Leveraging Existing DBMS Storage for XML DBMS

Mark Graves

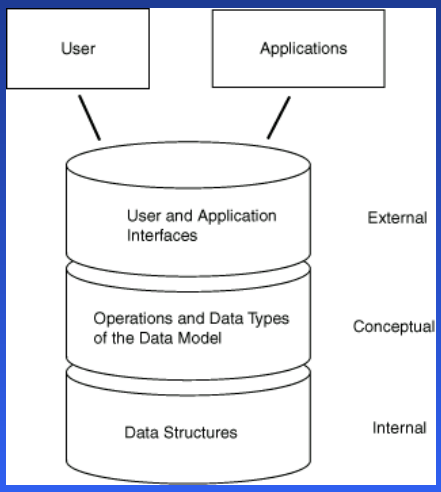
Agenda

- DBMS Architecture
- External Interface
- Data Model
- Storage Systems
  - Overview
  - Fine-grained RDBMS storage
  - Coarse-grained RDBMS storage
  - Medium-grained RDBMS storage

XML DBMS

- Create a DBMS to capture XML
- Access of document & elements
- Should support:
  - Storage
  - Querying
  - Editing

DBMS Architecture



External Interface

- User Interface -- HTML or Java
- URLs -- command access
- Java API -- used by servlet
- Command-line interface
- XML -- taglib, SOAP

Data Models

- **Type Constructors**
- **Operations**
- **Constraints**
- **Examples: relational, Entity-relational, object**

XML Data Model -- Types

- *Document* has one name and one (root) element.
- *Element* has
  - type name (which is a string),
  - collection of attributes, and
  - ordered collection of (interspersed) character data and elements.
- *Attribute* has a name and a value (both strings).
- *Character data* has a value (a string).

XML Data Model -- Constraints

- Each document name may occur only once. (*Thus, the document names are unique and may be queried.)*
- All elements other than the document element have an element node as a parent. The document element has no parent. (*Thus, the elements form a tree.)*
- No attribute name may appear more than once in an element.

XML Data Model -- Operations

- **Add and Delete**
- **Retrieve**
- **Replace**
- **Search**

Operations -- Add and Delete

- Add a document to the database.
- Delete a document from the database.
- Add an element to a specific location in the document.
- Delete an element from a specific location in the document.
- Add an attribute to an element.
- Delete an attribute from an element.

Operations -- Retrieve

- Retrieve a document from the database given its name.
- Retrieve an element from a specific location in the document.
- Retrieve all the elements and character data from a document in document order (in effect, regenerate the document).
- Retrieve an attribute from an element given its name.
- Retrieve the *nth* child of an element.
- Retrieve all children of an element.
- Retrieve the text of the character data.
- Retrieve the parent element of the character data.

Operations -- Replace

- Replace an element at a specific location with another element or character data.
- Replace character data at a specific location with other character data or elements.
- Replace the value of an attribute in an element given its name.
- Set the text of the character data.

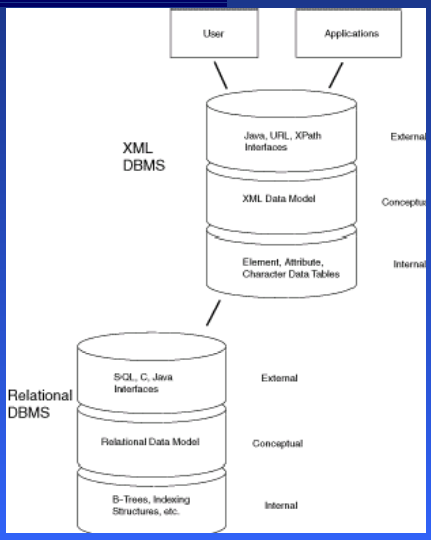
Operations -- Search

- Search for all documents in the database given a particular set of constraints.
- Search for all elements in a document that satisfy a particular set of constraints.
- Search the document for character data that matches a particular set of constraints (such as matching a string).
- Element type name equals (or does not equal) some value.
- Attribute name equals (or does not equal) some value.
- Character data equals (or does not equal) some value.
- Element has a specified number of children (or less than, or greater than, or not equal to).
- Character data contains a specified string as a substring.
- Query constraint consists of two query constraints that must both be true (or either be true).
- Query constraint consists of one query constraint that must not be true.

Storage System (Internal Interface)

- Native store
- Object-oriented
- Complex flat-file
- Relational DBMS

Leveraged Storage Systems



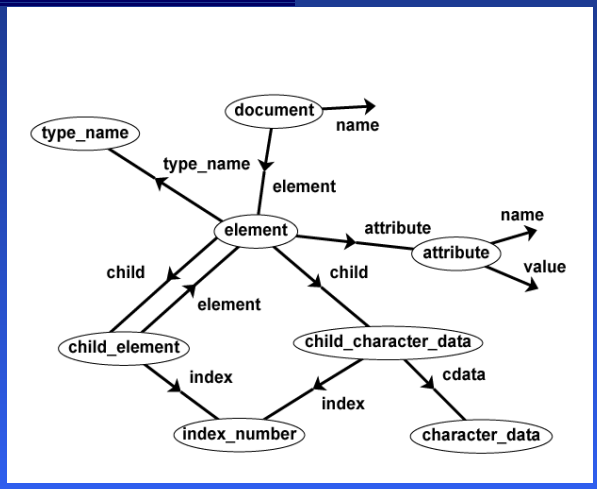
RDBMS Implementation

- Use a Relational DBMS to store XML documents
- Strategies
  - fine-grained -- store every piece of data separately (completely parsed)
  - coarse-grained -- store entire document together (no parsing)
  - medium-grained -- store some elements in coarse-grained storage, other in fine-grained storage (partial parsing)

Fine-grained Storage

- Approach: Completely parse data and store each element, attribute, and character data value in a relational table.
- Design
  - Conceptual Schema
  - Logical Schema (unnormalized & normalized)
  - Physical Schema
- Implementation (Java)

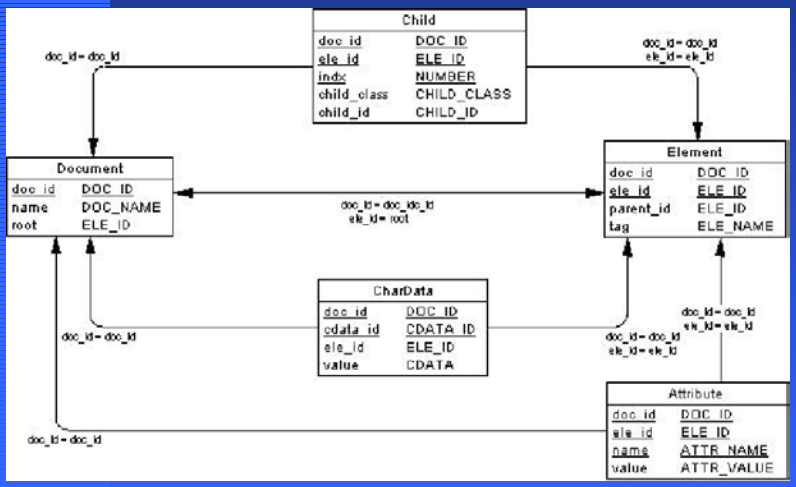
Conceptual Schema



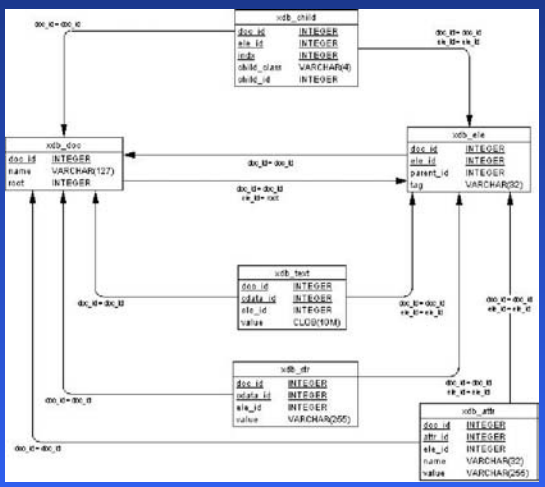
Fine-grained Logical Schema

- Document(name DOC_NAME, root ELEMENT)
- Element(doc DOCUMENT, parent ELEMENT, tag ELE_NAME)
- Attribute(doc DOCUMENT, element ELEMENT, name ATTR_NAME, value ATTR_VALUE)
- CharData(doc DOCUMENT, element ELEMENT, value CDATA)
- Child(doc DOCUMENT, element ELEMENT, index NUMBER, child_class CHILD_CLASS, child CHILD_NODE)

Fine-grained Logical Schema



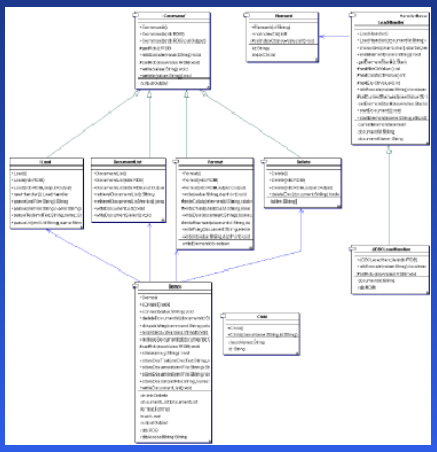
Fine-grained Physical Schema



Fine-grained Commands

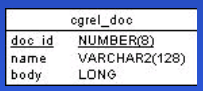
- Retrieve a document (with or without XML header)
- Store a document
- Delete a document
- List documents in database

Fine-grained Implementation



Coarse-grained Storage

- **Approach: Store each document in its entirety**
- **Logical Schema:**
  - Document (name STRING, body TEXT)
- **Physical Schema:**



Medium-grained Storage

- **Use both fine-grained (parsed) and coarse-grained (unparsed) storage as appropriate within a document**
- **Slice points**
- **Multiple slice points**
- **Specifying slice points**
  - **element type name**
  - **element type name & attributes**

Dictionary Example

```
<dictionary>
  <entry number="1" name="aardvark">
  ...
  </entry>
  <entry number="2" name="aadax">
  ...
  </entry>
  .
  .
  .
  <entry number="1200" name="zebra">
  ...
 </entry>
<dictionary>
```

Dictionary Example

DOCUMENT ID	ELEMENT ID	VALUE
1	1	<dictionary> <proxy document=”1” element=”2”/> <proxy document=”1” element=”3”/> … <proxy document=”1” element=”1201”/> </dictionary>
1	2	<entry number=”1” name=”aardvark”> … </entry>
…	…	…
1	1201	<entry number=”1200” name=”zebra”> … </entry>

Medium-grained Physical Schema