

# Web Schemas in WHOWEDA

Sourav S Bhowmick  
School of Computer  
Engineering  
Nanyang Technological  
University  
Singapore 639798  
assourav@ntu.edu.sg

Wee Keong Ng  
School of Computer  
Engineering  
Nanyang Technological  
University  
Singapore 639798  
awkng@ntu.edu.sg

Sanjay Madria  
Department of Computer  
Science  
University of Missouri-Rolla  
Rolla, MO 65409, USA  
madrias@umr.edu

## ABSTRACT

The term schema denotes whatever way a data model chooses to model its data. In this paper we discuss schemas of a set of HTML or XML documents retrieved from the Web in the context of our web warehousing system called WHOWEDA (*WareHouse Of WEb DAta*). *Web schemas* are used to bind a *web table* that contains a collection of interlinked web documents called *web tuples*. These schemas specify some of the metadata, content and structural properties (in the form of *predicates*) shared by some of the Web documents and hyperlinks in the web table. They also summarize the hyperlink structure of these documents using the notion of *connectivities*. We show how web schemas are generated in WHOWEDA and discuss different types of operation that may be performed on web schemas.

## Keywords

web schema, web warehouse, schema operations

## 1. INTRODUCTION

The standard database paradigm for schema generation involves first creating a schema to describe the structure of the database and then populating that database through the interface provided by the schema. However, due to the irregular, inconsistent nature of Web data, we do no longer have this luxury of a priori schema for data in the Web [1]. The dynamic nature of the Web further aggravates the problem of reusing traditional schema generation techniques in the context of Web data. As new data is added frequently to its sources in the Web, we may find a schema is incomplete or inconsistent. Consequently, the rigidity of traditional schemas become a stumbling block to reuse conventional techniques for generating and managing schemas of Web data. This has become the driving force behind increasing research activities in generating schemas for semistructured data [3, 9, 10, 13, 14]. For instance, the authors in [9]

present schemas for graph-structured databases. A formal definition of a *graph schema* is given, along with an algorithm to determine whether a database conforms to a specific schema. This work is presented with a more traditional view of a schema. In [13], schema information is used for semistructured data translation. In [3] the authors elaborate on the theoretical foundations of the schema model in [13]. They present two schema definition languages, SCMDL and VSCMDL, and illustrate their expressive power. DataGuides [10] is cast in the context of Lore system [2]. It provides a structural summary of a semistructured database. Rather than require an explicit schema that all data must follow, a DBMS can support free-form data, dynamically generating and maintaining a DataGuide that summarizes the structure of the data. In [11] Goldman and Widom relax the notion of DataGuides to *Approximate DataGuides* (ADG) to address the limitations on performance traps particularly for cyclic databases. In [14], a work on the extraction of implicit structure in semistructured data modeled in the style of [2] as directed, labeled graph is presented. To achieve this an algorithm for *approximate* typing of semistructured data based on patterns of incoming and outgoing edges is proposed. By *approximate* typings, the authors meant an object does not need to fit its type definition precisely.

In this paper, we introduce the notion of *web schema* to model instances of warehouse data in the context of our web warehousing system, called WHOWEDA (*WareHouse Of WEb DAta*) [7, 15]. Informally, our web warehouse can be conceived of as a collection of *web tables*. A web table contains sets of *web tuples* and *web schemas*. A web tuple is a directed graph consisting of sets of *node* and *link objects* (hereafter, referred to as *nodes* and *links* respectively for brevity). Intuitively, a *node* represents the metadata associated with a Web document and the content and structure of the document (excluding hyperlinks in the document). Specifically, it consists of two components: a set of *node metadata trees* to represent values of different metadata associated with the document (such as URL, size and date of last modification) and a *node data tree* (directed labeled tree) to represent the content and structure of the document. A *link* consists of a set of link meta-attribute/value pairs (such as **target URL**, **source URL** and **link\_type**) represented as *link metadata trees*, a *link data tree* and an unique reference identifier. Link data tree is a directed labeled tree to represent the structure and content of a HTML or XML link<sup>1</sup>. The ref-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM DOLAP '00 McLean, VA USA

Copyright 2000 ACM 0-89791-88-6/97/05 ..\$5.00

<sup>1</sup>We only consider simple and extended XML links.

erence identifier is used to associate the *location* of links in a particular Web document or node. Informally, one can think of a location as a portion of a document or a position in it. The reader may refer to [4] for complete discussion on how Web data is represented in the warehouse.

Schemas, in general, play a pivotal role in query formulation and evaluation. Without schemas these tasks become significantly harder. Traditionally, a schema provides structural summary of its data instances. However, we believe that to expedite query formulation and evaluation in a web warehouse, only structural summary of a set of Web documents is not enough. Query formulation and evaluation can be performed efficiently if some of the content and metadata properties shared by the Web documents and hyperlinks are highlighted in the schema. Although structural summary of XML documents may enable us to speculate the content of XML data due to the usage of meaningful user defined tags, such may not be the case for HTML documents. HTML tags are not used for describing the data segment enclosed in it and hence structural summary of HTML pages is not always very useful in subsequent query evaluation and formulation. Furthermore, capturing summary of the hyperlink structure of a set of Web documents may also help to formulate meaningful queries in the warehouse. Consequently, a *web schema* in WHOWEDA provides two types of information: First, it specifies some of the common properties shared by the documents and hyperlinks in the web table with respect to not only their structure, but also their metadata and content. For instance, given a set of documents in a web table, the web schema(s) may specify that the title of all these documents contain the keyword “genetic disorder”. It may also specify that these documents belong to the web site at [www.ninds.nih.gov](http://www.ninds.nih.gov) and contain the tags `symptom`, `treatment` and `drugs` inside the tag `disease`. Secondly, a web schema(s) summarizes the hyperlink structure of these documents in the web table. For instance, a web schema may specify that a set of documents containing the keyword “genetic disorder” are directly linked to a set of documents having the tags `drugs` and `side effects` via a set of hyperlinks whose label contain the keyword “drugs”. For brevity, a schema will mean a web schema (unless explicitly stated otherwise).

## 2. WEB SCHEMA

In WHOWEDA, *web schemas* are used to specify the *common characteristics* of a set of web tuples stored in the form of a web table. The main features of a web schema are as follows:

- Some of the nodes and links in a set of web tuples may share common characteristics with respect to their content, structure and metadata. Thus, these nodes and links can be represented collectively by a *node* or *link type identifier* that identifies some of their common properties. A web schema encapsulates such node or link type identifiers to express such common characteristics of some of the nodes and links.
- In a web tuple some of the nodes, if not all, are connected to other nodes by hyperlinks. A web schema summarizes these *connectivity* properties of the nodes in the web table. For instance, if all nodes of type *A* are directly connected to nodes of type *B*, then the schema may express this connectivity property involv-

ing nodes of types *A* and *B*.

- Due to the unstructured and irregular nature of the Web, there may exist a set of nodes or links in the web tuples whose common characteristics are not known ahead of time, or they may not simply share any similar properties. Moreover, there may also exist a collection of nodes and links which does not share any common characteristics with respect to their connectivities with one another or other nodes. Such nodes and links are called *free* nodes and links. A web schema is flexible enough to represent these nodes and links and their connectivities.

### 2.1 Components of a Web Schema

Informally, a web schema consists of four components: sets of *node* and *link type identifiers* to represent a collection of node and link objects respectively, a set of *connectivities* defined over these node and link type identifiers to express the interlinked structure of the documents and a set of *predicates* to express the common characteristics of some of the nodes and links with respect to their metadata, textual content or structure. We now briefly discuss these components.

A web schema, denoted by  $S = \langle X_n, X_\ell, C, P \rangle$ , contain sets of *node* and *link type identifiers* (denoted by  $X_n$  and  $X_\ell$  respectively) where each element in  $X_n$  or  $X_\ell$  is a nominal identifier of a set of node or link objects. That is, each nominal identifier represents a set of documents or hyperlinks (possibly empty) retrieved from the Web. Each identifier in  $X_n$  or  $X_\ell$  may be either *bound* or *free*. The set of node or link objects represented by a *bound* type identifier share some *common properties* in terms of their metadata, content or structure. Some of these properties are expressed explicitly in the schema using a set of *predicates* (discussed later). A *free* type identifier does not have any predicate defined over it in the schema. That is, there are no conditions in terms of metadata, content or structure imposed by the user on the nodes or links represented by the free type identifier. In WHOWEDA, we denote such free node and link type identifiers by using the special symbols ‘#’ and ‘-’ respectively.

A set of *predicates*  $P$  on the node and link type identifiers express constraints on the metadata, contents or structure of the instances of some of the node or link type identifiers. A predicate consists of the following components: *predicate qualifier*, *attribute path expression*, *predicate operator* and *value* for imposing various constraints over node or link objects. Formally, if  $x$  is a node or link type identifier then the following is the form of predicate on  $x$ :  $p(x) \equiv \text{predicate\_qualifier}::x\{\text{attribute\_path\_exp}\}$   $\text{predicate\_operator}$  "V".  $\text{predicate\_qualifier}$  determines the scope of the predicate. It can have any one of the following values: “METADATA”, “CONTENT”, “STRUCTURE” and “DTD”. It determines whether the predicate is to be used to impose constraints on the metadata, textual content, structure or DTD (for XML data only) of instances of  $x$ .  $\text{attribute\_path\_exp}$  is essentially a sequence of tags which may include wild cards and regular expression operators e.g., `table.tr.td`, `html(.%)+`, `disease.treatment|cure`. It is used to specify constraints on specific location of Web documents and hyperlinks. It may also be used to impose structural constraints on Web data.  $\text{predicate\_operator}$  represents operators such as EQUALS, ATTR\_ENCL, NON-ATTR\_ENCL, ATTR\_CONT, NON-ATTR\_CONT and CONT to test for string regular expression matching. The operators containing the strings

ATTR and NON-ATTR are used to distinguish between the attribute/value pairs associated with tags of HTML or XML elements and the textual content between tagged elements when desired. The strings CONT and ENCL in these operators are used to further distinguish between partial and complete data segments in an element or in the attribute set associated with an element. Operators such as SATISFIES and EXISTS\_IN are used to impose conditions on the structure and DTD. The operator SATISFIES checks if the instances of a node or link type identifier satisfies a particular element structure or DTD. The operator EXISTS\_IN is specifically used on link objects and checks if instances of a link type identifier exist in the specified portion of source documents.  $V$  is called the *value* of a predicate and is a regular expression over the ASCII character set when the *predicate\_qualifier* is either METADATA or CONTENT. Specifically, for CONTENT  $V$  may also be a regular expressions over a set of attribute name/value pairs. When the predicate qualifier is STRUCTURE,  $V$  may be an attribute path expression or a collection of tag element names or a DTD name. The curly brackets are used to indicate 0 or 1 occurrence of the components. That is the component *attribute\_path\_exp* is optional and may not occur in all types of predicates. *Attribute\_path\_exp* is mandatory for predicates with qualifier METADATA or CONTENT. However, *attribute\_path\_exp* may not appear in some cases when the predicate qualifier is STRUCTURE.

A web schema may contain a set of *connectivities*  $C$  to impose constraints on the hyperlink structure of the relevant documents. Note that the predicates can only impose conditions on the metadata, structure and content of Web documents and hyperlinks. However, they fail to impose constraints on how these documents are connected to one another. In order to exploit inter-document relationship, i.e., the hyperlink structure of relevant Web documents, we introduce the notion of *connectivities*<sup>2</sup>. A connectivity  $k$  of a web schema is an expression of the form:  $k \equiv s(\ell)t$  where  $s$  and  $t$  are node type identifiers (called *source identifier* and *target identifier* respectively) and  $\ell$  is a link type identifier. The angle brackets around  $\rho$  are used for delimitation purposes only. For instance,  $x(e)y$ ,  $\#_1(-)y$  etc. are examples of connectivities in a web schema. Note that the connectivity  $s(\ell)t$  specifies that the instances of the identifier  $s$  are connected to the instances of  $t$  via hyperlinks which are instances of  $\ell$ .

## 2.2 Types of Web Schema

A web schema is categorized into two types: *complex* and *simple* web schemas. We say that a web schema is *complex* if it contains only sets of connectivities in Disjunctive Normal Form (DNF). That is, if  $C_1, C_2, C_3, \dots, C_n$  be sets of connectivities such that  $C_1 \vee C_2 \vee \dots \vee C_n$  and each  $C_i$  contains a set of connectivities in conjunction to one another then  $S$  is a complex web schema. For example, if  $C \equiv (k_1 \wedge k_2) \vee k_3$  for a web schema  $S = \langle X_n, X_\ell, C, P \rangle$  where  $k_1 \equiv x(e)y$ ,  $k_2 \equiv y(f)z$  and  $k_3 \equiv x(e)z$  then  $S$  is a complex web schema. A simple web schema, on the other hand, contains only a set of connectivities where each connectivity is in

conjunction to another. That is  $S$  is simple if it contains a set of connectivities  $k_1, k_2, k_3, \dots, k_n$  such that  $k_1 \wedge k_2 \wedge \dots \wedge k_n$ .

## 2.3 Importance of Web Schema

We now discuss the benefits realized by the use of web schemas in a web warehouse. Beyond its use to define the structure of a set of data in the warehouse, a web schema serves two important functions. It helps user in query formulation and aids the query processor for efficient execution of query. We elaborate on these two functions.

A web schema enables user to understand the partial structure and content of a set of instances of the schema materialized in the form of web tuples and form meaningful query over it. Without a schema this task becomes significantly harder as the user does not know the structure and content of underlying data and this impedes his/her efforts to formulate reasonable queries. Although it may be possible to manually browse a small set of web tuples to formulate queries, in general forming a meaningful query when there exists large number of web tuples is difficult without a schema or some kind of summary information of the underlying data.

A web schema also facilitates efficient query processing in the web warehouse. A query processor relies on the schema to devise efficient plans for computing query results. It uses information available in the schema to prune the search. Note that a lack of information about the structure and content of the data in the warehouse can cause a query processor to resort to exhaustive searches. We illustrate this with an example given below.

Suppose we wish to combine treatment information of neurological disorders which are stored in two web tables. Assume that the query processor need to perform a *web join* operation [6] on the instances of node type identifiers  $y$  and  $b$  in these web tables to gather this information. Intuitively, web join is an operation to concatenate two web tuples over identical nodes having same URL and content. Also, for the moment ignore the existence of schema in the warehouse. Thus, in order to perform web join operation, the query processor has to compare each instances of  $y$  to the instances of  $b$  to determine if they have identical URL and content. However, if there exist schemas of these two web tables then it may be possible for the query processor to restrict his search for identical nodes by inspecting the schemas. The query processor may evaluate the predicates (defined in the schemas) associated with  $y$  and  $b$  before evaluating the instances of  $y$  and  $b$ . Suppose the predicates on  $y$  and  $b$  are as follows:

```

p1(y)  ≡  CONTENT:y[title] NON-ATTR_CONT
          ":START_STR: + neurological disorders
          + :END_STR:"
p2(b)  ≡  CONTENT:b[title] NON-ATTR_ENCL
          "Treatment"

```

From these predicates it is clear that the instances of  $y$  and  $b$  cannot be identical as the titles of the instances of  $y$  and  $b$  are not same. Consequently, web tuples containing instances of  $y$  and  $b$  cannot be joined over these identifiers and the joined web table will return no web tuples. Note that the query processor can efficiently determine the result of web join operation without operating on the web tuples. Observe that the importance of a web schema in query formulation and query evaluation depends on the quality or

<sup>2</sup>Traditionally, in the field of graph theory, the term connectivity of a connected graph is defined to be the minimum number of vertices whose removal disconnects the graph or reduces the graph to a single vertex. However, in this paper we do not use the notion of connectivities in this context.

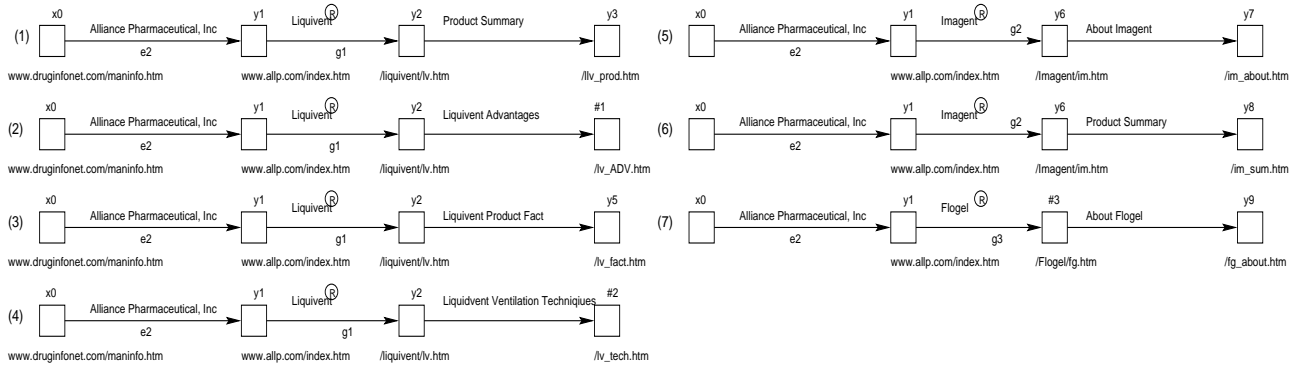


Figure 1: Web tuples.

*goodness* of the schema (the quality of non-trivial information provided by the schema). Thus, a very “weak” or low quality schema may not provide much help in query evaluation and formulation. For example, suppose in the above example the predicates defined on  $b$  and  $y$  be:

$$\begin{aligned}
 p'_1(y) &\equiv \text{CONTENT::}y[\text{title}] \text{ NON-ATTR\_CONT} \\
 &\quad \text{:START\_STR:} + \textit{neurological disorders} \\
 &\quad \text{:END\_STR:} \\
 p'_2(b) &\equiv \text{CONTENT::}b[\text{title}] \text{ NON-ATTR\_CONT} \\
 &\quad \text{:BEGIN\_WORD:} + \textit{Treatment} + \text{:END\_WORD:}
 \end{aligned}$$

These predicates do not guarantee that the URL and content of instances of  $b$  and  $y$  are not identical. Hence, these predicates do not aid the query processor in determining if the web tuples can be joined over the instances of  $b$  and  $y$  by inspecting the schemas alone. It has to inspect each instance of  $b$  and  $y$  to determine whether the web tuples can be joined.

Note that a web schema may not always aid in query formulation and evaluation as it is not always possible to provide a complete summary of a set of web tuples. But having a web schema is certainly advantageous in some situation, rather than having no structural and content summary of warehouse data instances.

## 2.4 Web Tuples and Web Table

*Web tuples* are instances of a web schema. Recall that a web tuple is a directed graph consisting of a set of node and link objects that are instances of node and link type identifiers respectively. That is, each web tuple contains a set of documents and hyperlinks. Informally, a web tuple  $t$  conforms to a schema  $S$  if the set of documents and hyperlinks in  $t$  satisfies the connectivities and predicates defined in  $S$ . The reader may refer to [5] for a detailed treatment on *schema conformity*. In WHOWEDA, web tuples are generated by two methods: First, by retrieving a set of Web documents from the Web that matches a user specified *coupling query* using *global web coupling* operation [12]. Note that in WHOWEDA, a user express a query in the form of coupling query [5] to retrieve relevant documents from the Web. Informally, a coupling query consists of five components: sets of node and link type identifiers, a set of connectivities, a set of predicates and a set of *coupling query predicates*. The components node and link type identifiers and set of predicates are identical to those of a web schema. However, a connectivity in a coupling query is more generalized version

of the connectivity in a web schema. A connectivity  $k$  in a coupling query is an expression of the form:  $k \equiv s\langle\rho\rangle t$  where  $s$  and  $t$  are the node type identifiers and  $\rho$  is called a *link path expression* which is essentially a sequence of link type identifiers and may include regular expressions, e.g.,  $e$ ,  $efg$ ,  $ef\{1,3\}$ . Recall that in a web schema the link path expression is simply a link type identifier. Further, the target identifier may also contain regular expression to express *certain* types of connectivities. Finally, a set of *coupling query predicates*  $Q$  may be used to control the execution of the query for retrieving relevant data. The coupling query is evaluated by global web coupling operation and a set of documents satisfying this query are garnered from the Web and stored in a web table in the form of web tuples. Each web tuple satisfies the coupling query. A set of new web tuples can also be generated by manipulating web tuples stored in web tables using a set of *web algebraic operators* [15]. In this case, the generation of such web tuples depends on the type of web operation performed on the web table(s) and query conditions.

A set of web tuples is stored in a *web table*. A set of simple schemas is used to bind these web tuples. Informally, a web table  $W$  consists of the following: *name* of the web table, a set of simple web schemas  $S$  and a set of web tuples  $T$  bound by these web schemas. Each simple web schema  $S_i \in S$  binds a set of web tuples  $T_i \subseteq T$ . The pair  $(S_i, T_i)$  is collectively called as *partition* (denoted by  $A$ ). Note that a partition may contain a web schema which does not bind any web tuples, i.e.,  $T_i = \emptyset$ . Such partition is called *noisy partition* and  $S_i$  is called *noisy schema*. Notice that a web table can be realized as a set of partitions, denoted as  $W = \langle N, \mathcal{P} \rangle$  where  $\mathcal{P} = \{A_1, A_2, \dots, A_n\}$ . Also observe that a web table is associated with a set of simple web schemas but not with complex schemas. In the web warehouse, a complex web schema binding a set of web tuples  $T$  is *decomposed* and *pruned* into a set of simple web schemas before assigning them as the schema(s) of the web table containing  $T$ . In Section 3 we elaborate on this. We now illustrate the notion of web schema with an example.

**EXAMPLE 1.** Consider the web page at [www.druginfonet.com/maninfo.htm](http://www.druginfonet.com/maninfo.htm). It provides a list of addresses, e-mail addresses, links and telephone numbers of many pharmaceutical manufacturing companies. Each link related to a company connects to the web site of the particular company. Suppose we wish to retrieve information of various products

of the company “Alliance Laboratories” at [www.allp.com](http://www.allp.com). Note that this web site use individual product names as anchor of hyperlinks to describe their products (i.e., “Liquid”, “Flogel” etc.). Also observe that each product name contains the symbol of encircled  $R$ . Note that in HTML this symbol is expressed by the special character `&reg`. Hence, the query to retrieve the set of interlinked documents may be expressed as follows. Let  $G = \langle X_{cn}, X_{cl}, C_c, P_c, Q_c \rangle$  be the coupling query such that  $X_{cn} = \{x, y\}$ ,  $X_{cl} = \{e, g, -\}$ ,  $C_c \equiv x(eg-)y^+$ ,  $Q_c = \emptyset$  and  $P_c = \{p_1, p_2, p_3, p_4\}$  where

```

p1(x)  ≡ METADATA:x[url] EQUALS
        "http://www.druginfonet.com/maninfo.htm"
p2(y)  ≡ CONTENT:y[html(.%)+] NON-ATTR_CONTENT
        ":BEGIN_WORD: + products? + :END_WORD:"
p3(e)  ≡ CONTENT:e[A(.%)*] NON-ATTR_ENCL
        "Alliance Laboratories"
p4(g)  ≡ CONTENT:g[A(.%)*] CONT
        "MATCH(. * &reg + :END_STR:)"

```

Note that the connectivity  $C_c$  specifies that starting from an instance of  $x$  one may encounter one or more nodes of type  $y$  while following the links of type  $e$ ,  $g$  and  $-$  respectively. The set of web tuples retrieved by the query is shown in Figure 1. The schema for these web tuples can be then expressed as follows. Let the web schema be  $S = \langle X_n, X_\ell, C, P \rangle$  such that  $X_n = \{x, y, \#1, \#2, \#3\}$ ,  $X_\ell = \{e, g, -\}$ ,  $C \equiv C_1 \vee C_2 \vee C_3 \vee C_4 \vee C_5 \vee C_6 \vee C_7$  where

$$\begin{aligned}
C_1 &\equiv x(e)\#1 \wedge \#1(g)\#2 \wedge \#2(-)y \\
C_2 &\equiv x(e)\#1 \wedge \#1(g)y \wedge y(-)\#3 \\
C_3 &\equiv x(e)y \wedge y(g)\#2 \wedge \#2(-)\#3 \\
C_4 &\equiv x(e)y \wedge y(g)y \wedge y(-)\#3 \\
C_5 &\equiv x(e)\#1 \wedge \#1(g)y \wedge y(-)y \\
C_6 &\equiv x(e)y \wedge y(g)\#2 \wedge \#2(-)y \\
C_7 &\equiv x(e)y \wedge y(g)y \wedge y(-)y
\end{aligned}$$

and  $P = P_c$ . Observe that the connectivities  $C_c$  in  $G$  is transformed into sets of simple connectivities in DNF in the schema. The connectivities in each set are in conjunction to one another. Hence, this is an example of complex web schema. ■

We now discuss some issues related to the modification of web tables. In WHOWEDA, a web table is not updated; i.e., no web tuples are inserted, deleted or modified. The only deletion occurs when we wish to delete the entire web table. We justify with reasons for this. In a relational database, insertion, deletion or update of a tuple in a relation is driven by the schema of the relation. A tuple can only be inserted or updated if and only if it strictly adheres to the schema of the relation. This assumption is reasonable in the traditional environment as the schema of the relation is relatively stable, rigid and does not change frequently. However, allowing the update, deletion or insertion of web tuples in a web table based on the conventional approach has some serious disadvantages as discussed below.

The change of structure and content of Web data is much more frequent than relational data. A consequent of this property is that a relevant web tuple may not be inserted in a web table simply because it does not satisfy the previously generated schema of the web table. One may argue that the insertion of web tuples may be possible by modifying the set of web schemas. However, such an operation has two major repercussions. First, the schema may have to be modified frequently due to frequent changes to Web data. Frequent

modification of web schema to incorporate new data may adversely affect the quality or *goodness* of existing web schema. Consequently, modification of web schema incurs overheads in maintaining the goodness of these schemas. Second, some of the existing web tuples may not satisfy the modified web schemas any more. Moreover, web tuples may now satisfy more than one schemas. Hence, web table modification may result in regenerating a new set of schemas for all the web tuples again. As a result, the maintenance overhead of web schemas and web tuples are considerably high. Note that such problems do not arise in relational databases as the schema of a relation is rigidly and completely defined and data are forced to comply to the schema. However, due to the very nature of Web data, such enforcement is not feasible.

The second issue is the manipulation of historical Web data. One of the objective of our web warehouse is to be able to store and manipulate historical data to facilitate data analysis over broad vistas of data over time. These historical data are generated due to changes to the Web. Web pages replace its antecedent, usually leaving no trace of the previous document. These rapid and often unpredictable changes to the information create a new problem of detecting, representing and querying these changes. These changes are reflected in the warehouse in the form of insertion, deletion and update of previous version of web tuples. However, if we delete or modify the previous versions of a web tuple then there is no way to retrieve these versions again for further analysis. Hence, data once deleted or updated is completely lost. If we wish to query these changes by comparing their snapshots at different instances of time then we need to store different versions of modified web tuples. Hence, it is convenient to store different snapshots of web tuples in different web tables and use a set of algebraic operators to manipulate them [7].

### 3. GENERATION OF WEB SCHEMAS

Web schemas in WHOWEDA are generated by two ways: (1) By manipulating the coupling query of the globally coupled Web data and (2) by manipulating the schemas of the input web table(s) during local operations to generate a new set of web schemas for resultant web table. Note that, unlike DataGuides [10, 11], which are inferred automatically from the data, our schema generation process first create a set of schemas without looking at the data retrieved by the coupling query (by manipulating coupling query or schemas of input web tables) and then prune the schemas by inspecting the hyperlink structure of the data instances. Hence, we begin the schema construction without inferring from the data and then refine the set of schemas, if possible, by inferring from the web tuples. Also, DataGuides are recomputed or incrementally updated when the data changes, the web schemas of a web table are not incrementally updated as the web tables in WHOWEDA are not modified. We store different snapshots of web tuples in different web tables [6]. Hence, the maintenance of web schemas is less complicated compared to DataGuides.

The first mechanism of schema generation, i.e., from the coupling query, consists of three stages as follows: (1) *Coupling Query to Schema Transformation Phase*: The gist of this stage is to transform the coupling query specified by the user to a simple or complex web schema by eliminated all *coupling query predicates* defined over it and *refining* the

set of predicates defined over the node and link type identifiers based on the *types* of coupling query predicates. Note that if the transformation generates a simple web schema then all the remaining stages are ignored. Note that we wish to bind a set of web tuples by a set of simple schemas which is the most simplest form of web schema. The advantages of expressing schemas in simple form is the *rigidity* it imposes on the web tuples structure and its contribution to efficient query formulation and evaluation. (2) *Complex Schema Decomposition Phase*: In the second stage, we convert the complex web schema that may result from the first phase into a set of simple web schemas. Recall that if  $C$  is the set of connectivities of a complex web schema  $S$  then  $C \equiv C_1 \vee C_2 \vee \dots \vee C_r$  for  $r > 1$ . The complex schema decomposition process takes as input such complex web schema  $S$  and generates as output a set of simple web schemas, i.e.,  $S_1, S_2, \dots, S_r$ . Observe that the decomposition of the complex schema  $S$  into simple schemas is driven by the sets of simple connectivities in  $S$ . That is, if there are  $r$  sets of simple connectivities in  $S$  where the connectivities in each set are in conjunction to one another then the schema decomposition process will generate  $r$  simple schemas. Each simple schema contains a set of simple connectivities where each connectivity is in conjunction to another. (3) *Schema Pruning Phase*: The third and the final stage of our method is about *pruning* the set of simple web schemas. The schema pruning process serves four important purposes: Firstly, it identifies the set of web tuples bound by each simple schema. Note that the schema decomposition process only generates a set of simple schemas. It does not relate each of these schemas to the corresponding set of web tuples. Secondly, it eliminates noisy simple schemas from the decomposed simple schemas. Next, it ensures that the number of web schemas binding the web tuples does not exceed the total number of web tuples in the web table. Lastly, it ensures that each web tuple is bound by only one simple web schema. Note that the first two stages are performed without inspecting the data instances or web tuples, whereas, the final stage requires us to inspect the web tuples for performing pruning operation. The reader may refer to [5] for detailed discussion on schema generation. We illustrate the schema generation process with an example.

EXAMPLE 2. Consider the complex web schema  $S$  in Example 1 and the set of web tuples it binds as depicted in Figure 1. This complex schema is the outcome of the first phase, i.e., coupling query to schema transformation. The complex schema decomposition phase then decompose  $S$  into the following set of simple web schemas:

1.  $S_a = \langle X_{n_a}, X_{l_a}, C_a, P_a \rangle$  where  $X_{n_a} = \{x, y, \#1, \#2\}$ ,  $X_{l_a} = \{e, g, -\}$ ,  $C_a = C_1$  and  $P_a = P$ .
2.  $S_b = \langle X_{n_b}, X_{l_b}, C_b, P_b \rangle$  where  $X_{n_b} = \{x, y, \#1, \#3\}$ ,  $X_{l_b} = \{e, g, -\}$ ,  $C_b = C_2$  and  $P_b = P$ .
3.  $S_c = \langle X_{n_c}, X_{l_c}, C_c, P_c \rangle$  where  $X_{n_c} = \{x, y, \#2, \#3\}$ ,  $X_{l_c} = \{e, g, -\}$ ,  $C_c = C_3$  and  $P_c = P$ .
4.  $S_d = \langle X_{n_d}, X_{l_d}, C_d, P_d \rangle$  where  $X_{n_d} = \{x, y, \#3\}$ ,  $X_{l_d} = \{e, g, -\}$ ,  $C_d = C_4$  and  $P_d = P$ .
5.  $S_e = \langle X_{n_e}, X_{l_e}, C_e, P_e \rangle$  where  $X_{n_e} = \{x, y, \#1\}$ ,  $X_{l_e} = \{e, g, -\}$ ,  $C_e = C_5$  and  $P_e = P$ .
6.  $S_f = \langle X_{n_f}, X_{l_f}, C_f, P_f \rangle$  where  $X_{n_f} = \{x, y, \#2\}$ ,  $X_{l_f} = \{e, g, -\}$ ,  $C_f = C_6$  and  $P_f = P$ .
7.  $S_g = \langle X_{n_g}, X_{l_g}, C_g, P_g \rangle$  where  $X_{n_g} = \{x, y\}$ ,  $X_{l_g} = \{e, g, -\}$ ,  $C_g = C_7$  and  $P_g = P$ .

Finally, the schema pruning operation relate these simple schemas with the set of web tuples it binds. Observe that the first, third, fifth and sixth web tuples in Figure 1 satisfy the schema  $S_g$ . However, they do not conform to the other simple schemas. On the other hand, the second and fourth web tuples conform to  $S_d$  and the last web tuple conforms to  $S_f$  only. Notice that the simple schemas  $S_a, S_b, S_c$  and  $S_e$  do not bind any web tuples. These schemas are noisy schemas and are eliminated during the schema pruning phase. Hence, we have the following set of partitions after performing schema pruning operation:  $A_d = \langle S_d, T_d \rangle$  where  $T_d = \{t_2, t_4\}$ ,  $A_f = \langle S_f, T_f \rangle$  where  $T_f = \{t_7\}$  and  $A_g = \langle S_g, T_g \rangle$  where  $T_g = \{t_1, t_3, t_5, t_6\}$ . Therefore, the web table can be represented as  $W = \langle N, \mathcal{P} \rangle$  where  $\mathcal{P} = \{A_d, A_f, A_g\}$ . ■

We now discuss how web schemas of a new web table, that results from manipulating existing web tables with a set of web operators, are generated. When a web table is generated locally by manipulating existing web table(s), the procedure of schema generation of the corresponding web table is distinct from that discussed above. Since the schemas of the input web table(s) are already transformed from the coupling queries and pruned when they were coupled from the Web, the coupling query to schema transformation and the complex schema decomposition phases are excluded in this case. Hence, the schema generation consist of the following two phases: *input schema(s) manipulation phase* and *schema pruning phase*. We now discuss these phases. In the first phase, the schema(s) and the set of resultant web tuples are generated first by manipulating the schema(s) and web tuples of the input web tables based on the *query conditions*. Note that the procedure for schema manipulation varies with the type of web operation. Once the set of simple schemas and the web tuples are computed from the schemas and tuples of the input web table(s), it may be necessary to prune these simple web schemas. The schema pruning phase is not exactly similar to that described earlier. The schema pruning operation discussed earlier has to be augmented to remove identical web schemas in the collection of simple web schemas. Recall that we do not perform removal of identical web schemas in the pruning operation for web tables generated by global web coupling operation. This is because identical schemas are never generated from a coupling query. However, such assumptions may not hold for web tables generated by local web operation such as *web project* [8]. For instance, a set of distinct web schemas may become identical after removal of some of the node or link type identifiers by a local web operation.

Finally, a simple web schema generated from a coupling query can always be represented by a directed connected acyclic graph. This is because a coupling query can be visualized as directed connected acyclic graph. At this moment we do not allow cycles in a coupling query. However, a simple schema generated by local web operations may not be connected. For instance, web project operation may generate disconnected simple schemas. Furthermore, a simple schema may also be cyclic. This may happen if we are interested in *selecting* those web tuples from a web table where each of these tuples contain a node that links back to any one of the predecessor node.

## 4. OPERATIONS ON WEB SCHEMA

We now discuss different types of operation that can be performed on web schemas.

### 4.1 Schema Goodness

*Schema goodness* operation quantifies how “good” a simple web schema is with respect to the web tuples it binds. Recall that each simple schema specifies the metadata, content and structural properties shared by some of the nodes and links in the form of predicates. It also specifies the inter-linked structure of the nodes in the web tuple set. However, it does not indicate the *goodness* of the schema with respect to the set of web tuples it binds. By goodness we mean how much *non-trivial* information related to the metadata, content and structure of the node and link objects in the tuple set is provided by the web schema. The more non-trivial information a web schema provides about its data instances, the more useful it is with respect to query evaluation and query formulation.

Note that the information provided by the predicates on the node and link type identifiers regarding the metadata, content and structure of the nodes and link objects may not be the optimum information the schema can provide about its data instances. We justify reasons behind this. Recall that the predicates in a simple web schema are actually defined by a user in a coupling query. There are two major factors that influence the predicates defined by the user: the information the user is looking for and his/her perception of the content and structure of the relevant web sites. For example, consider the following predicate in a schema:

$$p_1(b) \equiv \text{CONTENT}::b[\text{title}] \text{NON\_ATTR\_CONT} \\ \text{"Genetic Disorder + :END\_STR:"}$$

For simplicity, we assume that there is only one node type identifier  $b$  in the schema. That is, the schema can be expressed as  $S = \langle X_n, X_\ell, C, P \rangle$  where  $X_n = \{b\}$ ,  $X_\ell = \emptyset$ ,  $C = \emptyset$  and  $P = \{p_1\}$ . This predicate specifies that the title of the instances of  $b$  must contain the string “genetic disorder”. Of course, the title of these documents may also contain any other arbitrary string. But the question is whether this is the optimum information the web schema can provide regarding the instances of  $b$ ? We argue that this may not be true always. By inspecting the metadata, content and structural properties of instances of  $b$  we may specify additional properties satisfied by all the instances of  $b$ . For instance, suppose after inspecting the nodes of type  $b$  the following common properties are discovered: (1) The title of all the instances of  $b$  is equal to “Genetic Disorder”; (2) The instances of  $b$  either belongs to the web site at [www.ninds.nih.gov](http://www.ninds.nih.gov) or at [www.genetic.org](http://www.genetic.org). These properties may be expressed by the following predicates:

$$p_2(b) \equiv \text{CONTENT}::b[\text{title}] \text{NON\_ATTR\_ENCL} \\ \text{"Genetic Disorder"}$$

$$p_3(b) \equiv \text{METADATA}::b[\text{url.server}] \text{EQUALS} \\ \text{"(www.ninds.nih.gov)|(www.genetic.org)"}$$

Intuitively, we may conclude that the amount of information provided by the above predicates about  $b$  is more than that provided by  $p_1(b)$ . Hence, the quality or *goodness* of the schema with predicates  $p_2(b)$  and  $p_3(b)$  is better than that of the schema with  $p_1(b)$ . Clearly, the predicates  $p_2(b)$  and  $p_3(b)$  are more helpful in subsequent query formulation and evaluation compared to  $p_1(b)$ . This indicates that it is nec-

essary to be able to quantify the goodness of a web schema with respect to the information provided by the predicates. This will enable us to compare whether a web schema is a better than another with respect to the set of web tuples it binds. The higher is the value of goodness of a schema then better is the schema in terms of the information it provides regarding its data instances.

To computer goodness of a simple web schema we consider the following two issues: (1)*Existence of free node and link type identifiers*: A web schema may contain free node or link type identifiers. These identifiers does not provide any information related to the metadata, content and structure of its instances in the web tuple. Hence, the more the number of free node and link type identifiers compared to bound identifiers in a web schema the less informative is the schema. This indicates that the existence of free identifiers adversely affect the schema goodness value. The effect of free identifiers is quantified by *free ratio*. (2)*Information capacity of bound identifiers*: The second metrics for measuring schema goodness is the amount of metadata, content and structural information provided by the predicates on the bound node and link type identifiers. We call this metrics as *information capacity* of the web schema. Based on these factors the schema goodness of a simple schema is computed by the following formula: Let  $I(S)$  and  $\mathcal{F}$  be the information capacity and free ratio of a simple web schema  $S = \langle X_n, X_\ell, C, P \rangle$ . Then, the schema goodness of  $S$ , denoted by  $Goodness(S)$ , is given by the following equation:  $Goodness(S) = I(S) \times (1 - \mathcal{F})$  where  $0 \leq \mathcal{F} \leq 1$  and  $0 < I(S) < 1$ . Observe that the maximum value of  $Goodness(S)$  is  $I(S)$  and occurs when  $\mathcal{F} = 0$ . That is  $Goodness(S)$  is maximum when all the identifiers in the web schema are bound. Minimum value of  $Goodness(S)$  occurs when all the identifiers in  $S$  are free, i.e.,  $\mathcal{F} = 1$ . Then,  $Goodness(S) = 0$ . Therefore,  $0 \leq Goodness(S) \leq I(S)$ . As  $I(S) < 1$ , the value of the schema goodness of a simple web schema varies between 0 and 1. The goodness value increases as  $I(S)$  increases or  $\mathcal{F}$  decreases. We have developed techniques to compute the *free ratio* and *information capacity* of a simple web schema. Due to space constraints, we do not discuss it in this paper.

### 4.2 Schema Tightness

The *schema tightness* operation is used to improve the goodness of a web schema. Note that the higher is the goodness of a simple schema the more useful it is for subsequent query formulation and evaluation. A schema tightening operation on a web table may be triggered off by specifying to improve the goodness of those simple schemas in the web table whose goodness value is less than the *tightness threshold*  $\epsilon$ . After the tightening operation the simple schemas are modified, if possible, such that their goodness is greater than or equal to the tightness threshold. Formally, let  $\mathcal{S}_\epsilon$  be the set of simple schemas in a web table such that  $Goodness(S_i) < \epsilon$  where  $S_i \in \mathcal{S}_\epsilon$  for all  $0 < i \leq |\mathcal{S}_\epsilon|$ . Then  $\mathcal{S}_\epsilon = Tightness(\mathcal{S}_\epsilon)$  returns a set of modified simple schemas  $\mathcal{S}_\epsilon$  such that  $Goodness(S_j) \geq \epsilon$  where  $S_j \in \mathcal{S}_\epsilon$  for all  $0 < j \leq |\mathcal{S}_\epsilon|$ . Currently, we are developing techniques to perform schema tightness operation.

### 4.3 Schema Search

A web warehouse may contain numerous web tables which may not be related to one another. Thus, manually search-

ing for relevant web tables to perform web operation on them becomes significantly difficult and tedious task. The *schema search* operator enables a user to retrieve a set of relevant web tables autonomously whose schema satisfies some *search conditions*. To initiate a schema search operation, the user specifies the condition(s) to be satisfied by the schema(s) of the web table(s). Since the schemas of web tables are more complex than that of relational tables, *search conditions* have to be expressed as node and link type identifiers, connectivities between the node type identifiers and/or keywords in the predicates. Formally, we define schema search operator as follows:  $\Psi(\text{search-condition}(s))$  where  $\Psi$  is the schema search operator. A user may explicitly specify any one of the search conditions or any combination of the three conditions as follows to initiate a schema search operation: (1) To retrieve web tables whose schema(s) matches with the conditions defined by the user in the form of predicates. (2) To select those web tables whose schema satisfies some user defined connectivities. (3) To restrict the number of node type identifiers in the schemas of the set of web tables to be retrieved.

#### 4.4 Schema Match

A web warehouse may contain large number of web schemas. Some of these schemas may be related to one another. Given two schemas, a user may wish to find out the *degree of similarity* between these two schemas based on some *conditions*. The degree of similarity is measured as a number varying from 0 to 1. Higher degree of similarity indicates that the two schemas are highly related to each other based on those conditions. Let  $W_i$  and  $W_j$  be two web tables with schemas  $S_i = \langle X_{i,n}, X_{i,\ell}, C_i, P_i \rangle$  and  $S_j = \langle X_{j,n}, X_{j,\ell}, C_j, P_j \rangle$  respectively. Then,  $S_i \longleftrightarrow S_j$  returns a number between 0 to 1 based on the degree similarity between  $S_i$  and  $S_j$ . The symbol  $\longleftrightarrow$  denotes the schema match operator. The conditions based on which the degree of similarity between two schemas can be measured are as follows: (1) Similarity based on number of node type identifiers in the schemas. (2) Similarity based on connectivities of  $S_i$  and  $S_j$ . (3) Similarity based on predicates as defined in the schemas  $S_i$  and  $S_j$ . Currently, we are exploring techniques for quantifying degree of similarity between two schemas.

#### 5. CONCLUSIONS

In this paper, we have reported an overview of the notion of web schema in WHOWEDA and discussed some interesting research ideas in this context. In a nutshell, a web schema provides two types of information: First, it specifies some of the common properties shared by the documents and hyperlinks in the web table with respect to their structure, meta-data and content. Secondly, a web schema(s) summarizes the hyperlink structure of these documents in the web table. Our method of schema construction begins with manipulating a coupling query or schema(s) of input web table(s) (without looking at the data instances) and then refine the set of schemas, if possible, by inferring from the web tuples. As ongoing work, in addition to investigating different operations on web schemas as discussed in the preceding section, we are investigating the problem of reducing the number of simple schemas compared to the number of web tuples in a web table. Note if there are  $n$  web tuples then there can be at most  $n$  simple schemas. We are developing techniques to reduce the number of schemas in a web table if it exceeds

certain *threshold*. For instance, if two simple schemas  $S_1$  and  $S_2$  share high degree of similarity then it is judicious to combine these schemas into a single schema.

#### 6. REFERENCES

- [1] A. SILBERSCHATZ, STAN SDONIK. Database Systems - Breaking Out of the Box. *ACM Computing Surveys*, December 1996.
- [2] 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.
- [3] C. BEERI, T. MILO. Schemas for integration and translational of structured and semi-structured data. *Proceedings of the International Conference on Database Theory*, 1999.
- [4] S. S. BHOWMICK, W.-K. NG, S. MADRIA. Representing Web Data in a Web Warehouse. *Technical Report*, CAIS-TR-4-00, School of Applied Science, Nanyang Technological University, Singapore, 2000. *Submitted for publication*.
- [5] 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.
- [6] 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.
- [7] S. S. BHOWMICK, W.-K. NG, S. MADRIA. Web Warehousing: In the Search of Veritable El Dorado on the Web. *Submitted for publication*.
- [8] 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.
- [9] P. BUNEMAN, S. DAVIDSON, M. FERNANDEZ, D. SUCIU. Adding Structure to Unstructured Data. *Proceedings of the International Conference on Database Theory*, Delphi, Greece, 1997.
- [10] R. GOLDMAN, J. WIDOM. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. *Proceedings of the 23th International Conference on Very Large Data Bases*, Athens, 1997.
- [11] R. GOLDMAN, J. WIDOM. Approximate DataGuides. *Proceedings of Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, Jerusalem, Israel, 1999.
- [12] A. K. LUAH, W.-K. NG, E.-P. LIM. Locating Web Information Using Web Checkpoints. *Proceedings of the International Workshop on Internet Data Management (IDM'99)*, held in conjunction with the 10th International Conference on Database and Expert System Applications (DEXA'99), Florence, Italy, August 30-September 3, 1999.
- [13] T. MILO, S. ZOHAR. Using Schema Matching to Simplify Heterogeneous Data Translation. *Proceedings of the 24th International Conference on Very Large Data Bases*, 1998.
- [14] S. NESTOROV, S. ABITEBOUL, R. MOTWANI. Extracting Schema from Semistructured Data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, 1998.
- [15] 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.