# Data Modeling with Markup Languages: A LP Perspective

François Bry and Norbert Eisinger

University of Munich
Institute for Computer Science
http://www.informatik.uni-muenchen.de

WLP 2000, Aug. 26, 2000, Berlin

## CONTENTS:

# 1. From Document Modeling to Data Modeling

Modern markup languages such as SGML and XML are **generic**

➜ they serve to specify the **logical structure** not the layout

➜ they make it possible to freely specify **application dependent data structures**

## Markup

- ➜ **mark up:** typography annotation
- ➜ **markup:** refers to formatting or structuring languages

## Specific and generalized markup languages

- ➜ **specific:** specify the rendering of texts (PostScript, RTF)
- ➜ **generalized:** specify the logical structure and map it into formatting (DCF GML from IBM, 1970)

## Layout vs. Logical Structure

### Layout

- ➜ rendering of a document on paper, on screen, or with synthesized voice

### Logical Structure

- ➜ Application dependent data structuring = data modeling

Goals of generic markup languages

➜ Independence of data modeling from data usage

➜ Independence of data modeling from presentation devices

## Free Structures with Free Naming – Structure Conveying Data

```
<person>
   <first-name>Francois</first-name>
   <last-name>Bry</last-name>
   <geographical-address>
      <street>rue de Vincennes</street>
      <street-number>70</street-number>
      <city>Bordeaux</city>
      <state>France</state>
   </geographical-address>
   <email-address>bry@hotmail.com</email-address>
   <phone-number>0033556988223</phone-number>
</person>
```

Data Modeling Facilities of Generic Markup Languages:

- ➜ complex structures as they can be specified with object oriented knowledge representation formalisms
- ➜ data sharing like in object database models
- ➜ optional elements
- ➜ different kinds of pointers

Complex Modeling:

- ➜ Exceptions – Semistructured Data

## Pointers and Hypertext with XLink

- ➜ n-ary, directed or undirected hypertext links
- ➜ out-of-line hypertext links
- ➜ non-hypertext links

XML + XLink convenient for implementing ontologies

- ➜ data dictionary
- ➜ thesaurus
- ➜ commented data schema
- ➜ sort of Entity-Relationship model

# 2. An "XML Ontology"

From SGML to XML

- ➜ no "markup minimization" such as

```
<ul>
   <li> First list item
   <li> Second list item
</ul>
```

- ➜ Companion hypertext model (XLink)

The acronym XML denotes

- ➜ a markup language (XML proper)
- ➜ a family of related languages and formalisms (DTD, XML Schema, XPath, XPointer, XLink, style sheet, CSS, XSL, XSLT, XSL-FO, XQL, etc.)

Special Purpose Markup Languages defined in XML, e.g.:

- ➜ Bioinformatic Sequence Markup Language, BSML
- ➜ Channel Definition Format, CDF
- ➜ Extensible HyperText Markup Language, XHTML 1.0
- ➜ Mathematical Markup Language, MathML
- ➜ Open Software Description, OSD
- ➜ Platform for Privacy References, P3P
- ➜ Precision Graphic Markup Language, PGML
- ➜ Synchronized Multimedia Integration Language, SMIL
- ➜ Scalable Vector Graphics, SVG
- ➜ Vector Markup Language, VML
- ➜ HTTP Extensions for Distributed Authoring, WEBDAV

Uses of XML within the XML Specification

- ➜ XML Schema
- ➜ XLink

Uses of XML in XML Related Specifications

- ➜ XSL Transformations, XSLT 1.0
- ➜ Formatting Objects of XSL, XSL-FO
- ➜ Document Content Description for XML, DCD
- ➜ Resource Description Framework, RDF

# 3. XML BASICS

## Document Prolog and Document Instance

```
<?XML version="1.0" encoding="iso-8859-1"?>
<?XML:stylesheet href="http://..." type="text/xsl"?>
<address-book>
   <card>
      <last-name>Eisinger</last-name>
      <first-name>Norbert</first-name>
      <email>eisinger@informatik.uni-muenchen.de</email>
      <phone>08921782218<phone>
   </card>
   <card>
      <last-name>Bry</last-name>
      <first-name>Francois</first-name>
      <fax>08921782211</fax>
   </card>
</address-book>
```

## Opening Tags, Closing Tags, and Elements

```
<address-book>
   <card>
      ...
   </card>
   <card>
      <last-name>Eisinger</last-name>
   </card>
</address-book>
```

## Empty Elements

```
 <a></a>

 <a/>
```

## Document Tree Nodes vs. Elements

```
<doc>
   <title>A Document</title>
   This is element free text.
   <chapter>This is a chapter</chapter>
</doc>
```

## Well-Formed Documents

- Illegal:

```
<a> ... <b> ... </a> ... </b>
```

- Legal:

```
<a> ... <b> ... </b> ... </a>
```

## Attributes

```
<card status="important">
   <name>Eisinger</name>
   <phone country="Germany">08921782218<phone>
</card>
```

## Attributes are controversial

```
<card>
   <status>important</status>
   <name>Eisinger</name>
   <phone>
      <country>Germany</country>
      <number-in-country>...</number-in-country>
   <phone>
 </card>
```

## Quoting in XML

- `title="ma'am"`
- `title='ma"am'`

## Defining Special Purpose Markup Languages in XML

- ➜ Document Type Definition (DTD)
- ➜ XML Schema

## Example of a DTD:

```
<!DOCTYPE address-book
    [<!ELEMENT address-book (card*)>
     <!ELEMENT card          (last-name, first-name,
                              (email | fax), phone?)>
     <!ELEMENT last-name     #PCDATA>
     <!ELEMENT first-name    #PCDATA>
     <!ELEMENT email         #PCDATA>
     <!ELEMENT fax           #PCDATA>
     <!ELEMENT phone         #PCDATA>
    ]>
```

- ➜ Internal DTD vs. External DTD
- ➜ #PCDATA
- ➜ Valid Documents
- ➜ Abstract Syntax vs. Concrete Syntax
- ➜ DTDs and Context-Free Grammars
- ➜ Namespaces

## Example of an XML Schema Specification

```
<xsd:complexType name="address">
  <xsd:element name="first-name"    type="xsd:string"/>
  <xsd:element name="last-name"     type="xsd:string"/>
  <xsd:element name="street"        type="xsd:string"/>
  <xsd:element name="street-number"
                           type="xsd:positiveInteger"/>
  <xsd:element name="city"          type="xsd:string"/>
  <xsd:element name="postal-code"   type="xsd:string"/>
  <xsd:attribute name="country"    type="xsd:NMTOKEN"/>
</xsd:complexType>
```

Element compelling to the previous specification

```
<address country="fr">
   <first-name>Francois</first-name>
   <last-name>Bry</last-name>
   <street>rue de Vincennes</street>
   <street-number>70</street-number>
   <city>Bordeaux</city>
   <postal-code>F-31000</postal-code>
</address>
```

- Localization Formalisms of XML
  - XPath for nodes
  - XPointer for nodes and ranges

- Unix-like path notation such as
  `~bry/XML/DocTypes/dtd/addbook.dtd`

- The XPath expression

  ```
  address[street='rue de Vincennes']
  ```

  characterizes elements of type "`address`" that have a subelement of type "`street`" containing the string "`rue de Vincennes`"

- XPointer was intended as an extension of XPath. Both formalisms evolved independently from each other. XPath is *not* a subset of XPointer.

## Style Sheets

➜ attach formatting, or rendering properties to documents

➜ Cascading Style Sheets, CSS1,CSS2
Extensible Stylesheet Language, XSL

## CSS

```
em     {font-style: italic; font-weight: bold; }
strong {text-transform: uppercase; font-weight: bold; }
h2     {text-transform: lowercase; }
h1     {font-size: 200%; }
body   {font-family: Verdama, sans-serif; }
```

## XSL consists of

- XSL transformation, XSLT, a simplification of DSSSL
- XSL Formatting Objects, XSL-FO

## XLink: A Kind of Entity Relationship Model

- link = relationship between n resources
- links are not necessarily directed
- arcs specify "traversal semantics"
- links are not necessarily hypertext links
- elements of all types can be links

## Hypertext traversal semantics

- `actuate="onRequest" "onLoad"`
- `show="replace" "new" "embed"`

## Linkbases

- "outbound"
- "inbound"
- "third-party"

## Query Languages à la SQL

- ➜ XML-QL
- ➜ YATL
- ➜ Lorel
- ➜ Quilt
- ➜ XQL

First 4 languages: **selector** (or pattern) and **constructor** parts

Constructors make **complex restructuring** possible, e.g.

Bibliography sorted by years $\longrightarrow$ Bibliography sorted by authors

# 4. Data Modeling with Markup Languages and LP

eXtended Logic Programming (XLP) Scheme

- ➜ Viewing XML elements and documents as extended terms
- ➜ Using rules similar to Prolog clauses for expressing element and data item transformations, views, and queries

A Language for Rigid and Flexible Terms

The **rigid term** `a(..., b(...), ...)` is similar to the element

```
<a>
    ...
    <b>
        ...
    </b>
    ...
</a>
```

**Flexible terms/elements** for data selection, e.g.:

```
<a>
   ...
   <meta type="_" occurrences="*">
      <b>
         ...
      </b>
   </meta>
   ...
</a>
```

Issues investigated:

- ➜ Term-Oriented vs. Path-Oriented Element Localization
- ➜ Unification for Term-Oriented Localization and Parsing
- ➜ Negation as Failure in Query and Transformation Languages

Further Research Issues Related to Logic Programming:

- ➜ Proof Methods and Strategies
- ➜ Meta-level Data Modeling and Meta-Programming
- ➜ Applications of Logic Grammars

# Final Remarks

- ➜ Data Modeling with Markup Languages (DM$^2$L) http://www.pms.informatik.uni-muenchen.de/forschung/datamodeling-markup.html
- ➜ comments are welcome
- ➜ visitors are welcome