Storing XML Data In a Native Repository

Kamil Toman
ktoman@ksi.mff.cuni.cz

Dept. of Software Engineering
Faculty of Mathematics and Physics
Charles University

Introduction

- Since 1998 XML has become a very popular standard for electronic interchange and application data
- XML documents don't need a rigid schema but they still offer a logical structure
- XML data originate from many different sources and are very heterogenous
- Greater flexibility creates a strong demand of *XML Databases*

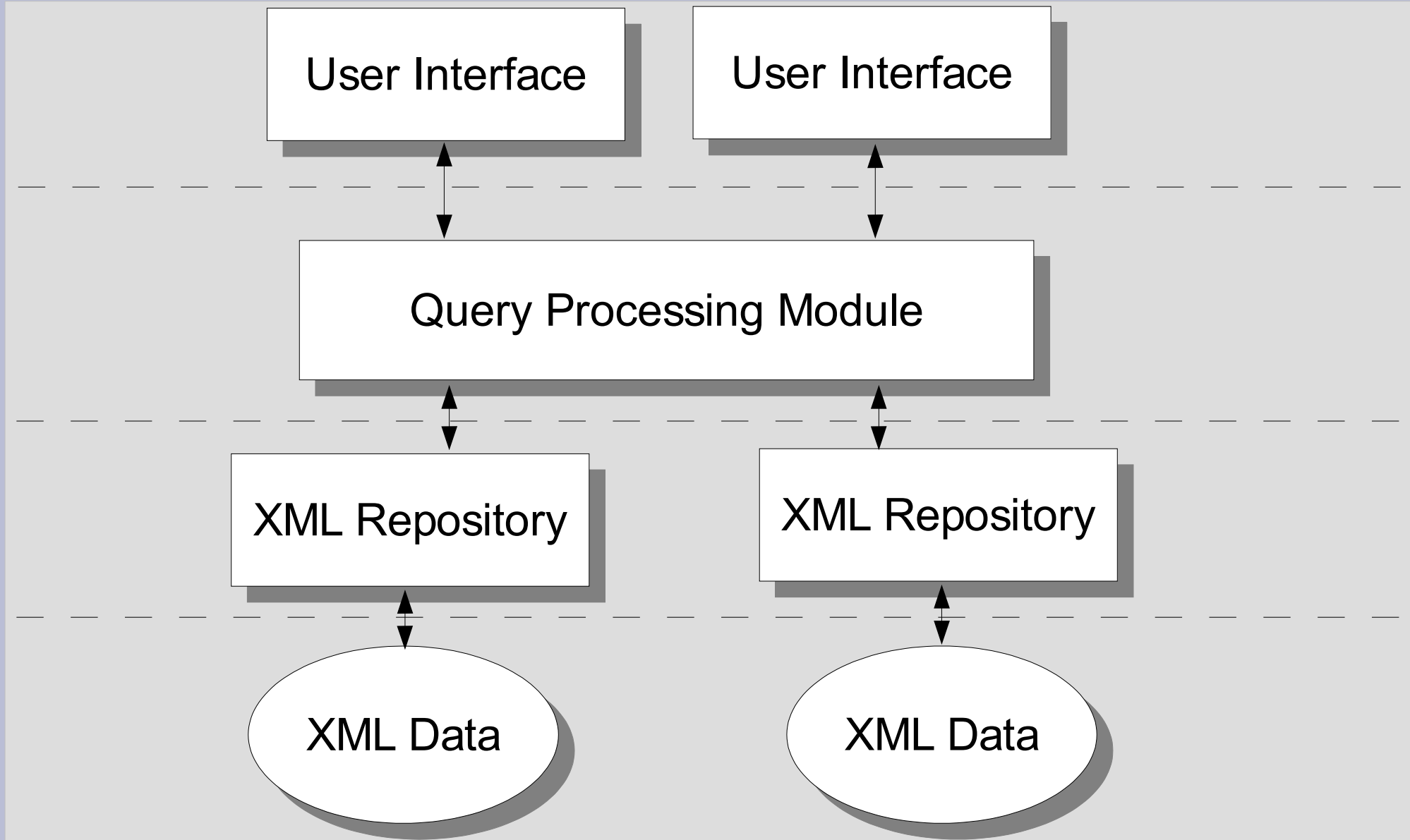
XML Querying

- New XML query languages have been proposed – *XPath* and *Xquery*
- Both languages use the basic concept of *path expressions*
- Implementation of these languages on top of traditional relational and object-relational database systems is problematic
- Storing XML in object-oriented databases is ineffective
- Native XML databases are being developed

SXQ-DB

- Experimental native XML DB to store and manage collections of XML documents with a common DTD
- As the query language, *SXQ* (Simple Xquery) querying language is implemented
- The general and extensible modular architecture is built up on XMLCollection framework

SXQ-DB, Overall Architecture

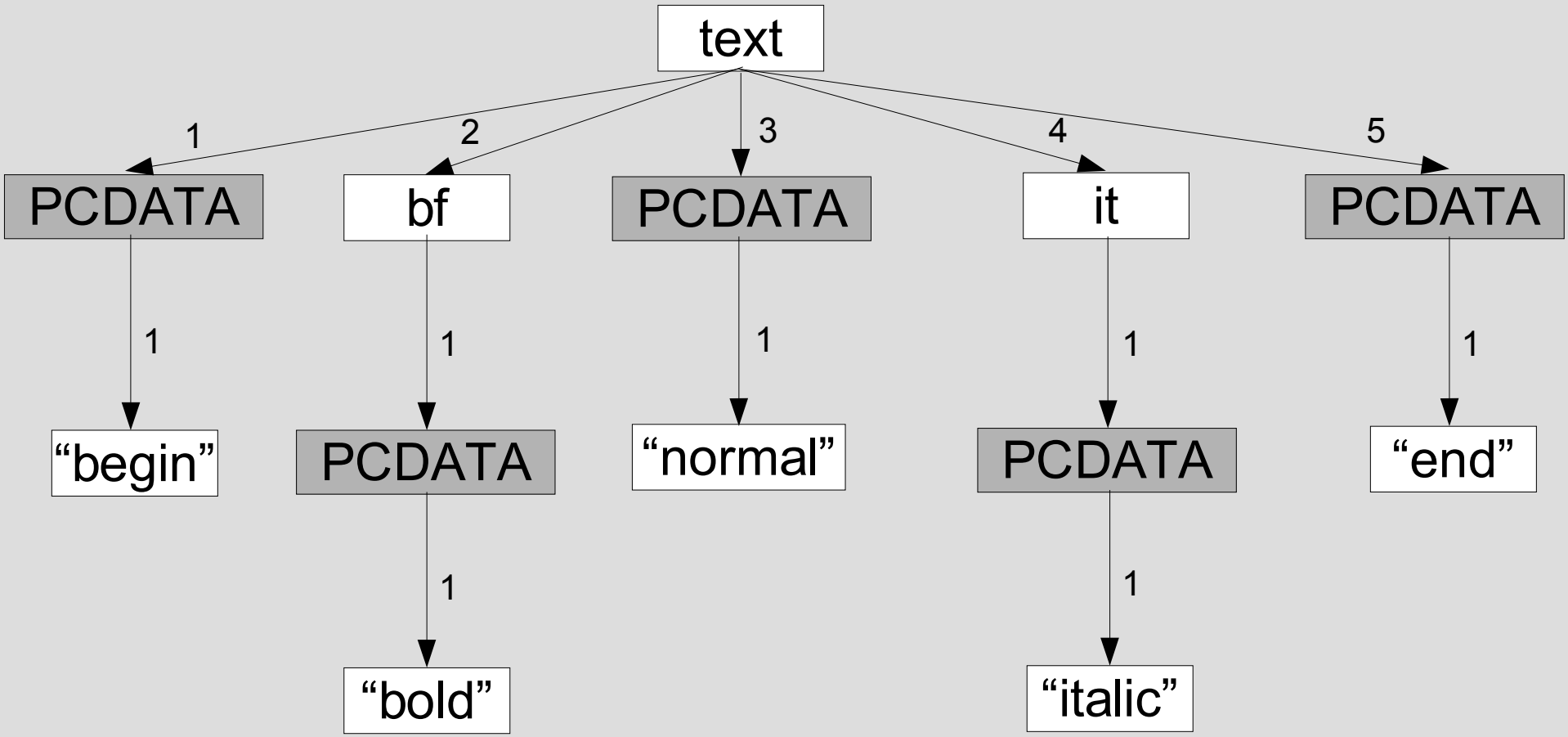


Document Representation

- *XML Information Set* augmented by relevant parts of *XQuery Data Model*
- Oriented tree where to each node is associated a type and a label, vertices with a common parent ordered left-to-right
  - Text values of elements or attributes are represented as artificial nodes
  - Mixed contents elements are modeled as trees

Document Representation

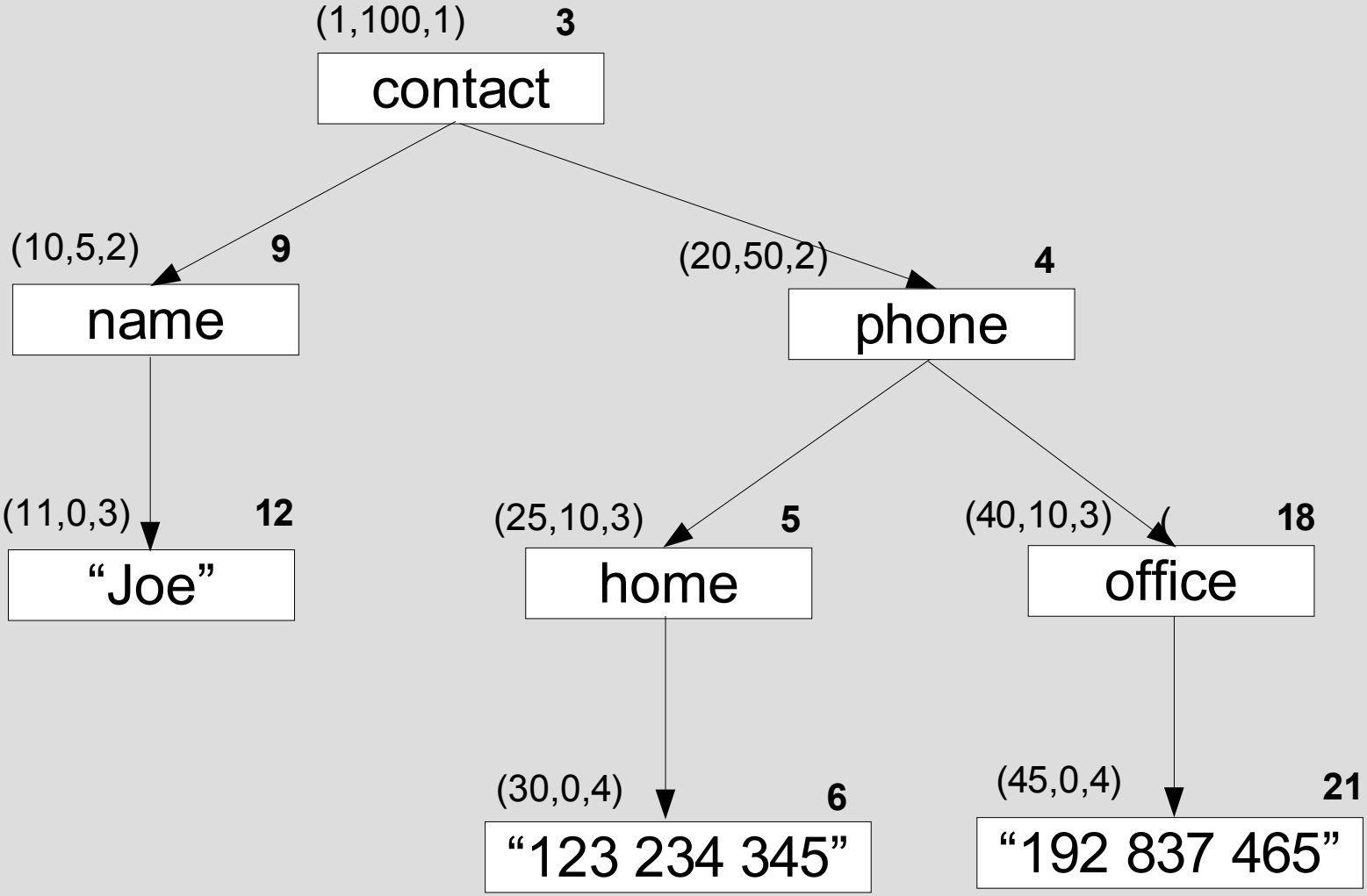
<text>begin<bf>bold</bf>normal<it>italic</it>end</text>



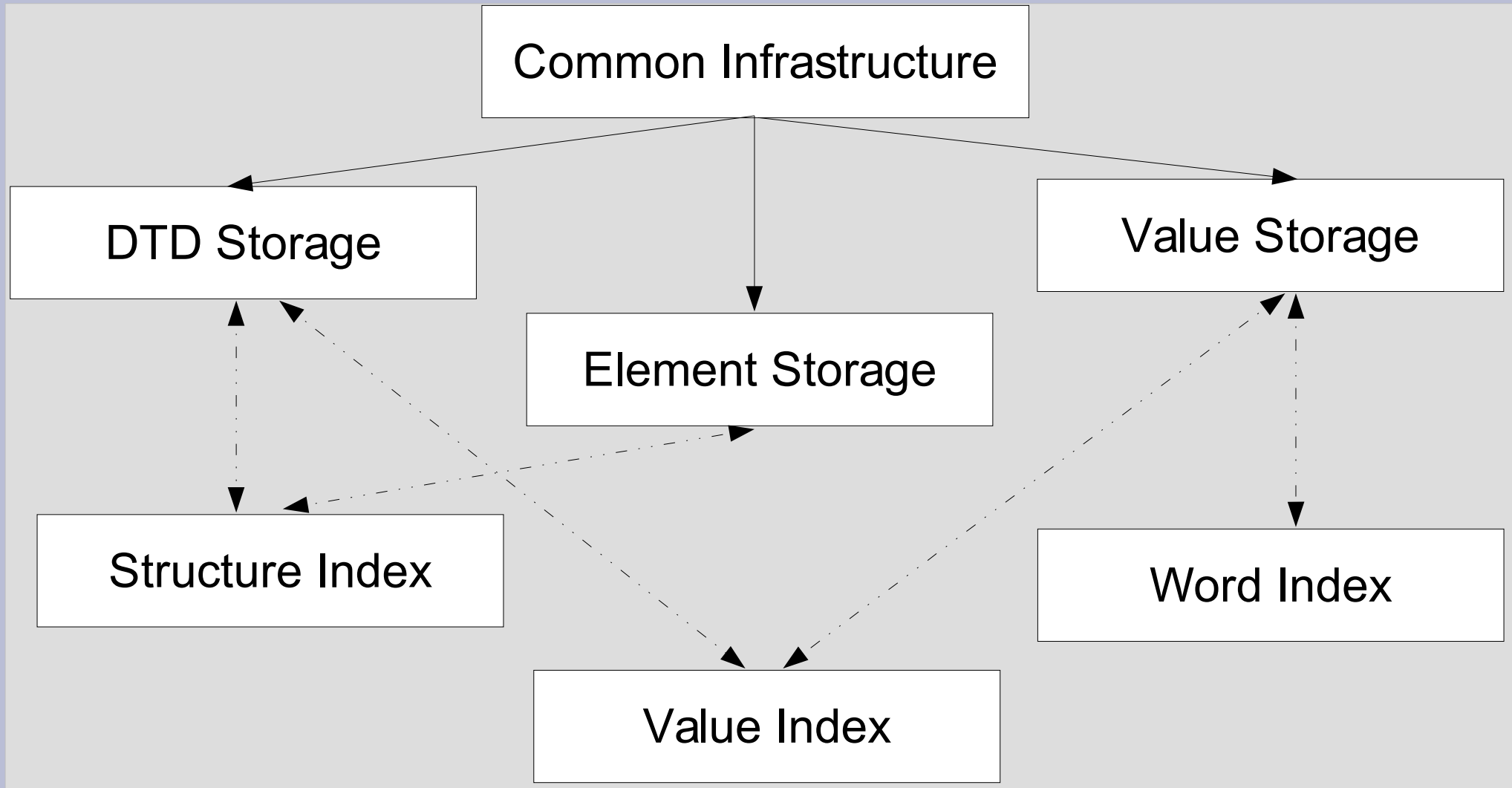
Node Identification

- Numbering scheme: a function that assigns a unique binary identifier to each node
  - This id can be used as a reference in an index or while query evaluation
  - Can be used as on document updates
- Primary: *sequential* numbering scheme
- Secondary: *structural* numbering scheme
  - Allows effective query evaluation utilizing *structural joins*

Node Identification



XML Repository Architecture



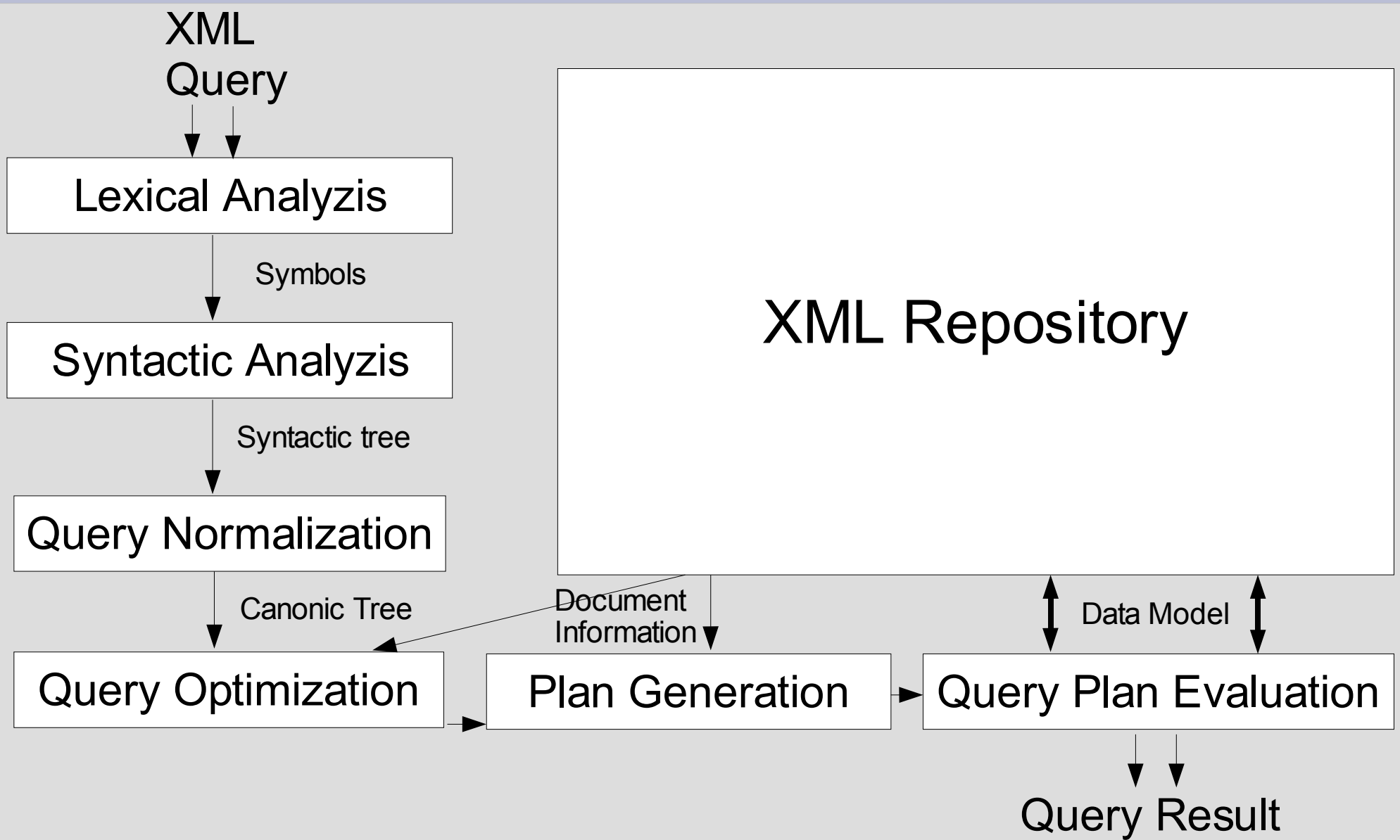
Physical Access To External Memory

- All XML nodes identifiers, their types and adjacent node identifiers are stored into individual fixed-length records in a binary file
- For effective access all records are indexed in a B+-tree
- Better representation of more complex relations between nodes is left to structural indices
- The system resources are limited – *paging mechanism* is used

Object Cache

- XML nodes are accessed frequently but
  - the information is mostly short-lived
  - Every node must be first looked up in an index (possibly unbuffered), its respective page has to be computed and fetched
- To avoid this, secondary object cache is implemented
- All cache objects are kept in main memory at all times and only reinitialized with new data

Query Processing Module



Sources of Difficulties

- Size of indices
  - Besides common word or value indices, additional indices are needed for structural joins or effective tree traversals
- Slow updates:
  - Not only data but even the structure of XML documents may change significantly
  - Expensive index updates may be needed
- Generality of XML query languages
  - Both XPath and XQuery are Turing-complete

Other Native XML Databases

- TIMBER
  - XML tree algebra (TAX) approach
  - XQuery subset translated to TAX operations
- eXist
  - Lightweight, can manage only small to medium sized XML documents
  - XPath subset + fulltext extensions
- dbXML
  - Using B-trees, fully updatable
  - Navigational approach + large indices
- Xindice
  - XPath fully implemented, navigational approach
  - XUpdate supported

Conclusion & Future Work

- Efficient XML database is achievable
  - Chosen data model is sufficient for implementation of the most important parts of XQuery
  - Managing dynamic XML data is much harder than static XML documents
- Future work should be probably focused on
  - Finding a more general way how to express and evaluate the most common XML queries
  - Reducing space needed for structural and term indices of the database

References

- M. Kopecny (2002): Implementacni prostredi pro kolekce XML dat (thesis, in Czech). MFF UK.
- K. Toman(2003): XML data na disku jako databaze (thesis, in Czech). MFF UK.
- J. Cowan, R. Tobin (2001): XML Information Set. *http://www.w3.org/TR/xml-infoset*
- J. Clark, S. DeRose (1999): XML Path Language (XPath 1.0) *http://www.w3.org/TR/xpath*
- M. Marchiori (2003): XML Query Specifications. *http://www.w3.org/XML/Query#specs*
- E. Cohen, H. Caplan, T. Milo (2002): Labeling XML Trees. Symposium on PODS, p. 271-281