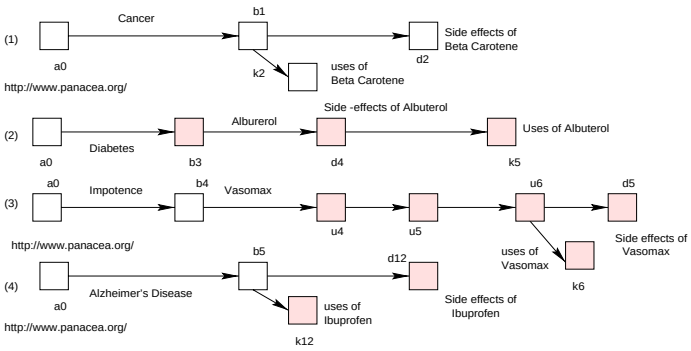

Figure 6. Left outer web join.

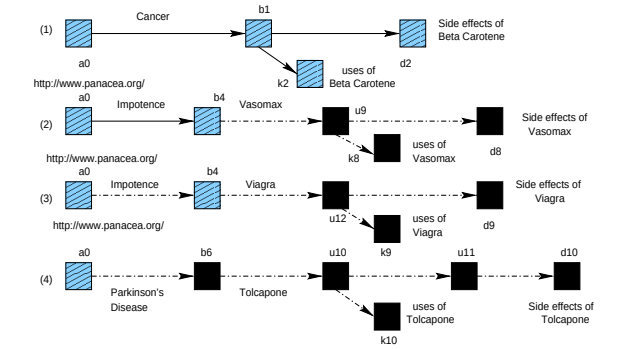

Figure 7. Right outer web join.

We define a structure called *delta web table* for representing these changes. Delta web tables encapsulate the relevant changes that have occurred in the Web with respect to a user's query. We define the following three types of delta web tables to represent the above types of change operations: (1)Δ^+**-web table** (denoted as W_{Δ^+}): contains a set of web tuples containing new nodes inserted into W_1 for transforming it into W_2. (2) Δ^-**-web table** (denoted as W_{Δ^-}): contains a set of web tuples containing nodes which are deleted from W_1. and (3)Δ^M**-web table** (denoted as W_{Δ^M}): represents web tuples containing old and new versions of nodes modified during the transition.

3 Web Join and Outer Web Join

In this section, we briefly introduce the web join and outer web join operators which we will be using to generate delta web tables. We discuss only those issues which are relevant to web delta generation.

Web Join: Informally, web join is used to combine *identical* data residing in two web tables. In web join, web tuples from two web tables containing *identical nodes* are *concatenated* into single web tuple over the *identical nodes* that can be materialized in a web table. We consider two nodes or Web documents identical when they have the same URL and last modification date. Note that nodes which are not joinable are called *dangling nodes*. For example, consider the first web tuple of Drugs and New Drugs in Figures 3 and 4. Since the nodes b_0, d_0, k_0 and u_0 remained unchanged during t_1 and t_2, these nodes are identical in both the web tables. Consequently, these nodes are joinable, the tuples are joinable. All joinable nodes in Figure 5 are depicted as patterned boxes. The first web tuple in Figure 5 is the joined tuple of these two web tuples. Observe that we remove one of the nodes from each pair of joinable nodes in the joined web tuple.

Outer Web Join: Web tuples in either input web table that do not participate in web join are called *dangling web tuples*. These tuples are absent from the joined web table. In certain situations it is necessary to identify dangling web tuples from the input web tables. The outer web join operation enables us to identify them. Depending on whether the outer-joined web table must contain the non-participant web tuples from the first or second web table, we define two kinds of outer web join: the *left-outer web join* and the *right-outer web join* respectively. Given two web tables W_1 and W_2, the resultant web tables from left-outer and right-outer web join on these two web tables will contain the dangling web tuples from W_1 or W_2 respectively.

Example 3 Consider the web tables Drugs and New

Drugs in Figures 3 and 4 respectively. The web tuples in Drugs and New Drugs, which are associated with the side effects and uses of "Beta Carotene", a drug used for cancer (third web tuple) do not participate in the web join process as the content of all the nodes in the web tuple in New Drugs has changed with respect to those in Drugs. The link structure of the web tuple related to "Vasomax" has been modified after 15th January, 2000 and none of the nodes in this web tuple in Drugs are joinable to the corresponding web tuple in New Drugs. The web tuple related to "Alzheimer's Disease" in Drugs is not materialized again in New Drugs as the new set of documents do not satisfy the coupling query anymore. Similarly, the web tuple containing documents related to "Diabetes" in Drugs has been removed from the Web site and is not materialized once again in New Drugs. These four web tuples in Drugs are dangling web tuples. Performing a left outer join on these two web tables enables us to identify these dangling web tuples (Figure 6).

Now consider the web table New Drugs. The last two web tuples dealing with "Viagra" and "Tolcapone" did not exist in the previous version (Drugs) as these drugs were added to the Web site after 15th January, 2000. Moreover, all the nodes in the web tuples related to "Vasomax" and "Beta Carotene" are modified. Thus, these tuples are dangling web tuples. Performing a right-outer web join on these two web tables enables us to identify these dangling web tuples in New Drugs (Figure 7). Observe that although the web tuple related to "Niacin" in New Drugs does not appear in Drugs, it is not a dangling web tuple as the node b_2 in this tuple is joinable with the corresponding node in the web tuple related to the drug "Hirudin" in Drugs. ∎

4 Generating Delta Web Tables

We now describe how, given two web tables W_1 and W_2 created by a polling global coupling operation at polling times t_1 and t_2, we compute a set of delta web tables corresponding to various types of changes to transform W_1 to W_2. This can be best described by the following four phases: *join tables generation phase*, the *delta node identification phase*, the *delta tuples identification phase* and the *delta table generation phase*. We discuss these phases in turn. A detailed discussion including the formal algorithms is given in [2].

Phase 1: Join Tables Generation Phase: In this phase the joined, right and left outer joined web tables are generated. It takes as input the two web tables, new and old versions, and generated the joined, right outer joined and left outer joined web tables (denoted by W_j, W_r and W_ℓ respectively). For instance, after this phase the web tables in Figures 5, 7 and 6 are generated from the web tables Drugs and New Drugs.

Note that right outer join operation on W_1 and W_2 may create three categories of dangling web tuples: (1) Web tuples which are added to W_1 during the polling times t_1 and t_2. These tuples may contain some new nodes and remaining nodes content are changed. (2) Tuples in which all nodes have undergone content modification and (3) Tuples in which some of the nodes are new and remaining ones content has changed but these tuples existed W_1. For example, consider the web table in Figure 7. The last two web tuples belong to the first category. The first web tuple belongs to the second category and the second web tuple is an example of a tuple whose all nodes have undergone content modification. Similarly, the left outer joined table may contain the following three categories of web tuples: (1) Web tuples which are deleted from W_1. These tuples do not occur in W_2. (2) Tuples in which every node has undergone content modification and (3) tuples in which some nodes are deleted from W_1 and remaining ones have been modified. The new and old versions occurs in both the tables in W_1 and W_2. For instance, the second and fourth web tuple in Figure 6 belongs to the first category of web tuples. The first and the third web tuples belong to the second and third categories respectively.

The web join operation on W_1 and W_2 may contain the following two types of web tuples: (1) Web tuples in which all the nodes are joinable nodes. These tuples are the results of joining two versions of web tuples in W_1 and W_2 in which all the nodes have remained unchanged during t_1 and t_2. (2) Web tuples in which some of the nodes are joinable nodes and remaining nodes are the result of insertion, deletion or modification operations during the transition. While generating delta web tables, we ignore the first category of web tuples in the joined web table as it does not reflect any change. For instance, in the joined web table in Figure 5, all the web tuples represent the second category. Specifically, the first three and the last three tuples contain nodes whose content are modified. The fourth web tuple contain nodes whose content are modified as well as nodes k_4, u_3 and d_7 which are inserted during t_1 and t_2. Finally, the fifth tuple contain a node u_8 which is deleted during the transition as well as a set of nodes which are modified.

Phase 2: Delta Nodes Identification Phase: In this phase, the nodes which are added, deleted or modified during t_1 and t_2 are identified. This phase takes as input the web tables W_1 and W_2 and the set of joinable nodes from the joined table and generate sets of nodes which are added, deleted or modified during the time interval. Note that in WHOWEDA, each node has a *node* and *version id*. The node ids of two nodes are different if their URLs are dissimilar. Two nodes has same node ids but different version ids if the URLs of these nodes are same but the last modification dates are different. Thus, node ids which exist in W_2 but not in W_1 are the new nodes that are added

to W_1. Similarly, node ids which only exist in W_1, but not in W_2 are the nodes that are removed from W_1. Furthermore, the nodes which are not joinable nodes, but they exist in W_1 as well as W_2 are essentially the nodes that have undergone content modification during t_1 and t_2. For instance, $\{b_3, u_4, u_5, u_6, u_8, d_4, d_5, d_{12}, k_5, k_6, k_{12}\}$ are the ids of nodes which appears in Drugs but not in New Drugs. Hence, these nodes were removed during transition. Similarly, $\{u_3, k_4, d_7, u_9, k_8, d_8, u_{12}, d_9, k_9, u_{10}, u_{11}, d_{10}, k_{10}\}$ are the ids of nodes that exist in New Drugs but not in Drugs. Hence, these nodes were added during t_1 and t_2. Finally, the nodes with ids a_0, b_1, d_2, k_2, u_2, d_3, k_3, b_4 and u_7 appears in both the web tables but are not joinable. Hence, these nodes has undergone content modification. These sets of node ids are denoted as $addNodeSet$, $delNodeSet$ and $updateNodeSet$ respectively.

Phase 3: Delta Tuples Identification Phase: To determine the association of nodes (represented by the identifiers in $addNodeSet$, $delNodeSet$ and $updateNodeSet$) with each other and with other relevant nodes in W_1, we identify the web tuples in W_j, W_r and W_ℓ containing these nodes and store them in the sets of web tuples denoted as $insertTupleSet$, $deleteTupleSet$ and $updateTupleSet$. Each element in $insertTupleSet$ and $deleteTupleSet$ are web tuples containing nodes that are inserted to or deleted from W_1 during t_1 and t_2. Each element in $updateTupleSet$ is an integrated web tuple containing old and new versions of nodes which has undergone content modification. The procedure of identifying delta web tuples can be decomposed into the following four steps.

1. Node Id Sets Generation: First, we identify the ids of all the nodes in the joined web table W_j (denoted by N_j) Next, we compute those nodes which are updated during the transition but are not captured by the joined web table. We denote this set of node ids as $K = updateNodeIdSet - N_j$. K represents those updated nodes which are not present in W_j but in W_r or W_ℓ. If $K = \emptyset$ then all updated nodes are captured by the joined web table. Otherwise, it is necessary to identify the tuples in W_r and W_ℓ which contains any nodes in K. Next, we identify those modified nodes which occurs only in the joined web table W_j, denoted as $U = updateNodeSet \cap N_j$.

2. Scanning Right Outer Joined Web Table: For each web tuple w_b in the right outer joined table the following steps are executed: **(1)** Identify the set of node ids, denoted as $tupleNodeIdSet[b]$, in W_b. **(2)** Check if any of the node ids in $tupleNodeIdSet[b]$ is contained in $addNodeSet$. If it is then the tuple contain node(s) which were inserted during the transition. Consequently, w_b is inserted in $insertTupleSet$. **(3)** If none of the nodes in w_b is an element of $addNodeSet$ then w_b contain nodes whose content were updated during the transition only. Hence, w_b is stored in a tuple set $temp1$. Note that $temp1$ stores those web tuples in right or left outer joined tables that do not contain any new or deleted nodes. Note that the tuples in $temp1$ represent web tuples in which all the nodes content are modified during the transition. For instance, $temp1$ will store the first web tuples in Figures 6 and 7. **(4)** Next, identify web tuples which contain modified nodes that are not present in the joined table. That is, check if there exist any node in w_b which is an element of K. If there is such node(s) then w_b is inserted in the tuple set $temp2$. Note that $temp2$ contain tuples from left or right outer joined tables that has at least one updated node not captured by the joined web table. That is, $temp2$ contain tuples from W_r and W_ℓ where each tuple contains at least one node that is an element of K. Observe that $temp2$ enables us to capture web tuples containing modified nodes that cannot be identified from the joined web tables. **(5)** Finally, the set of new nodes which are already identified form the right outer joined web table are removed from the $addNodeSet$.

3. Scanning Left Outer Joined Web Table: The procedure is similar to that of the scanning of right outer joined table. The only difference being that the tuples in W_ℓ are inspected now to check if they contain at least one node which is an element of $delNodeSet$. Hence, we do not elaborate on this step.

4. Scanning Joined Web Table: Finally, we proceed to inspect the joined web table W_j. Consider the case when all the inserted and deleted nodes are already identified from W_r and W_ℓ. If it is then the joined web table will only contain the updated nodes. Consequently, for each joined web tuple $w_a \in W_j$, the following steps are executed: **(1)** The set of node ids, denoted as $tupleNodeIdSet[a]$, in web tuple w_a are identified. **(2)** Next, the dangling node ids (denoted by $X = tupleNodeIdSet[a] \cap updateNodeSet$) in w_a are computed. Note that X represents those nodes which are not joinable in the web tuple w_a. **(3)** Identify the dangling node ids in w_a which belong to W_1 and W_2, denoted by $N_1(w_a)$ and $N_2(w_a)$ respectively. The purpose of this step is to identify the relevant tuples in the input web tables that contains these updated nodes. Naively, it may seem that each tuple containing dangling node(s) may represent old and new version of modified nodes. However, such assumption is not true. Note that we are specifically interested in those joined web tuples which contain both the old and new version of the updated nodes only. Note that not all joined web tuples may satisfy this condition. For example, consider the joined web table in Figure 5. The fourth web tuple contains the dangling nodes u_2, k_3, d_3 etc.. However, both the old and new version of these nodes are missing in this tuple. Specifically, these old and new nodes are found in the last joined web tuple. Hence, the last web tuple is inserted in $updateTupleSet$ but not the fourth web tuple. Therefore if $X - (N_1(w_a) \cap N_2(w_a)) = \emptyset$ then the w_a contains only the old and new version of the updated nodes. For

instance, for the last web tuple, $N_1(w_a) = \{a_0, u_2, d_3, k_3\}$, $N_2(w_a) = \{a_0, u_2, d_3, k_3\}$ and $X = \{a_0, u_2, d_3, k_3\}$. Hence, $\{X - (N_1(w_a) \cap N_2(w_a))\} = \emptyset$. However, for the fourth web tuple, $N_1(w_a) = \{a_0, u_3, k_3, d_3\}$, $N_2(w_a) = \{a_0, u_3, d_7, k_4\}$ and $X = \{a_0, u_2, d_3, k_3\}$. Hence, $\{X - (N_1(w_a) \cap N_2(w_a))\} = \{u_2, k_3, d_3, k_4\}$. Consequently, the condition is not satisfied. If the condition is true then the joined tuple is inserted in $updateTupleSet$ and the set of dangling nodes in w_a are removed from $updateNodeSet$.

Consider now the case when all the new or deleted nodes are not identified yet after scanning W_r and W_ℓ. In this case we are not only looking for web tuples containing updated nodes but also tuples containing the new or deleted nodes. Similar to the previous step, for each joined web tuple w_a the following additional steps are executed to identify web tuples containing new or deleted nodes: **(4)** If not all the dangling nodes in w_a represent new and old version of the nodes then we check if the remaining dangling nodes, i.e, $(D - (N_1(w_a) \cap N_2(w_a)))$ (where D is the set of all dangling nodes in w_a including new or deleted nodes), are actually new or deleted nodes. That is if $Y = D - (N_1(w_a) \cap N_2(w_a))$ then $Y \subset (addNodeSet \cup delNodeSet)$. If it is then the web tuple is inserted in $updateTupleSet$. Also the original web tuples in W_1 or W_2, which after join operation has generated w_a are extracted. The purpose is to identify the web tuples in W_1 or W_2 which contains the deleted or new node. For example, consider the fifth web tuple in Figure 5. Here $D = \{a_0, b_4, u_7, u_8\}$, $N_1(w_5) = N_2(w_5) = \{a_0, b_4, u_7\}$. Therefore, $D - (N_1(w_5) \cap N_2(w_5)) = \{u_8\}$. As u_8 represents a deleted node, i.e, $u_8 \in delNodeSet$, w_5 is inserted into $updateTupleSet$. Further, the seventh web tuple in Drugs is extracted and insert into $deleteTupleSet$. Observe that this web tuple contains the node u_8. **(5)** At this point the web tuples containing new nodes k_4, u_3 and d_7 in the joined web table (fourth joined tuple) has not been identified yet. Note that for this web tuple $Y = D - (N_1(w_4) \cap N_2(w_4)) = \{u_2, k_3, d_3, k_4, u_3, d_7\}$. Note that $Y \subset (addNodeSet \cup delNodeSet)$ or $Y \subset delNodeSet$ is not satisfied. Consequently, the Step (4) cannot identify these nodes from the joined web table. To identify these nodes, the conditions $Y \cap addNodeSet \neq \emptyset$ and $Y \cap delNodeSet \neq \emptyset$ are used. As $Y \cap addNodeSet = \{k_4, u_3, d_7\}$, the condition is true and the original web tuple (fifth web tuple in New Drugs) is retrieved and inserted in $insertTupleSet$. Note that If $Y \cap delNodeSet \neq \emptyset$ then is executed and the original web tuple from W_1 is retrieved and inserted in $deleteTupleSet$. **(6)** Finally, the $addNodeSet$ and $delNodeSet$ are updated by removing those inserted or deleted nodes which are already identified in w_a.

At this point we have identified all the web tuples containing new or deleted nodes. We have also identified some of the tuples containing updated nodes. The remaining tuples containing updated nodes are identified from $temp1$ and $temp2$. The web tuples in $temp1$ representing old and new version of modified nodes are inserted in $updateTupleSet$ as a single web tuple. For instance, the first web tuples in Figures 6 and 7 are contained in $temp1$. These two web tuples are combined together and inserted as a single web tuple (represented by the fourth web tuple in Figure 9). Similarly, the new and old version of the web tuples in $temp2$ are determined and inserted in $updateTupleSet$.

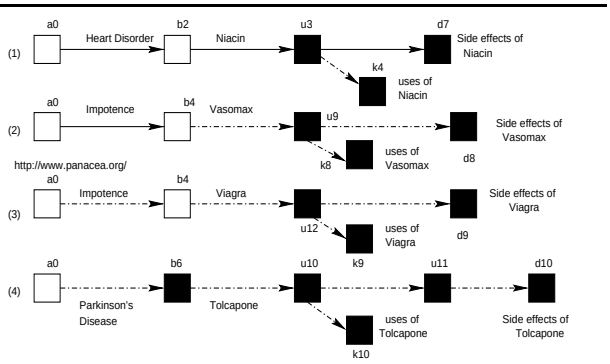

Figure 8. Δ^+-web table.

Phase 4: Delta Web Tables Generation Phase: Finally, the three types of delta web tables are generated in this phase. It takes as input the three sets of tuples, i.e, $insertTupleSet$, $deleteTupleSet$ and $updateTupleSet$ generated in the previous phase and generates the delta web tables from these sets. The procedure to generate these tables is straightforward. The tuples in $insertTupleSet$ are stored in Δ^+-web table. The tuples in $deleteTupleSet$ and $updateTupleSet$ are stored in Δ^- and Δ^M-web tables respectively. We now illustrate the generation of delta web tables with an example given below.

Example 4 Consider the two web tables Drugs and New Drugs in Figures 3 and 4. We would like to find the various change operations that transform Drugs into New Drugs and generate W_{Δ^+}, W_{Δ^-} and W_{Δ^M} tables. We discuss the generation of each delta web tables in turn.

Figure 9 depicts the Δ^M-web table. The patterned boxes in this figure in each web tuple are the old and new version of the nodes. For example, the second web tuple in Figure 9 contains the old and new version of the nodes a_0, u_2, d_3 and k_3, along with the joinable node u_2 (content of u_2 has remained unchanged during the transition). Each web tuple shows how the set of modified nodes are related to one another and with the joinable nodes. Observe that the first three web tuples are extracted from the joined web table in Figure 5. The last web tuple (enclosed in a dotted box) is the result of the integration of two web tuples - one from the

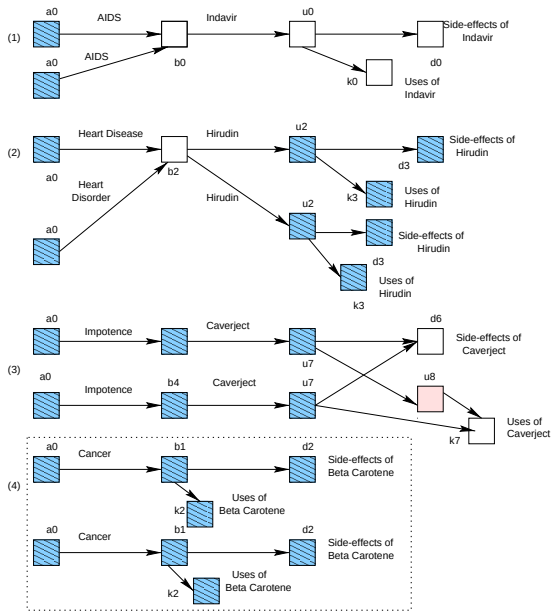

Figure 9. Δ^M-web table.

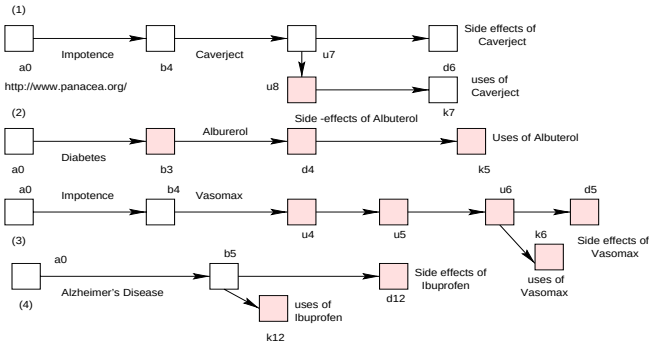

Figure 10. Δ^--web table.

left outer joined web table in Figure 6 and another from the right outer joined table in Figure 7.

Figure 8 illustrate the Δ^+-web table. The black boxes in each web tuple are the new nodes inserted into Drugs during 15th January, 2000 and 15th February, 2000. Similar to Δ^M-web table, each web tuple in Δ^+-web table shows how the new nodes are related to other relevant nodes in the web table. Note that the last three web tuples in Figure 8 are extracted from the right outer joined web table in Figure 7. However, as the node b_2 in the first web tuple is a joinable node, the new nodes d_7, k_4 and u_3 in this tuple are identified from the fourth web tuple of the joined web table in Figure 5.

Finally, Figure 10 depicts the Δ^--web table containing all the nodes that are deleted from Drugs. The last three web tuples are extracted from the web table in Figure 6. However, the tuple containing the deleted node u_8 is extracted from the fifth web tuple in the joined web table in Figure 5. ∎

5 Summary and Future Work

We have motivated the problem of detecting changes to Web data, i.e., web deltas, relevant to a user's query. To solve this problem, we have presented an approach that is based on representing two versions of Web data as web tables and manipulating these web tables using a set of web algebraic operators for detecting changes. We have represented the web deltas in the form delta web tables. As ongoing work, we are addressing the following issues: (1) Analytical and empirical studies of the algorithms for generating delta web tables. (2) Currently, the delta web tables contain tuples where only some of the nodes represents the insertion, deletion or update operation during the transition. This is because we wish to show how these nodes are related to one another and to other nodes which has remained unchanged during the transition. Therefore, we need a mechanism to distinguish between the modified, new or deleted nodes from each other and from the nodes which has remained unchanged in each delta web tables. We are currently building a data model over our warehouse data model to allow *annotation* on the affected nodes to represent these changes. (3) As we represent the web deltas in the form of web tables, these tables can be further manipulated using existing set of web operators and queried. We are designing and implementing powerful query languages for a change management system in the context of our web warehouse.

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] S. S. BHOWMICK, S. MADRIA, W.-K. NG, E.-P. LIM. Detecting and Representing Relevant Web Deltas in a Web Warehouse. *Technical Report*, CAIS-TR-2-99, School of Applied Science, Nanyang Technological University, Singapore, 1999.

[3] 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.

[4] S. CHAWATHE, S. ABITEBOUL, J. WIDOM. Representing and Querying Changes in Semistructured Data. *Proceedings of ICDE 98*, Orlando, Florida, February 1998.

[5] F. DOUGLIS, T. BALL, Y-F CHEN, E. KOUTSOFIOS The AT & T Internet Difference Engine: Tracking and Viewing Changes on the Web. *World Wide Web Journal*, 1(1), pp. 27–44, January 1998.

[6] D. FLORESCU, A. LEVY, A. MENDELZON. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Records*.