XML DATA WAREHOUSE: MODELLING AND QUERYING

Jaroslav Pokorný

Department of Software Engineering
Faculty of Mathematics and Physics
Malostranske nam. 25
118 00 Prague - Czech Republic
email: pokorny@ksi.ms.mff.cuni.cz

Abstract

A large amount of heterogeneous information is now available in enterprises. Some their data sources are repositories of XML data or they are viewed as XML data independently on their inner implementation. In this paper, we study the foundations of XML data warehouses. We adapt the traditional star schema with explicit dimension hierarchies for XML environment. We propose the notion of XML-referential integrity for handling XML-dimension hierarchies. For querying XML data warehouses, we introduce an operation semijoin based on approximate matching XML data and discuss its effective evaluation.

1 Introduction

A large amount of heterogeneous information is now available in enterprises. Such data stores may be classical formatted databases but also data collections coming from e-mail communication, e-business, or from inner digital documents that are produced by applications in enterprise.

A data warehouse (DW) is an integrated repository of data generated from many sources and used by the entire enterprise. The dimensional model (DM) is a logical representation of a business process whose main features are user understandability, query performance and usefulness for reporting, issue resolution, and predictive modelling. As the viable technique in DW environment, DM is widely accepted [Ki+98]. A usual approach to DM is based on dimension and fact tables grouped into a structure called a star schema.

In a real environment of the enterprise, it is not too hard to imagine that some its data sources are repositories of XML data or that they are viewed as XML data independently on their inner implementation. An increasing interest appears to conceive XML data as database data [Bo01] or, particularly, as DW data. In [Po01] we tried to build a DW over XML data, i.e. we supposed that dimensional data is XML data. The notion of DM was modified and accommodated substantially and a star schema structure with explicit hierarchies [Po99] was used as a basic data structure in the new approach to XML-based DW. We assumed XML data equipped by Document Type Definitions (DTD). Then, on the schema level, a particular dimension is modelled as a sequence of DTDs that are logically associated by so called XML-referential integrity. Obviously, facts may be modelled as XML data as well.

XML data is traditionally divided into two categories: document centric and data centric. The former has only few, interspersed mark-ups, the latter is solely created and interpreted by some application logic. We suppose rather data centric XML data for our DW. However, the information in an XML DW is never complete. It follows from the possibilities of XML data specification via regular expressions allowed by DTDs. Consequently, only partial (or approximate) matching of XML data is appropriate for testing if the referential integrity is satisfied by two XML collections.

Having such a theoretical framework for XML-based DW, a natural question is arising: how to query such data in practice. In other words, usual approaches to querying stars should be reformulated. We design a basic algorithm for this purpose. It uses a semijoin operation accommodated for XML data.

The paper is organized as follows. Section 2 shortly introduces the main concepts of DM with tables and states some restrictions chosen for the approach in the paper. In Section 3 we give a brief overview over XML and present a tree model for XML data and associated DTD. Section 4 defines notions needed for characterization of XML collections, for specifying XML-referential integrity, and for establishing dimensions over XML data. We define XML-star schemes with explicit dimension hierarchies and dimensional XML-databases. We also design an algorithm which makes it possible to extract fact data from a dimensional XML-database, considering a simple query language over XML-star schema. We complete this section by a discussion of some decisions chosen in the approach. Finally, we summarize the approach and point out further research issues.

2 Dimensional modelling with tables

Informally, a *DM-schema* is a description of *dimension* and *fact tables*. An associated diagram is called *DM-diagram*. A variant of this approach is called a *star schema*, i.e. the case with one fact table surrounded by dimension tables. Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multi-part key in the fact table. Attributes of dimension tables (*dimension attributes*) are used as a source of constraints usable in DW queries. A *fact* is a focus of interest for the enterprise. It is modelled by values of non-key attributes in the fact table that functionally depend on the set of key attributes. Each fact is "measured" by values of a tuple of dimension values.

2.1 Dimension hierarchies

On the conceptual level, particular members of each dimension hierarchy are sets of entities that can be described as entity types. Although more complex hierarchies are distinguished in literature, we keep to *simple hierarchies*, whose members compose a path in a directed graph (see Figure 1). Consequently, we can model such hierarchies by a chain of tables connected by logical references (foreign keys).

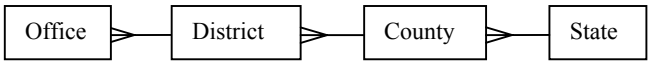

Figure 1: Dimension hierarchy as a chain of entity types

The notions can be formalized following [Po99]. We consider *dimension table schemes* $D_i(\Omega_i)$, i=1,…,n, n≥1, and a *fact table schema* $F(\Omega)$, where Ω_i and Ω are sets of attributes. One attribute from Ω_i is called the *key* of D_i table and is denoted KD_i. The *key of* F is $\cup_{i=1..n} KD_i$[1]. Other (non-key) attributes of F are usually called *facts*. We call such fact table schema F *specified w.r.t.* dimension table schemes $D_i(\Omega_i)$, i=1,…,n. Extensions of table schemes are sets of rows similarly as in relational databases. We call them *tables*.

Definition 1 (Dimension Hierarchy): Let $\boldsymbol{D}$ be a set of dimension table schemes. Then a *(simple) dimension hierarchy* N is a pair $<H, CC>$, where H is specified as

(a) $H \subseteq \boldsymbol{D} \times \boldsymbol{D}$ or $\{D\}$, $D \in \boldsymbol{D}$,

(b) H is an acyclic path,

and $CC = \{CC_{ij} \mid (D_i, D_j) \in H\}$. Each CC_{ij} is defined syntactically as follows:

(c) if KD_j is the key of D_j, then KD_j is also an attribute of D_i.

Ds in H are called *members* of N. We write usually N: $D_1 \rightarrow \ldots \rightarrow D_k$. From the condition (b) we can observe that KD_j in D_i is a foreign key in the same sense as in the logical connection of F to a dimension table. Condition (a) implies the existence of a unique *root* member of H. The root plays a significant role. Actually, facts in F table are usually directly dependent on data stored in root tables.

[1] No dependencies are supposed among dimensions.

We use upper letters D, F, ... for table schemes and D^*, F^*, ... for tables. Let D_i^* and D_j^* be tables. The *cardinality constraint* CC_{ij} is *satisfied* by these tables when for each row u from D_i^* there is only one row v in D_j^*, such that $u.\mathrm{KD}_j = v.KD_j$. Let N: $D_1 \rightarrow \ldots \rightarrow D_k$ be a dimension hierarchy and $D_1^*,\ldots, D_k^*$ tables. The tables are *admissible for* N if they satisfy all cardinality constraints from CC.

2.2 Stars with explicit dimension hierarchies

We will use explicit hierarchies in star schemes [Po99]. This approach, known also as snowflakes, is useful for querying a database equipped by such schema.

Definition 2 (Star Schema with Explicit Dimension Hierarchies): Let $\boldsymbol{N}$ be a non-empty set of dimension hierarchies, $\boldsymbol{D}_N$ the set of their hierarchy roots, and F a fact table schema specified w.r.t. $\boldsymbol{D}_N$. A *star schema with explicit dimension hierarchies* is a pair $<\boldsymbol{N}, F>$.

A fragment of a DM-diagram is depicted in Figure 2. It follows from definition F that the cardinality constraint is defined between F and each root table of N, for each $N \in \boldsymbol{N}$.

A *dimensional database* S^* *over a star schema with explicit hierarchies* S is a set of dimension tables and a fact table. For each $N \in \boldsymbol{N}$ the associated dimension tables must be admissible for N and the cardinality constraint must be satisfied by F and the root table of N.

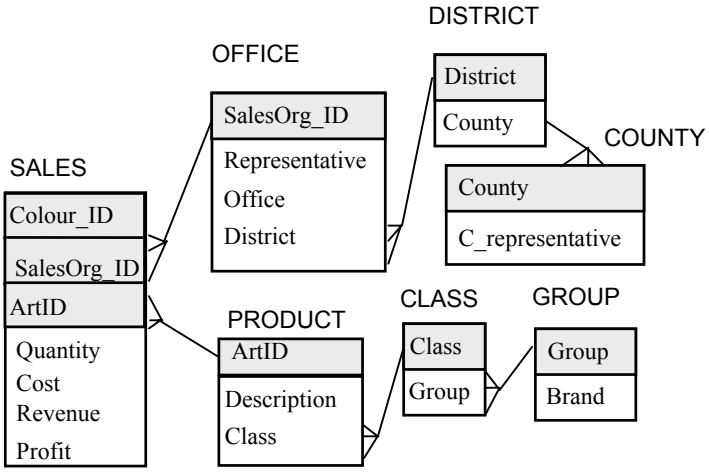

Figure 2: Fragment of DM-diagram for a star schema with explicit dimension hierarchies

The crow's feet notation expresses that rows of the fact table and of its any dimension table are in many-to-one relationship[2]. Moreover, dimensions are independent of facts, facts can not exist without dimensions. CCs also imply an expected observation that each KD is a foreign key in F, e.g. there is a referential integrity between F and each N root as well as between two adjacent members of N.

3 Basics for XML

The data in XML [W3C98] is grouped into *elements* by *tags*. Figure 3 shows a simple XML document. XML elements may contain attributes. These attributes characterize the elements. There is no common agreement when to use elements and when to use attributes. One pragmatic rule says that attributes are useful for expressing metadata about data in elements. An unordered set of attributes can be placed in the start-tag of the element.

[2] Real world DWs include also many-to-many relationships between a dimension and a fact table [So+01]. We do not suppose them in this paper.

A *DTD* specifies how elements can be nested. Subelements nesting is specified by regular expressions. An XML document *valid* w.r.t a DTD can be the root in any element specified in the DTD. An example of DTD is in Figure 4. The document in Figure 3 is valid w.r.t. this DTD.

```
<catalogue>
    <product pid = "PA312">
        <name> Canon 246/V </name>
        <description> A thing …<description>
        <class> camera </class>
        <dealer did = "K2OP1">
            <d_name> J. Smith </d_name>
            <address >
                <locality> Kings Buildings,  Edinburgh
                </locality>
                <ZIP> E12 8QQ </ZIP>
            </address>
        <dealer>
    </product>
    <product pid = "PA108">
        <name> Sony III </name>
        <description> A tool …<description>
        <class> CD player </class>
    </product>
</catalogue>
```

Figure 3: XML document describing a catalogue

```
<!DOCTYPE catalogue[
<!ELEMENT catalogue(product)*>
<!ELEMENT product(name, description, class, dealer?)>
<!ATTLIST product pid ID #REQUIRED>
<!ELEMENT dealer (d_name, contact?, address+)>
<!ATTLIST dealer did ID #REQUIRED>
<!ELEMENT name PCDATA>
<!ELEMENT d_name PCDATA>
<!ELEMENT description  PCDATA>
<!ELEMENT address(locality, ZIP)>
<!ELEMENT locality PCDATA >
<!ELEMENT ZIP PCDATA >
<!ELEMENT contact(fax |phone)
<!ELEMENT fax PCDATA >
<!ELEMENT phone PCDATA >
]>
```

Figure 4: DTD catalogue

Applications of XML require appropriate models both of XML data and DTDs. For example, XML-graphs [De+99] are well-known as a tool for describing semantics of XML-based query languages. Tree- and graph-oriented models are used also for DTDs (e.g. [LC00], [PV00]). For the sake of simplicity, most of the formal models neglect differences between attributes and subelements and the ordering of elements.

In [Po01] we modified types of labelled ordered tree objects [PV00]. Suppose a finite alphabet Σ of tags. *A labelled ordered tree object* (*loto*) over Σ is a finite tree such that each node has an associated tag from Σ and the set of children of a given node is totally ordered. For example, the loto corresponding to the XML document in Figure 3 is in Figure 5. A *loto type definition* (*ltd*) over Σ consists of a root type in Σ and a mapping associating a language L_m over Σ with each $m \in \Sigma$. For the root the mapping assigns the root element of the modelled DTD. The empty language is denoted ε. Notice that all these languages are regular in XML. Lotos can be associated with a ltd. Informally, a loto *satisfies* an ltd over Σ if its root has the type of ltd's root and for each of its nodes m the sequence of tags associated with children of m is a word of L_m. A set of lotos satisfying the ltd is denoted T(ltd). Observe that ltds do not contain PCDATA elements. In this model we do not take data in leaves of lotos into account. An example of ltd is in Figure 6 (ε languages are omitted). For example, for product the language L_{product} contains sequences (words) (name, description, class, dealer) and (name, description, class).

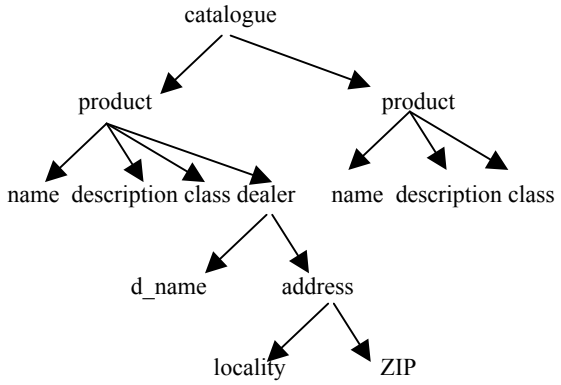

Figure 5: Loto corresponding to XML data in Figure 3

```
root: catalogue;
catalogue:product*;
product:(name, description, class, dealer?);
dealer:(d_name, contact?, address+);
address(locality, ZIP?);
contact(fax |phone);
```

Figure 6: Ltd for DTD catalogue

A key feature of our approach is the usage of XML views of XML data. XML views help us to see different data in the same way. They can be evaluated by standard view mechanisms, and used for integrity checking and query processing in XML DWs. For purposes of this paper, a *view V over a collection C* is given by a *view query* in a query language[3] for XML data. By *materialization* of *V* in *C* we mean a set of XML data, denoted *V*(*C*), which is obtainable by evaluating *V* on *C*. In particular, *V*(*C*) may be empty. In other words, *V* is a partial function defined on a set of *C* states, when we assume the collection *C* dynamic.

4 Re-building a dimension hierarchy from XML data

We suppose XML collections C_1, …, C_n, $n \geq 1$, and their respective DTDs, DTD_1… DTD_n, $n \geq 1$. DTD_C denotes the DTD of *C*. A collection *C* contains valid XML data rooting in a DTD_C element. The collections may be independent in some sense, i.e. the same information can be represented in two documents with different DTDs in multiple ways, similarly as in distributed databases. Each collection can be a source for one or more dimension members of one or more dimensions. The overall architecture of XML-star schema is drawn in Figure 7.

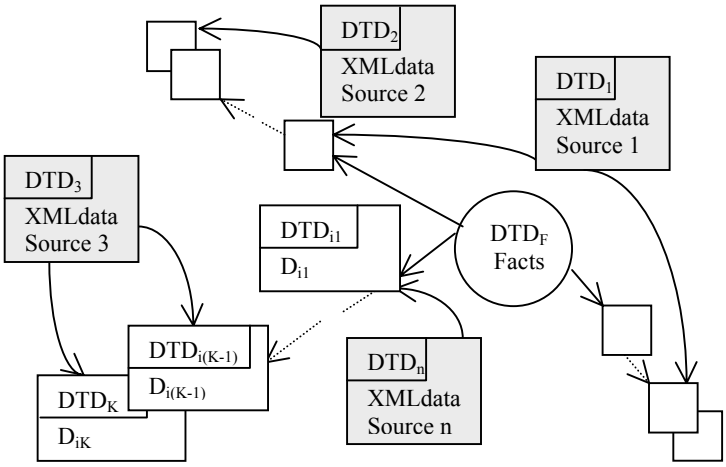

Figure 7: XML-star schema

The design of a dimension hierarchy means to explore dimension members and then logical associations among them. Three tasks are to be solved (see table 1).

	task	method	result
1	finding the data in *C* which is essential for a dimension member *D*	design (informally from DTDs and/or with a conceptual schema)	DTD_D specification
2	choice of information (*D*-core) characterizing, perhaps uniquely, most of documents in *D*	design (informally from DTDs and/or with a conceptual schema)	$DTD_{D\text{-core}}$
		view specification of *D*-core by an XML query language	$V_{D\text{-core}}$
		finding DTD specification of *D*-core	$DTD_{D\text{-core}}$
3	forming the notion of the XML-referential integrity based on *C*-cores between collections C_1 and C_2	view specification of C_2-core in C_1 by an XML query language	$V_{(C1\rightarrow C2)\text{-core}}$

Table 1: Development of XML-dimension hierarchy

[3] The same can be described more formally, independent on any query language. In practice, a query language is the simplest choice.

4.1 How to describe a dimension member

In task 1, we only restrict the source DTD to a DTD_D that is more feasible for manipulation. We use so-called *subDTD* of the given DTD. This subDTD is DTD and describes XML data sufficient for the dimension member *D* specification. An extension of *D*, D^*, on *C* contains XML data valid w.r.t. DTD_D and having the same root element tag given in DOCTYPE clause DTD_D. For example, DTD dealer in Figure 8 is a subDTD of DTD catalogue in Figure 3. For example, the addresses that are not dealer addresses are not in $Dealer^*$. Observe that elements of subDTD dealer are also elements of catalogue. Obviously, a more general approach might be possible, i.e. to restrict regular expressions in the element definitions and construct a structurally different DTD_D that is meaningful for defining a dimension member and is not subDTD of DTD_C in our sense.

```
<!DOCTYPE dealer[
<!ELEMENT dealer(d_name, contact?, address+)>
<!ELEMENT d_name PCDATA>
<!ELEMENT address(locality, ZIP?)>
<!ELEMENT locality PCDATA >
<!ELEMENT ZIP PCDATA >
<!ELEMENT contact(fax |phone)
<!ELEMENT fax PCDATA >
<!ELEMENT phone PCDATA >]
```

Figure 8: $SubDTD_1$ for dealer

```
<!DOCTYPE dealer[
<!ELEMENT dealer(d_name, address+)>
<!ELEMENT d_name PCDATA>
<!ELEMENT address(locality, ZIP)>
<!ELEMENT locality PCDATA >
<!ELEMENT ZIP PCDATA >]
```

Figure 9: $SubDTD_2$ for dealer

4.2 Characterization of a dimension member

In task 2, we actually solve the problem of keys for XML data. In principle, the problem is associated with hierarchical keys [Bu01]. First, we can forget ID attributes as possible adepts for such keys. ID attributes are not scoped. Moreover nobody can ensure their uniqueness in the environment of more XML collections. Similarly, attributes like SSN, InvoiceNo, etc. need not be also reliable. Moreover, XML data is repeating usually in the collection. Another problem concerns non-completeness of XML data. For example, the pair (name, address) of a dealer can serve as a key for dealers. But, due to the regular expressions used in DTD dealer, some dealer addresses contain ZIP data only optionally. Thus, our characterizations should meet weaker requirements that those for keys in relational databases. They should

- distinguish any two XML documents in *D* as best as possible,
- be suitably simple,
- be XML data,
- be valid w.r.t. a DTD.

In [Bu01], a key specification is a pair $(Q, \{P_1,\ldots,P_n\})$, where *Q* is a path expression and $\{P_1,\ldots,P_n\}$ is a set of simple path expressions[4]. The path expression *Q* identifies a set of nodes, which refer to as the target set, on which the key constraint is to hold. Paths P_i specify sets of nodes. Consider the following key specification:

(product, {name, description})

The target set is given by product element. If two products have the same name, then they must distinguish in their description.

Our approach is similar but not identical. We "cut of" paths *Q* via the subDTD concept, i.e. *Q* is empty (ε). We also do not use only simple paths. For example, for books such a key consists of ISBN, but probably (title, author*) is sufficient. In notation of [Bu01] we would expressed the key as (ε, {title, author*}). Further, key

[4] Simple paths are merely sequences of elements tags.

specifications of [Bu01] are directly on XML data and not on the schema level, in the database terminology. There is an approach to keys specified on the DTD level [FL01]. However, its authors consider keys based only on XML attributes.

The characterization of D in our approach is called *core* of D, shortly *D-core*. We do not use the notion of key, because vagueness of above requirements. On the extension level we talk about *core elements*. Core elements must satisfy the following condition:

- Each document from D generates one core element at most. */weak identifiability/*

It is on the designer responsibility to ensure this integrity constraint. Unfortunately, it is not always possible to require the function to be total and injective. The rigidity of these properties inherited from the notion of primary key is here lost due to the incompleteness of XML data in D and its possible duplicates. Thus we speak only about a *weak identifiability* of D.

In [Po01] we have shown how

- to define the D-core data as a view V over D^*, and
- to find its $DTD_{C\text{-}core}$.

D-core should be expressible by a DTD[5]. We use $DTD_{D\text{-}core}$ to denote such a DTD. XML data extracted from D and valid w.r.t. $DTD_{D\text{-}core}$ is called *D-core data*. We denote this data set by $T(DTD_{D\text{-}core})$.

For example, only one fax or one phone number is sufficient to identify a dealer. Then the associated view can be expressed in XML-QL language [De+99] as follows.

```
WHERE    <dealer> <contact> $e </contact>
         </dealer> IN "http://kocour.cuni.cz/…"
CONSTRUCT  <dealer_core> $e </dealer_core>
```

It provides "heterogeneous" core elements, with a fax or a phone for each dealer, who has the contact non-empty. The only user-defined tag used in queries will be dealer_core. The associated DTD is

```
<!DOCTYPE dealer_core[
<!ELEMENT dealer_core (fax |phone)>
<!ELEMENT fax PCDATA >
<!ELEMENT phone PCDATA >]
```

Figure 9: $DTD_{Dealer\text{-}core}$

4.3 XML-referential integrity

The idea how to define the constraint XML-referential integrity for collections C_1 and C_2 is to connect logically those documents $d_1 \in C_1$ and $d_2 \in C_2$ that match on the core data described by $DTD_{C2\text{-}core}$. In other words, d_1 contains data, which is the same (in some sense), as the core data of d_2. C_2-core plays for DTD_{C1} a similar role as a foreign key in a relational database.

We specify a view, we denote it $V_{(C1\to C2)\text{-}core}$, that generates XML data from C_1 of the same structure as the data valid with respect to $DTD_{C2\text{-}Core}$. We could also specify $DTD_{(C1\to C2)\text{-}core}$. Note that since DTD_{C1} distinguishes from DTD_{C2}, and $V_{C2\text{-}core}$ and $V_{(C1\to C2)\text{-}core}$ are different view queries, $DTD_{C2\text{-}core}$ and $DTD_{(C1\to C2)\text{-}core}$ would be also different in general. On the other hand, we do not need to construct $DTD_{(C1\to C2)\text{-}core}$ explicitly for specification of the XML-referential integrity. Any checking referential integrity is based on data from $V_{(C1\to C2)\text{-}core}(C_1)$. Given $V_{(C1\to C2)\text{-}core}$ and $DTD_{C2\text{-}core}$, it is decidable whether lotos associated with data from $V_{(C1\to C2)\text{-}core}(C_1)$ are also lotos from $T(ltd_{C2\text{-}core})$.

[5] In [Po01] we discuss that it is not always guaranteed that the data valid w.r.t. $DTDC_{\text{-}core}$ is exactly the data in materialization of V.

The problem of checking a referential integrity can be mapped to the problem of embedding a pattern tree in the tree determined by $V_{C2\text{-}core}(d_2)$. Obviously, unlike lotos the content of leaves must already be considered. We prefer partial matching of trees only and denote the relation by symbol $\leq:$. Certainly, if $d_1 \leq: d_2$ and $d_2 \leq: d_1$ then $d_1 = d_2$. This equality is based on the fact that associated trees distinguish only in ordering of their elements.

There are more reasons why we do not prefer an exact matching. For example, different orders of elements in d_1 and d_2 can be supposed or $V_{C1 \to C2\text{-}core}(C_1)$ data often occurs only partially in its pattern in C_2. A justification for the choice follows from the intuition that the dimension member D_i^* contains more information about an entity e than the linkage data of e in elements of D_{i-1}^*.

Definition 3 (XML-Referential Integrity): Let C_1 and C_2 be collections with DTD_{C1} and DTD_{C2} respectively. Let $V_{C1\text{-}core}$ and $V_{C2\text{-}core}$ be views defining C_1- and C_2-core data, respectively. Then an *XML-referential integrity based on C-cores* is satisfied by C_1 and C_2 iff

$$\forall\, d_1 \in C_1\, \exists\, d_2 \in C_2\, (V_{(C1 \to C2)\text{-}core}(d_1) \leq: V_{C2\text{-}core}(d_2))$$

Observe, that the matching is automatically true if $V_{(C1 \to C2)\text{-}core}(d_1)$ is empty. Thus, the definition behaves according to the notion of referential integrity given for relational databases in the SQL language [ISO99]. For some d_1 there is no document d_2 having a link through C_2-core.

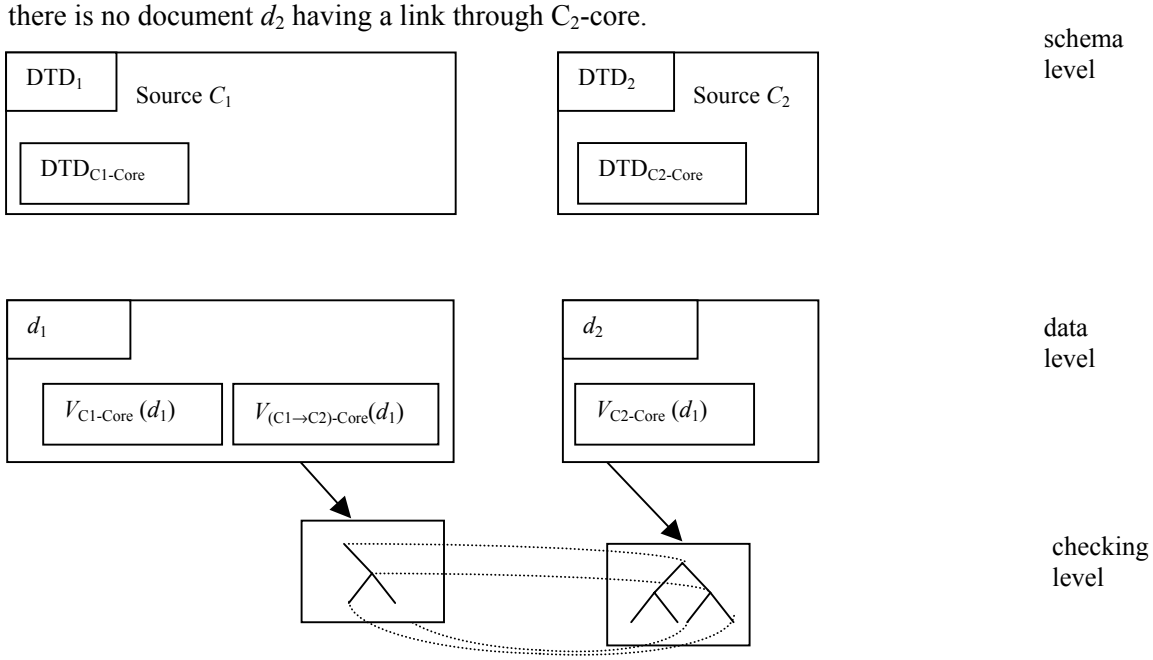

Figure 10: XML-referential integrity

For collections C_1 and C_2, we denote the statement XML-referential integrity based on C-cores by

$$C_1 \subseteq: C_2$$

Note that comparing the XML-referential integrity to the relational referential integrity, there may be more d_2 documents satisfying the condition from Definition 3. The partial (unordered) tree embedding problem, generally NP-complete, has an effective solution in practice [SN00]. The principle of the XML-referential integrity and its checking is shown in Figure 10.

5. XML-star schema with explicit hierarchies

The notion of XML-star schema with explicit dimension hierarchies is formulated according to [Po01] as follows.

Definition 4 (XML-Dimension Hierarchy): Let $C_1,\ldots, C_n$, $n \geq 1$, be a set of XML collections with their respective DTDs $DTD_1,\ldots,DTD_n$. Let $\boldsymbol{D}$ be a set of DTDs whose each member is a subDTD of a DTD_i, $i \in \langle 1,n \rangle$, $DTD_{Di\text{-}core}$ describes its D_i-core. Then a *(simple) XML-dimension hierarchy N* is specified as

(a) $N \subseteq \boldsymbol{D} \times \boldsymbol{D}$ or $\{D\}$, $D \in \boldsymbol{D}$,

(b) N is an acyclic path.

(c) If $(D_i, D_{i+1}) \in H$, then $D_j \subseteq: D_{i+1}$.

Dimensional data for N: $D_1 \rightarrow\ldots\rightarrow D_K$, is given by the union of D^*_i, $i = 1,\ldots,K$, where for (D_i^*, D_{i+1}^*) the statement $D_j \subseteq: D_{i+1}$ is satisfied.

For the sake of simplicity, we represent fact data similarly to rows of a fact table, i.e. the resulted XML data is composed from homogenous elements each dimensional component of which will be in accordance with core data of the root of the associated dimension N. We say that DTD_F of such fact data *is specified* w.r.t. a set of hierarchy roots. Fact data is not discussed here.

Definition 5 (XML-Star Schema with Explicit Dimension Hierarchies): Let $\boldsymbol{N}$ be a non-empty set of XML-dimension hierarchies, $\boldsymbol{D}_N$ the set of their hierarchy roots, and F a DTD_F specified w.r.t. $\boldsymbol{D}_N$. Then an *XML-star schema with explicit dimension hierarchies* is a pair $\langle \boldsymbol{N}, F \rangle$.

A *dimensional XML-database* S^* over an XML-star schema with explicit hierarchies S is a collection of dimensional data for N, for all $N \in \boldsymbol{N}$, and XML fact data F^* valid w.r.t. DTD_F. F^* satisfies the constraint $F \subseteq: D$ for each $D \in \boldsymbol{D}_F$.

5.1 Querying a dimensional XML-database

We will suppose a simple query language whose queries are expressed by restrictions on members of dimension hierarchies. For a D, we denote such restriction φ and write in relation algebra style $D(\varphi)$. In general the restriction can be by any predicate (possibly empty) on the element content according to the respective DTD_D. Typical are range expressions besides equality expressions. For example, path expressions as they are used in XQL language [Ro+98] are just enough to illustrate our approach. For example, the expression

Dealer(dealer[contact/phone = '21914265'])

restricts dealer elements to that ones that have the subelement <phone> 21914265</phone>. The next operation we need is an XML semijoin. This operation, we call it *C-seminjoin*, applied to two collections of XML data C_1 and C_2 extracts XML data from C_1, which successfully matches data from C_2. The matching is done by matching the data from $T(DTD_{(C1\rightarrow C2)\text{-}core})$ and $T(DTD_{C2\text{-}core})$. The matching condition is given by predicate $\leq:$. In relational environment, we would do the semijoin over a foreign key and the associated primary key of two relations.

The basic idea how to evaluate a query over a dimensional XML-database is to navigate each dimension hierarchy N from its last member with non-empty φ to the root of N. For the sake of simplicity, we will suppose that the member is the last member of N.

Algorithm: Evaluation of a query over a dimensional XML database

Input: an XML-star schema with explicit dimension hierarchies S, S^*, a query q

Output: a subcollection of F^* satisfying q

1. **for** i:=1 **to** $|\boldsymbol{N}|$ **do**
   restricted$_2$:= $D_{Ki}(\varphi_{Ki})$; //start with the last member of N and restrict it
   j:=K_i-1;
   while j ≠ 0 do

restricted$_1$:= $D_j^*(\varphi_j)$; //continue with its left neighbour and restrict it
restricted$_2$:= C-semijoin(restricted$_1$, restricted$_2$); //restrict D_j^* by semijoin
j:= j-1;
end while
root$_i$: = restricted$_2$
end for //each root$_i$ contains dimensional data necessary for processing F^*

2. result:= (…(C-semijoin(F, root$_1$),…), root$_{|N|}$); //result contains a subcollection of F^* obtained by //subsequent applications of the C-semijoin operation

We describe the semantics of C-semoijoin in the procedural manner. Let C_1 and C_2 be XML collections. Then the result of C-semoijoin(C_1, C_2) is obtainable by the following steps:

1. core$_2$:=$V_{C2\text{-core}}(C_2)$ //materialize $V_{C2\text{-core}}$ on C_2
2. C-semijoin := $\{d \mid d \in C_1 \wedge \exists\, d' \in \text{core}_2\,(V_{(C1 \rightarrow C2)\text{-core}}\,(d) <: d')\}$ //matching cores

Obviously, various optimization techniques can make a real query evaluation more feasible. Indexing methods for XML data can significantly reduce the scan required for selections or naive nested loops implementation for semijoins. The existence of an index on elements important for D-cores is a simple example of such strategy.

5.2 Discussion

This section contains a brief list of remarks to choices of alternatives used in our approach:

(a) The definition of D-cores we have adopted here is quite weak. It mirrors the fact that we need not uniqueness of keys as in relational databases. Our choice is in accordance to the semistructured nature of XML data contained in our DWs. In some applications of data centric DWs we can design D-cores that are based on real foreign and primary keys. This situation occurs if XML data originate from relational databases.

(b) Consider the principle of our approach to XML-referential integrity. In contrast to (a), even the embedding a pattern tree in a tree can be too restrictive for some applications. Return, e.g., to (title, author*) for books. Suppose that it contributes to $\text{DTD}_{\text{Di-core}}$ and $\text{DTD}_{(\text{D(i-1)}\rightarrow\text{Di})\text{-core}}$ of dimension members D_i and D_{i-1}, respectively. For two sets of authors A and A' of respective documents $d \in D_{i-1}$ and $d' \in D_i$, the non-empty $A \cap A'$ can be sufficient for a successful matching. In our approach we require $A \subseteq A'$.

(c) The choice of what language we use to specify restrictions on dimension hierarchies is important to the expressive power of the approach. Path expressions from XQL fulfil reasonable minimal requirement for such a language.

6 Conclusions

We have investigated the approach to data warehousing based on XML data. Specifically, we have focused on the possibility to create XML DWs with a star architecture and explicit dimension hierarchies. We have shown that in contrast to the simple foreign key-primary key linkage of dimension members, a form of approximate matching must be used in XML environment. Furthermore, we have shown that querying such DW in a usual way is possible. We can use partially recent XML query languages for this purpose and a new query operation C-semijoin, which is based on an approximate matching predicate. In future development, various possibilities of such approximations should be studied. Certainly, their choice will depend on the application requirements.

An important question is how to design D-cores. Starting from XML data and its DTDs, we develop, in fact, the DW from the bottom-up starting on the representation level. It is the same as to design dimensions from a set of relational schemes. Without doubts, to have a conceptual view on XML data seems to be useful at least for specifying referential integrities in XML collections. In future we would like to show how the dimensional design

for DW could be carried out transforming XML metadata (DTDs, XML schemes) into equivalent conceptual schemes. First steps already exist to this goal (e.g. [Go+01]).

References

[Bo01] Bourret, R.: XML and Databases. Available at http://www.rpbourret.com/xml/XMLAndDatabases.htm.

[Bu+01] Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.-Ch.: Keys for XML. In: Proceedings of WWW10, May 1-5, Hong-Kong, ACM Press, 2001

[De+99] Deutsch, D., Fernandez, M.F., Florescu D., Levy, M. A., Suciu D.: A Query Language for XML. WWW8/Computer Networks 31 (11-16): 1155-1169, 1999.

[FL01] Fan, W., Libkin, L.: On XML Integrity Constraints in the Presence of DTDs. In: Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2001, pp. 114-125.

[Go+01] Golfarelli, M., Rizzi, S., Vrdoljak, B.: Data warehouse design from XML sources. In: Proceedings of ACM Fourth International Workshop on Data Warehousing and OLAP (DOLAP 2001), J. Hammer Ed., ACM Press, 2001, Atlanta, pp. 40-47.

[ISO99**]** ISO/IEC 9075:1999: Information Technology --- Database Languages --- SQL. Part 2: Foundations. 1999.

[Ki+98] Kimball, R., Reeves, L., Ross, M., Thorthwaite, W.: The Data Warehouse Lifecycle Toolkit. New York: John Wiley & Sons, Inc., 1998.

[LC00] Lee, D., Chu, W.W.: Constraint-preserving transformation from XML Document Type Definition to Relational Schema. In: Proc. of 19the ER Conf., Salt Lake City, 2000.

[Ma+01] Mani, M., Lee, D., Munz, R.: Semantic Data Modelling using XML Schemes. In: Proceedings of E-R Conf., Yokohama, 2001.

[PV00] Papakonstantinou, Y., Vianu, V.: DTD Inference for Views of XML Data, Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems 2000.

[Po99] Pokorny, J. "Data Warehouses: a Modelling Perspective." In: Evolution and Challenges in System Development (Eds. W.G.Wojtkowski, S. Wrycza, J. Zupancic), Kluwer Academic/Plenum Press Publ., 1999.

[Po01] Pokorny, J.: Modelling Stars Using XML. In: Proceedings of ACM Fourth International Workshop on Data Warehousing and OLAP (DOLAP 2001), J. Hammer Ed., ACM Press, 2001, Atlanta, pp. 24-31.

[Ro+98] Robie, J., Lapp, J., Schach, D.: XML Query Language (XQL). http://www.w3.org/TandS/QL/QL98/pp/xql.html

[SN00] Schlieder, T., Naumann, F.: Approximate Tree Embedding for Querying XML Data. ACM SIGIR Workshop on XML and IR, Athens, 2000.

[So+01] Song, I.-Y., Rowen, W., Medsker, C., Ewen, E.: An Analysis of Many-to-Many Relationship Between Facts and Dimension Tables in Dimensional Modeling. In: Proceedings of the Int. Workshop on Design and Management if Data Warehouses (DMDW 2001), Interlaken, 2001, pp. 6-5 - 6-13.

[W3C98] Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998.

[W3C01] XQuery 1.0: An XML Query Language. W3C Working Draft 20 December 2001http://www.w3.org/TR/2001/WD-xquery-20011220/