Chapter 8.7
Reverse Engineering from an XML Document into an Extended DTD Graph

Herbert Shiu
City University of Hong Kong, Hong Kong

Joseph Fong
City University of Hong Kong, Hong Kong

ABSTRACT

The extensible markup language (XML) has become a standard for persistent storage and data interchange via the Internet due to its openness, self-descriptiveness, and flexibility. This article proposes a systematic approach to reverse engineer arbitrary XML documents to their conceptual schema, extended DTD graphs, which are DTD graphs with data semantics. The proposed approach not only determines the structure of the XML document, but also derives candidate data semantics from the XML element instances by treating each XML element instance as a record in a table of a relational database. One application of the determined data semantics is to verify the linkages among elements. Implicit and explicit referential linkages are among XML elements modeled by the parent-children structure and ID/IDREF(S), respectively. As a result, an arbitrary XML document can be reverse engineered into its conceptual schema in an extended DTD graph format.

INTRODUCTION

As the extensible markup language (XML; Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2004) has become the standard document format, the chance that users have to deal with XML documents with different structures is increasing. If the schema of the XML documents in a document type definition (DTD; Bosak, 1998) is given or derived from the XML documents right away (Kay, 1999; Moh, Lim, & Ng, 2000), it is easier

to study the contents of the XML documents. However, the formats of these schemas are hard to read and not very user friendly.

XML has been the common format for storing and transferring data between software applications and even business parties as most software applications can generate or handle XML documents. For example, a common scenario is that XML documents are generated and based on the data stored in a relational database; there have been various approaches for doing so (Fernandez, Morishima, & Suciu, 2001; Thiran, Estiévenart, Hainaut, & Houben, 2004). The sizes of XML documents that are generated based on the data stored in databases can be very large. Most probably, these documents are stored in a persistent storage for backup purposes as XML is the ideal format that can be processed by any software applications in the future.

In order to handle the above scenario, it is possible to treat XML element instances in an XML document as individual entities, and the relationships from the different XML element types can be determined by reverse engineering them for their conceptual models, such as extended DTD graphs with data semantics. As such, users can have a better understanding of the contents of the XML document and further operations with the XML document become possible, such as storing and querying (Deutsch, Fernandez, & Suciu, 1999; Florescu & Kossmann, 1999; Kanne & Moerkotte, 2000).

This article proposes several algorithms that analyze XML documents for their conceptual schema. Two main categories of XML documents exist: data centric and narrative. As the contents of narrative XML documents, such as DocBook (Stayton, 2008) documents, are mainly unstructured and their vocabulary is basically static, the necessity of handling them as structured contents and reverse engineering them into conceptual models is far less than that of handling data-centric ones. Therefore, this article will concentrate on data-centric XML documents.

Referential Integrity in XML Documents

XML natively supports one referential integrity mechanism, which is the `ID/IDREF(S)` type of attribute linkages. In every XML document, the value of an `ID` type attribute appears at most once and the value of the `IDREF(S)` attribute must refer to one `ID` type attribute value. An `IDREF(S)` type attribute can refer to any XML element in the same document, and each XML element can define at most one `ID` type attribute. Due to the nature of `ID/IDREF(S)` type attributes in XML documents, relationships among different XML element types can be realized and it is possible to use them to implement data semantics.

This article will discuss the various data semantics and the possible ways to implement them. The algorithms presented are based on observations of the common XML document structures:

1. Due to the nested structure of an XML document (the relationship between a parent element and its child elements), the child elements implicitly refer to their parent element.
2. For an `IDREF` or `IDREFS` type attribute, the defining element is referred to the element(s) with an `ID` type attribute by the referred value. Such linkages are similar to the foreign keys in a relational database. The two associated element types are considered to be linked by an explicit linkage.
3. As an `IDREFS` type attribute can refer to more than one element, there is a one-to-many cardinality from the referring element type and the referred element type(s).

The schema of an XML document can restrict the order of the XML elements, which may be significant; the order depends on the intentions of the original XML document designer. For example,