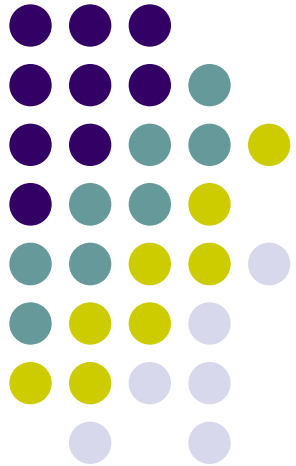
NPRG039

Advanced Aspects and New Trends in XML (and Related) Technologies

RNDr. Irena Holubová, Ph.D.

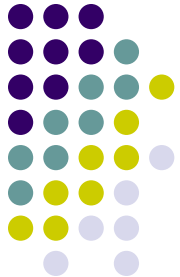
holubova@ksi.mff.cuni.cz



Lecture 6. Native XML databases

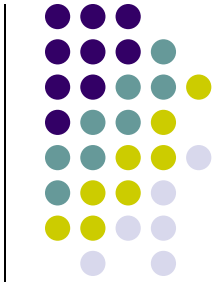
http://www.ksi.mff.cuni.cz/~holubova/NPRG039/

XML Databases



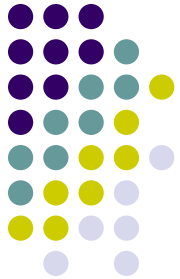
- A software system that enables data to be stored in XML format
- Types:
  - XML-enabled – a traditional database extended with XML support
    - Typically relational databases
      - Oracle DB, MS SQL Server, IBM DB2
  - Native XML database (NXD) – internal model depends on XML and it uses XML documents as the fundamental unit of storage
    - Native does not have to mean text format

Native XML Database

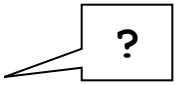


- Defines a (logical) model for an XML document and stores and retrieves documents according to that model
  - Minimum model = elements, attributes, PCDATA, and document order
  - XPath data model, XML Infoset, DOM, SAX, …
- Has an XML document as its fundamental unit of (logical) storage
  - Relational database has a row in a table
- Is not required to have any particular underlying physical storage model
  - Can be built on a relational/hierarchical/object-oriented database, indexed compressed files, …

Native XML Databases – Categories

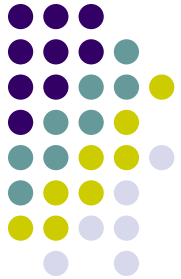


- Document-based storage
  - Stores the entire document in a text or binary form
  - Provides database functionality in accessing the document
  - Examples: to store XML document as a BLOB in a relational database, file in a file system + XML-aware indexes
  - Advantages:
    - Better **round tripping level**
    - Fast reconstruction of the whole document



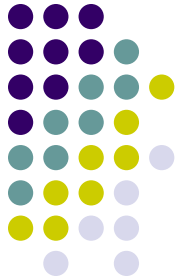
- Node-based storage
  - Stores individual nodes of the document in an existing/custom data store
  - Example: to map the DOM to relational tables such as Elements, Attributes, Entities

Main Differences From XML-Enabled Databases

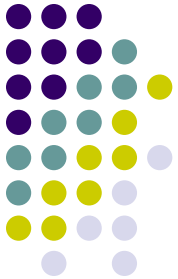


- Native XML databases
  - Can (usually) preserve physical structure (entity usage, CDATA sections, etc.), comments, PIs, DTDs, …
    - XML-enabled databases can do this in theory, not in practice
      - Or it is an expensive feature
  - Can store XML documents without knowing their schema (DTD), assuming one even exists
  - Provide only XML APIs to the data, such as XPath, DOM, …
    - XML-enabled databases offer direct access to the data
      - i.e., tables, objects, indices, …

Current Native XML Databases



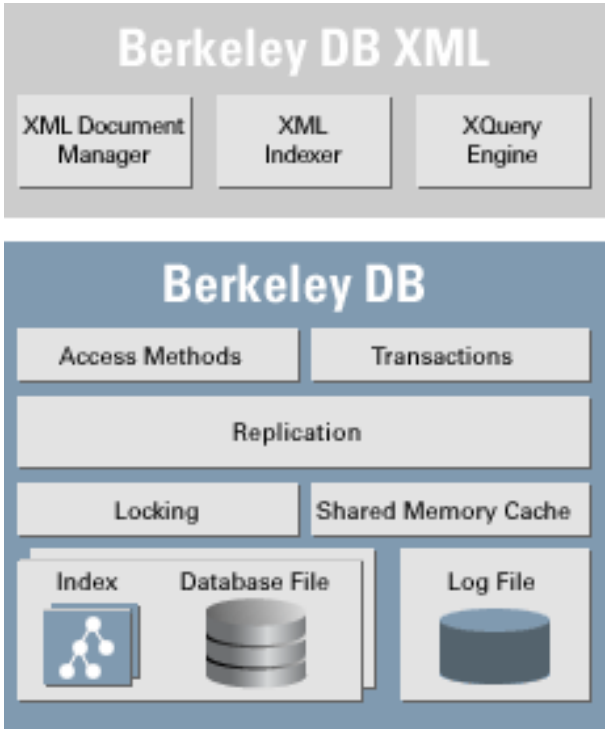
- Product list:
  - http://www.rpbourret.com/xml/ProdsNative.htm
    - Comprehensive list
      - 40 native XML databases
    - Various types of XML products
- Examples:
  - Oracle Berkeley DB XML
  - TIMBER
  - Sedna, BaseX, Tamino, eXist-db, …



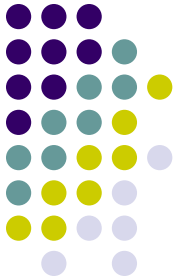
Oracle Berkeley DB XML

Oracle Berkeley DB XML

- Developer: Oracle
  - Formerly owned by Sleepycat Software
  - Originally created at University of California, Berkeley
- http://www.oracle.com/us/products/database/berkeley-db/xml/overview/index.html
- Open source
- Built on top of Berkeley DB
  - Adds an XML document parser, XML indexes, XQuery engine, …
  - Inherits storage engine, transaction support, automatic recovery, …



Berkeley DB (BDB)



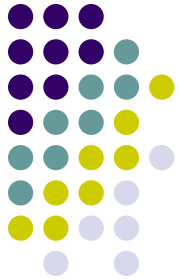
- High-performance embedded database for key/value data
  - Multiple data items for a single key
- Supports:
  - Thousands of simultaneous threads
  - Databases up to 256 terabytes
- Written in C
- OS: most Unix-like and Windows systems
- Embedded database
  - A library embedded into client application
  - Bindings in C, C++, Java, Perl, Python, Ruby, Tcl, Smalltalk
- Three parts:
  - Oracle Berkeley DB
  - Berkeley DB Java Edition
  - Berkeley DB XML

http://www.oracle.com/us/products/database/berkeley-db/overview/index.html

Berkeley DB XML Data Storage

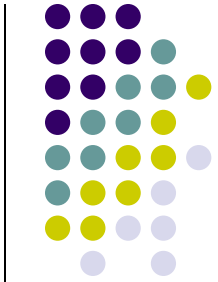
- Stores XML documents in logical groups
  - Containers
- Users can specify a number of properties on a per-container basis
  - e.g., whether to validate documents, store documents as a whole/individual nodes, what indexes to create, …
- Can store also non-XML documents, metadata for XML documents, …
  - Thanks to underlying Berkeley DB data store
- XML document metadata
  - User-specified property-value pairs
  - Can be queried as if they were child elements of the root element

Querying and Updates



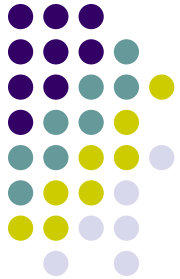
- Querying:
  - XQuery 1.0 and XPath 2.0
- Updating:
  - Performs updates at the node level
    - Uses XQuery to identify a set of nodes to update
  - Allows to
    - Append a new child to a target node
    - Insert a new node before or after a target node
    - Remove a target node
    - Rename a target node
    - Change the value of a target node
  - Supports the XQuery Update Facility

Indexing



- Dynamic indexing system
  - Targeted indexes
    - Indexes particular paths
  - Statistical, cost-based query planning
- Indexes for:
  - XML nodes
    - Elements
    - Attributes
    - Meta-data

Deployment

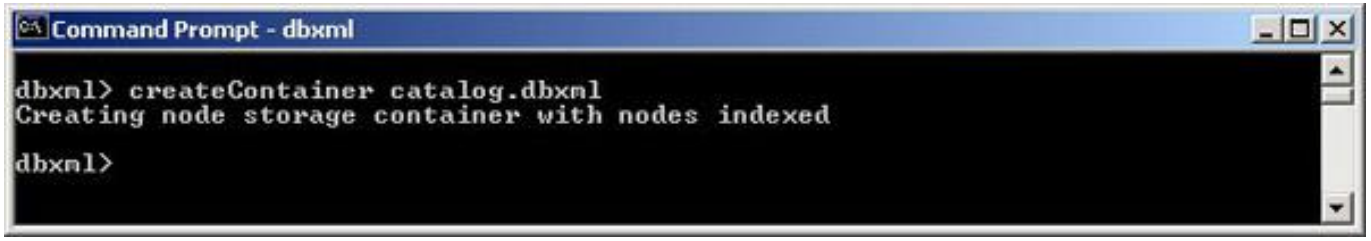


- Embedded database = a library that is linked directly to applications
  - Not a client-server mode
  - Shared memory
  - OS primitives for mutual exclusion
- OS: Windows, Linux, BSD UNIX, Mac OS/X and any POSIX-compliant operating system
  - Installer for Microsoft Windows
- A command-line interface
- Support for C++, Java, Perl, Python, PHP, Tcl, Ruby, …
  - Third-party APIs for other languages

Usage

- Download:
  - http://www.oracle.com/technology/software/products/berkeley-db/xml/index.html
    - Sets path and classpath
- Command shell:

  `dbxml`




```
help
```

```
help createContainer
```

Command	Description
createContainer	Creates a container – name, type, document validation 'in' (default) = node storage container with node indexes 'd' = wholedoc container 'n' = node storage container 'id' = wholedoc container with node indexes
getDocuments	Gets all the documents in the default container, or, if the docName is specified, gets the specified document
openContainer	Opens a container and sets the container as the default container. The validate/novalidate option specifies if documents are to be validated on insertion.
print	Prints the most recent results to stdout. Results may be printed to a file by specifying path to a file. A specified number of results may be output with the 'n' argument.

Command	Description
putDocument	Adds an XML document to the default container. The first argument specifies the document name. The second parameter specifies the XML document string. The third argument specifies the content of the second argument: 'f' = file name 's' (default) = string 'q' = XQuery expression
query	Evaluates an XQuery expression.
removeContainer	Removes a container. The `.dbxml` file corresponding to the container gets deleted.
removeDocument	Removes a document from the default container.
run	Runs the specified file as a script.

Containers and Documents

- Creating a container

```
createContainer catalog.dbxml
```

- Adding a document

```
putDocument catalog1 '<catalog title="Oracle Magazine"
   publisher="Oracle Publishing">
 <journal date="March-April 2006">
  <article>
   <title>Using Bind Variables</title>
   <author> Steve Muench </author>
  </article>
 </journal>
</catalog>' s
```

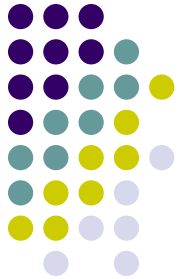
```
putDocument catalog2 C:\catalog2.xml f
```

```
getDocuments
```

Types of Containers

Container Type	Storage Mode	Query Performance	Load Performance	Application
Node Container	XML document stored in nodes	Faster to query	Lower load performance	Use Node container if faster query performance is required. Use Node container if document size is more than 1MB.
Wholedoc Container	Whole XML document stored	Lower query performance, because complete document has to be navigated	Faster document loading, because an XML document does not have to be deconstructed into nodes	Use Wholedoc container if load performance is more important than query performance. Use Wholedoc container if document is relatively small and requires to be frequently retrieved.

Querying



- Querying

```
query 'collection("catalog.dbxml")/
  catalog/journal/article/title/text()'
print
```

```
query 'collection("catalog.dbxml")'
print
```

- We can query:
  - Multiple containers
  - Multiple documents
  - Specific documents

    `doc("catalog.dbxml/catalog1")`

Indexing

- By default BDB XML turns several indexes on

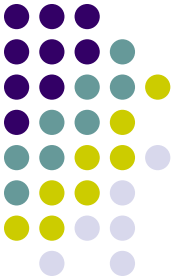
```
setAutoIndexing off
```

```
time query 'collection("catalog.dbxml")/
  catalog[journal]'
print
```

```
Time in seconds for command 'query': 0.437096
```

- Searches for presence of `journal` in all documents
  - We can create an index for such documents

```
addIndex "" journal node-element-presence-none
```

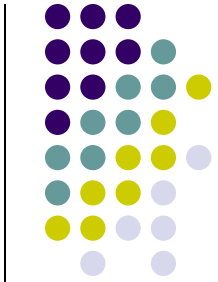


Indexing

- Parts of key index:
  - unique – indicates whether the indexed value must be unique (`unique`) within the container
  - path type – enables to index nodes (`node`) or edges (`edge`)
    - Edge is used when nodes occur in different contexts (e.g., name of a person vs. name of a country)
  - node type – `element` / `attribute` / `metadata`
  - key type – `equality` (searching for value) / `presence` (existence of a node) / `substring` (values containing a string)
  - syntax type – syntax to use for the indexed value (`none`, `string`, `double`, `time`, `date`, ...)
- Example 1: `node-element-presence-none`
  - Index of the node elements
  - To determine if something is present or not, not expected to be unique

http://docs.oracle.com/cd/E17276_01/html/gsg_xml/cxx/indices.html

Indexing



- Example 2: `node-element-equality-double`
  - We index the node as double value, not string

- Default indices:
  - `node-element-equality-string`
  - `node-attribute-equality-string`
  - `node-element-equality-double`
  - `node-attribute-equality-double`

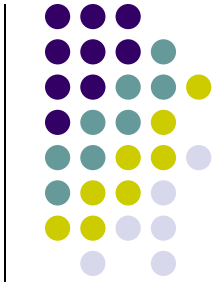
  for every attribute and leaf node

Updating

```
query 'insert nodes
       <title>Commanding ASM</title>
after doc("dbxml:/catalog.dbxml/catalog1")/
  catalog/journal[article/title="Using Bind Variables"]'
```

- Rules:
  - Update queries never return a result
  - The query to select a document for updating must not itself be an update query
  - Update queries can only work on one document at a time
- Other options: `before`, `into`, `as first into` and `as last into`

Updating



- Deleting:

```
query 'delete nodes
doc("dbxml:/catalog.dbxml/catalog1")/catalog/title'
```

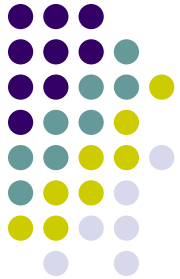
- Renaming:

```
query `rename node
doc("dbxml:/catalog.dbxml/catalog1")
  /catalog/journal as "magazine"'
```

- Replacing:

```
query `replace node
doc("dbxml:/catalog.dbxml/catalog1")
  /catalog/magazine/article/title with
doc("dbxml:/catalog.dbxml/catalog2")
  /catalog/magazine/article/title'
```

Updating



- Replacing:

```
query `replace value of node
doc("dbxml:/catalog.dbxml/catalog1")
  /catalog/magazine/article/title with
"New title"'
```

```
query '
for $i in
doc("dbxml:/catalog.dbxml/catalog1")
  /catalog/magazine/article/title
return
  replace value of node $i with "New title"'
```

XML Schema Validation

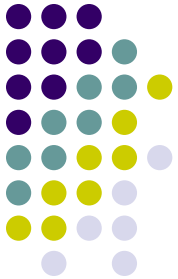
- Suppose we have a schema available at: `http://www.oracle.com/fakeschema.xsd`
- Creating a container with validation:

```
createContainer validate.dbxml d validate
```

- Document inserting:

```
putDocument phone1 '
<phonebook xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://www.oracle.com/fakeschema.xsd">
  ...
</phonebook>' s
```

Metadata

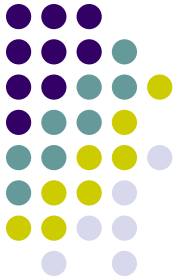


- Additional information about a document
  - Examples: name of a document, time a document was modified, name of the person who modified, …
- Not part of document
  - But can be queried

```
openContainer catalog.dbxml
setMetaData catalog1 '' modifyuser string john
```

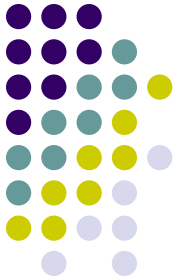
```
query 'collection("catalog.dbxml")/
catalog/dbxml:metadata("modifyuser")'
print
```

C++ API



```
#include <string>
#include <fstream>
#include "dbxml/DbXml.hpp"
using namespace std;
using namespace DbXml;

int main(int argc, char **argv) {
  try {
    XmlManager mgr;
    XmlContainer cont = mgr.createContainer("phone.dbxml");
    XmlUpdateContext uc = mgr.createUpdateContext();
    cont.putDocument("phone1",
                     "<phonebook>...</phonebook>", uc);
    cont.putDocument("phone2",
                     "<phonebook>...</phonebook>", uc);

    ...
```

C++ API

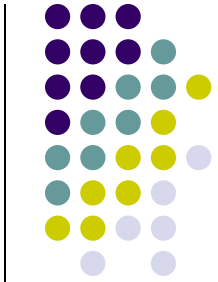


```
    ...

    XmlQueryContext qc = mgr.createQueryContext();
    XmlResults res = mgr.query(
      "collection('phone.dbxml')/phonebook[name/first =
       'Lisa']/phone[@type = 'home']/string()", qc);

    XmlValue value;
    while (res.next(value))
       cout << "Value: " << value.asString() << endl;
  } catch (XmlException &e) {
    std::cout << "Exception: " << e.what() << std::endl;
  }
  return 0;
}
```

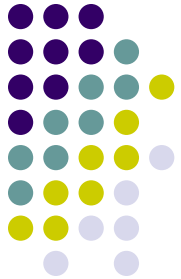
Java API

```
package dbxml.gettingStarted;
import com.sleepycat.dbxml.XmlContainer;
import com.sleepycat.dbxml.XmlException;
import com.sleepycat.dbxml.XmlManager;

class doDbXml {
  public static void main(String args[]) throws Throwable {
    XmlContainer myContainer = null;
    XmlManager myManager = null;
    String theContainer = "container.dbxml";

    try {
      myManager = new XmlManager();
      myContainer = myManager.openContainer(theContainer);
    } catch (XmlException e) {
      // Error handling goes here
    }
  }
}
```

Oracle Berkeley DB XML

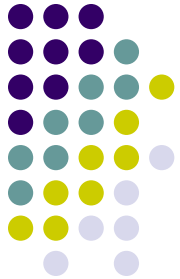


- Resources:
  - Introduction to Berkeley DB XML
    - http://docs.oracle.com/cd/E17276_01/html/intro_xml/index.html
  - Getting Started with Berkeley DB XML for Java
    - http://docs.oracle.com/cd/E17276_01/html/gsg_xml/cxx/index.html
  - Getting Started with Berkeley DB XML for C++
    - http://docs.oracle.com/cd/E17276_01/html/gsg_xml/java/index.html
  - Berkeley DB XML Reference Guide
    - http://docs.oracle.com/cd/E17276_01/html/ref_xml/toc.html



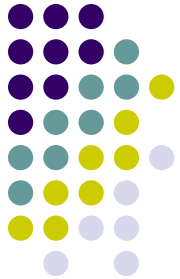
TIMBER

TIMBER



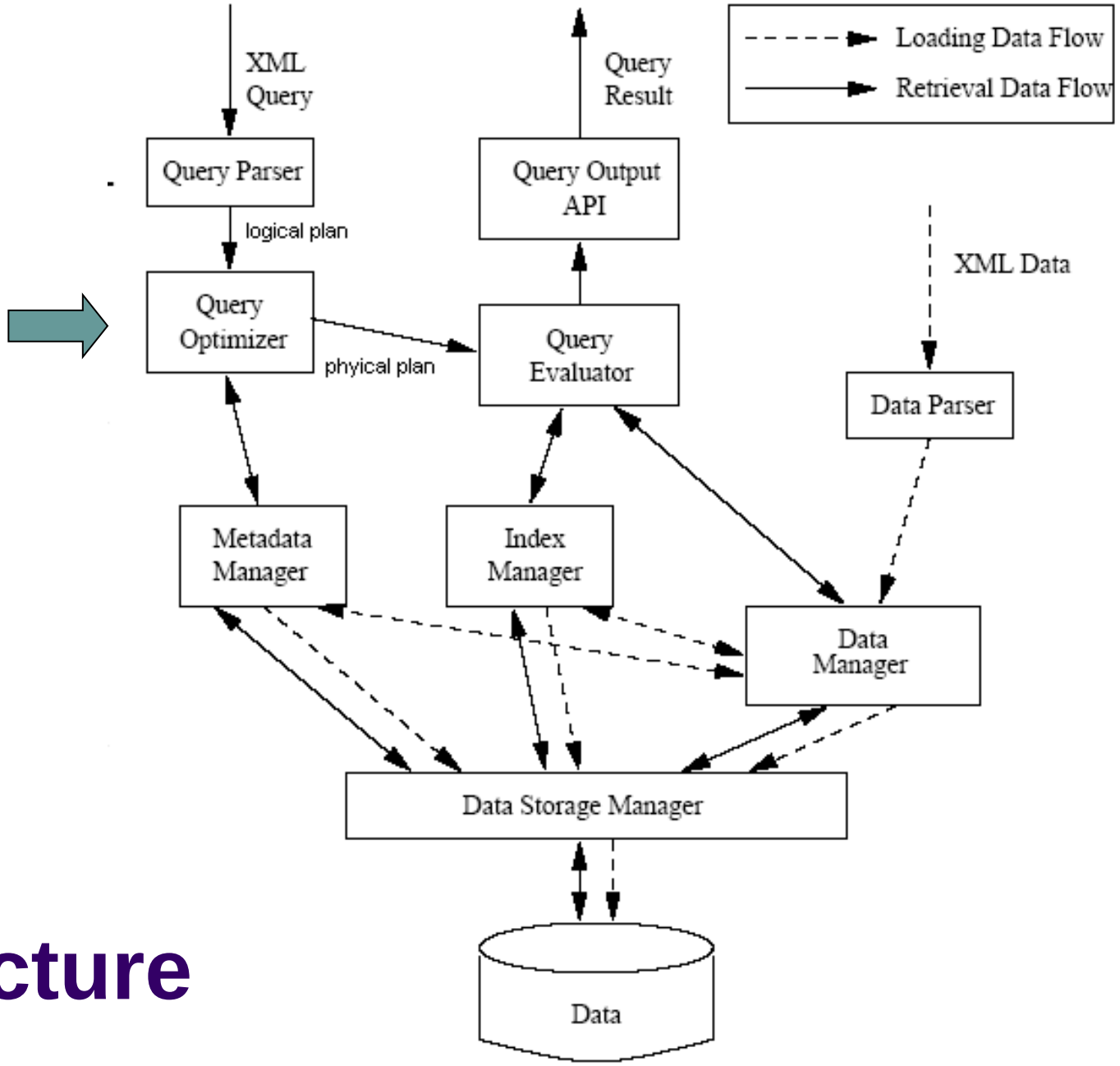
- Developer: University of Michigan
  - **T**ree-structured native XML database **I**mplemented at the University of **M**ichigan by **B**right **E**nergetic **R**esearchers
- http://www.eecs.umich.edu/db/timber
- Open source for non-commercial users
- Architecture as close as possible to a relational database
  - Aim: to reuse the technologies developed for relational databases
- Basis: XML algebra that manipulates sets of ordered, labelled trees
  - Issues: complex and variable structure of trees, ordering, …

Installation

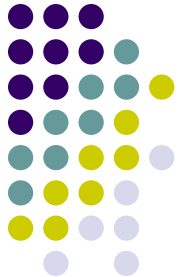


- Access options:
  - Command line interface
    - Download a solution and compile in Visual Studio
  - GUI interface
    - -//-
  - Web service for accessing Timber
    - Apache Tomcat, Apache Ant
- Can use Berkeley DB for physical storage
  - Instead of default system Shore

Architecture



Data Storage

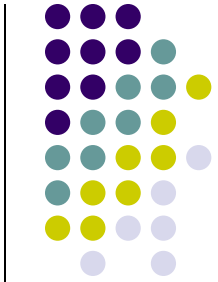


- Data Parser takes an XML document as input, and produces a parsed tree as output
- Data Manager takes each node of the parsed tree as it is produced, transforms it into internal representation and stores as an atomic unit of storage
  - A node corresponding to each element
  - Attributes of an element node are clubbed together into a single child node
  - Content of an element node is a separate child node
    - Mixed content: Each content part is a separate child node
  - Processing instructions, comments, … are ignored

!!!

- Data Storage Manager – a set of navigation interfaces and scan interfaces

Query Evaluation

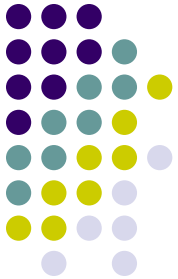


- Emphasis on efficient evaluation
- **Tree algebra for XML trees** (TAX)
  - Like relational algebra for relations
- Parts:
  - Logical
    - e.g., in the relational algebra:
      - Involves Cartesian product
      - Does not permit sorting, deals with unordered sets
  - Physical
    - e.g., in relational algebra:
      - Involves natural join and sorting
      - Manipulates ordered sets (and exploits ordering)

http://www.eecs.umich.edu/db/timber/files/tax_full.pdf

TAX Example

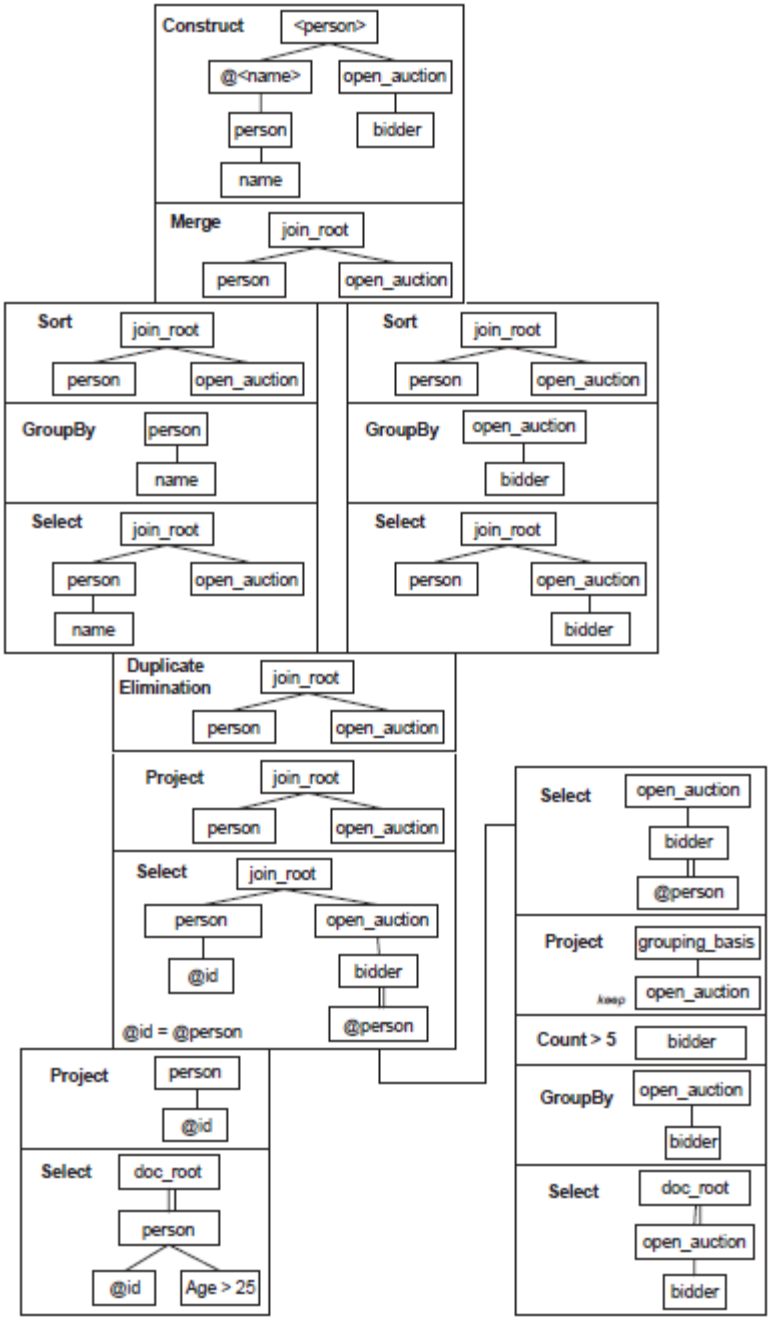
Query in XQuery

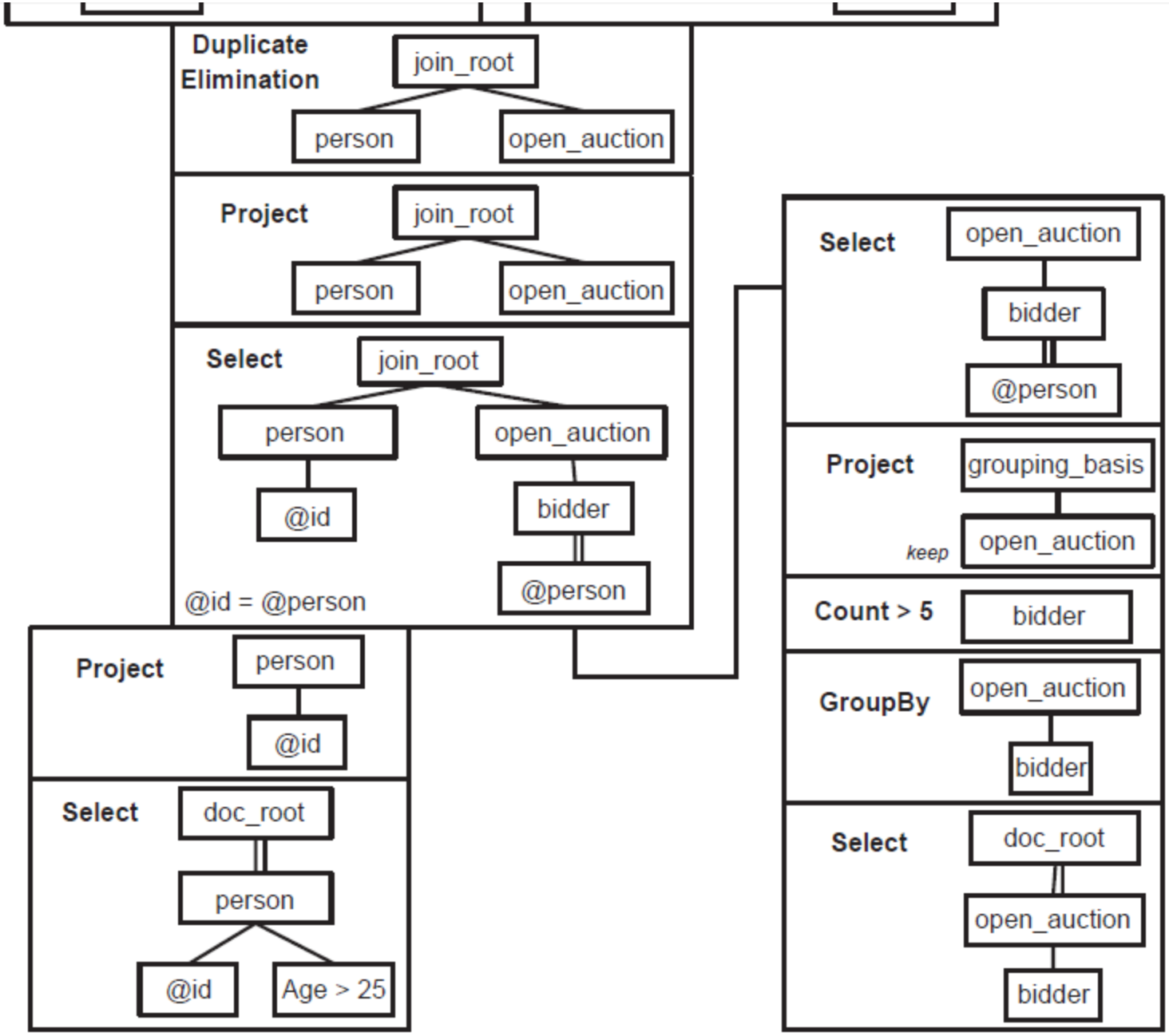


```
FOR $p IN document("auction.xml")//person
FOR $o IN document("auction.xml")//open_auction
WHERE count($o/bidder) > 5 AND $p/age > 25
        AND $p/@id = $o/bidder//@person
RETURN
<person name={$p/name/text()}> $o/bidder </person>
```

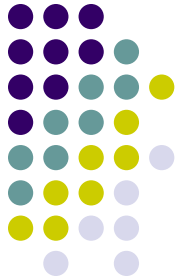
TAX Example

Bottom-up Evaluation




Duplicate Elimination
join_root
person
open_auction
Project
join_root
person
open_auction
Select
join_root
person
@id
open_auction
bidder
@person
@id = @person
Project
person
@id
Select
doc_root
person
@id
Age > 25
Select
open_auction
bidder
@person
Project
grouping_basis
keep
open_auction
Count > 5
bidder
GroupBy
open_auction
bidder
Select
doc_root
open_auction
bidder

Updating



- Deleting:

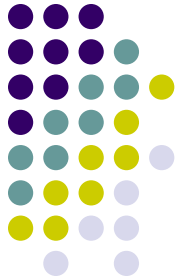
```
for $b in document("sbook.xml")//book
for $a in $b/author
return timber-delete($a)
```

- Updating:

```
for $b in document("sbook.xml")//book
for $a in $b/author
return timber-update($a, "Stelios")
```

```
for $b in document("sbook.xml")//book
for $i in $b/@id
return timber-update($i, $b/isbn)
```

Updating



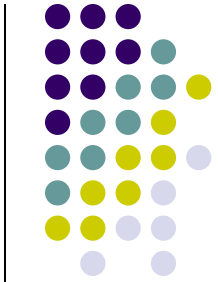
- Insertion of element:

```
for $b in document("sbook.xml")//book
for $a in $b/author
return timber-insertelement($a, "name", "Stelios")
```

- Insertion of attribute:

```
for $b in document("sbook.xml")//book
for $c in $b/chapter
where $b/editor = "Stelios"
return timber-insertattribute($c,"num","")
```

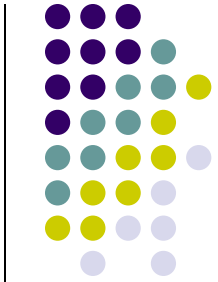
Updating



- Without updates, all nodes of a document are laid out on the disk in the document order
  - Efficient query execution
- Updates may require to put nodes into a separate unordered overflow portion of the file
  - Insert of new nodes, modification of a node, …
  - Query performance will suffer
- Function reorganize

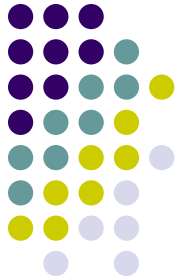
```
timber -m reorganize -d sbook.xml
```

Schema-free XQuery

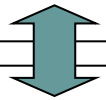


- Standard XQuery requires knowledge of the document structure
  - e.g., the user must know that nodes `price` and `title` are children of nodes `book` in order to correctly write the queries
- Function timber-mlca
  - Users with limited knowledge
  - Posing complex queries and geting meaningful results
  - Returns **M**eaningful **L**owest **C**ommon **A**ncestor (MLCA) of the variables inside the function

Schema-free XQuery

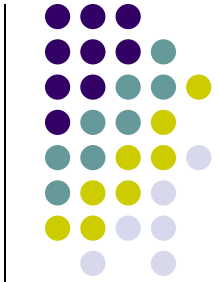


```
for $d in timber-mlca($b, document("sbook.xml")//book,
                      $p, document("sbook.xml")//price,
                      $t, document("sbook.xml")//title)
where $p < 60 and $p > 20
return $t
```



```
for $b in document("sbook.xml")//book
where $b/price < 60 and $b/price > 20
return $b/title
```

References



- Oracle Berkeley DB XML
  - http://www.oracle.com/us/products/database/berkeley-db/xml/overview/index.html
  - http://docs.oracle.com/cd/E17276_01/html/toc.htm
- TIMBER
  - http://www.eecs.umich.edu/db/timber