INTRODUCTION

The World Wide Web is of strategic importance as a global repository for information and a means of communicating and sharing knowledge. Its explosive growth has caused deep changes in all the aspects of human life, has been a driving force for the development of modern applications (e.g., Web portals, digital libraries, wrapper generators, etc.), and has greatly simplified the access to existing sources of information, ranging from traditional DBMS to semi-structured Web repositories. The adoption by the WWW consortium (W3C) of XML (eXtensible Markup Language) as the new standard for information exchange among Web applications has led researchers to investigate classical problems in the new environment of repositories containing large amounts of data in XML format.

Great attention has also been recently devoted to the introduction of integrity constraints and the definition of normal forms for XML (Arenas & Libkin, 2003, 2004; Fan & Libkin, 2002; Vincent & Liu, 2003). XML allows a simple form of constraints to describe references obtained through ID/IDREF, but it does not actually provide a general mechanism for expressing semantic constraints like those commonly used in relational databases. The need of enriching the semantics of XML is so deep as a large amount of XML data originates in object-oriented and relational databases, where different forms of integrity constraints are used to add semantics to the collected information.

This work stems from the need of enriching the semantics of XML documents. This need is attested by several new works which introduce different forms of constraints to XML documents (Arenas, Fan & Libkin, 2002, 2004; Buneman et al., 2001, 2002; Fan & Libkin, 2002; Fan & Simeon, 2000; Vincent et al., 2004; Yang, Yu & Wang, 2001). Most of them introduce a simple form of constraints such as keys and foreign keys, whereas some others attempt to extend the class of integrity constraints associated with XML documents.

Obviously, reasoning about constraints in the presence of an incomplete knowledge of the data structure is rather complex so that some of these attempts are likely to be a purely theoretical exercise. In fact, their practical applicability follows the solution of non-trivial problems such as the implication and interaction among constraints which are far from being solved. In the presence of constraints on data, an XML document may result in being inconsistent; that is, it does not respect some constraint. The following example shows the case of an inconsistent XML document.

Example 1. Consider the following XML document representing a book collection:

```xml
<bib>
  <book isbn="0-451-16194-7"
    <title> A First Course in Database Systems </
      <author> Ullman </author>
      <author> Widom </author>
    </book>
  <book isbn="0-451-16194-7"
    <title> Principles of Database and Knowledge-Base Systems </
      <author> Ullman </author>
      <publisher> Computer Science Press </publisher>
  </book>
</bib>
```
and the functional dependency \( \text{bib.book.@isbn} \rightarrow \text{bib.book.title} \)'\(^1\), stating that two books with the same isbn must have the same title.

The above document does not satisfy this functional dependency, as the first and the second book have the same isbn attribute but different titles.

The above example shows that, generally, the satisfaction of constraints cannot be guaranteed, thus, in the presence of an XML document which must satisfy a set of constraints, we have to manage potential inconsistencies of data. This problem has been recently investigated for relational databases and several techniques based on the computation of repairs (minimal sets of insert/delete operations), and consistent answers have been proposed in this context (Arenas, Bertossi & Chomicki, 1999; Greco & Zumpano, 2000). However, these techniques cannot easily be extended to XML data because of the different structure of data and the different nature of constraints.

The document of the previous example can be repaired by performing one of the following minimal sets of update operations:

- replace the string “A First Course in Database Systems” with the title “Principles of Database and Knowledge-Base Systems”;
- replace the string “Principles of Database and Knowledge-Base Systems” with the title “A First Course in Database Systems”;
- assign a new, different value to one of the two isbn attributes so that there are no two books with the same isbn.

This means that the violation of a functional dependency could be repaired by applying several sets of possible update operations, yielding a consistent scenario of the information. In our framework, we prefer the repairs performing minimal sets of changes to the original document, in the same way as well known approaches proposed for relational database repairing.

We also address the problem of extracting “reliable” information from inconsistent documents. To this end, we define two different semantics for queries, which are evaluated on the repaired version of the given document, instead of the (inconsistent) original one. A wider discussion of the notions and contributions introduced in this article is provided in Flesca et al. (2003).

**BACKGROUND**

A functional dependency \( A \rightarrow B \) in a relational database \( D \) models the correspondence between \( A \) and \( B \) values in the tuples of \( D \). However, there is no standard concept for tuple in the XML context. In this section, we recall the notion of functional dependency in the XML setting proposed in Arenas and Libkin (2004) and Arenas et al. (2002)\(^2\), which will be used in the following as a basis for our framework. Before introducing functional dependencies for XML, we present the tree-based representation model which will be adopted in the rest of this work, and then we provide the concept of tree tuple, corresponding to the concept of tuple in relational databases.

From now on, XML documents will be represented by means of labeled trees (called “XML trees”) whose nodes correspond to either elements, attributes, or string values. In particular, nodes corresponding to elements have a single label (representing the tag name), whereas nodes corresponding to attributes have two labels, representing the attribute name and value, respectively. The text content of an element will be represented by a distinguished node (marked with the symbol “S”) labeled with a string equal to the text content of the element. An example of XML tree is shown in Figure 1.

The information represented in an XML document can be extracted by means of path expressions identifying nodes of the corresponding XML tree. A path expression is a sequence of symbols (i.e., tag names, attribute names, or the symbol “S”) occurring in the XML tree, identifying traversals over it. In more detail, the result of a path expression ending with an element name is the set of node identifiers which can be reached, starting from the root, walking through a sequence of nodes whose labels satisfy the given expression. Otherwise, if the path expression ends with an attribute name or the symbol “S”, the result is a set of strings representing the attribute values (or the text content of elements) which can be reached by means of path satisfying the given expression. For instance, the path expression bib.book.title applied on the XML tree of Figure 1 returns the set \( \{v_{12}, v_{22}\} \), whereas bib.book.written_by.author.name.S returns the set \( \{“Ullman”, “Widom”\} \).

Informally, a tree tuple groups together nodes of the document which are semantically correlated, according to the structure of the tree. For instance, a tree tuple of the XML tree \( XT \) of Figure 1 consists of a sub-tree which contains information about a book. Observe that each book is possibly described by more than one tree tuple, as each tree tuple contains the information of only one author.

**Definition 1.** Given an XML tree \( XT \), a tree tuple \( t \) of \( XT \) is a maximal sub-tree of \( XT \) such that, for every path expression \( p \) defined on \( XT \), \( t.p \) contains at most one element.

**Example 2.** Consider the XML tree \( XT \) of Figure 1. The sub-tree of \( XT \) shown in Figure 2(a) is a tree-tuple, whereas the sub-tree in Figure 2(b) is not a tree tuple (it contains
Related Content

Methods for the Identification of Data Outliers in Interactive SQL
[www.igi-global.com/article/methods-identification-data-outliers-interactive/51083?camid=4v1a](www.igi-global.com/article/methods-identification-data-outliers-interactive/51083?camid=4v1a)

Retrieval Optimization for Server-Based Repositories in Location-Based Mobile Commerce
[www.igi-global.com/chapter/retrieval-optimization-server-based-repositories/4302?camid=4v1a](www.igi-global.com/chapter/retrieval-optimization-server-based-repositories/4302?camid=4v1a)

Relational Techniques for Storing and Querying RDF Data: An Overview
[www.igi-global.com/chapter/relational-techniques-storing-querying-rdf/52305?camid=4v1a](www.igi-global.com/chapter/relational-techniques-storing-querying-rdf/52305?camid=4v1a)

Human Factors Research on Data Modeling: A Review of Prior Research, An Extended Framework and Future Research Directions
Heikki Topi and V. Ramesh (2002). *Journal of Database Management* (pp. 3-19).
[www.igi-global.com/article/human-factors-research-data-modeling/3276?camid=4v1a](www.igi-global.com/article/human-factors-research-data-modeling/3276?camid=4v1a)