

PB138 — XML databases, NoSQL databases

(C) 2019 Masaryk University -- Tomáš Pitner, Luděk Bártek, Adam Rambousek

NoSQL databases

- non relational databases, **flexible schema**
- often used for big data applications, clusters
- different storage structure than SQL databases
- **give up constraints**/transactions to improve performance
- low-level interface

NoSQL types

- **key-value**
 - Redis, Memcached, Amazon SimpleDB...
- **document** (JSON, XML...)
 - CouchDB, Elasticsearch, MongoDB...
- **graph / RDF triple**
 - Virtuoso, Neo4j...
- **object**
 - Caché, GemStone...

RDF databases / triple store

- standard data model (RDF)
- standardized interchange format (N-Triples, N-Quads, XML,...)
- query language (SPARQL), Linked Data
- native
 - Apache Jena, Sesame/RDF4J...
- RDF layer to relational database
 - Virtuoso, IBM DB2...

SPARQL

- SPARQL Protocol and RDF Query Language
- W3C Recommendation SPARQL 1.1, March 2013
- *SELECT* - values as table
- *CONSTRUCT* - extract RDF
- *ASK* - true/false
- *DESCRIBE* - extract RDF graph

- inferencing

SPARQL example

Ontology

```
ex1:FullProfessor rdf:subClassOf ex1:Professor.  
ex1:AssistantProfessor rdf:subClassOf ex1:Professor.  
ex1:Professor owl:equivalentClass ex2:Teacher
```

Data

```
ex1:Bob rdf:type ex1:FullProfessor .  
ex1:Alice rdf:type ex1:AssistantProfessor .  
ex2:Mary rdf:type ex2:Teacher
```

SPARQL example

Data

```
ex1:Bob rdf:type ex1:FullProfessor .  
ex1:Alice rdf:type ex1:AssistantProfessor .  
ex2:Mary rdf:type ex2:Teacher
```

SPARQL query

```
SELECT ?x  
WHERE {  
  ?x rdf:type ex1:Professor  
}
```

- noone is *Professor*, but inferencing will find Bob, Alice, Mary

XML databases, when to use

- **working with documents** or metadata in XML format
- data format/**schema changes** over time
- complex and variable schema
- structure queries

XML database concepts

- basic element = *document*

- documents gathered in *collections* ("tables")
- query on document structure
- output is document, document fragment, or constructed XML

XML database types

- **XML-enabled** databases
 - mapping XML data to own data model (relational, object...)
 - character large object
 - fragmented to series of tables/objects
 - stored in XML Type
 - **ISO SQL/XML** - element construction, data mapping, enhanced SQL with XQuery
 - Oracle, IBM DB2, MS SQL, PostgreSQL

XML database types

- **native** XML databases
 - using XML data model directly
- open-source
 - eXist, Sedna, BaseX, MonetDB, Oracle/Berkeley DB XML
- commercial
 - MarkLogic, Virtuoso, Qizx

Interface

XQuery

```
<titles>{
  for $book in collection("books")/book where $book/year="1990"
  return $book/title
}</titles>
```

Interface

XQuery Update

```
update delete collection("books")/book/isbn
```

```
update insert <title>XQuery Guide</title>
into collection("books")/book[isbn="42"]
```

Interface

SQL/XML (result-table)

```
SELECT
    id, vol,
    xmlquery('$j/name', passing journal as "j") as name
FROM
    journals
WHERE
    xmlexists('$j[licence="CreativeCommons"]',
    passing journal as "j")
```

Interface

SQL/XML (result-XML)

```
SELECT XMLELEMENT (NAME "saleProducts",
    XMLNAMESPACES (DEFAULT 'http://posample.org'),
    XMLAGG (XMLELEMENT (NAME "prod",
        XMLATTRIBUTES (p.Pid AS "id"),
        XMLFOREST (p.name AS "name",
            i.quantity AS "numInStock"))))
FROM PRODUCT p, INVENTORY i
WHERE p.Pid = i.Pid
```

- *XSLT* - output transformation
- *XML Schema* - input validation

Interface

- *XQJ* - XQuery API for Java
 - unified query layer between application and XML Datasource
 - prepared statements
 - binding variables

Interface

```

XQDataSource xqs = new ExistXQDataSource();
XQDataSource xqs = new SednaXQDataSource();
xqs.setProperty("serverName", "localhost");
XQConnection conn = xqs.getConnection();
XQExpression xqe = conn.createExpression();
String xqueryString = "for $x in doc('books.xml')//book
    return $x/title/text()";
XQResultSequence rs = xqe.executeQuery(xqueryString);
while(rs.next())
    System.out.println(rs.getItemAsString(null));
conn.close();

```

Interface

- *XML:DB*
- similar concept to JDBC, abstract interface to XML database
 - *Driver* - access to given database
 - *Collection* - document collection in database
 - *Services* - support database features, e.g. XPathQueryService, XUpdateQueryService
 - *Resource* - data stored in database
 - *ResourceSet* - data as result of query

Storage

- **intact document storage**
 - unique identifier for document
 - preferably **parse and index** on storage
 - on query: fast, if application need **access to whole document** and index can select the right document
 - slow, if additional parsing needed

Storage

- **parsing documents**
 - document parsed on save and **stored in own data model** (eg. DOM)
 - addressable **numbered nodes** - some operations faster based on numbers
 - no need for parsing on query, more efficient
 - retrieved document **not 100% same**
 - fine granularity for addressing

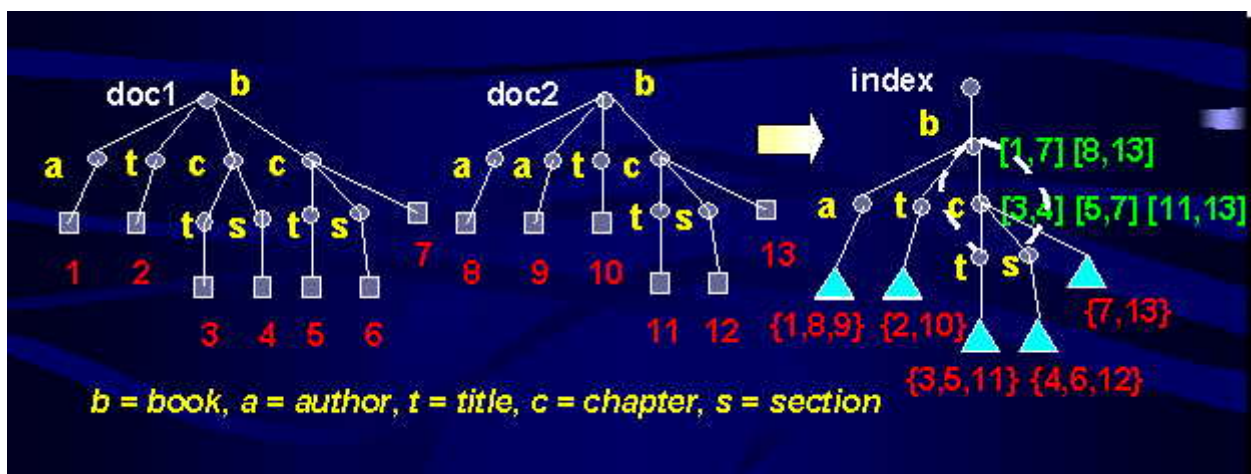
- **partial modifications**

Indexing

- index scope - collection, database, document
- index target - document, node
- **value** index
 - store all combinations of element/attribute values
- **substring** index
 - for contains() etc., store n-grams

Indexing

- **structural** index
 - track existing paths, enriched tree, trie, DataGuide etc.



Benchmark

- compare database performance
- mostly XQuery speed, less often Update
- data generator (up to GBs) and a set of XQueries
- *XMark*, *XBench*, *XMach-1*
- *TPoX* - complex database testing, XQuery and SQL/XML, indexing, XML Schema, XQuery Update