

ERX: A Data Model for Collections of XML Documents

Giuseppe Psaila

Politecnico di Milano

Dipartimento di Elettronica e Informazione

Piazza L. Da Vinci, 32 - I-20133 Milano, Italy

psaila@elet.polimi.it

Abstract

XML is able to represent any kind of structured or semi-structured document, such as papers, web pages, database schemas and instances, style-sheets, etc.. However, the tree-structure of XML documents, induced by mark-up nesting, does not provide a sufficiently expressive and general conceptual model of data in the documents, particularly when multiple source documents are processed at the same time.

This paper proposes the ERX (Entity Relationship for XML) data model, an evolution of the Entity Relationship model that copes with the peculiar features of XML. ERX is devised to provide an effective support to the development of complex XML processors for advanced applications.

1 Introduction

XML, the *eXtensible Mark-up Language* ([3]) proposed by the World Wide Web Consortium (W3C), is object of an increasing interest by researchers and industries. XML has been designed to become the format to exchange documents over the Internet, but is becoming a standard to represent any kind of documents in any application domain. This is due to the fact that it is able to represent any kind of structured or semi-structured document, such as papers, manuals, technical reports, web pages, database schemas and instances (see XML-Data [9]), style-sheets (see XSL [1]), meta-data (see RDF [8]), etc.. As a consequence, new complex and advanced applications designed to operate in Internet/Intranet environments are being developed, which use XML both to exchange data with other applications and as an internal representation [5]. In many cases, such applications deal with large repositories of documents, that must be maintained, consulted (see query languages [4, 7]), and published (for instance via Internet) with different user-dependent styles.

Since the amount of data in such collections of documents may be large and the structural complexity of data

may be significant, a conceptual model of data is necessary before starting the implementation of those software components, called *XML Processors* [3], that process the XML documents. However, the tree-structure of XML documents, induced by mark-up nesting and usually adopted by XML parsers, does not provide a sufficiently expressive and general conceptual model of data in the documents, because concepts of a given type are dispersed in the tree and it is not possible to view them as collections of homogeneous concepts. This is particularly evident in the case of XML processors that process multiple source documents at the same time: in fact, relationships between different document classes may be intricate to model and manage using the tree representation.

This paper proposes the ERX (*Entity Relationship for XML*) model, an evolution of the Entity Relationship model (see [2]) that copes with the peculiar features of XML. ERX is devised to provide an effective support to the development of complex XML processors for advanced applications, by giving a conceptual model to the collection of documents that puts in evidence classes of concepts and their relationships, even across different classes of documents.

The paper is organized as follows. Section 2 discusses a simplified, but inspired to reality, case concerning the treatment of papers and style-sheets. Section 3 introduces the ERX data model. Section 4 draws the conclusions.

2 An Example: Papers and Style-sheets

In order to illustrate the ERX data model in Section 3, we introduce a practical example.

Consider the problem of defining and processing style-sheets applied to documents. The goal of a style-sheet is to provide style rules that are applied to a given class of documents; the style-sheet defines the way each element (tag) of the document will appear on the output device (e.g. a HTML page displayed by a browser, but not necessarily). The aim of style-sheets is to separate the presentation of a document and the content of the document.

```

<!-- The DTD -->
<!ELEMENT BIBITEM (#PCDATA)>
<!ATTLIST BIBITEM label ID #IMPLIED>
<!ELEMENT BIBLIOGRAPHY (BIBITEM)+ >
<!ELEMENT CITE EMPTY >
<!ATTLIST CITE label IDREF #REQUIRED >
<!ELEMENT EMPH (#PCDATA) >
<!ELEMENT SECTION (#PCDATA | CITE | EMPH)*>
<!ATTLIST SECTION title CDATA #REQUIRED >
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT PAPER (TITLE,(SECTION)*,
(BIBLIOGRAPHY)?) >
<!ATTLIST PAPER name ID #REQUIRED>

```

Figure 1. The DTD for structured papers

In our example, the class of source documents is constituted by papers. We show a sample paper in Figure 3. The XML structure of such a document is defined by the DTD (*Document Type Definition*) reported in Figure 1. A paper is identified by its name (attribute name of tag PAPER). It has a mandatory title as content of tag TITLE. The body of the paper is constituted by a possibly empty list of *sections*; each tag SECTION has a mandatory attribute title, describing the section title. The content of a section can be a possibly empty sequence of generic text (#PCDATA), emphasized text (tag EMPH) and citations (tag CITE). While EMPH has only the content (the text to emphasize), CITE is an empty tag where attribute label specifies the label of the cited bibliographic item. Finally, the *bibliography* is described by a possibly missing and a non-empty tag BIBLIOGRAPHY, which contains a non empty list of *bibliographic items* (tag BIBITEM). The BIBITEM tag has associated a possibly missing (#IMPLIED) attribute named label: if specified, this attribute identifies the bibliographic item and can be referenced by CITE tags inside sections. The content of tag BIBITEM is the description of the bibliographic reference.

The XML source of the paper displayed in Figure 3 is reported in Figure 2.

The style-sheet. In the paper of Figure 2, nothing is said about the presentation style of the document, which is specified by a style-sheet: it defines the presentation style of papers, independently of the particular paper instance.

We adopt a sample style-sheet model, introduced by discussing the style-sheet of Figure 4. This style-sheet is a toy case, defined for the sake of clarity; however, we inspired to fundamental concepts of XSL [1].

The body of the style-sheet is constituted by a sequence of TARGET-ELEMENT tags. Each of them defines the pre-

```

<PAPER name="sample doc">
  <TITLE>
    Using bibliography citations.
  </TITLE>
  <SECTION title="Introduction">
    This paper shows how to use a
    hypothetical
    <EMPH> XML language </EMPH>
    to describe
    structured documents.
  </SECTION>
  <SECTION title="A Citation">
    Here we have an example of
    citation. We refer to Knuth's
    paper which introduced the
    concept of attribute grammar.
    This paper has the number
    <CITE label="Knuth68"/> in
    our bibliography.
  </SECTION>
  <BIBLIOGRAPHY>
    <BIBITEM label="ASU85">
      A.V. Aho, R. Sethi,
      J. D. Ullman, "Compilers:
      Principles, Techniques,
      Tools", Addison-Wesley, 1985.
    </BIBITEM>
    <BIBITEM label="Knuth68">
      D. E. Knuth, "Semantics of
      Context Free Languages",
      Mathematical System Theory,
      Vol. 2, pp. 127-145, 1968.
    </BIBITEM>
  </BIBLIOGRAPHY>
</PAPER>

```

Figure 2. The XML version of the toy paper.

sentation style for a specific tag (or element) in the source document, called the *target element*. The name of the target element is specified by the attribute name.

Consider the style for the element TITLE in the paper. The first ALIGN tag changes the current style settings, indicating that the title is centered in the page, while the tag FACE specifies that a bold face font, with size 18 points, is used. Tag CHILDREN indicates that the children of the tag are processed, included the textual content of TITLE, that will be displayed using the current style settings (TITLE contains only generic text). To end the processing of the title, the alignment is set to *justified* and the font is set to *normal* with size of 10 points.

The style for the target element SECTION specifies that the section title is displayed using a bold character with size of 14 points; observe the tag WRITE-ATTR, denoting that

Introduction

This paper shows how to use a hypothetical *XML language* to describe structured documents.

A Citation

Here we have an example of citation. We refer to Knuth's paper which introduced the concept of attribute grammar. This paper is labeled [Knuth68] in our bibliography.

BIBLIOGRAPHY

[ASU85] A. V. Aho, R. Sethi, J. D. Ullman, "Compilers: Principles, Techniques, Tools", Addison-Wesley, 1985.

[Knuth68] D. E. Knuth, "Semantics of Context Free Languages", Math. System Theory, Vol. 2, pp. 127-145, 1968.

Figure 3. The formatted paper.

the value of the specified attribute is displayed with the current style settings. Then the tag `FACE` sets again the character style to *normal* with size of 10 points. A new line (tag `NEWLINE`) is inserted and the children of the section are processed (the textual content of `SECTION` is displayed using the current style settings). If a section contains emphasized text (tag `EMPH`), this text is displayed in *italic* style. If a section contains a citation (tag `CITE`), its label is shown between braces. Observe the use of tag `WRITE`, which writes a constant string, specified as attribute `string`. The style for the target element `BIBLIOGRAPHY` is similar to the style for the target element `SECTION`, with the difference that the title is the constant string "BIBLIOGRAPHY".

Finally, the style for the target element `BIBITEM` specifies that the label is displayed in italic font between a pair of braces, followed by the reference displayed in normal font

3 The ERX Data Model

This section introduces the ERX data model. Concepts are discussed by means of the ERX model reported in Figure 5, the data model for papers and related style-sheet introduced in Section 2. We consider the case of a processor that manages a collection of papers, all referring to the same style-sheet.

Entities. An entity describes a complex (structured) concept of the source XML documents. Entities are represented as solid line rectangles; the entity name is inside the rectangle. For example, a paper is an entity (the processor manages a collection of papers), as well as a bibliographic item and a section are entities.

An *instance* of an entity X is a particular occurrence of a concept in the source documents. For example, an instance of entity `BIBITEM` is the first (or the second) bibliographic item of Figure 2.

Relationships. A relationship describes correlations existing between entities X and Y . A relationship is represented as a rhomb labeled with the name of the relationship, con-

nected to X and Y by solid lines; these lines are labeled with a *cardinality constraint* ($l:u$), which specifies for each instance of entity X (resp. Y) the minimum number l and the maximum number u of associated instances of Y (resp. X). For example, the relationship named `CITED`, between the entities `CITE` and `BIBITEM`, says that an instance of `CITE` is associated to one and only one instance of `BIBITEM` (cardinality $(1:1)$), while an occurrence of `BIBITEM` can be associated to an unlimited number of occurrences of `CITE`, even none (cardinality $(0:n)$).

A complex form of relationship is represented by a *relationship with alternatives*: an instance of an entity X is associated with instances of alternative entities Y_1, Y_2, \dots, Y_n ; the cardinality constraint for X considers all associations of an instance of X with instances of any Y_1, Y_2, \dots, Y_n . An example is the case of an instance of entity `SECTION`, that can be associated to zero or more instances (cardinality $(0:n)$) of entities `CITE`, `EMPH` or `PAR` (this latter describing text in sections) by the relationship `CONTAINS`.

Orthogonally, a relationship can be a *containment relationship*. Given two entities X and Y , a containment relationship from X to Y denotes that in the source XML document each instance of X contains instances of Y . Containment relationships are represented as normal relationships, except for the line on the X side, which becomes a dashed arrow entering the rhomb. In figure 5, `CONTAINS` is a containment relationship; it denotes that entities `CITE`, `EMPH`, and `PAR` are contained in a `SECTION`. Observe that the nesting of tags in XML documents would suggest to extensively use containment relationships in the ERX model; however, during the definition of the model, one should try to abstract as much as possible the actual document structure. This is why in Figure 5 only one containment relationship is used: in fact, from a conceptual point of view it is relevant to evidence that sections are structurally composed of paragraphs, emphasized text and citations; conversely, for bibliographic items and style specifications this is not relevant, so that we obtain a more general model.

Attributes. Entities can have *attributes*: they represent ele-

```

<STYLE-SHEET>
  <TARGET-ELEMENT name="TITLE">
    <ALIGN type="centered"/>
    <FACE type="bold" size="18"/>
    <CHILDREN/>
    <ALIGN type="justified"/>
    <FACE type="normal" size="10"/>
  </TARGET-ELEMENT>
  <TARGET-ELEMENT name="SECTION">
    <FACE type="bold" size="14"/>
    <WRITE-ATTR name="title"/>
    <FACE type="normal" size="10"/>
    <NEWLINE/> <CHILDREN/>
  </TARGET-ELEMENT>
  <TARGET-ELEMENT name="EMPH">
    <FACE type="italic"/> <CHILDREN/>
    <FACE type="normal"/>
  </TARGET-ELEMENT>
  <TARGET-ELEMENT name="CITE">
    <WRITE string="["/>
    <WRITE-ATTR name="label"/>
    <WRITE string="]"/>
  </TARGET-ELEMENT>
  <TARGET-ELEMENT name="BIBLIOGRAPHY">
    <FACE type="bold" size="14"/>
    <WRITE string="BIBLIOGRAPHY"/>
    <FACE type="normal" size="10"/>
    <NEWLINE/> <CHILDREN/>
  </TARGET-ELEMENT>
  <TARGET-ELEMENT name="BIBITEM">
    <NEWLINE/> <FACE type="italic"/>
    <WRITE string="["/>
    <WRITE-ATTR name="label"/>
    <WRITE string="]"/>
    <FACE type="normal"/> <CHILDREN/>
  </TARGET-ELEMENT>
</STYLE-SHEET>

```

Figure 4. The style-sheet of the example.

mentary concepts associated to an entity. Attributes are represented as small circles, labeled with the name, and connected to the entity they belong to by a solid line.

Usually, attributes of XML tags correspond to attributes of entities, as for attribute `name` of `PAPER`, or attribute `label` of `CITE`. But entity attributes can correspond to the textual content of tags as well. For example, the content of tag `EMPH` is represented by the `content` attribute of entity `EMPH`. Differently, the content of tag `TITLE` is represented by attribute `title` of entity `PAPER`: this is correct, since in the DTD of Figure 1 one and only one instance of tag `TITLE` is mandatory, and this tag contains only generic text.

A special category of attributes is constituted by *key attributes*, represented as black filled circles. Key attributes

are the entity features that uniquely identify each instance of the entity. For example, a paper is identified by attribute `name`, as well as a style specification for a target (entity `TARGET`) is identified by the attribute `t_name`.

Attribute names are associated to a qualifier, which indicates specific properties of the attribute. Qualifiers `(R)` and `(I)` denotes that the attribute is *required* or *implied* (i.e. optional), respectively; as an example, attribute `title` of entity `PAPER` is required, while attribute `num` of entity `NEWLINE` is implied (`I=1` denotes that, if no specified, the default number of new lines is 1). The qualifier `(O)` denotes that the attribute is an *order attribute*, i.e. its value is the order with which the instance of the entity appears in the document. The additional qualifier `U` (unique) can be specified for a non-key attribute (for key attributes it is implicit), meaning that only one instance can have a given value for the attribute (e.g. attribute `label` of `BIBITEM` is implied, but if specified its value must occur for one bibliographic item only). Finally, key attributes are qualified only by `(R)` or `(O)`.

Strong and Weak Entities. There are two types of entities. *Strong Entities* (denoted by thin lines) represent primary concepts in the source documents. This is the case, e.g., of entity `PAPER` (recall that we are considering the case of multiple papers processed at the same time) or style-sheet targets (entity `TARGET`).

Weak Entities (denoted by thick lines) are concepts defined in the context of another concept. For example, a section and a bibliographic item exist in a specific paper, hence they are modeled by the weak entities `SECTION` and `BIBITEM`, that are correlated with the strong entity `PAPER`. A weak entity can be defined in the context of another weak entity: weak entities `PAR`, `CITE`, and `EMPH` are correlated with the weak entity `SECTION`, since they are defined in the context of sections¹. The implicit uniqueness constraint of key attributes of a weak entity X applies in the scope of the stronger entity Y : this is denoted by an edge connecting the key attribute(s) of X to the edge of the relationship connecting X to Y . For instance, the *order* attribute of entity `SECTION` indicates the position of a given section inside a given `PAPER`, thus the same number of section can be associated to multiple sections, provided that they belong to different papers. This is valid also in case of alternatives: inside a section, only one child can have a specific position, either a citation or an emphasized text or a generic text (`PAR`). Finally, consider attribute `label` of entity `BIBITEM`, qualified as `(IU)`: since it is connected to the relationship `BIBLIO` between `BIBITEM` and `PAPER`, the uniqueness constraint is applied only in the scope of a paper (a specific label cannot appear in multiple bibli-

¹Containment relationships are orthogonal w.r.t. weakness of entities. In fact, we might have strong entities (describing primary concepts) whose instances appear inside other entity instances.

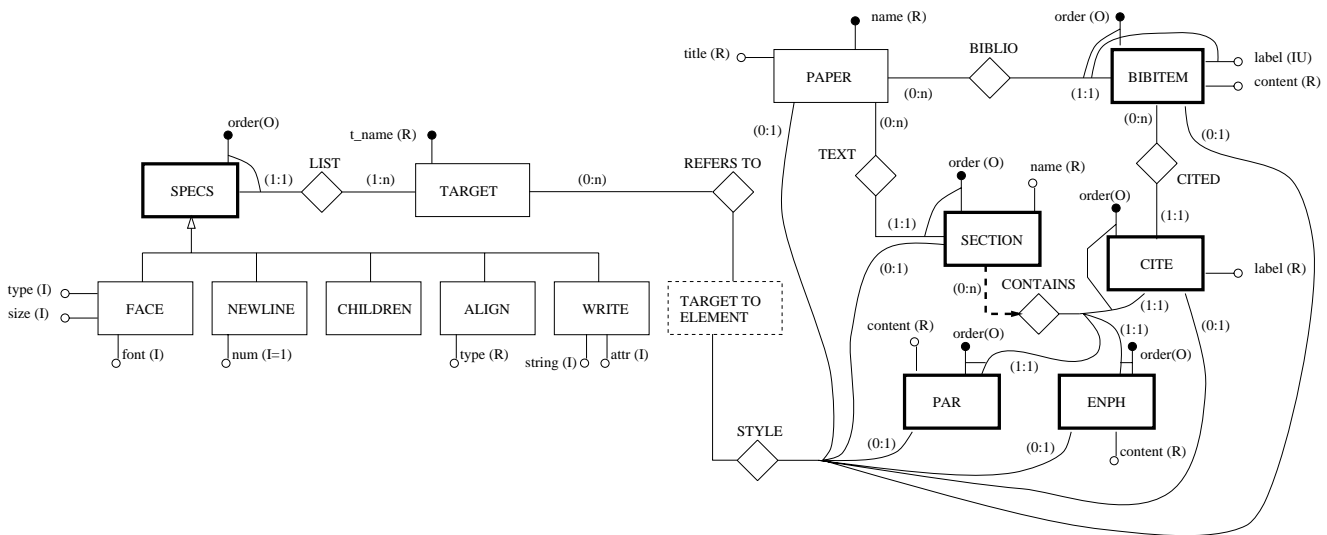


Figure 5. ERX conceptual model for papers and style-sheets.

ographic items of the same paper, but can appear in several papers).

Hierarchies. Specialization hierarchies are possible in ERX. If an entity X is specialized into several sub-entities Y_1, Y_2, \dots, Y_n , this means that an instance of the super-entity X is actually an instance of one of the sub-entities Y_1, Y_2, \dots, Y_n . Key attributes can be specified only in the super-entity X , and all the attributes of X are common to (or inherited from) all the sub-entities, which can have specific attributes. For example, the style for a target is constituted by a sequence of *style specifications* (entity SPECS), which in turn are either face specifications (FACE), or new line specifications (NEWLINE), or references to children (CHILDREN), or alignment specifications (ALIGN), or write instructions (entity WRITE, in which both tags WRITE and WRITE-ATTR are fused); notice the definition of specific attributes for sub-entities, which correspond to attributes of the corresponding tags.

Interfaces. The last feature considered by ERX is the concept of *interface*. An interface divides two parts of the conceptual model that correspond to different classes of source documents, or parts that are homogeneous and semantically different w.r.t. the rest of the model². In the example of Figure 5, the right half of the figure contains the model for papers, the left part of the figure contains the model for the style-sheet. The interface TARGET TO ELEMENT (represented by a dashed line rectangle) plays the following roles: styles for targets (entity TARGET) are correlated with ele-

²This is the case of SOX [6]. SOX documents define the structure of XML documents, by means of a higher level description formalism than DTD. SOX specifications can be completed by remarks, written in HTML. Hence, there are two homogeneous sets of tags: those defining the structure of documents, and those used for remarks.

ments in the paper through the interface, that hides to styles the actual model of targets; on the other side, an element in the paper is correlated with the corresponding style through the interface, that hides the actual model of styles. Note that there is no cardinality constraint on relationship edges connected to the interface: in fact, an interface is not an entity, but a place-holder for entities in the two parts of the model.

This solution provides several advantages. At first, interfaces clearly maintain a semantic distinction between the different parts of the model, but nevertheless they clearly put in evidence the correlations between these parts. Second, interfaces give rise to the concept of *view*: the XML processor may need operate only on a restricted view of the model, ignoring the rest of the model. For example, a sub-task of the processor may need to work only on the view concerning the styles, e.g. to build HTML templates that will be successively instantiated.

Notes. Note that not necessarily all the tags present in the source XML documents have a corresponding entity in ERX. This is the case of tag TITLE: only one occurrence of it is certainly present in the paper, and its informative content is described by a single attribute. Similarly, there is not a BIBLIOGRAPHY entity in the model: tag BIBLIOGRAPHY is in effect a container of bibliographic items; thus, entity BIBITEM suffices to represent this concept.

ERX does not provide specific constructs for concepts like links, navigation, etc., which are concepts typical of web sites and thus too specific. This is coherent with XML, which is meant to provide a common syntactic format to documents and data; web pages and site contents constitute a particular application, but they are not the only one. In fact observe that the three components of a web site (data schema, presentation and navigation) can be designed us-

ing a high level, site-oriented, conceptual model [5]; this model can be described by three distinct, although correlated, classes of XML documents. The XML processors on the server-side may manage such documents by means of the ERX data model; this way ERX can be exploited as document oriented, server-side, data model, supporting the specific site-oriented conceptual model.

4 Conclusions and Future Work

In this paper, we propose ERX, a data model for collections of XML documents; ERX evolves the classical Entity Relationship model, providing specific features suitable to model large collections of XML documents.

ERX is devised to be effective for building advanced XML processors, specifically processors that have to manipulate complex XML documents, or multiple classes of documents at the same time. In particular, the database-like view of the modeled documents offered by ERX is a valuable factor of the proposal, both because it determines a better understanding of the data in the documents, and because it opens the way to the development of applications that exploit the well known and reliable technology of relational DBMSs to manage and manipulate large collections of documents, because that it can easily mapped to the relational model. Furthermore, the database-like view of the collection of documents allows the development of processors that easily access the data from different entry points w.r.t. the unique entry point (the root) provided by the tree structure; this solution avoids the waste of processing time in navigating the tree to search the desired data items, e.g. for building indexes, table of contents, etc; a better specification of the data extraction and manipulation process is another advantage..

Another advantage is that a unique general conceptual model can be used to model document classes which are conceptually similar but structurally different, because defined by different DTDs; this is very important because enhance the data portability.

Future work. We plan to follow several directions. At first, we are going to test the effectiveness of ERX on complex cases coming from industry and/or bank applications; these tests will be the occasion to validate ERX, specifically as far as its completeness is concerned.

From the implementation point of view, we are realizing several versions of the ERX data manager; in particular, we are investigating both a main memory solution and an implementation that exploits a relational DBMS; in fact, this latter solution seems very promising to build complex and distributed information systems based on XML.

A third research line will consider the definition of a formalism to specify XML to ERX transformation rules; this

formalism is the first step toward the development of CASE tools that automatically generate XML-ERX mappers.

References

- [1] S. Adler, A. Berglund, J. Clark, I. Cseri, P. Grosso, J. Marsh, G. Nicol, J. Paoli, D. Schach, H. S. Thompson, and C. Wilson. A proposal for XSL. Technical Report NOTE.XSL.html, World Wide Web Consortium, August 1997.
- [2] C. Batini, S. Ceri, and S. Navathe. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin Cummings, Menlo Park, California, 1992.
- [3] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (xml). Technical Report PR-xml-971208, World Wide Web Consortium, December 1997.
- [4] A. Deutsch, M. Fernandez, D. Florescu, , A. Levy, and D. Suciu. Xml-ql: A query language for xml. Technical Report NOTE-xml-ql-19980819, World Wide Web Consortium, August 1998.
- [5] P. Fraternali, S. Ceri, and S. Paraboschi. Data-driven, one-to-one web site generation for data-intensive applications. In *Proc. 25th VLDB Conference*, Edinburgh, Scotland, September 1999.
- [6] M. Fuchs, M. Maloney, and A. Milowski. Schema for object-oriented xml. Technical Report NOTE-SOX-19980930, World Wide Web Consortium, September 1998.
- [7] L. Lakshmanan, F. Sadri, and I. Subramanian. A declarative language for querying and restructuring the web. In *RIDE-NDS, Computer Society Press*, 1996.
- [8] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification. Technical Report REC-rdf-syntax-19990222, World Wide Web Consortium, February 1999.
- [9] A. Layman, E. Jung, E. Maler, N. H. Mikula, J. Paoli, J. Tigue, H. S. Thompson, and S. DeRose. Xml-data. Technical report, World Wide Web Consortium, December 1997.