Structured Documents and their Manipulation in Database Systems

Karl Aberer
GMD-IPSI

aberer@darmstadt.gmd.de

http://www.darmstadt.gmd.de/IPSI

joint work with Klemens Böhm, Erich Neuhold, Yangjun Chen, Dunren Che, Michael Fuchs, ....
1. **Database support for SGML/HyTime documents**
2. **Management of document objects**
   - Document object modelling
   - Implementation of the object model
   - Object-Relational implementation and experiments
3. **Generic application framework**
   - Document parsing and structure creation
   - Document modification
   - Declarative access
   - Document query optimization
   - Document rendition
4. **HyTime support**
5. **Conclusions**
SGML/XML

<?XML version="1.0"?>
<!DOCTYPE play PUBLIC "-//Free Text Project//DTD Play//EN">
<PLAY>
<TITLE>The Tragedy of Hamlet, Prince of Denmark</TITLE>
<fm>
<p>This work may be freely copied and distributed worldwide.</p>
</fm>

<Personae>
<TITLE>Dramatis Personae</TITLE>
<Persona>CLAUDIUS, king of Denmark. </Persona>
<Persona>HAMLET, son to the late, and nephew to the present king. </Persona>
</Personae>
<Scndescr>SCENE  Denmark. </Scndescr>
<Playsubt>HAMLET</Playsubt>
<Act>ACT I</Act>
<Scene>SCENE I.  Elsinore. A platform before the castle. </Scene>
<Stagedir>FRANCISCO at his post. Enter to him BERNARDO</Stagedir>
<Speech>BERNARDO</Speech>
<Line>Who's there?</Line>
<Speech>FRANCISCO</Speech>
<Line>Nay, answer me: stand, and unfold yourself.</Line>
....
Documents as structured data

© Karl Aberer, 1999
<!-- DTD for Shakespeare  J. Bosak  1994.03.01, 1997.01.02 -->
<!ELEMENT play     - - (title, fm, personae, scndescr, playsubt, induct?,
                                prologue?, act+, epilogue?)>
<!ELEMENT title    - - (#PCDATA)>  <!ELEMENT fm       - - (p+)>
<!ELEMENT personae - - (title, (persona | pgroup)+)>  <!ELEMENT pgroup  - - (persona+, grpdescr)>
<!ELEMENT induct   - - (title, subtitle*, (scene+|(speech|stagedir|subhead)+))>
<!ELEMENT act      - - (title, subtitle*, prologue?, scene+, epilogue?)>
<!ELEMENT scene    - - (title, subtitle*, (speech | stagedir | subhead)+)>
<!ELEMENT prologue - - (title, subtitle*, (stagedir | speech)+)>
<!ELEMENT epilogue - - (title, subtitle*, (stagedir | speech)+)>
<!ELEMENT speech   - - (speaker+, (line | stagedir | subhead)+)>
<!ELEMENT line     - - (stagedir | #PCDATA)+>
Document type definitions

Document type definition (DTD)
- element type: play, act, scene, speech, line, ...
- attributes
- content model

\[ c \rightarrow <\text{element-type name}> \]
\[ c_1, c_2 \quad \text{(SEQ connector)} \]
\[ c_1 | c_2 \quad \text{(OR connector)} \]
\[ c_1? \]
\[ c_1^* \quad \text{(optional occurrence indicators)} \]
SGML/XML summary

Two views on SGML/XML documents
• text with markup (e.g. presentation, consistency)
• structured data (e.g. metadata, querying)

Consequences
• applicable data management techniques
• interpretation context dependent

Success of XML
• text/data duality
• flexible use of schemas
• standardization

our focus: structural aspects of SGML/XML documents
Data management for structured documents

Goal
• navigational access
• declarative access
• document modification
• document presentation

taking into account structure, constraints, behavior of SGML documents (HyTime/XML/XLL)

Approach
• object modelling
• flexible physical representation
• configurable, generic application
• database management system
2.1 Document object modelling

SGML document elements represented by DB objects

- structure
  - content
  - SGML attributes
  - structural relation to other elements

- behavior
  - navigation in the document graph (up, down, left, right)
  - querying content/attributes (regular expressions)
  - basic element queries (getFirst, getAll)
  - physical properties (size)
  - textual retrieval
  - content interpretation (references, HyTime)
  - updates
Basic object model (VML/VODAK)
2.2 Implementation of the object model

Straightforward:
elements = database objects, element types = classes
  • individual semantics of different element types easy to reflect
  • modifications are feasible at fine granularity
  • inefficient for many operations

Alternatives?
  • Storing documents as uninterpreted bytestrings
  • on-demand interpretation by means of parsing
  • use of text indexing structures, e.g. PAT
Straightforward object representation

The Tragedy of Hamlet, Prince of Denmark

Scene Denmark

Act I

Scene 1

PLAY

PLAYTITLE

THE TRAGEDY OF HAMLET, PRINCE OF DENMARK

ACT

ACTTITLE

Act I

SCENE

SCENETITLE

Scene 1

STAGEDIR

STAGEDIR

SPEECH

SPEAKER

BERNARDO

Who is there?

PERSONAE

CLAUDIUS

VOLTIMAND

curtiers

GROUP

GROUP

GRPDSCR

HAMLET

TEXT PLACED

.......

Worldwide

P

P

Dramatis Personae

PLAY

PLAYSUBT

Scene

SCENEDESC

Scene Denmark

Spanish

Line

Line

Line

Line

Line

Line

Line
The Tragedy of Hamlet, Prince of Denmark

Dramatis personae
Claudius

Act I
Scene

<play>
<playtitle>The Tragedy of Hamlet, Prince of Denmark</playtitle>
<act title>Act I</act>
<scene>Scene</scene>
</play>

<html>
<body>
<xml>
<play>
<playtitle>The Tragedy of Hamlet, Prince of Denmark</playtitle>
<act title>Act I</act>
<scene>Scene</scene>
</xml>
</body>
</html>
Modified database schema

Virtual instances

Virtual classes

Metaclasses

Application Classes

Objects

<acttitle> Act I </acttitle>
<scene> ... </scene>

© Karl Aberer, 1999
Object identification

Use element position for identification
  • requires completion of markup
  • elements accessible by linear parsing
  • method interpreter handles extended OID's
Access methods for flat elements

Completion of markup allows to compute navigational methods without use of DTD
  • Example: get next element

Optimized method implementations for flat elements
  • Example: getAll (get all elements of certain type contained in the element receiving the method call)
  • original implementation: recursive
  • on flat elements: linear scan
2.3 Object-relational implementation

Analogous object model realized as Informix DataBlade
- simplified relational, structural model
- same fragmentation technique
- same functional interface
## Relational Database Schema

### FlatElemTable
- **own_id**: 0
- **doc**: 1
- **up**: -
- **succ**: -
- **pred**: -
- **own_id**: 1
- **name**: Dok1
- **dtdref**: 2
- **root**: 0
- **own_id**: 2
- **name**: Dok2
- **dtdref**: 1
- **root**: 2
- **own_id**: 3
- **name**: Dok3
- **dtdref**: 1
- **root**: 9

### NonFlatElemTable
- **down**: 1
- **etName**: E0
- **own_id**: 3
- **etName**: E2
- **own_id**: 4
- **etName**: E3
- **own_id**: 6
- **etName**: E5
- **own_id**: 8
- **etName**: E7
- **own_id**: 10
- **etName**: E9

### DocElemTable
- **own_id**: 0
- **doc**: 1
- **up**: -
- **succ**: -
- **pred**: -
- **own_id**: 1
- **doc**: 1
- **up**: 0
- **succ**: -
- **pred**: -
- **own_id**: 2
- **doc**: 2
- **up**: 2
- **succ**: -
- **pred**: -
- **own_id**: 3
- **doc**: 2
- **up**: 2
- **succ**: 5
- **pred**: -
- **own_id**: 4
- **doc**: 2
- **up**: 3
- **succ**: -
- **pred**: -
- **own_id**: 5
- **doc**: 2
- **up**: 2
- **succ**: 7
- **pred**: 3
- **own_id**: 6
- **doc**: 2
- **up**: 3
- **succ**: 7
- **pred**: -
- **own_id**: 7
- **doc**: 2
- **up**: 2
- **succ**: -
- **pred**: 5
- **own_id**: 8
- **doc**: 7
- **up**: -
- **succ**: -
- **pred**: -
- **own_id**: 9
- **doc**: 3
- **up**: -
- **succ**: -
- **pred**: -
- **own_id**: 10
- **doc**: 3
- **up**: 9
- **succ**: -
- **pred**: -

### DocumentTable
- **own_id**: 1
- **name**: Dok1
- **dtdref**: 2
- **root**: 0
- **own_id**: 2
- **name**: Dok2
- **dtdref**: 1
- **root**: 2
- **own_id**: 3
- **name**: Dok3
- **dtdref**: 1
- **root**: 9

### attrRecTable
- **element**: 2
- **name**: Attr2
- **value**: AW2
- **element**: 3
- **name**: Attr3
- **value**: AW3

### DocumentTypeTable
- **own_id**: 1
- **name**: E0
- **etypes**: DTD1
- **config**: {...}
- **root**: "...",
- **own_id**: 2
- **name**: E2
- **etypes**: DTD2
- **config**: {...}
- **root**: "...",
- **own_id**: 3
- **name**: E3
- **etypes**: DTD3
- **config**: {...}
- **root**: "..."
The Tragedy of Hamlet, Prince of Denmark

Scene Denmark

Act I

Scene I...

Scene II...

Playwright

Who is there?
Experimental Results

Documents from Hamlet play, one document ~ 300 KB

• retrieve all elements of a single document

<table>
<thead>
<tr>
<th>getAll (1 document)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
<th>fine fragm. (5000 x 0.22 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE (20 elements)</td>
<td>2.0 s</td>
<td>16.7 s</td>
<td>52.0 s</td>
<td>4609.0 s</td>
</tr>
<tr>
<td>STAGEDIR (134 elements)</td>
<td>1.0 s</td>
<td>16.3 s</td>
<td>53.3 s</td>
<td>4546.0 s</td>
</tr>
<tr>
<td>LINE (4014 elements)</td>
<td>1.7 s</td>
<td>17.0 s</td>
<td>53.3 s</td>
<td>4556.0 s</td>
</tr>
</tbody>
</table>

• retrieve all elements from all documents

<table>
<thead>
<tr>
<th>getObjs (9 documents)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT (60 KB)</td>
<td>21.0 sec</td>
<td>3.5 sec</td>
<td>3.5 sec</td>
</tr>
<tr>
<td>SCENE (15 KB)</td>
<td>21.0 sec</td>
<td>71.0 sec</td>
<td>4.0 sec</td>
</tr>
</tbody>
</table>

• navigation at element level

<table>
<thead>
<tr>
<th>10 x succ(succ(succ(succ()))))</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE</td>
<td>12.0 sec</td>
<td>5.0 sec</td>
<td>15.0 sec</td>
</tr>
<tr>
<td>LINE</td>
<td>8.0 sec</td>
<td>4.0 sec</td>
<td>3.0 sec</td>
</tr>
</tbody>
</table>
Documents from Hamlet play, one document ~ 300 KB

- retrieve all elements of a single document

<table>
<thead>
<tr>
<th>getAll (1 document)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
<th>fine fragm. (5000 x 0.22 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE (20 elements)</td>
<td>2.0 s</td>
<td>16.7 s</td>
<td>52.0 s</td>
<td>4609.0 s</td>
</tr>
<tr>
<td>STAGEDIR (134 elements)</td>
<td>1.0 s</td>
<td>16.3 s</td>
<td>53.3 s</td>
<td>4546.0 s</td>
</tr>
<tr>
<td>LINE (4014 elements)</td>
<td>1.7 s</td>
<td>17.0 s</td>
<td>53.3 s</td>
<td>4556.0 s</td>
</tr>
</tbody>
</table>

- retrieve all elements from all documents

<table>
<thead>
<tr>
<th>getObj (9 documents)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT (60 KB)</td>
<td>21.0 sec</td>
<td>3.5 sec</td>
<td>3.5 sec</td>
</tr>
<tr>
<td>SCENE (15 KB)</td>
<td>21.0 sec</td>
<td>71.0 sec</td>
<td>4.0 sec</td>
</tr>
</tbody>
</table>

- navigation at element level

<table>
<thead>
<tr>
<th>10 x succ(succ(succ(succ())))</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE</td>
<td>12.0 sec</td>
<td>5.0 sec</td>
<td>15.0 sec</td>
</tr>
<tr>
<td>LINE</td>
<td>8.0 sec</td>
<td>4.0 sec</td>
<td>3.0 sec</td>
</tr>
</tbody>
</table>
Experimental Results

Documents from Hamlet play, one document ~ 300 KB

<table>
<thead>
<tr>
<th>Type</th>
<th>Elements</th>
<th>Time (s)</th>
<th>Time (s)</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>totally flat (1 x 300 KB)</td>
<td>2.0</td>
<td>16.7</td>
<td>52.0</td>
<td>4609.0</td>
<td></td>
</tr>
<tr>
<td>rough fragm. (5 x 60 KB)</td>
<td>1.0</td>
<td>16.3</td>
<td>53.3</td>
<td>4546.0</td>
<td></td>
</tr>
<tr>
<td>medium fragm. (22 x 15 KB)</td>
<td>1.7</td>
<td>17.0</td>
<td>53.3</td>
<td>4556.0</td>
<td></td>
</tr>
<tr>
<td>fine fragm. (5000 x 0.22 KB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- retrieve all elements of a type either above the fragmentation level at nearly constant cost, or below proportional to number of DB operations (as in 1)

<table>
<thead>
<tr>
<th>Type</th>
<th>Elements</th>
<th>Time (s)</th>
<th>Time (s)</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>getObj (9 documents)</td>
<td>totally flat (1 x 300 KB)</td>
<td>21.0 sec</td>
<td>3.5 sec</td>
<td>3.5 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rough fragm. (5 x 60 KB)</td>
<td>21.0 sec</td>
<td>71.0 sec</td>
<td>4.0 sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td>medium fragm. (22 x 15 KB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACT (60 KB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCENE (15 KB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- navigation at element level

<table>
<thead>
<tr>
<th>Type</th>
<th>Elements</th>
<th>Time (s)</th>
<th>Time (s)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 x succ(succ(succ(succ())))</td>
<td>totally flat (1 x 300 KB)</td>
<td>12.0 sec</td>
<td>5.0 sec</td>
<td>15.0 sec</td>
</tr>
<tr>
<td></td>
<td>rough fragm. (5 x 60 KB)</td>
<td>12.0 sec</td>
<td>4.0 sec</td>
<td>3.0 sec</td>
</tr>
<tr>
<td></td>
<td>medium fragm. (22 x 15 KB)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCENE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results

Documents from Hamlet play, one document ~ 300 KB

- retrieve all elements of a single document

<table>
<thead>
<tr>
<th>getAll (1 document)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
<th>fine fragm. (5000 x 0,22 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE (20-elements)</td>
<td>2,0 s</td>
<td>16,7 s</td>
<td>52,0 s</td>
<td>4609,0 s</td>
</tr>
<tr>
<td>STAGEDIR (134 elements)</td>
<td>1,0 s</td>
<td>16,3 s</td>
<td>53,3 s</td>
<td>4546,0 s</td>
</tr>
<tr>
<td>LINE (4014 elements)</td>
<td>1,7 s</td>
<td>17,0 s</td>
<td>53,3 s</td>
<td>4556,0 s</td>
</tr>
</tbody>
</table>

- retrieve all elements from all documents

<table>
<thead>
<tr>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21,0 sec</td>
<td>3,5 sec</td>
<td>3,5 sec</td>
</tr>
<tr>
<td>21,0 sec</td>
<td>71,0 sec</td>
<td>4,0 sec</td>
</tr>
</tbody>
</table>

- navigation at element level

<table>
<thead>
<tr>
<th>10 x succ(succ(succ(succ(succ()))))</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE</td>
<td>12,0 sec</td>
<td>5,0 sec</td>
<td>15,0 sec</td>
</tr>
<tr>
<td>LINE</td>
<td>8,0 sec</td>
<td>4,0 sec</td>
<td>3,0 sec</td>
</tr>
</tbody>
</table>

trade-off between number of DB operations and parsing cost within the fragment: optimum found for the shortest fragment size that allows to perform the operation with one DB access.
Experimental Results

Documents from Hamlet play, one document ~ 300 KB
• retrieve all elements of a single document

<table>
<thead>
<tr>
<th>getObj (9 documents)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE (15 KB)</td>
<td>21,0 sec</td>
<td>3,5 sec</td>
<td>3,5 sec</td>
</tr>
<tr>
<td>ACT (60 KB)</td>
<td>21,0 sec</td>
<td>71,0 sec</td>
<td>4,0 sec</td>
</tr>
</tbody>
</table>

Each experiment has a different winner!

• navigation at element level

<table>
<thead>
<tr>
<th>10 x succ(succ(succ(succ())))</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE</td>
<td>12,0 sec</td>
<td>5,0 sec</td>
<td>15,0 sec</td>
</tr>
<tr>
<td>LINE</td>
<td>8,0 sec</td>
<td>4,0 sec</td>
<td>3,0 sec</td>
</tr>
</tbody>
</table>
3.1 Document insertion

Super-DTD
- representation of DTD's as SGML documents
- used to configure the application

Fragment of Super DTD instance

```xml
<DOCTYPE docName=shakespeare ...>
  <ELEM elemName = play...
    contentModel = ' (title, fm, personae, scndescr, playsubt,
      induct?, prologue?, act+, epilogue?)' ... ><?ELEM>
  <ELEM elemName = title ... contentModel = ' (#PCDATA?)'...><?ELEM>
  <ELEM elemName = fm ... contentModel = ' (p+)' ...><?ELEM>
  ....
```
Configuring the database application

Step 1: generate from the application DTD the super-DTD instance

Step 2: insert configuration information into super-DTD instance

Step 3: insert super-DTD instance into database

Step 4: generate internal database structures (bootstrap)

Step 5: insert application documents into database
3.2 Document modification

Ensure conformance to DTD
- modified document fragment consistent with DTD
- document fragment may be inserted at specified position

Approach
- modified DTD (example): generates database structures
- parser checks consistency of content model at insertion point -> inclusions/exclusions!

Consistency
- index structures
- ID/IDREF values
3.3 Querying

Three levels of querying

- configurable query templates
- PAT query language
- SQL/OQL-type query language
## Query Templates

<table>
<thead>
<tr>
<th>SURNAME</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rothli</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LANGUAGE/LANGQUAL</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION/SECQUAL</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARTICLE</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SURNAME</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LANGUAGE/LANGQUAL</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>EsEnglish</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SECTION/SECQUAL</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>News</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ARTICLE</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PAT query language

**structural queries (A, B element types)**

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A INCLUDES B = { a ∈ A</td>
<td>∃ b ∈ B such that b contained in a}</td>
</tr>
<tr>
<td>B INCL-IN A = { b ∈ B</td>
<td>∃ a ∈ A such that b contained in a}</td>
</tr>
</tbody>
</table>

**simple predicates**
- ATTR_SELECT, CONTENT_SELECT

**set operations**
- INTERSECT, UNION, DIFF, EMPTY

**extensions for**
- SGML references, hyperlinks, text retrieval

querying independent of particular DTD!
"Select all plays where Bernardo appears as a speaker in a scene" as PAT query:

```
play INCLUDES (scene INCLUDES CONTENT_SELECT("Bernardo", speaker))
```
3.4 Optimization of document queries

Optimization issues
• optimizing complex queries
• exploit physical characteristics of DB
• exploit document type definition

Approach
• use of PAT query algebra
• rule-based optimization
Optimization goals

Query simplification
- generated by graphical user interfaces
- federated document databases and document transformations

Documents with DTD 1

Documents with DTD 2

Documents with DTD 3

Global query

local queries

local results

global result
Optimization goals

Physical optimization
- index structures
  - document paths
  - element properties
- path traversal
- physical document representation

```
<fm> ... </fm>
<personae>
<personae title>
Dramatis personae
</personae title>
<playsubt>
....
</playsubt>

<acttitle>
Act I
</acttitle>
<scene>
...
</scene>
```

The Tragedy of Hamlet, Prince of Denmark
Exploiting DTD for query optimization - example 1

query: Select all plays where Bernardo appears as a speaker in a scene.

result: same as: Select all plays where Bernardo appears as a speaker.
Exploiting DTD for query optimization - example 2

query: Select all acts within a prologue where Bernardo appears as a speaker.

result: empty
Query transformations

set theoretic equivalences
- A INCLUDES EMPTY ⇔ EMPTY
- (A UNION B) UNION C ⇔ A UNION (B UNION C)

consistency with DTD
- A INCLUDES B ⇔ EMPTY
  if A or B do not occur in the DTD, or
  B is not contained within A in the DTD

transformations exploiting the DTD structure
- exclusivity
- obligation
Optimization based on exclusivity

Element type graph
Edge \( (A,B) \) in element type graph iff \( B \) occurs in the content model of \( A \)

Exclusivity
Firstname INCL-IN Name ⇔ Firstname

Name entrance location, for (Author, Surname)
Surname INCL-IN Author ⇔
Surname INCL-IN (Name INCL-IN Author) ⇔
(Surname INCL-IN Name) INCL-IN Author

B is *exclusively contained* in \( A \) iff every document element \( b \in B \) within a document is contained in a document element \( a \in A \). Then

\[ B \text{ INCL-IN } A \leftrightarrow B \]
Optimization based on obligation

A *obligatorily contains* B iff every document element \( a \in A \) within a document contains an element \( b \in B \). Then:

\[
A \text{ INCLUDES } B \iff A
\]

Problem: determine the obligatory containment

- optional occurrence indicators (?, *)
- OR - connectors (|)

Normalization of content models

- remove all optional occurrence indicators (1)
- SEQ/OR normal form (2)
SEQ/OR normalization

\[
\begin{align*}
\text{SEQ} & \quad \rightarrow \\
T_1 \ldots T_k & \quad \rightarrow \\
T_{k+1} \ldots T_l & \\
\text{SEQ} & \quad \rightarrow \\
T_{l+1} \ldots T_m & \\
\text{OR} & \quad \rightarrow \\
T_1 \ldots T_k & \quad \rightarrow \\
T_{k+1} \ldots T_l & \\
\text{OR} & \quad \rightarrow \\
T_{l+1} \ldots T_m & \\
\text{SEQ} & \quad \rightarrow \\
T_1 \ldots T_k & \quad \rightarrow \\
T_{k+1} \ldots T_l & \\
\text{SEQ} & \quad \rightarrow \\
T_{l+1} \ldots T_m & \\
\text{OR} & \quad \rightarrow \\
T_1 \ldots T_k & \quad \rightarrow \\
T_{k+1} \ldots T_l & \\
\text{OR} & \quad \rightarrow \\
T_{l+1} \ldots T_m & \\
\end{align*}
\]
Index structures

Primary structure index
- supports A INCLUDES B, A INCL-IN B

Secondary structure index
- ID/IDREF tables

Content indices
- attribute values
- textual content of elements

Information retrieval index
- coupling with Inquery
- configuration of document granularity
Combined rule application

Example: exploiting a structure index for (C,D)

\[ \text{SI}_{(C,D)} = \{ [d, C_d] \mid c \in C_d \text{ contained in } d \in D \} \]

\[ \text{A INCL-IN B } \iff \text{ A INCL-IN (C INCL-IN D)} \]

if \ A \text{ exclusively contained in C}
\ B \text{ exclusively contained in D}
\ B \text{ entrance location for C and D}
Implementation of an optimizer

Extend an (object-) relational rule-based optimizer
  • mapping of PAT algebra to SQL-like query language
  • complexity difficult to control

Rule-based optimizer for the PAT algebra

Heuristic simplification of PAT expressions
  • no exploration of search space
  • simplify using the standard rules
  • identify beneficial entrance locations
3.5 Document rendition

Style-sheets for conversion to HTML

Hyperlink rendition
  • preview to referenced documents
  • specified by means of a query
  • content, physical information

Materialized HTML views of rendered documents
4 Hytime support

HyTime

- predefines patterns for element type definitions (so-called architectural forms)
- modelling of composite media documents
- semi-formal specification

The DBMS application needs to be extended to

- recognize HyTime constructs in SGML documents
- generate HyTime-related database objects
- ensure conformance of these structures to the HyTime standard
- model HyTime semantics in application independent manner
Reference Mechanisms in HyTime

Comparison to SGML

• granularity of objects (elements vs. arbitrary granules)
• scope (same document vs. arbitrary documents)
• object addressing (explicit document components vs. arbitrary document fragments)

HyTime Reference Mechanisms

• name space addressing (same as with SGML)
• coordinate addressing (arbitrary coordinate systems as a basis for locating hyperobjects)
• semantic addressing (objects referenced may be described using query language)
Extended database schema
## Consistency of HyTime DTDs and documents

<table>
<thead>
<tr>
<th></th>
<th>DTD</th>
<th>documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements</td>
<td>Attribute HyTime declared</td>
<td>Attribute value of HyTime is name of architectural form</td>
</tr>
<tr>
<td>documents</td>
<td>Root element type has attribute HyTime</td>
<td>Attribute value is HyDoc</td>
</tr>
<tr>
<td>Content model of architectural form</td>
<td>(may be compared)</td>
<td>Must conform</td>
</tr>
<tr>
<td>Comments in architectural forms</td>
<td></td>
<td>Must conform</td>
</tr>
<tr>
<td>Required attributes</td>
<td>Must be declared</td>
<td>Must be instantiated</td>
</tr>
<tr>
<td>Optional attributes</td>
<td>-</td>
<td>(may assign default value)</td>
</tr>
<tr>
<td>Fixed attributes</td>
<td>'#FIXED-inDtd' becomes '#FIXED'</td>
<td>Identical for all instances</td>
</tr>
<tr>
<td>Variable attributes</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Declared values in architectural forms</td>
<td>Declaration in DTD more restrictive</td>
<td>-</td>
</tr>
<tr>
<td>Default value in architectural form</td>
<td>-</td>
<td>Optional attributes not in DTD, declared attributes with keyword #IMPLIED</td>
</tr>
</tbody>
</table>
Current Status of the Informix DataBlade

SGML-parser

<table>
<thead>
<tr>
<th>SP-component to generate the SDTD-instance</th>
<th>SP-component to insert parsed documents into the database</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>ESQL/C -</td>
</tr>
</tbody>
</table>

"execute function" or "SQL-query"

DataBlade API

- indices
- documents
- SuperDTD-instances

interface
Conclusions

Issues not covered

• Integration with information retrieval [5]
• Consistency and performance of HyTime [2, 6]

Current work

• Packaging and certification of the SGML/XML DataBlade
• Indexing & query optimization
• coupling with document structure extraction tools (JEDI)
• Interoperability of SGML databases
• applications: catalogues, on-line information, biology
Selected Publications


Storage of semi-structured/document data
Option 1: Bulk Data Type

Flatten object-network with mark-ups

```xml
<Dbgroup id=1><Member id=2><Name id=7>“Peter“</Name></Member><Member id=3><Name id=8>“Heidi“</Name><Office id=9>46</Office>....</Member><Member id=4><Name id=12>“Paul“</Name><Age id=13>28</Age><Office id=14><Department id=19>“Other“</Department><Room id=252><Project id=#5>....</Project></Room></Member>...
</Dbgroup>
```

Pros & Cons

+ generic: no need for source specific schema
+ efficient bulk-access (parsing, indexing), structure/content-retrieval
- costly change (index creation/maintenance, locking)
- needs extensions to query processor for structure access
Option 2: Edge-Table(s)

Table with record for each edge (or table for each label)

<table>
<thead>
<tr>
<th>Parent</th>
<th>Label</th>
<th>Type</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Member</td>
<td>Object</td>
<td>#2</td>
</tr>
<tr>
<td>#1</td>
<td>Member</td>
<td>Object</td>
<td>#3</td>
</tr>
<tr>
<td>#3</td>
<td>Name</td>
<td>String</td>
<td>„Mary“</td>
</tr>
</tbody>
</table>

Pros & Cons

+ generic: no need for source specific schema
+ efficient edge-wise access
- costly assembly of „documents“
- needs extensions to query processor

-> OEM model
Option 3: Node-Table(s)

Table(s) with record for each node - possibly partitioned

<table>
<thead>
<tr>
<th>Oid</th>
<th>Member</th>
<th>Project</th>
<th>Name</th>
<th>Age</th>
<th>Office</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>{2,3,4}</td>
<td>{5,6}</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>#2</td>
<td>null</td>
<td>null</td>
<td>„Peter“</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>#3</td>
<td>null</td>
<td>null</td>
<td>„Mary“</td>
<td>null</td>
<td>{D252,11}</td>
<td>null</td>
</tr>
</tbody>
</table>

Pros & Cons

+ little extensions to query-processor (schema reflects structure)
+ efficient node-wise access
- need for schema-evolution
- costly assembly of documents
2.1 Handling of inclusions/exclusions

- inclusions lead to problems during the creation of the database internal representation with flat element types
- dynamic creation of an expanded DTD tree
- a formal verification has been carried out
Example architectural form

<!element dataloc -- Locates string and token data objects in data --
-- Constraint: dimlists are concatenated into one list --
(dimlist*)>
<!attlist dataloc HyTime NAME dataloc
id ID #REQUIRED
quantum -- Data quantum: bit comb. or token --
(str|norm|word|name|sint|date|time|utc) str
catsrc -- Concatenate multiple source objects into a
single object before applying dimlist to it --
(catsrc|nocatsrc) nocatsrc
catres -- Concatenate results of applying dimlist,
in the order of the dimlist --
(catres|nocatres) nocatres
locsrc IDREFS #IMPLIED

...>
HyTime Architectural Form clink

Element type clink

<!element clink (%HyBrid;)*>
<!attlist clink HyTime NAME clink
id ID #IMPLIED
linkend IDREF #REQUIRED ... >

Element type footnote, derived from clink

<!ELEMENT footnote (#PCDATA)>
<!ATTLIST footnote HyTime NAME #FIXED "clink"
id ID #IMPLIED
linkend IDREF #REQUIRED
author NAME #IMPLIED ... >
Example (new methods reflecting semantics of HyTime)

- `resolveDatalocContent ()`: LIST OF INT
- `determineLocsrcs ()`: LIST OF STRING
- `determineLocation (dimlistMarker: LIST OF INT, locations: LIST OF STRING)`: LIST OF STRING
HyTime Architectural Forms vs. Concrete Element_Type Definitions

different element_type names
  • relationship is established by means of attribute `HyTime'
different content model
additional attributes
optional attributes may be omitted
different attribute types
  • e.g., `IDREFS' may become `IDREF', but not vice versa
different keywords for attributes
  • e.g., `#IMPLIED' may become `#REQUIRED', but not vice versa
Conformance Checks

before generation of HyTime element type classes
  • attributes required by architectural form exist
  • attributes have proper categories
  • attribute range subset of the one from the architectural form
  • Comparison of content models not mandatory

before generation of database objects
  • content conforms to abstract and concrete content model
  • required attributes are instantiated, fixed attributes have proper value

after generation of database objects
  • assigning default values to certain attributes
  • other requirements on document structure resulting from the standard, e.g., dimspec resolves to a list of numbers.
Measures to improve performance

Materialization of HyTime_specific views

• configuration is by means of super_DTD instance
  <!ELEMENT ELEM ... >
  <!ATTLIST ELEM ...
  HYTIMEINDEX (ON|OFF) OFF ... 

• requires non-flat representation

Conformance checks

• dynamic generation of data structures representing information on the architectural forms
• shifting conformance checks from instance to type level
• using external HyTime checkers
Architecture

WWW Clients → WWW Browser

WWW Server

Database Clients → VODAK Client

Database Server

SGML/HyTime Schema

VODAK DBMS

QUERY

ObjectStore
Extended database schema

Metaclasses

Classes

Objects

Application-independent part of modeling

HyTime-Modeling

Document hierarchy

_Instance

dataloc

dimspec

marklist

INSTANCE

5 3

dataloc

Instance with attribute

markerlist = "5 3"

Semantic relationship

‘role specialization’

InstanceOf

Marker

Dataref

DatalocIndex

Marklist

Karl Aberer, 1999
Modified DTD

<!-- Modified DTD for Shakespeare  -->

<!ELEMENT play_ext     - - (DUMMY|(title, fm, personae, scndescr, playsubt, induct?, prologue?, act+, epilogue?))>

<!ELEMENT DUMMY - - (title|fm|personae|pgroup|induct| .... |line)>

<!ELEMENT title    - - (#PCDATA)>  
<!ELEMENT fm       - - (p+)>

<!ELEMENT personae - - (title, (persona | pgroup)+)>  
<!ELEMENT pgroup   - - (persona+, grpdescr)>

<!ELEMENT induct   - - (title, subtitle*, (speech|stagedir|subhead)+)>  
<!ELEMENT act      - - (title, subtitle*, prologue?, scene+, epilogue?)>

<!ELEMENT scene   - - (title, subtitle*, (speech | stagedir | subhead)+)>  
<!ELEMENT prologue - - (title, subtitle*, (stagedir | speech)+)>  
<!ELEMENT epilogue - - (title, subtitle*, (stagedir | speech)+)>

<!ELEMENT speech   - - (speaker+, (line | stagedir | subhead)+)>  
<!ELEMENT line     - - (stagedir | #PCDATA)+>

back
Example Document Base

Dok1

0

1

Text of Elem1

Dok2

2

3

5

7

4

6

8

Text of Elem4

Text of Elem6

Text of Elem8

Dok3

9

10

Text of Elem10
Hytime - more detail
## Example architectural form

```xml
<!element dataloc -- Locates string and token data objects in data --
-- Constraint: dimlists are concatenated into one list --
(dimlist*)>
<!attlist dataloc
  HyTime
  id
  NAME
  dataloc
  ID
  #REQUIRED
  #IMPLIED
quantum -- Data quantum: bit comb. or token --
(str|norm|word|name|sint|date|time|utc) str
  (catsrc|nocatsrc) nocatsrc
  catsrc -- Concatenate multiple source objects into a
  single object before applying dimlist to it --
  (catsrc|nocatsrc) nocatsrc
  catres -- Concatenate results of applying dimlist,
in the order of the dimlist --
  (catres|nocatres) nocatres
locsrc IDREFS #IMPLIED
...>
```
HyTime Architectural Form clink

Element type clink

```xml
<!element clink (%HyBrid;)*>
<!attlist clink HyTime NAME clink
id ID #IMPLIED
linkend IDREF #REQUIRED ... >
```

Element type footnote, derived from clink

```xml
<!ELEMENT footnote (#PCDATA)>  
<!ATTLIST footnote HyTime NAME #FIXED "clink"
id ID #IMPLIED
linkend IDREF #REQUIRED
author NAME #IMPLIED ... >
```
HyTime - methods

Example (new methods reflecting semantics of HyTime)

• resolveDatalocContent (): LIST OF INT
• determineLocsrcs (): LIST OF STRING
• determineLocation (dimlistMarker: LIST OF INT,
  locations: LIST OF STRING): LIST OF STRING
HyTime Architectural Forms vs. Concrete Element_Type Definitions

different element_type names
  • relationship is established by means of attribute `HyTime'
different content model
additional attributes
optional attributes may be omitted
different attribute types
  • e.g., `IDREFS' may become `IDREF', but not vice versa
different keywords for attributes
  • e.g., `#IMPLIED' may become `#REQUIRED', but not vice versa
Conformance Checks

before generation of HyTime element type classes
  • attributes required by architectural form exist
  • attributes have proper categories
  • attribute range subset of the one from the architectural form
  • Comparison of content models not mandatory

before generation of database objects
  • content conforms to abstract and concrete content model
  • required attributes are instantiated, fixed attributes have proper value

after generation of database objects
  • assigning default values to certain attributes
  • other requirements on document structure resulting from the standard, e.g., dimspec resolves to a list of numbers.
Measures to improve performance

Materialization of HyTime_specific views

- configuration is by means of super_DTD instance

```
<!ELEMENT ELEM ... >
<!ATTLIST ELEM ...
       HYTIMEINDEX (ON|OFF) OFF ... 
```

- requires non-flat representation

Conformance checks

- dynamic generation of data structures representing information on the architectural forms
- shifting conformance checks from instance to type level
- using external HyTime checkers