Storage Techniques and Mapping Schemas for XML

Sihem Amer-Yahia

sihem@research.att.com

AT&T Labs – Research
Motivation and Outline

Motivation:
Flexibility of representation in XML induces multiple possible storagemappings into relational/object back-ends. Applications that need access to stored data would benefit from transparency.

Using Schemas (Schema-less, DTD, XML Schema).

Capturing Identity, Structure and Order.

Commercial Tools:
• MXM.

Mapping usually hand-coded into storage system and not

Flexible or representation in XML induces multiple possible

Outline:
Identity, Structure, and Order

KFO: Foreign key in child element, ordinal value for sibling order.

INTERVAL: Records at node path from the node to document root.

DEWEY: Records at node path from the node to document root.

KEY/Foreign Key Joins with substring comparisons to
documents. Stack-based joins with interval inclusion to
document IDs to distinguish nodes from different
descendants. Level number to distinguish children from
descendants.

EDGAR: Attribute, Universal, LedgerDB commercial solutions.

LDAP: Stack-based joins with substring comparison to
document structure. Most complete but depends on depth in
document structure.
Identity, Structure and Order:

Path stored as a string and substring matching used in joins:

- XRel
- Path [pathID, pathexp]
- Element [docID, pathID, start, end, index, reindex]
- Attribute [docID, pathID, start, end, value]
- Text [docID, pathID, start, end, value]

Reduces number of joins needed to recover document structure. Uses B+ trees and R trees.

Index records document order and reverse index records reverse document order.

Between shared paths:

- Index records document order and reverse document order to distinguish start and end region of each node.

Path stored as a string and substring matching used in joins.
Generic Mappings: No schema (Edge, Attribute, Universal).

Fixed Mappings: DTD-driven (Basic, Shared, Hybrid). One relation per element. Variations in inlining/outlining elements.


Generic Mappings: No schema (Edge, Attribute, Universal).

Using Schemas.
Commercial Tools

Relational back-ends:

- Tailored to one particular system. Cannot be used for other systems.
- IBM and Oracle can validate stored documents. Microsoft cannot.

IBM supports storing XML documents in CLOBs with side tables to index structure.

- IBM supports storing XML documents in CLOBs with side tables to index structure.
- Hard-code defaults such as KFO for document structure and attribute initialization.
- Use a mapping language.
can be used to validate XML documents. Users can access this table to insert their own DTDs. DTDs XML DTD Repository stores meta information on mappings.

•

XML DTD Repository stores meta information on mappings.

•

path, value).

•

&image;

•

Annotate a simplified XML Schema with mapping

•

IBM DB2 XML Extender
<DAD>
  <Xcollection>
    <root_name>
      <element_nodename=''PATIENT''>
        <attribute_nodename=''IDNum''>
          <RDB_node>
            <tablename=''Patient_tab'' />
            <columnname=''Patient_key'' />
            <condition>IDNum>635</condition>
          </RDB_node>
        </attribute_node>
      </element_node>
    </root_name>
  </Xcollection>
</DAD>
Microsoft SQL Server

3 publishing modes: RAW, AUTO, and EXPLICIT.

3 storage solutions: Edge, XSD, and OpenXML.

Annotated schemas XSD (successor to the XML-Data Reduced (XDR) schema definition language). Implemented in SQLXML which includes an XDR to XSD converter tool.

XPath used to decompose documents into tables.

OpenXML compiles XML documents into DOM. XPath used to decompose documents into tables.

Select * from OpenXML(@pat, '/HL7/PATIENT', 1)

Used both for publishing and for storage.

Describes table and column names. Contains embedded SQL.

WITH IDNum int, GtNA varchar(20)
(SELECT * FROM OpenXML( @pat, '/HL7/PATIENT', 1) )
<schema>
  <$annot>
    <$appinfo>
      <$sql:relationship name="Patient_OBX" parent="Patients" parent-key="PatKey" child="OBX" child-key="OBXKey"/>
    </appinfo>
  </annot>

  <$element name="PATIENT" sql:relation="Patients">
    <$complexType>
      <$sequence name="PaNa">
        <$element name="FaNa" sql:field="LastName" type="string"/>
        <$element name="GiNa" sql:field="FirstName" type="string"/>
      </sequence>
      <$attribute name="IDNum" sql:field="PatientID" type="integer"/>
    </complexType>
  </element>
</schema>
on demand. XPath optimized using Brees.

XML Attributes. Lazy materialization reads DOM nodes.

SOL to generate documents: XMLTELEMENT, XMLMAP, XMLTEMPLATE.

System or user-generated (by annotating an XML Schema).

Oracle 9iR2 introduced Oracle XML DB. Mappings are

Oracle 9iR2 introduced Oracle XMLDB. Mappings are

XPath expression, extractNode().

Operators associated to XMLType: extract() evaluates

URI-Ref) Functions to shred documents into tables.

New datatypes for XML (XMLType) and for logical pointers

XPath processors, XSLT Processor, XSLT to generate documents.


Oracle 9iR2
Oracle9iR2

http://www.oracle.com/HL7.xsd', doc) end;

BEGIN dbms_xmlescema.registerschema(

, schema/>

complexType>

<sequence/>

</complexType>

<element name="FaNa" type="string"/>

<element name="OBX" type="string"/>

<element name="OBZ" type="string"/>

declare doc varchar(5000) :=

DECLARE doc VARCHAR(5000) :=

Oracle9iR2
MXM: Goal and Design Choices


Orthogonal Design. Express existing mappings and more!

Map schema-less documents, documents conforming to an XML Schema, and documents conforming to a DTD.

Capture existing XML-to-Relational mappings and express them in a declarative language. Make them accessible through an interface. Do not hard-code defaults.
MXM: Architecture

XML applications

XMl

XQUERY to SQL

RS

RDB

Loading

XML (relational schema)

Creation

APi

Repository mapping

Mapping

Mapping

Processing

Mapping

DTD or XML Schema

MIS
<!DOCTYPE addressBook[
<!ELEMENT addressBook (#PCDATA)[]>
<!ELEMENT fullname (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
<!ENTITY addressBookContent (fullname, telephone)>
<!ENTITY phone (fullname, telephone)>
<!ENTITY phone (fullname, telephone)>
<!ENTITY phone (fullname, telephone)>
<!ENTITY owner CDATA>
<!ELEMENT addressBook owner CDATA>
<!ELEMENT addressBook addressBookContent>
<!ELEMENT addressBook addressBookContent>
]>

MXM: Example DTD
mapping information related to a particular element).

Granularity of API depends on application (e.g., return all

Query defaults (e.g., `getDefTabNaming()`).

- `getTableName(TableName)`
- `getFieldType(FieldName)`
- `getFields(TableName)`
- `getStructMap()`
- `getCLOBName(TableName)`
- `getElemIds(TableName)`
- `getElemIds(TableName|ElemName)`
- `getInstance(TableName|ElemName)`
- `getElemIdMap()`
- `getEnumName|ElemName`
MXM: Implementation

- Implementation
- XML document
- SAX parser
- ASCII file
- ASCII file
- Batch loading
- MySQL
- RDB
- RS loading programs
- IMXM calls
- IMXM generation

Repository

RTM

IMXM

Relational schema

Relational schema
Identity fragments of documents to exchange:

XML data exchange: Avoid building entire XML documents and test different XML-to-relational mappings.

Tuning mappings: Modify declarative mapping specification and implement mappings on top of commercial solutions.

• In XML Schema.
• Incorporate constraints on schemas (e.g., cardinality constraints in native systems).
• Extend mapping specification to other backends (e.g., LDAP).

Implement mappings on top of commercial solutions.