Representing and Querying Semistructured Web Data Using Nested Tables with Structural Variants*

Altigran S. da Silva[1,†] Irna M.R.Evangelista Filha[1]

Alberto H. F. Laender[1] David W. Embley[2]

[1] Department of Computer Science
Federal University of Minas Gerais
31270-901 - Belo Horizonte MG - Brazil
{alti,imre,laender}@dcc.ufmg.br

[2]Department of Computer Science
Brigham Young University
Provo, Utah 84602 USA
embley@cs.byu.edu

Abstract

This paper proposes an approach to representing and querying semistructured Web data. The proposed approach is based on nested tables, which may have internal nested structural variations to accommodate semistructured data. Our motivation is to reduce the complexity found in typical query languages for semistructured data and to provide users with an alternative for quickly querying data obtained from multiple-record Web pages. We show the feasibility of our proposal by developing a prototype for a graphical query interface called QSByE (*Querying Semistructured data By Example*). For QSByE, we define a particular variation of nested tables and propose a set of QBE-like operations that extends typical nested-relational-algebra operations to handle semistructured data. We show examples of how users can pose interesting queries using QSByE.

1 Introduction

In recent years, the demand for technologies to manage data available on the Web has increased considerably. Motivated by such a demand, several researchers have proposed data models for representing [4, 27] and query languages for manipulating [2, 4, 5] Web data. These proposals generally adapt database techniques for dealing efficiently and flexibly with the particularities of this kind of data [16]. The advent of XML [31] has also offered a perspective on Web data management. Indeed, XML has been largely used as a data model for representing semistructured data in general [25], and Web data in particular. This has led to the development of several systems, query languages, and tools, all to deal with XML objects as data containers.

Unfortunately, the inherent freedom common in generic semistructured data models and XML inhibits the deployment of metaphors and paradigms like the ones that have been used extensively in typical data management tasks. For instance, the problem of storing arbitrary XML objects in relational databases involves finding regular structures in XML files—a problem known to be NP-complete [11]. More to the point of this paper, XML query languages [7, 10] have become much more complex, making it even more difficult for less skilled users to tap into the newly available and rapidly growing body of data on the Web.

*This work was partially supported by Project SIAM (MCT/CNPq/PRONEX grant number 76.97.1016.00) and by CNPq (grant number 467775/00-1). The first and second authors are supported by scholarships from CAPES. The fourth author is supported by NSF (grant number IIS-0083127).

†On leave from the University of Amazonas, Brazil.

These problems have motivated work in the management of semistructured data where flexibility is restrained in favor of simplicity. STORED [11], for example, is a query language that maps a semistructured data model to the relational data model. Since semistructured data does not usually fit in a regular structure, STORED uses an "overflow" graph to accommodate irregular portions of the data. Buneman et al. propose in [6] a data model that is more limited than the usual semistructured data models. Their goal is to provide a data modeling process for semistructured data that is as close as possible to conventional database systems. Dataguides [17], an interactive tool for querying OEM [27] databases, presents users with a dynamic structural summary of semistructured data. The goal is to reduce the level of complexity in query formulation by exempting users from dealing (possibly unnecessarily) with the entire structure of the data to be queried.

In harmony with the work of others that favors simplicity over flexibility, we propose in this paper the use of nested tables allowing internal structural variations for representing and querying semistructured Web data, formalizing the ideas introduced in [14, 21]. We show throughout the paper how nested tables can be nicely adapted for this kind of data and can provide a simple and intuitive representation close to record-based database representations. As Makinouch [24] explains, nested tables naturally accommodate regular hierarchical data. The distinction, and main contribution, of our proposal is that it also allows the representation of variations and irregularities typical of semistructured data.

As an example, consider the data implicitly present in Web pages generated as a result of a query in the main page of the Amazon.com Web site. Figure 1 shows an example of such a page, for the Brazilian composer Antonio Carlos Jobim. In Figure 2, we present a nested table containing data of interest extracted from this page. Observe that the information about items from each Amazon.com store is different. For instance, in row 1, where the value of Store is "Popular Music", the information consists of Item, By, and Format, whereas in row 5, where the value of Store is "Auctions", the information consists of Item, Bid, and Time.

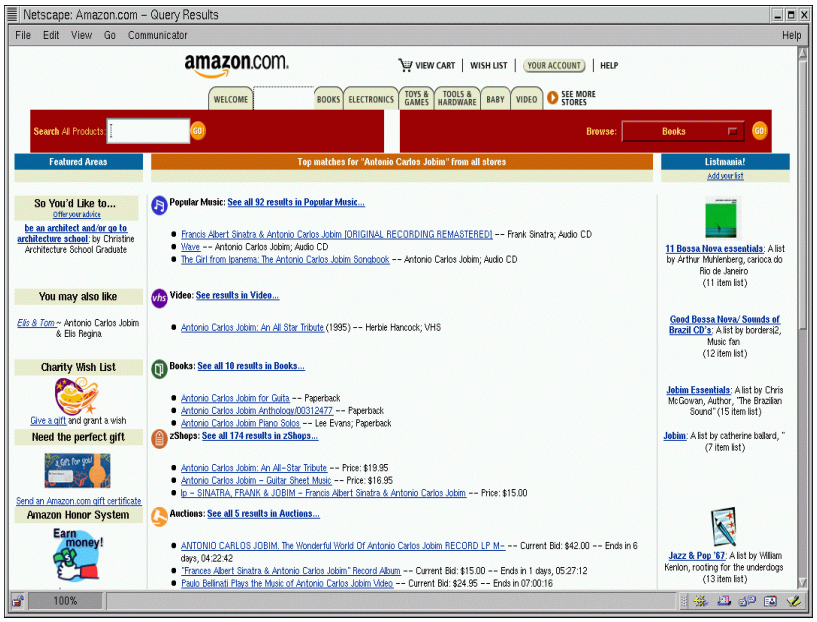
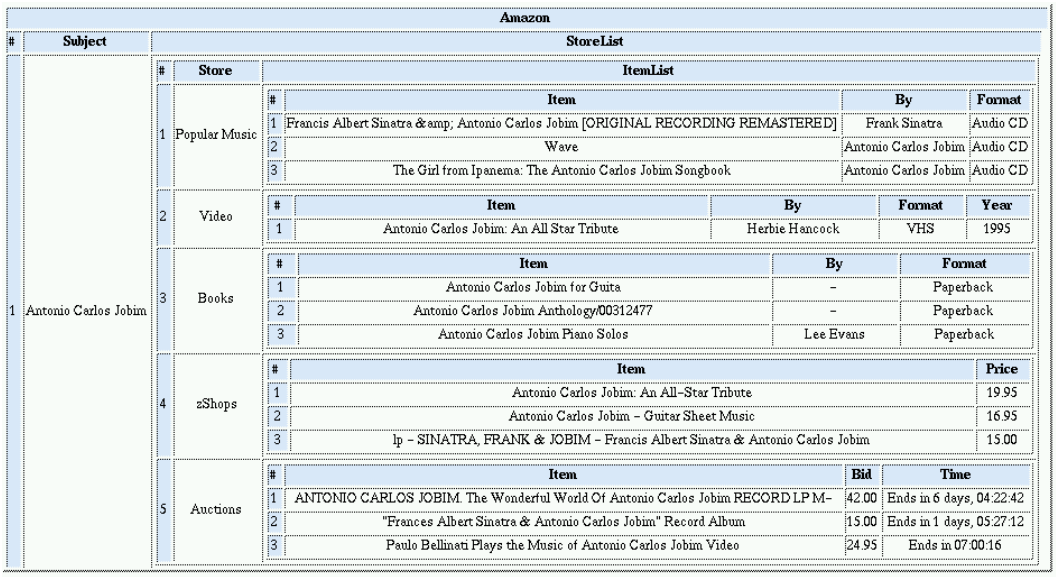


Figure 1: Excerpt of a page from the Amazon.com Web site.

In addition to formally defining nested tables with internal variations, this paper also formally defines a set of query operations for semistructured data represented as an internally varying nested table. We base this set of operations on several proposals found in the literature [1, 8, 18, 29, 30], particularly on the algebra proposed by Colby [8]. As a further contribution, this paper



Amazon

#	Subject	StoreList
1	Antonio Carlos Jobim	

1 Popular Music

#	Item	By	Format
1	Francis Albert Sinatra & Antonio Carlos Jobim [ORIGINAL RECORDING REMASTERED]	Frank Sinatra	Audio CD
2	Wave	Antonio Carlos Jobim	Audio CD
3	The Girl from Ipanema: The Antonio Carlos Jobim Songbook	Antonio Carlos Jobim	Audio CD

2 Video

#	Item	By	Format	Year
1	Antonio Carlos Jobim: An All Star Tribute	Herbie Hancock	VHS	1995

3 Books

#	Item	By	Format
1	Antonio Carlos Jobim for Guita	-	Paperback
2	Antonio Carlos Jobim Anthology/00312477	-	Paperback
3	Antonio Carlos Jobim Piano Solos	Lee Evans	Paperback

4 zShops

#	Item	Price
1	Antonio Carlos Jobim: An All-Star Tribute	19.95
2	Antonio Carlos Jobim - Guitar Sheet Music	16.95
3	lp - SINATRA, FRANK & JOBIM - Francis Albert Sinatra & Antonio Carlos Jobim	15.00

5 Auctions

#	Item	Bid	Time
1	ANTONIO CARLOS JOBIM. The Wonderful World Of Antonio Carlos Jobim RECORD LP M-	42.00	Ends in 6 days, 04:22:42
2	"Frances Albert Sinatra & Antonio Carlos Jobim" Record Album	15.00	Ends in 1 days, 05:27:12
3	Paulo Bellinati Plays the Music of Antonio Carlos Jobim Video	24.95	Ends in 07:00:16

Figure 2: Example of a nested table.

also describes QSByE (*Querying Semistructured data By Example*) [13, 15], which shows the feasibility of using nested tables for representing and querying semistructured Web data*. QSByE combines features of QBE (*Query By Example*) [32] with typical features of query languages for semistructured data [2, 5]. As implemented, QSByE provides the structure of the data as a nested table skeleton [23] so that users do not have to uncover the structure of the data by themselves. It also lets users restructure provided nested tables to their own liking.

Despite the relative simplicity of dealing with semistructured data in the form of nested tables, it is easy to see that such a representation is not as expressive as general semistructured data models or XML. We cannot, for example, have different structures at the top level. Thus, we sacrifice some flexibility for greatly increased simplicity. However, in this work we are mainly concerned with representing data in Web pages, like the one in Figure 1. This kind of page is said to be a *data rich*, *ontological narrow*, *multiple-record* Web page [12]. Examples of such pages are found in Web sites such as bookstores, electronic catalogs, travel agencies, and classified ads and include pages composed of data whose overall structure is naturally hierarchical, but exhibits a modest degree of variation. In particular, QSByE makes it possible to manipulate data resulting from the extraction of data from these kinds of Web pages by DEByE (*Data Extraction By Example*) [21, 28], a tool that extracts data from multiple-record Web pages and organizes the extracted data in the form of nested tables.

The remainder of the paper is organized as follows. In Section 2 we present the terminology and modeling concepts used to represent semistructured data in the form of nested tables. In Section 3, we discuss the expressiveness of nested tables as a model for representing semistructured Web data. In Section 4 we define the set of operations used to query nested tables that comply with our definitions. In Section 5, we describe QSByE and illustrate how it can be used by showing some examples. Finally, we present our conclusions and an interesting direction for future work in Section 6.

*Previous query languages for nested tables, such as QBEN [18], have been defined, but none deal with semistructured data as does QSByE, as described in this paper.

2 Basic Concepts and Notation

In this section, we present the data-modeling concepts we adopt for representing semistructured Web data. These concepts are based on the notion of *nested tables* [24], augmented with the concept of *variants* [22]. As we discuss in Section 5, this kind of representation allows us, for instance, to adapt the QBE paradigm for querying this kind of data.

The main problem we have in achieving this goal is dealing with irregularities typical of semistructured data. As a solution, we propose a generalization of regular nested tables in which we allow a column to have two or more distinct substructures. An example of this solution is presented in the nested table in Figure 2. Note that the internal structures of the objects in the column ItemList are distinct for each of the Store rows.

In what follows, we characterize these ideas more precisely. We begin by defining the notion of a *table scheme*.

Definition 1 *A **table scheme** τ is defined using the notation $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}], C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}]), (m \geq 2,\ n_k \geq 1,\ k = 1, \ldots, m)$, where C_j is called a **column** and τ_j^i denotes exactly one of the following: (i) an atomic value, represented by* atom*, (ii) a set of atomic values, represented by* {atom} *or (iii) a table scheme. For the sake of simplifying the notation, if $n_j = 1$, we can use $C_j : \tau_j^1$ instead of $C_j : [\tau_j^j]$.*

Intuitively, a table scheme describes the structure of a kind of nested table in which a column C_j may store "values" or objects with distinct structure in distinct tuples. The structures of the possible objects are given by the alternatives $\tau_j^1, \ldots, \tau_j^{n_j}$ which can be either atomic values, lists of atomic values, or other nested tables.

Consider the page excerpt illustrated in Figure 1. The structure of the objects implicitly present can be described as follows:

$$\tau = (\text{Subject:atom,StoreList:(Store:atom,ItemList:}[\tau_2^1;\tau_2^2;\tau_2^3]))$$
$$\tau_2^1 = (\text{Item:atom,By:atom;Format:atom;Year:atom})$$
$$\tau_2^2 = (\text{Item:atom,Price:atom}) \quad \tau_2^3 = (\text{Item:atom,Bid:atom,Time:atom})$$

The nested table Amazon in Figure 2 is an instance of the table scheme τ defined above. In this table scheme, for the first level two columns are defined: Subject and StoreList. For the column Subject, only atomic values are allowed, while the column StoreList stores nested tables. For these inner tables, we have a column Store, whose values are atomic, and a column ItemList with three distinct possible structures (nested tables), each one corresponding to a type of store in Figure 1. We next define precisely the notion of an instance of a table scheme.

Definition 2 *Let $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}],\ C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}]),\ (m \geq 2,\ n_k \geq 1,\ k = 1 \ldots, m)$, be a table scheme. An **instance** T of τ, denoted by $T : \tau$, is a set $T = \{\langle C_1 : v_1^1, C_2 : v_2^1, \ldots, C_m : v_m^1\rangle, \ldots, \langle C_1 : v_1^n, C_2 : v_2^n, \ldots, C_m : v_m^n\rangle\}, (n \geq 0)$, where v_j^k is: (i) an atomic value, if any $\tau_j^i =$* atom*, (ii) a list of atomic values, if any $\tau_j^i =$* {atom}*, or (iii) an instance of any τ_j^i that is a table scheme. An instance of a table scheme is referred to as a **table**.*

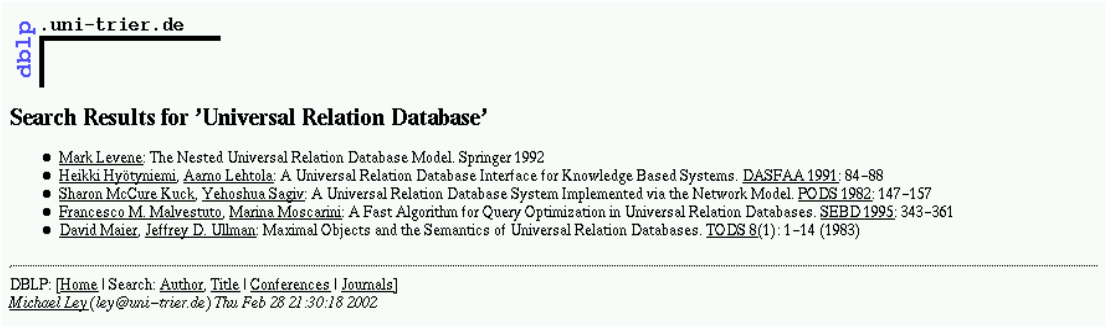
According to the notation introduced in Definition 2, a possible instance of our example table scheme of is as follows:

$$\text{Amazon} = \{\langle \text{Subject} : \text{“Antonio Carlos Jobim”}, \langle \text{StoreList} : S\rangle\}$$
$$S = \{\langle \text{Store} : \text{“Popular Music”}, \text{ItemList} : I_1\rangle, \ldots, \langle \text{Store} : \text{“Auctions”}, \text{ItemList} : I_5\rangle\}$$
$$I_1 = \{\langle \text{Item} : \text{“Francis Albert...”}, \text{By} : \text{“Frank Sinatra”}, \text{Format} : \text{“Audio CD”}\rangle, \ldots\}$$
$$\ldots$$
$$I_5 = \{\langle \text{Item} : \text{“ANTONIO CARLOS ...”}, \text{Bid} : \text{“42.00”}, \text{Time} : \text{“Ends in 6 days, 04:22:42”}\rangle, \ldots\}$$

Observe that the notation above incorporates structural information along with the data itself; thus we have a self describing representation for semistructured data. As a consequence, instead of using this notation, we could easily describe such data by means of XML, as we actually do in our data extraction tool DEByE and in the QSByE interface discussed in Section 5. A formal description of a possible XML representation, however, is beyond the scope of this paper, and we omit it.

3 Expressiveness of Nested Tables for Representing Semistructured Web Data

In this section, we discuss the expressiveness of nested tables as a data model for representing semistructured Web data. In particular, we make a brief comparison between our nested-tables model and typical semistructured data models. For the discussion that follows, consider the Web page resulting from the query "Universal Relation Database" in the DBLP Web site, which is shown in Figure 3.


dblp.uni-trier.de

Search Results for 'Universal Relation Database'

- Mark Levene: The Nested Universal Relation Database Model. Springer 1992
- Heikki Hyötyniemi, Aarno Lehtola: A Universal Relation Database Interface for Knowledge Based Systems. DASFAA 1991: 84–88
- Sharon McCure Kuck, Yehoshua Sagiv: A Universal Relation Database System Implemented via the Network Model. PODS 1982: 147–157
- Francesco M. Malvestuto, Marina Moscarini: A Fast Algorithm for Query Optimization in Universal Relation Databases. SEBD 1995: 343–361
- David Maier, Jeffrey D. Ullman: Maximal Objects and the Semantics of Universal Relation Databases. TODS 8(1): 1–14 (1983)

DBLP: [Home | Search: Author, Title | Conferences | Journals]
Michael Ley (ley@uni-trier.de) Thu Feb 28 21:30:18 2002

Figure 3: A sample Web page from DBLP.

Figures 4 and 5 show the data extracted from this page organized into two distinct labeled trees according to OEM, a well known semistructured data model [27]. In the following discussion, we refer to these trees as M and N, respectively.

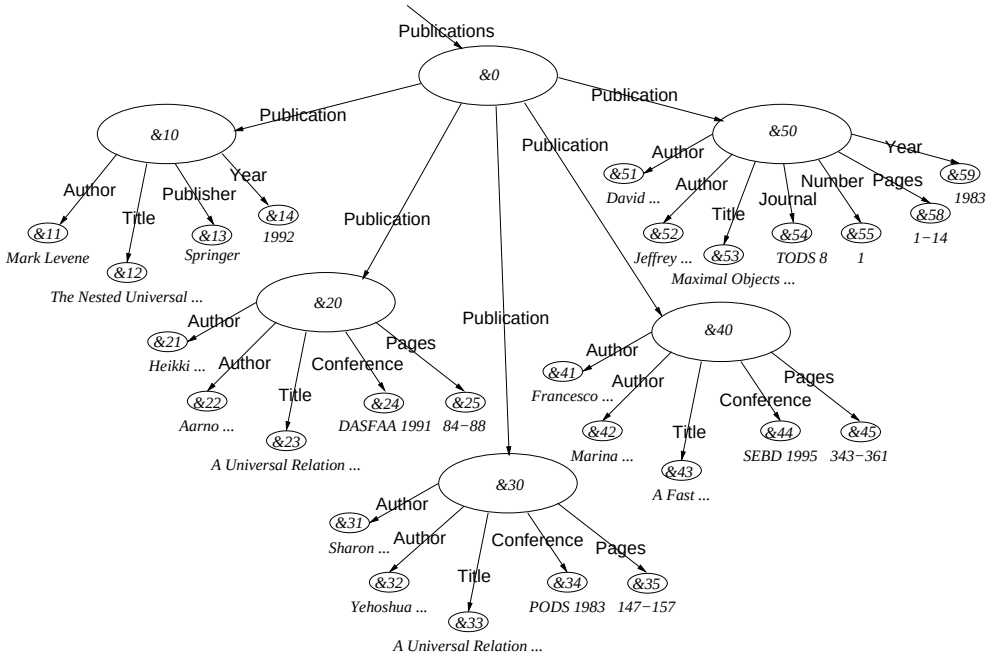


Figure 4: An OEM tree for the data in the page of Figure 3.

Trees M and N can be considered as semistructured databases and, intuitively, they are equivalent, since the relationship between atomic values is maintained. However, while in M each Publication subtree is composed of distinct atomic components, in N we introduce two additional

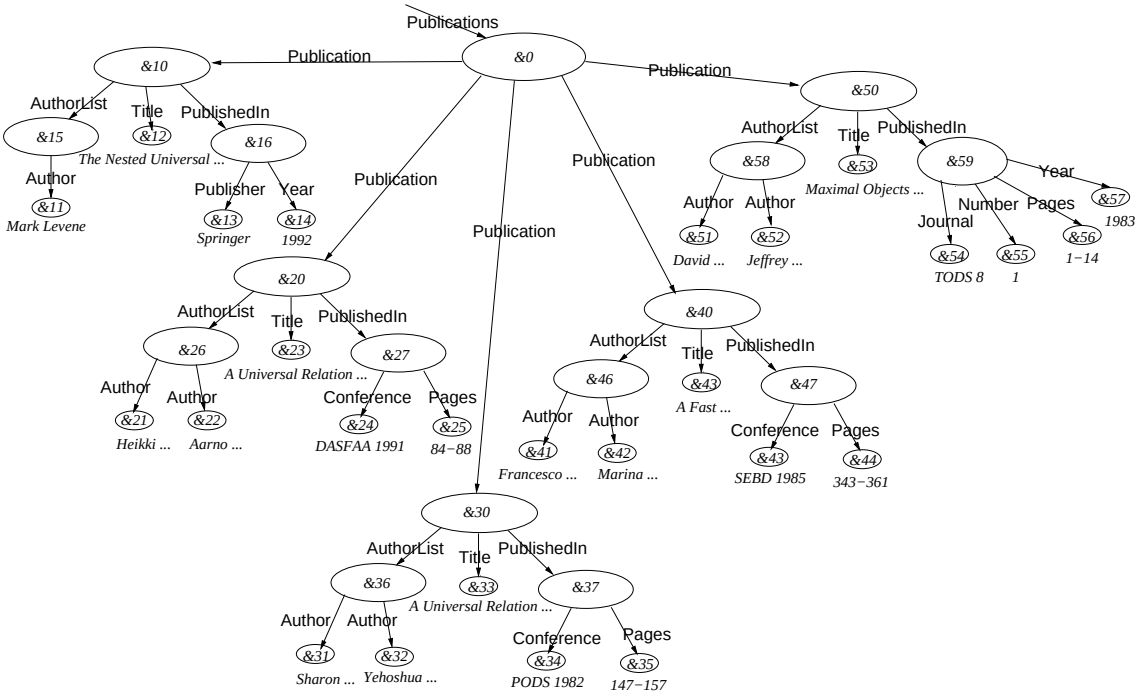


Figure 5: An alternative OEM tree for the data in the page of Figure 3.

nodes (AuthorList and PublishedIn) in Publication sub-trees with the goal of making these sub-trees uniform in their first levels. This alternative representation preserves the semantics of the objects, but it is less concise than the first one. On the other hand, for our purposes, N presents an important property: it can be directly mapped into a nested table, such as the one presented in Figure 6.

Publication			
#	AuthorList	Title	PublishedIn
1	Mark Levene	The Nested Universal Relation Database Mode	# | Publisher | Year 1 | Springer | 1992
2	Heikki Hyötyniemi Aarno Lehtola	A Universal Relation Database Interface for Knowledge Based Systems	# | Conference | Pages 1 | DASFAA 1991 | 84–88
3	Sharon McCure Kuck Yehoshua Sagiv	A Universal Relation Database System Implemented via the Network Model	# | Conference | Pages 1 | PODS 1982 | 147–157
4	Francesco M. Malvestuto Marina Moscarini	A Fast Algorithm for Query Optimization in Universal Relation Databases	# | Conference | Pages 1 | SEBD 1995 | 343–361
5	David Maier Jeffrey D. Ullman	Maximal Objects and the Semantics of Universal Relation Databases	# | Journal | Number | Pages | Year 1 | TODS 8 | 1 | 1–14 | 1983

Figure 6: Data from the DBLP page of Figure 3 organized into a nested table.

The table in Figure 6 makes explicit an interesting characteristic of nested tables for the representation of semistructured Web data. Traditionally, nestings have the role of representing in a single column complex objects, i.e., non-atomic values. In our approach, we "overload" this structural feature by using it to also accommodate structural variations. This is what happens for the column PublishedIn, in which rows 1 and 5 store tables that have a structure distinct from the structure of the tables stored in rows 2, 3 and 4. Notice that, for the case of this specific example, each row corresponds to a single publication. Thus, this representation can be considered

imprecise, since all tables stored under PublishedIn will actually have one single tuple each. ¿From this simple example, we can conclude that nested tables are indeed less expressive than OEM for representing semistructured data, but in situations where typical Web data is to be represented, nested tables are a viable alternative.

To go further in this discussion, we now present a brief comparison with XML, which is currently the predominant formalism for representing Web data. We notice that most of the discussion we have presented so far in this section also applies to XML, since it is, essentially, a notation for representing labeled trees.

In Figure 7(a) we present a DTD that declares the structure of an XML document corresponding to the labeled tree M of Figure 4. Similarly, in Figure 7(b) we present a DTD that declares the structure of an XML document corresponding to the labeled tree N of Figure 5. Let us refer to these DTDs as D_M and D_N, respectively.

```
<!DOCTYPE dpub [
 <!ELEMENT Publications (Publication*)>
 <!ELEMENT Publication  (Author*, Title,
           ((Publisher,Year)|
           (Conference,Pages)|
           (Journal,Number,Pages,Year))>
 <!ELEMENT Publisher    (#PCDATA)>
 <!ELEMENT Author       (#PCDATA)>
 <!ELEMENT Title        (#PCDATA)>
 <!ELEMENT Conference   (#PCDATA)>
 <!ELEMENT Journal      (#PCDATA)>
 <!ELEMENT Number       (#PCDATA)>
 <!ELEMENT Pages        (#PCDATA)>
 <!ELEMENT Year         (#PCDATA)>
]>
```

(a)

```
<!DOCTYPE dpub [
 <!ELEMENT Publications (Publication*)>
 <!ELEMENT Publication  (AuthorList,Title,PublishedIn)>
 <!ELEMENT Authorlist   (Author*)>
 <!ELEMENT PublishedIn  (PublishedIn1|
                         PublishedIn2|
                         PublishedIn3)>
 <!ELEMENT PublishedIn1 (Publisher,Year)>
 <!ELEMENT PublishedIn2 (Conference,Pages)>
 <!ELEMENT PublissheIn3 (Journal,Number,Pages,Year)>
 <!ELEMENT Title        (#PCDATA)>
 <!ELEMENT Author       (#PCDATA)>
 <!ELEMENT Publisher    (#PCDATA)>
 <!ELEMENT Year         (#PCDATA)>
 <!ELEMENT Conference   (#PCDATA)>
 <!ELEMENT Pages        (#PCDATA)>
 <!ELEMENT Journal      (#PCDATA)>
 <!ELEMENT Number       (#PCDATA)>
]>
```

(b)

Figure 7: Two DTDs for XML documents storing data extracted from the DBLP page of Figure 3.

Notice that D_M and D_N define XML documents that are equivalent in the same sense as trees M and N are. Considering that DTDs are indeed context-free grammars [3] and that XML documents (or OEM labeled trees) are derivations of such grammars, we can see D_N as the grammar that results from including in D_M a number of productions (or `ELEMENT` declarations) to ensure that the resulting documents or trees take a form similar to N and, thus, can be mapped to nested tables. More precisely, such trees would be considered as tables, according to Definition 2.

Indeed, nested tables are less expressive than XML for representing Web data precisely because they can be described by a sub-class of DTDs such as D_N, which we refer to as *Tabular DTDs* or *TDTDs*. In DTDs, `ELEMENT` declarations are restricted to some pre-defined forms that guarantee that XML documents (or labeled trees) correspond to nested tables. In particular, TDTDs non-terminal `ELEMENT` declarations are restricted to be of one of the following forms.

- An ***aggregating*** (or ***tuple-generating***) declaration has the form `<!ELEMENT` X_0 $(X_1 \ldots X_n)$`>` ($n \geq 2$), where $X_i \neq X_j$, for every $0 \leq i, j \leq n$ except $i = j$. Further, each X_k, $k = 1 \ldots n$, must appear on the left-hand side of exactly one iterating or terminal declaration;
- An ***iterating*** (or ***list-generating***) declaration has the form `<!ELEMENT` X $(Y*)$`>`, where $X \neq Y$. Further, Y must appear on the left-hand side of exactly one aggregating, varying, or terminal declaration;

- A ***varying*** (or ***variant-generating***) declaration has the form <!ELEMENT X_0 $(X_1|\ldots|X_n)$> ($n \geq 2$), where $X_i \neq X_j$, for every $0 \leq i,j \leq n$ except $i = j$. Further, each X_k, $k = 1 \ldots n$, must appear on the left-hand side of exactly one aggregating or iterating declaration.

It is easy to see that limiting the possible ELEMENT declarations as described above considerably restricts the possible derivations (i.e., labeled trees or XML documents) that can be generated. However, it must be observed that formats such as XML are intentionally non-restrictive, since they do not aim any application in particular. Indeed, XML can be used to represented typical Web data, such as the data found in pages of Figures 1 and 3, but it is also flexible enough to describe, for instance, DNA sequences, communication protocols or stylesheets. In this paper, we claim that, for representing data typically found in *data rich, ontological narrow, multiple-record* Web pages, it is possible to use nested tables as we define them, without compromising the accuracy of the representation. Indeed, despite their relative lack of expressiveness, nested tables are expressive enough to represent a vast collection of different data available in Web pages, such as those in Figures 1 and 3. As additional evidence, much of the recent work on data extraction [9, 19, 20, 26] confirms that nested tables are an effective paradigm for describing Web data.

4 Query Operations for Nested Tables

In this section we present a set of operations for querying semistructured Web data represented as nested tables. Other operations for manipulating table schemes are omitted here but are part of a preliminary proposal we made and are described in [21].

The query operations presented here are *selection, projection, nest,* and *unnest.* The particular version of these operators can be regarded as extensions of the recursive query operations proposed by Colby [8] to query data in regular nested tables. In his work, Colby defines a recursive nested relational algebra where every query operation can be specified directly over attributes at any nesting level of a nested table. We extended these operations to deal with semistructured data so that these operations adequately deal with irregularities such as structural variations and absence of values. We note that the operations presented here do not form a complete operation set; that is, they do not provide an expressiveness equivalent to relational algebra. As future work, we can extended this set and make it complete.

In the following, we describe each operation and present its formal definition. For the remainder of this section, we assume that all operations apply to an instance of a table scheme: $\tau = (C_1 : [\tau_1^1; \ldots; \tau_1^{n_1}], C_2 : [\tau_2^1; \ldots; \tau_2^{n_2}], \ldots, C_m : [\tau_m^1; \ldots; \tau_m^{n_m}])$ $(m \geq 2,\ n_k \geq 1,\ k = 1, \ldots, m)$. Before proceeding, we define two necessary auxiliary functions.

Definition 3 *We denote by $\Upsilon(T)$ a function that when applied to table $T : \tau$ returns its table scheme τ. Otherwise, if T is not a table, it returns a null value.*

Definition 4 *Let $t = \langle C_1 : v_1, ..., c_m : v_m \rangle$ be a tuple of a table $T : \tau$. Let $\{D_1 : u_1, ..., D_p : u_p\} \subseteq \{C_1 : v_1, ..., c_m : v_m\}$ be a subset of the columns of T. We use the notation $t[D_1, ..., D_p]$ to refer to tuple $\langle D_1 : u_1, ..., D_p : u_p \rangle$, composed of the values of the columns specified in t.*

Informally, Definition 4 captures the notion of a "sub-tuple" or "restriction" of a given tuple. This notation is common in the context of the traditional relational model.

Selection (σ). The selection operation allows selection conditions to be specified at any level of nesting. A selection condition is a boolean expression defined over the value in a column or over the existence of a column in an internal subtable (i.e., a *structural condition*). The operators used in selection conditions are $\{\supset, \supseteq, \in, <, >, =, \sim, *\}$. The symbols $\sim$ and $*$ are used to denote the pattern matching operator and the structural condition, respectively. The other symbols correspond to the operators of the nested relational algebra.

The general form of the selection operation is $\sigma_{sel-cond}(T)$, where *sel-cond* represents a selection condition specified on an instance T of a table scheme τ. To formally define the selection operation, we first consider the following definition.

Definition 5 *A selection condition* S *specified on a table scheme* τ *is of the form* $\mathrm{s} + \mathsf{L}$*, where* s *represents the condition specified over the columns of* τ *and* L *is a list.* L *is called an* **internal list of conditions** *of* τ*, if and only if, (i)* L *is empty, or (ii)* L *is of the form* $\{s_1 + C'_1.\mathsf{L}_1; s_2 + C'_2.\mathsf{L}_2; \ldots; s_l + C'_l.\mathsf{L}_l\}$*, where* $\{C'_1, C'_2, \ldots, C'_l\} \subseteq \{C_1, C_2, \ldots, C_m\}$ *and* $\forall C'_i$ $(0 \leq i \leq l)$*,* $\Upsilon(C'_i)$ *is a table scheme, each* s_i *is a condition specified on* $\Upsilon(C'_i)$*, and* L_i *is an internal list of conditions of* $\Upsilon(C_i\prime)$*.*

Intuitively, each element $s_i + C'_i.\mathsf{L}_i$ of an internal list of conditions limits the scope of a selection condition to C'_i.

As an example, assume that we are interested in instances of the table in Figure 2 that represent popular music items by "Antonio Carlos Jobim". A possible internal list for conditions of the table Amazon is Store = *Popular Music* + {ItemList.By $\sim$ *Antonio Carlos Jobim*}, where the condition involves the column Store and an internal list of conditions is recursively specified on the column ItemList.

For the condition s_i to be satisfied by a row of the table, the row must contain all the columns involved in s_i and the corresponding boolean expression must evaluate to true. Based on this, we present the definition of the selection operation as follows:

Definition 6 *Let* T *be a table such that* $T : \tau$*,* L *an internal list of conditions for* τ*, and* S *a selection condition. The selection operation* $\sigma_{\mathsf{S}}(T)$ *is defined as follows:*

(i) if L *is empty, then* $\sigma_s(T) = \{t \in T \mid s(t) = \mathsf{true}\ \}$.
(ii) if L *is not empty, then* $\sigma_{s+\{s_1+C'_1.\mathsf{L}_1;\ldots;s_l+C'_l.\mathsf{L}_l\}}(T) = \{t \mid \exists t_r \in T \wedge$
$(t[\{C_1, \ldots, C_m\} - \{C'_1, \ldots, C'_l\}] = t_r[\{C_1, \ldots, C_m\} - \{C'_1, \ldots, C'_l\}] \wedge$
$(s(t_r) = \mathsf{true} \wedge ((t[C'_1] = \sigma_{s_1+C'_1.\mathsf{L}_1}(t_r[C'_1])) \wedge (t[C'_1] \neq \emptyset)) \wedge \ldots \wedge$
$((t[C'_l] = \sigma_{s_l+C'_l.\mathsf{L}_l}(t_r[C'_l])) \wedge (t[C'_l] \neq \emptyset)))\}$

According to Definition 6, selection conditions are evaluated throughout the nested table, so that tables at any level of nesting can be addressed. In general, the selection operation generates a table with the same structure as the original table. Because of the semistructured nature of the queried data, however, the selection operation can generate a table whose structure is distinct from the structure of the original table. For instance, if we have specified a selection operation on the attribute Bid for the table of Figure 2, only rows of the nested table ItemList that have this attribute would be considered. All other structural variations for the column ItemList would be discarded. As a consequence, the resultant table would bare a structure distinct from the original table.

Projection (π). The projection operation is similar to the projection operation defined in the relational algebra. We can say that this operation horizontally reduces a table by maintaining only the columns specified by the user (referred to as *projected columns*). The general form of the projection operation is $\pi_{col-set}(T)$, where *col-set* represents the set of projected columns of an instance T of τ. To formally define the projection operation, we first consider the following definition.

Definition 7 *A set of projected columns* A *is of the form* $\mathsf{A}_\tau + \mathsf{P}$*, where* A_τ *is a subset of the columns of* τ *and* P *is a list.* P *is called an* **internal list of projections** *of* τ*, if and only if, (i)* P *is empty, or (ii)* P *is of the form* $\{\mathsf{A}_{C'_1}+C'_1.\mathsf{P}_1; \mathsf{A}_{C'_2}+C'_2.\mathsf{P}_2; \ldots; \mathsf{A}_{C'_l}+C'_l.\mathsf{P}_l\}$*, where* $\{C'_1, C'_2, \ldots, C'_l\} \subseteq \{C_1, C_2, \ldots, C_m\}$ *and,* $\forall C'_i$ $(0 \leq i \leq l)$*,* $\Upsilon(C'_i)$ *is a table scheme, each* $\mathsf{A}_{C_i\prime}$ *is a set of columns in* $\Upsilon(C'_i)$*, and* P_i *is an internal list of projections of* $\Upsilon(C_i\prime)$*.*

Similar to the internal list of conditions, we can say that each element $\mathsf{A}_{C'_i} + C'_i.\mathsf{P}_i$ of an internal list of projections determines that the instances of the columns represented by $\mathsf{A}_{C'_i}$ will be retrieved from the internal table defined in column C'_i. As an example, suppose that we want to project on the columns Store, Item, and By of the table in Figure 2. A possible internal list for these projected columns is {{Store} + ItemList.{Item,By}}, where A_τ is a set composed only of the column Store and a list of internal projections is recursively specified on the column ItemList.

Definition 8 *Let T be a table such that $\Upsilon(T) = \tau$ and P be an internal list of projections of τ. The projection operation $\pi_{\mathsf{A}}(T)$ is defined as follows:*

(i) if P is empty , then $\pi_{\mathsf{A}_\tau}(\mathsf{T}) = \{t \mid (\exists t_r \in T \wedge \mathsf{A}_\tau \subseteq \tau \wedge t = t_r[\mathsf{A}])$.
(ii) if P is not empty, then $\pi_{\mathsf{A}_\tau} + \{\mathsf{A}_{C'_1+C'_1.\mathsf{P}_1;\ldots;\mathsf{A}_{C'_l}+C'_l.\mathsf{P}_l}\}(\mathsf{T}) =$

$$\{t \mid \exists t_r \in T \wedge (t[\mathsf{A}_\tau] = \pi_{\mathsf{A}_\tau}(\mathsf{T}) \wedge t[C'_1] = \pi_{\mathsf{A}_{C'_1}+C'_1.\mathsf{P}_1}(t[C'_1]) \wedge \ldots \wedge t[C'_l] = \pi_{\mathsf{A}_{C'_l}+C'_l.\mathsf{P}_l}(t[C'_l]))\}$$

According to Definition 8, to deal with the semistructured nature of the data, the projection operation recursively verifies the existence of each projected column in all rows of the table. As a consequence of the irregularities that might occur in a nested table, the result of a projection operation sometimes includes an "artificial" nesting level to accommodate the structural variations inherent in the projected columns. For instance, consider the nested table in Figure 2. If we project on the column ItemList, we cannot group the resulting instances in a table because of the different structure in each variation. We would thus need an additional level of nesting to create a single table.

Nest (ν). Similar to the operation proposed by Thomas and Fischer [30], our nest operation has two distinct semantics. When applied to a single column, this operation groups the values in the column that are associated with equal values occurring in all other columns. Otherwise, when the nest operation is applied to a set of columns, it creates a new table scheme that groups the values of these columns and, consequently, generates a new level of nesting in the table. The general form of the nest operation is $\nu_{col-set}(T)$, where *col-set* represents a set of columns of T that will be nested.

Definition 9 *Let T be a table such that $\Upsilon(T) = \tau$ and A be a set of columns to be nested in T. The nest operation $\nu_{\mathsf{A}}(T)$ is defined as follow:*

$$\begin{aligned}\nu_{\mathsf{A}}(T) = \{t \mid \exists t_r \in T \wedge (((\mathsf{A} = \{c_1,\ldots,c_o\},\ (1 \le o \le m) \wedge \mathsf{A} \subseteq \{C_1,\ldots,C_m\}) \wedge \\ \tau_{\mathsf{A}} = (c_1 : [\tau^1_{\mathsf{A}_1};\ldots;\tau^{n_1}_{\mathsf{A}_1}], c_2 : [\tau^1_{\mathsf{A}_2};\ldots;\tau^{n_2}_{\mathsf{A}_2}],\ldots, c_o : [\tau^1_{\mathsf{A}_o};\ldots;\tau^{n_o}_{o}]) \wedge \\ ((t[\{C_1,\ldots,C_m\} - \{c_1,\ldots,c_o\}] = t_r[\{C_1,\ldots,C_m\} - \{c_1,\ldots,c_o\}]) \wedge \\ (t[\tau_{\mathsf{A}}] = \{s[c_1,\ldots,c_o] \mid s \in T \wedge \\ \qquad s[\{C_1,\ldots,C_m\} - \{c_1,\ldots,c_o\}] = \\ \qquad t_r[\{C_1,\ldots,C_m\} - \{c_1,\ldots,c_o\}]))) \vee \\ (\exists C'_k\ (1 \le k \le m) \wedge \Upsilon(C'_k) \textit{ is a table scheme} \wedge \Upsilon(C'_k) \supseteq \mathsf{A} \wedge \nu_{\mathsf{A}}(C'_k)))\}\end{aligned}$$

According to Definition 9, the nest operation is recursively applied to a nested structure. This recursion navigates the nesting structure of the table until the columns to be nested are reached. Both of the two semantics are addressed by this definition. It is important to note that when a single column is nested, the table structure does not change, but all the values of this column are grouped, eliminating repeated values in other columns. In addition, this definition shows that when the nest operation is applied to a set of columns, it can only be executed if all of the chosen columns are defined in the table scheme τ.

Unnest (μ). The unnest operation is the inverse of the nest operation. It also has two distinct semantics according to [30], and it must always be applied to a column whose contents is either a list of atoms or an internal table. If it is applied to a list of atoms, it will "ungroup" its elements, i.e. it will split them into different rows of the table. Otherwise, if it is applied to an internal table it will eliminate a level of nesting. The general form of the unnest operation is $\mu_{col}(T)$, where *col* is the column to be unnested in T.

Definition 10 *Let T be a table such that $\Upsilon(T) = \tau$ and a be a column to be unnested in T. The unnest operation $\mu_{\mathsf{a}}(T)$ is defined as follows:*

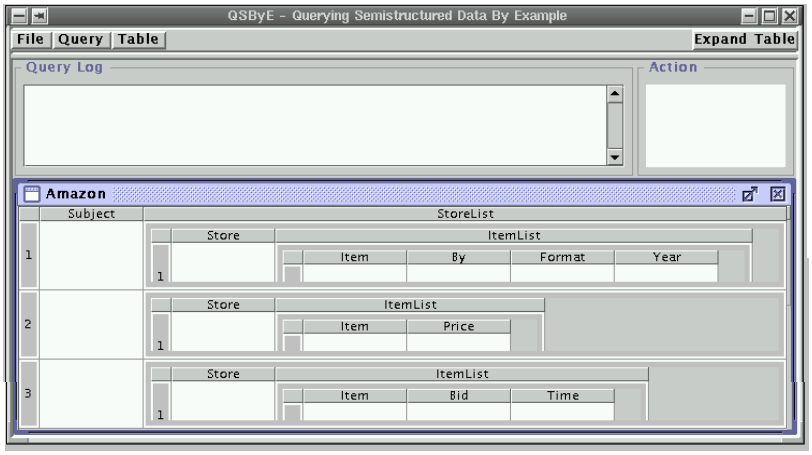


Figure 8: Skeleton for the repository used in the query examples.

$$\begin{aligned}\mu_{\mathsf{a}}(T) = \{t \mid \exists t_r \in T \wedge\\ &\mathsf{a} = C_i\ (0 \le i \le m)\ \textit{such that}\ C_i\ \textit{is not an atom} \wedge\\ &(t[C_1, C_{i-1}, C_{i+1}, C_m] = t_r[C_1, C_{i-1}, C_{i+1}, C_m] \wedge\\ &\tau_i^j = \Upsilon(C_i)\ (0 \le j \le n)\ \textit{such that}\\ &\qquad \tau_i^j = (c_1 : [\gamma_1^1; \ldots; \gamma_1^{n_1}], \ldots, c_l : [\gamma_l^1; \ldots; \gamma_l^{n_l}]) \wedge\\ &\qquad t[c_1, \ldots, c_l] = t_s\ \textit{for each}\ t_s \in C_i) \vee\\ &\exists C'_k (1 \le k \le m) \wedge \Upsilon(C'_k)\ \textit{is a table scheme} \wedge \Upsilon(C'_k) \supseteq \mathsf{A} \wedge \mu_{\mathsf{a}}(C'_k))\}\end{aligned}$$

According to Definition 10, the unnest operation is recursively defined in a way similar to the nest operation. The only restriction refers to the situation where the chosen column presents internal tables with different structures. In this case, the unnest operation cannot be executed because it would generate a result whose scheme would not conform with Definition 1.

5 QSByE

In this section, we present QSByE (*Querying Semistructured data By Example*) [13, 15], a query interface that adopts a variation of the QBE paradigm [32]. QSByE combines features of query languages for semistructured data, such as type coercion and path expressions, with features of the original QBE language to query semistructured Web data extracted by the DEByE tool [20, 28]. When using QSByE, users formulate their queries by means of graphical facilities provided by the interface, which makes the syntax of the operations defined in Section 4 completely transparent to them.

To show the suitability of QSByE for semistructured data, and to illustrate the process of query formulation, we use three examples. We assume that these queries are executed over a repository containing data extracted from a set of Web pages similar to the page Figure 1. The data in this repository are structured according to the example in Figure 2.

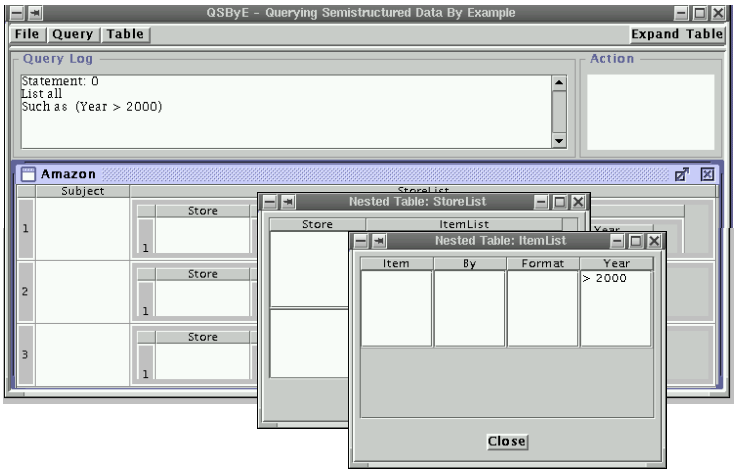
To formulate queries using QSByE, the user first selects a repository containing the data of interest. This immediately provides the structure of the stored data in this repository as a nested table skeleton. Figure 8 shows the skeleton for the repository used in our examples.

Our first example query is as follows:

QA: *List all information on items released after 2000.*

Using the skeleton provided as an aid to query formulation, users just have to choose the operations required to formulate the query. For query **QA**, the user specifies the selection condition on the column Year, as Figure 9(a) shows. This yields the result in Figure 9(b). Notice that, for

(a)



(b)

Amazon

#	Subject	StoreList
1	Woody Allen	# | Store | ItemList
		1 | DVD | # | Item | By | Format | Year: 1 | The Curse of the Jade Scorpion | Woody Allen | DVD | 2001
		2 | Movie Showtimes | # | Item | By | Format | Year: 1 | The Curse of the Jade Scorpion | Woody Allen | Theatrical Release | 2001; 2 | Hollywood Ending | - | Theatrical Release | 2002
2	Paul Mccartney	# | Store | ItemList
		1 | DVD | # | Item | Year: 1 | Paul McCartney - Wingspan - An Intimate Portrait | 2001; 2 | The Concert for New York City | 2001
		2 | Video | # | Item | Year: 1 | Paul McCartney - Wingspan - An Intimate Portrait | 2001
3	caetano veloso	# | Store | ItemList
		1 | DVD | # | Item | By | Format | Year: 1 | Prenda Minha | Caetano Veloso | DVD | 2001
4	Paul Simon	# | Store | ItemList
		1 | Movie Showtimes | # | Item | By | Format | Year: 1 | Crocodile Dundee in Los Angeles | Paul Hogan | Theatrical Release | 2001; 2 | Rollerball | Chris Klein, Jean Reno | Theatrical Release | 2002

Figure 9: Specification and result of Query **QA.**

this query, no projection conditions were specified. Thus, the result shows the complete objects satisfying the selection condition.

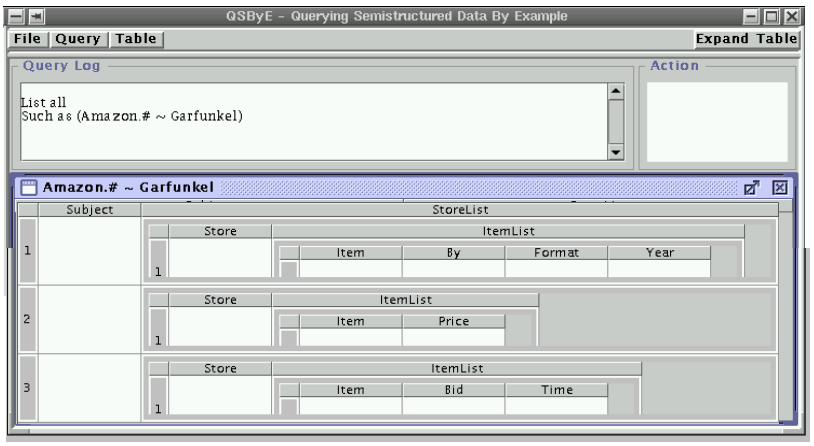
By looking into the contents of the "Query Log" window in the screen shown in Figure 9(a), we see a textual description of the operation executed. Such descriptions are generated automatically by QSByE for each query executed, as a way of logging the queries. They can be also observed in the execution of the next example queries.

An important requirement of query tools and languages for semistructured data is allowing the specification of selection conditions disregarding the details of the scheme. Our next example query shows how this is accomplished with QSByE.

QB*: Show information on all items related to "Garfunkel".*

As Figure 10(a) illustrates, to formulate this query we specify the selection condition on the outermost table header (labeled Amazon), meaning that this selection condition must be applied to all columns under this header. This feature explores the whole nested structure of the table in a way similar to what is done in regular path expressions on typical semistructured query languages such as Lorel [2]. Graphically, in QSByE, this condition is represented by the symbol

(a)



(b)

Amazon

#	Subject	StoreList

1 – Art Garfunkel

#	Store	ItemList

1 – DVD

#	Item	Format	Year
1	Art Garfunkel – Across America	DVD	–
2	Steely Dan – Two Against Nature – DTS 5.1	DVD	2000
3	Boxing Helena	Julian Sands, Sherilyn Fenn	1993

2 – zShops

#	Item	Price
1	lp – GARFUNKEL, ART – Scissors Cut	10.00
2	ART GARFUNKEL ~ Across America ~	14.99
3	lp – GARFUNKEL, ART – Angel Clare	10.00

3 – Auctions

#	Item	Bid	Time
1	Short Fuse ~ Starring Art Garfunkel and Robert Doqui (New VHS)	14.99	Ends in 13:58:44
2	STILL WATER BY ART GARFUNKEL 11390	4.00	Ends in 6 days, 16:05:48
3	SHORT FUSE (1986) ART GARFUNKEL B7108	2.49	Ends in 13 days, 13:35:36

1 – Popular Music

#	Item	By	Format
1	Breakaway	Art Garfunkel	Audio CD
2	Songs From a Parent to a Child	Art Garfunkel	Audio CD
3	Garfunkel: Best Of	Art Garfunkel	Audio CD

2 – Paul Simon

1 – Video

#	Item	Format	Year
1	Simon and Garfunkel – The Concert in Central Park	VHS	1982

Figure 10: Specification and result of Query **QB**.

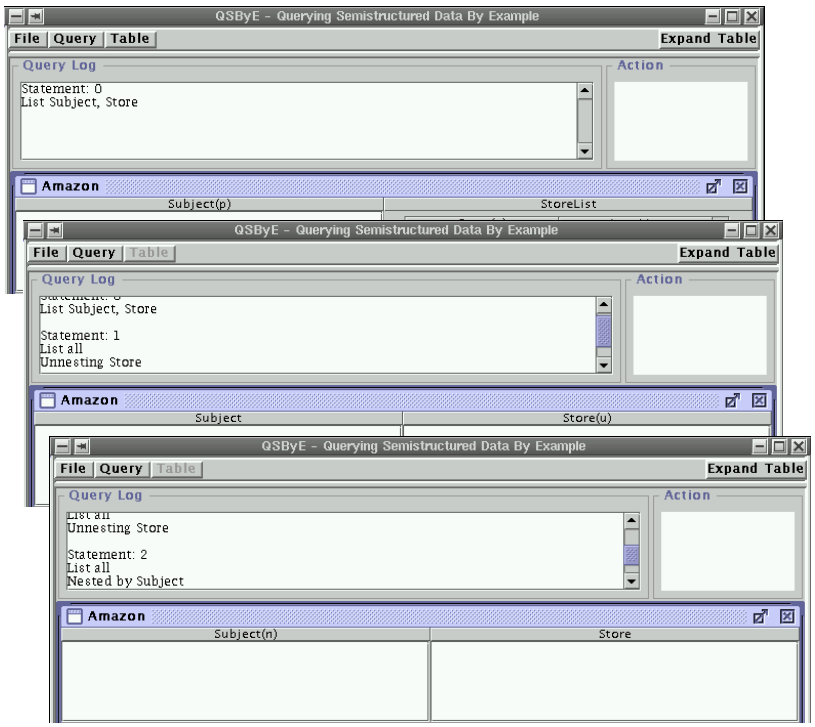
"#", as indicated in the header of the table presented in Figure 10(a). This flexibility is suitable for semistructured data because of possible distinct occurrences of the same value in different places in a data source. Also, in this query selection condition we have used the ~ operator, which provides the flexibility of pattern matching and allows errors, therefore making it possible to retrieve the required data even if the condition value has been misspelled. Figure 10(b) shows the query result.

Our next query, described in Figures 11(a) to (f), exemplifies the use of the projection and nest/unnest operations. The query is as follows:

(a)

(b)

(c)



(d)

#	Subject	Store
1	Woody Allen	DVD
		Popular Music
		Books
		Video
		zShops
		Movie Showtimes
		Auctions
2	John Grisham	Books
		zShops
		Auctions
3	Alfred Hitchcock	DVD
		zShops
		Books
		Video
		Auctions

(e)

#	Subject	Store
1	Woody Allen	DVD
2	Woody Allen	Popular Music
3	Woody Allen	Books
4	Woody Allen	Video
5	Woody Allen	zShops
6	Woody Allen	Movie Showtimes
7	Woody Allen	Auctions
8	John Grisham	Books
9	John Grisham	zShops
10	John Grisham	Auctions
11	Alfred Hitchcock	DVD
12	Alfred Hitchcock	zShops
13	Alfred Hitchcock	Books
14	Alfred Hitchcock	Video
15	Alfred Hitchcock	Auctions
16	Alfred Hitchcock	Movie Showtimes

(f)

#	Store	Subject
1	DVD	Woody Allen
		Alfred Hitchcock
		Paul Mccartney
		caetano veloso
		Paul Simon
		Art Garfunkel
2	Popular Music	Woody Allen
		Alfred Hitchcock
		Antonio Carlos Jobim
		Paul Mccartney
		caetano veloso
		Paul Simon
		Art Garfunkel
		Woody Allen
		John Grisham

Figure 11: Specification and result of Query **QC**.

QC*: For each store, list the subjects for whom products are available in the store.*

For this query, three steps are required. In the first step, we specify a projection operation over columns Subject and Store. As Figure 11(a) shows, we accomplish this by clicking on the header of these columns, what causes them to be marked with a "(p)". The result is the table in Figure 11(d), where each subject is associated with the list of stores having its products. However, the statement of the query requires this table to be rearranged. For this, we first apply an unnest operation over the table of Figure 11(d). This is accomplished by pressing the button "Table" to select the the unnest operation and then clicking on the header of the Store column, what causes it to be marked with a "(u)". The result is the table in Figure 11(e). Finally, we apply a nest operation over this table by again pressing the button "Table" to select the nest operation and then clicking on the header of the Subject column, what causes it to be marked with a "(n)". The result of this last operation is shown in Figure 11(f).

6 Conclusions and Directions for Future Work

We have formalized nested tables with varying internal structure. This formalization generalizes nested tables for semistructured Web data by allowing distinct rows to contain objects with distinct structures. We have shown how to use these nested tables to represent and query semistructured data extracted from the Web. We extended nested relational operators to manage the variable nestings we defined, and we implemented QSByE to allow a user to query tables with variable internal structure in a QBE-like manner. Preliminary experiments with QSByE [13] demonstrate that with a small amount of training even unskilled users can use our interface to query semistructured data extracted from Web pages.

An exciting direction for future work is to rearrange our query interface to allow a user to directly query a semistructured, multiple-record Web document. Instead of first extracting data into nested tables using DEByE [20, 28], we can allow a user to pose a query directly to a Web page. To pose a direct query, a user would specify a nested table, choosing header names and a nesting structure to suit the query and would fill in selection conditions as explained in this paper. To complete the query, the user would specify *real examples* from the Web page by highlighting data values on the page with a mouse and dragging them into the nested query table. This action would invoke DEByE, causing it to analyze the Web page and extract data values as explained in [20, 28], and then invoke the backend selection processing described in this paper to limit the extracted values to those satisfying the query specification. Thus, users would be able to directly specify a QBE-like query over a Web page and obtain an answer. Carrying this future work a bit further, we could obtain the results of a specified QBE-like query from several Web pages containing similar application data and merge obtained results into one final answer.

References

[1] Abiteboul, S., and Bidoit, N. Non First Normal Form Relations to Represent Hierarchical Organized Data. In *Proceedings of the 3rd ACM Symposium on Principles of Database Systems* (Waterloo, Canada, 1984), pp. 191–200.

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

[3] Abiteboul, S., Suciu, D., and Buneman, P. *Data on the Web: From Relations to Semistructured Data and XML*, 1st ed. Morgan Kaufmann, San Francisco, 2000.

[4] Arocena, G. O., and Mendelzon, A. O. WebOQL: Restructuring Documents, Databases, and Webs. In *Proceedings of the International Workshop on Database Programming Languages* (Scotland, UK, 1999), pp. 208–223.

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

[6] Buneman, P., Deutsch, A., and Tan, W. A Deterministic Model for Semistructured Data. In *Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats* (Jerusalem, Israel, 1999).

[7] Chamberlin, D., Florescu, D., Robie, J., Siméon, J., and Stefanescu, M. XQuery: A Query Language for XML. *http://www.w3.org/TR/xquery/*.

[8] Colby, L. S. A Recursive Algebra and Query Optimization for Nested Relations. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data* (Portland, Oregon, 1989), pp. 273–283.

[9] Crescenzi, V., Mecca, G., and Merialdo, P. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Rome, Italy, 2001), pp. 109–118.

[10] Deutsch, A., Fernandez, M. F., Florescu, D., Levy, A. Y., and Suciu, D. A Query Language for XML. *International Journal on Digital Libraries 31*, 11-16 (1999), 1155–1169.

[11] Deutsch, A., Fernandez, M. F., and Suciu, D. Storing Semistructured Data with STORED. In *Proceedings the 1999 ACM SIGMOD International Conference on Management of Data* (Philadephia, Pennsylvania, 1999), pp. 431–442.

[12] Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Ng, Y.-K., Quass, D., and Smith, R. D. A Conceptual-Modeling Approach to Extracting Data from the Web. In *Conceptual Modeling - ER'98, 17th International Conference on Conceptual Modeling, November 16-19, Singapore* (1998), Springer, pp. 78–91.

[13] Evangelista-Filha, I. M. R. A Graphical Interface for Querying Semistructured Data Through Examples. MSc. Dissertation, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 2001. (In Portuguese).

[14] Evangelista-Filha, I. M. R., da Silva, A. S., Laender, A. H. F., and Embley, D. W. Using Nested Tables for Representing and Querying Semistructured Data. In *Proceedings of 14th International Conference on Advanced Information Systems Engineering* (Toronto, Canada, 2002). Short Paper. To appear.

[15] Evangelista-Filha, I. M. R., Laender, A. H. F., and Silva, A. S. Querying Semistructured Data By Example: The QSByE Interface. In *Proceedings of the International Workshop on Information Integration on the Web* (Rio de Janeiro, Brazil, 2001), pp. 156–163.

[16] Florescu, D., Levy, A., and Mendelzon, A. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record 27*, 3 (1998), 59–74.

[17] Goldman, R., and Widom, J. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases* (Athens, Greece, 1997), pp. 436–445.

[18] Jaeschke, G., and Schek, H.-J. Remarks on the Algebra of Non First Normal Form Relations. In *Proceedings of the ACM Symposium on Principles of Database* (Los Angeles, California, 1982), pp. 124–138.

[19] Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal 118*, 1-2 (2000), 15–68.

[20] Laender, A. H. F., Ribeiro-Neto, B., and da Silva., A. S. DEByE – Data Extraction By Bxample. *Data and Knowledge Engineering 40*, 2 (2002), 121–154.

[21] Laender, A. H. F., Ribeiro-Neto, B., da Silva, A. S., and Silva, E. S. Representing Web Data as Complex Objects. In *Proceedings of the 1st International ConferenceElectronic Commerce and Web Technologies* (London, UK, 2000), pp. 216–228.

[22] Libkin, L. A Relational Algebra for Complex Objects Based on Partial Information. In *Proceedings of the 3rd Symposium on Mathematical Fundamentals of Database and Knowledge Bases Systems* (Rostock, Germany, 1991), pp. 29–43.

[23] Lorentzos, N. A., and Dondis, K. A. Query by Example for Nested Tables. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications* (Vienna, Austria, 1998), pp. 716–725.

[24] MAKINOUCHI, A. A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model. In *Proceedings of the 3rd International Conference on Very Large Data Bases* (Tokyo, Japan, 1977), pp. 447–453.

[25] MCHUGH, J., ABITEBOUL, S., GOLDMAN, R., QUASS, D., AND WIDOM, J. Lore: A Database Management System for Semistructured Data. *SIGMOD Record 26*, 3 (1997), 54–66.

[26] MUSLEA, I., MINTON, S., AND KNOBLOCK, C. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems 4*, 1/2 (2001), 93–114.

[27] PAPAKONSTANTINOU, Y., GARCIA-MOLINA, H., AND WIDOM, J. Object Exchange Across Heterogeneous Information Sources. In *Proceedings of the 11th International Conference on Data Engineering* (Taipei, Taiwan, 1995), pp. 251–260.

[28] RIBEIRO-NETO, B., LAENDER, A. H. F., AND DA SILVA, A. S. Extracting Semi-Structured Data Through Examples. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management* (Kansas City, Missouri, 1999), pp. 94–101.

[29] ROTH, M. A., KORTH, H. F., AND SILBERSCHATZ, A. Extended Algebra and Calculus for Nested Relational Databases. *ACM Transactions on Database Systems 13*, 4 (1988), 389–417.

[30] THOMAS, S. J., AND FISCHER, P. C. Nested Relational Structures. *Advances in Computing Research 3* (1986), 269–307.

[31] Extensible markup language (XML). http://www.w3.org/XML/.

[32] ZLOOF, M. M. Query-by-Example: A Data Base Language. *IBM Systems Journal 16*, 4 (1977), 324–343.