Full Text Retrieval Systems, XML and Databases - Where are Future Information Architectures heading?

Stefan Baerisch
GESIS / IZ Social Science Information Centre
Bonn, Germany

EuroCRIS IR Workshop, November 9, 2006
Outline

- Local data integration
  - Challenges of data integration
  - Reasons for XML
  - The SOWIPORT schema
- XML retrieval
  - Retrieval scenarios
  - Full-text retrieval systems
- Conclusion
Data Integration > Situation

• Local data integration is one way to offer users integrated search
• Partners use different data models
  • Text
  • Relational
  • XML
• Schemas are changing over time
• Data conversion by partners proved problematic
  • Postprocessing was necessary
  • Details were hard to communicate
  • Changes had to be coordinated
Data Integration > SOWIPORT

- Example SOLIS & FORIS
  - Currently about 400,000 documents
  - Focus on research (FORIS) and publications (SOLIS)
  - Extensive background information
    - Institutions, persons
- 370 tables
  - 10+ references per table
- Business rules
• Text is most common format for data deliveries by partners
• Encoding differs
  • UTF-8 best suited
  • ISO 8859-1 most common
  • Encoding errors are still frequent
• Flat as well as structured
• Lack of schema information
  • Print-to-file used for export
  • Different fields used for the same attribute
  • Relational information is often lost
• No documentation exists
The common subset for publications is small
  - Title
  - Person
  - Year
  - Source
  - Subject / keywords

Many optional fields exist
  - Institutions
  - Alternative titles, etc
  - Abstracts
  - Full-texts
  - Citations
Data Integration > Schema Evolution

- Most changes to the database require schema modifications
- Data integration requires frequent updates
- Alternatives exist, but are problematic
  - Storing documents in one field
  - Storing single values as tuples

<table>
<thead>
<tr>
<th>ID</th>
<th>Docid</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Title</td>
<td>Rain in Ireland</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Author</td>
<td>Schmidt</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Issued</td>
<td>1995</td>
</tr>
</tbody>
</table>
Data Integration > Hierarchical Values

- One-to-Many relationships are common within one document
  - Authors, controlled terms
  - The child-elements are often single values
- Relational databases are suboptimal for hierarchical data
  - References are required
  - SQL is not well suited for quick data exploration
  - Object relational mappings may help

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Youth and Work</td>
<td>Meier</td>
</tr>
<tr>
<td>2</td>
<td>The Effects of Weather</td>
<td>Miller</td>
</tr>
<tr>
<td>3</td>
<td>Rain in Ireland</td>
<td>Schmidt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Docid</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Meier</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Müller</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Miller</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Schmidt</td>
</tr>
</tbody>
</table>
Data Integration > Null Values

• Schema with many optional fields are common
• Relational schemas have to include every field
  • for every document
• Different relations can be used for co-occurring subsets of fields but this raises the complexity of the schema

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Author</th>
<th>Institution</th>
<th>Journal</th>
<th>Volume</th>
<th>Issue</th>
<th>Doctype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Journal on Waterways</td>
<td>NULL</td>
<td>London Publishing</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>Journal</td>
</tr>
<tr>
<td>2</td>
<td>Water and Snow</td>
<td>NULL</td>
<td>Research Foundation</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>Book</td>
</tr>
<tr>
<td>3</td>
<td>Rain in Ireland</td>
<td>Schmidt</td>
<td>NULL</td>
<td>Meteorology Review</td>
<td>10</td>
<td>8</td>
<td>Article</td>
</tr>
</tbody>
</table>
Data Integration > Semi Structured Data

• Schema is optional, implicit
• Self-describing
• Types are supported

```xml
<Publications>
  <Article>
    <Title>Rain in Ireland</Title>
    <Author>Schmidt</Author>
    <Journal>
      <Title>Meteorology Review</Title>
      <Volume>10</Volume>
      <Issue>8</Issue>
    </Journal>
  </Article>
  <Book>
    <Title>Water and Snow</Title>
    <Institution>Research Foundation</Institution>
  </Book>
  <Journal>
    <Title>Journal on Waterways</Title>
    <Institution>London Publishing</Institution>
  </Journal>
</Publications>
```
Data Integration > The Role of XML

• XML can express other data models
  • Relational
  • Flatfiles
• It makes a good format for data exploration

XMarc example record
Data Integration > Goals for the SOWIPORT Schema

• A common schema / set of fields
  • For all collections
  • For all entities
• Include all elements used for:
  • Search for / display of information
  • Linking to full-text
• Ability to scale
  • Full-text
  • New entities
  • Different levels of detail
Data Integration > Existing Metadata Standards

• Dublin Core
  • Creator and contributor <-> person and institution
  • Lacks structure

• XMarc
  • Does not support all entity types
  • Not really structured
  • Not human readable / not self documenting

• Mods
  • Does not support all entity types
  • Lacks necessary fields
  • Controlled vocabulary was not sufficient
Different focus than our existing legacy databases
Data from legacy / harvested databases are not suitable for automatic conversion to CERIF
Not all legacy formats can be expressed
But: CERIF as a exchange and view format
Data Integration > The SOWIPORT Schema

• One common set of fields for all entities
• New entity: existing fields are reused, only new fields are added to the schema
Different use cases have to be considered for retrieval on XML-databases

- Structured queries by experts
- Ad-hoc queries by novices
- Automated queries for statistical purposes

Different views on the same data are possible

- Search for publications
- Search for persons
In combination, modern full-text retrieval systems and XML-databases can fulfill all use cases.

Full-text retrieval systems offer ranking in accordance to the users information need.

- Ranking by different criteria is supported
- Fielded search allows views with different levels of granularity

XML-databases allow for complex queries using the XQuery language.
XML Retrieval > Technology

X-Hive

- Native XML database
- XQuery support
- Versioning
- Full-text indices

Fast

- Scalable
- Structured query support
- Strong relevance concepts
• Modern full-text retrieval systems support a wide range of relevance concepts:
  • Freshness
  • Authority
  • Quality
  • Proximity
  • Content

• The weight of the concepts and the weight of document parts can be modified
Many systems support the Extended Boolean retrieval model

- AND
- OR
- NOT

Additional operators are available

- RANK
- FILTER
- PROXIMITY
- RANGE
Modern full-text retrieval systems allow field-based retrieval

Some systems support composite fields

```xml
<Field name="Content">
  <Field name="Person">
    <Field name="Author"/>
    <Field name="Editor"/>
  </Field>
</Field>
<Field name="Text">
  <Field name="Abstract"/>
  <Field name="Title"/>
</Field>
<Field name="ControlledTerms">
  <Field name="Subject"/>
  <Field name="Classification"/>
</Field>
</Field>
```
Some retrieval systems allow queries over arbitrary XML

Find all texts with a person named Meier:

- Text: Person:"Meier"

Find texts contain the term ‘influence’

- Text:"influence"
XQuery Retrieval

• XQuery FLOWR expressions allow powerful queries and transformations
  • For
  • Let
  • Order By
  • Where
  • Return

```xml
let $person := $book/person
where $person/role = "author"
order by $person/name/surname
return <authors>{$person/name}</authors>
```

• Since XQuery is a superset of XPath 2.0, queries can also be written in XPath.
XML Retrieval > XQuery and Full-Text Retrieval

• The W3C XQuery full-text extension defines a rich set of search operations for full-text
  • Thesaurus support
  • Relevance ranking/scoring
  • Wildcard search
• Implementations are still lacking
• Most XML databases offer simple, proprietary full-text extension.
• Most offer a Boolean retrieval model
<authlist>{
    for $a in fn:distinct-values($bib/book/author)
    order by $a
    return
        <author>
            <name> {$a} </name>
            <books>{
                for $b in $bib/book[author = $a]
                order by $b/title
                return $b/title
            }</books>
        </author>
} </authlist>

Example from the W3C XQuery Specification
Conclusion

XML is well suited for data integration tasks
  • Other data models can be expressed
  • Continuous refinement of the transformation process is possible

With XML Databases and full-text retrieval systems, current use cases for XML-retrieval are well supported
  • Complex XQuery requests
  • Ranked full-text queries
  • Boolean expressions and field-based search