# XML Systems & Benchmarks

Christoph Staudt
Peter Chiv

Saarland University, Germany
July 1st, 2003

# Main Goals of our talk Part I

- Show up how databases and XML come together
- Make clear the problems that arise when dealing with XML in databases
- Show up possible solutions on a concrete example: XML database management system Timber

# XML goes database Motivation

- Why do we need to put XML data into databases
  - Growing popularity of XML
- How to save the actual data
  - XML document has graph structure
- Own standardized query language XQuery

# Two Different Approaches

- Store XML Data in a Relational Database
  - Data has to be modified
  - Must match the relational structure
- Use a native XML Database

# Relational databases

- Use Relations containing tuples
  - Store data in a flat design
- Many tried and true systems available

Tradeoffs:

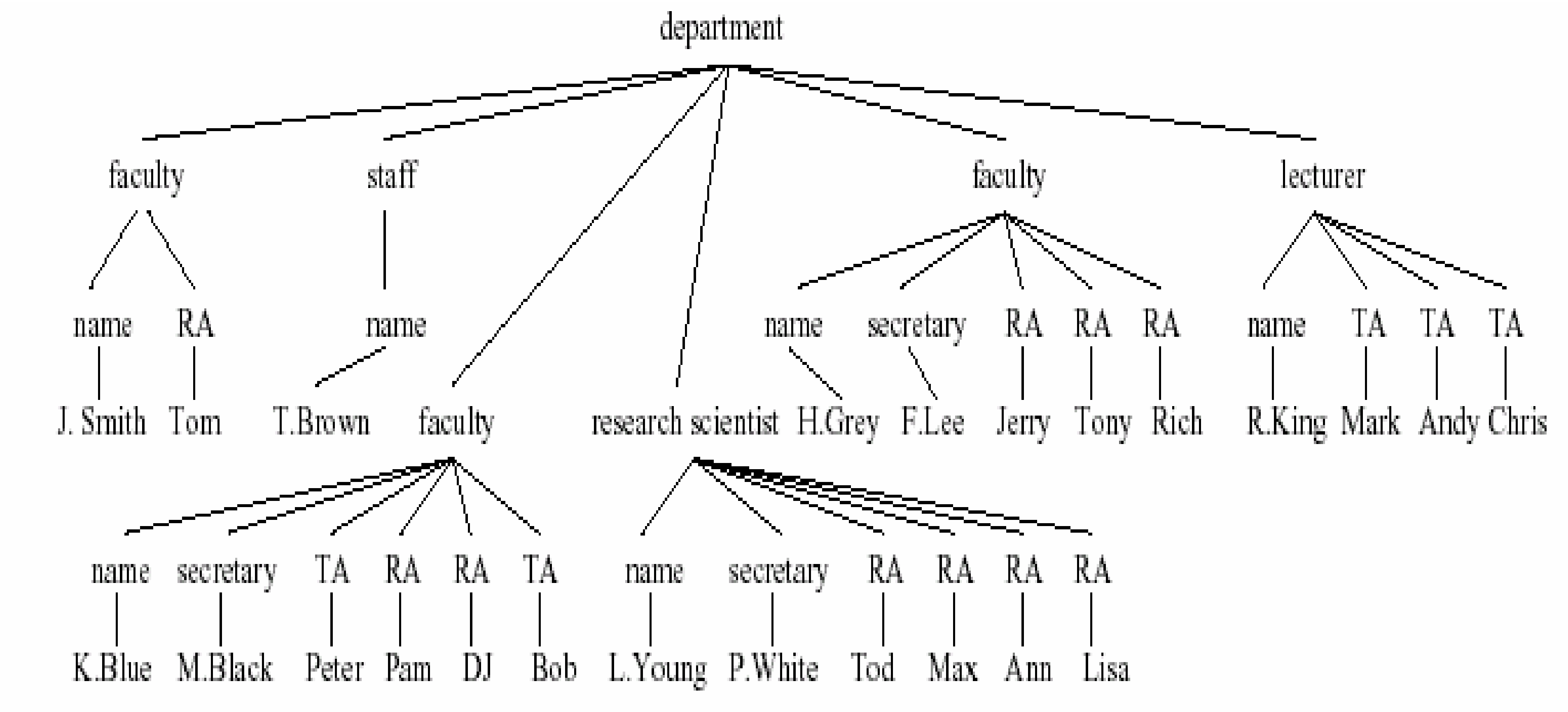
- Flat structure can‘t represent hierarchical design
  - Graph Edges, Elements, Attributes
- Expensive Join Operations necessary to reconstruct the dokuments structure

# Relational Data

| name | phone |
|---|---|
| John | 3634 |
| Sue | 6343 |
| Dick | 6363 |

{ row: { name: “John”, phone: 3634 },
row: { name: “Sue”, phone: 6343 },
row: { name: “Dick”, phone: 6363 }
}

# Sample XML Dokument tree



# Syntax for Semistructured Data

```
Department: &o1
        { staff: &o12 { … },
          lecturer:  &o24 { … },
          faculty: &o29
              { Name: &o52 "Abiteboul",
                Secretary: &o96 { firstname: &243 "Victor",
                                  lastname: &o206 "Vianu"},
                TA: &o93 "Regular path queries with constraints",
                RA: &o12,
              }
        }
```

# Problems handling Semistructured Data

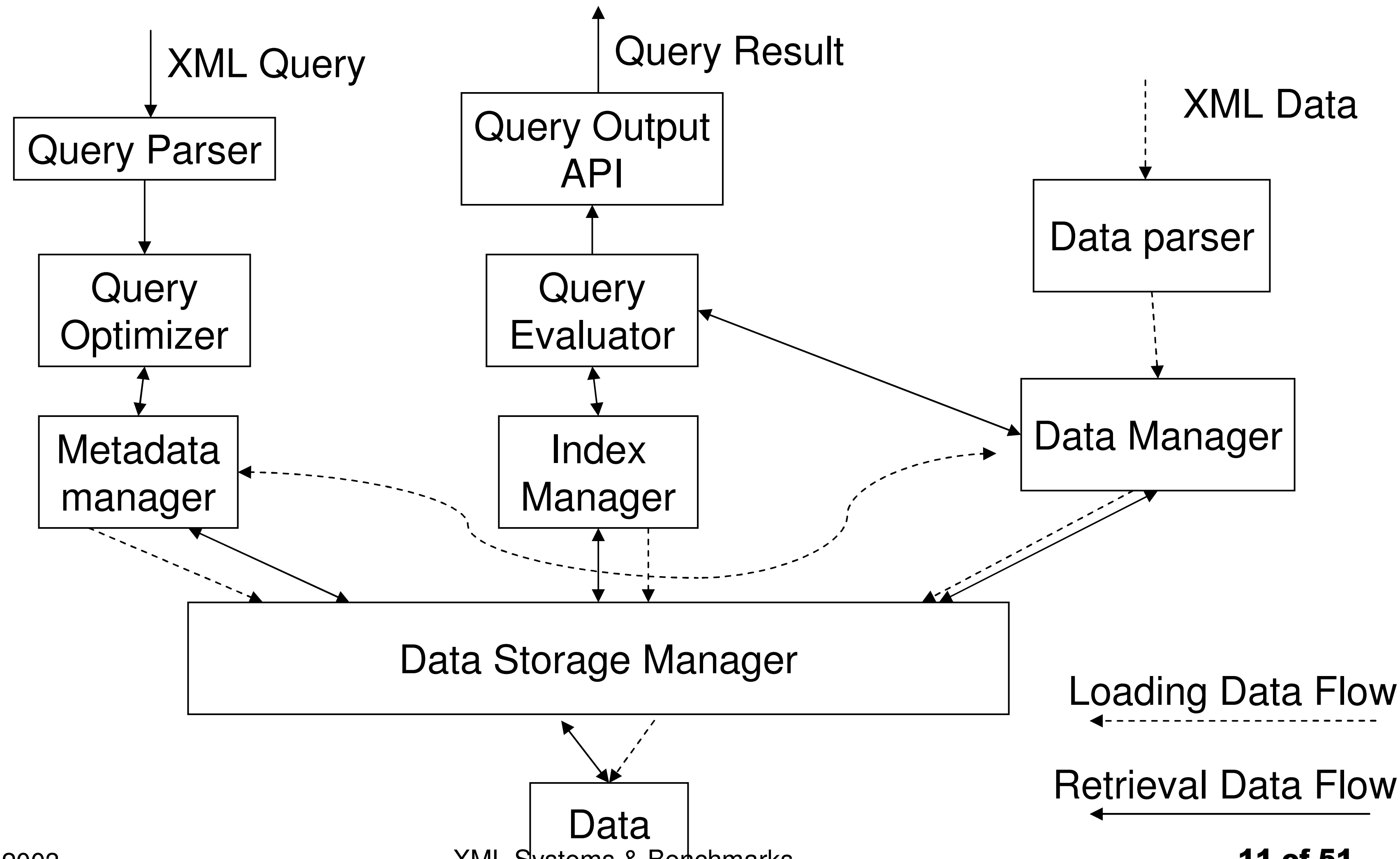
- missing or additional attributes
- multiple attributes
- different types in different objects
- heterogeneous collections

# Touch and feel with a native XML database system

- Timber is a scientific open source native XML database
- Leading commercial system in this field is Tamino. Many more available e.g. Oracle…
- Only an open source System Provides enough implementation information to be studied here.

# Digging deeper into Timber

## Timber system Architecture



# Data Storage (1)

- Uses Shore as backend store
- Loading data
  - Document saved as atomic unit
  - Saved in internal representation:
    - One Node per element
    - Child nodes for sub-elements
    - All attributes clubbed into one child node
    - Content of element node pulled out into child node

# Data Storage (2)

- Labeling nodes
  - Ancestor-descendant relationship and
  - Parent-child relationship
  - Updates are an issue in labeling
    - Leave gaps between successvice labels

# Index / Metadata storage

- Index Storage
  - At the current time only node indices implemented
  - Path indices are being studied
- Metadata Storage
  - Designed to do a good job in the absence of Schema or DTD Information
  - Goal is to use this information when available to advantage

# Query Processing

- XML Query in XQuery
- Parsed into algebraic operator tree by Query Parser
- Query Optimizer reorganizes the tree
- Resulting query plan evaluated by Query Evaluator
- Query Evaluator calls Data- /Index- Manager which in turn call shore

# Tree Algebra

- Guideline: Requirement of an algebra which can manipulate sets of ordered labeled trees
- Tree algebra in Timber: TAX

  Operators: selection, projection, product, set union, set difference, renaming, reordering, grouping

In this talk:

- Handling the Heterogeneity of XML Data
- Selection
- Projection

# Heterogeneity

- XML Dokuments have flexible Structure
  - No direct referencing of nodes possible

- Use of Pattern-Trees to homogeneize
  - Only relevant portions of the tree remain

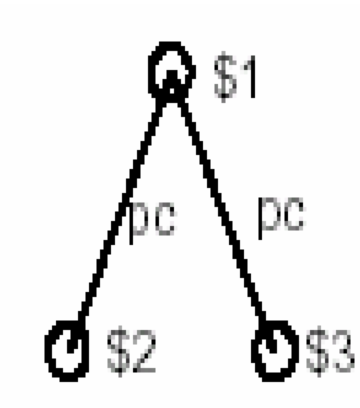
- Results in witness-trees
  - All witness-trees are homogeneous

# Selection

- Takes as an Input:
  - Collection of trees C
  - Selection Predicate: Pattern P
  - List SL
- Returns trees that satisfy selection predicate
- Resulting trees not necessarily homogeneous
  - But: All witness trees are!

# Selection example

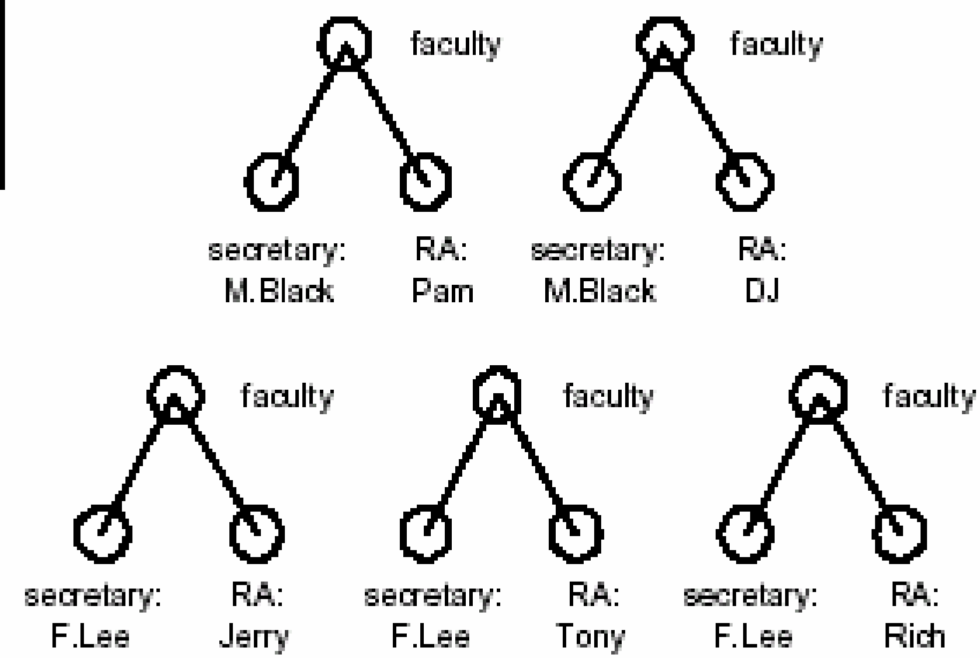
- Pattern tree:




$1.tag = faculty &
$2.tag = secretary &
$3.tag = RA

- Empty adronment list SL

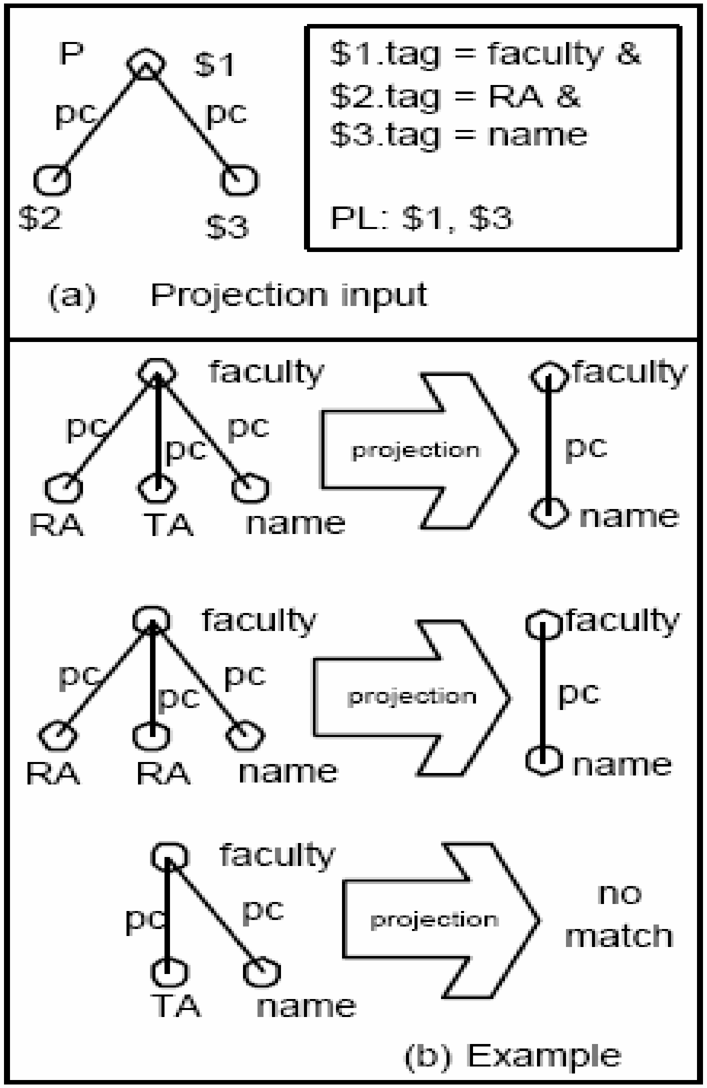
Selection Result:



# Projection

- Takes as an Input:
  - Collection of trees C
  - Pattern tree P
  - Projectionlist PL
- Removes nodes not specified as interesting
- Comparison zu relational projection

# Projection example




- (a) shows a sample projection input
- (b) shows three application examples

# Query Processing

- XML Query in XQuery
- Parsed into algebraic operator tree by Query Parser
- Query Optimizer reorganizes the tree
- Resulting query plan evaluated by Query Evaluator
- Query Evaluator calls Data- /Index- Manager which in turn call shore

# Query Optimization (1)

- Structural Join Order Selection
  - Pattern Matching requires Structural Joins
  - Order in which structural Joins are computed makes differences to cost of query evaluation
- Query Optimizer‘s tasks:
  - Enumerate all evaluation plans
  - Estimate their costs
  - Choose the one with lowest estimated cost

# Query Optimization (2)

- **Result Size Estimation**
- To estimate cost are required:
  - Accurate estimate of cardinality of final query result Example: Computation of upper bound
- Example: Computation of upper bound
  - Pattern tree: faculty-TA (parent-child)
  - Lack of information about structure:
  - Product of cardinality: 3 faculty x 5 TA = 15

# Query Optimization (3)

- Position Histogramm
  - Using Start- and Endlabels to create Histogramm
  - Left column: 1. Half of nodes
  - Right column: 2. Half of nodes

faculty

| | |
|---|---|
| 0 | 1 |
| 2 | |

TA

| | |
|---|---|
| 0 | 3 |
| 3 | |

⇨ (2x2 + 1x3) = 7 upper bound (compared to 15)

# Query Processing

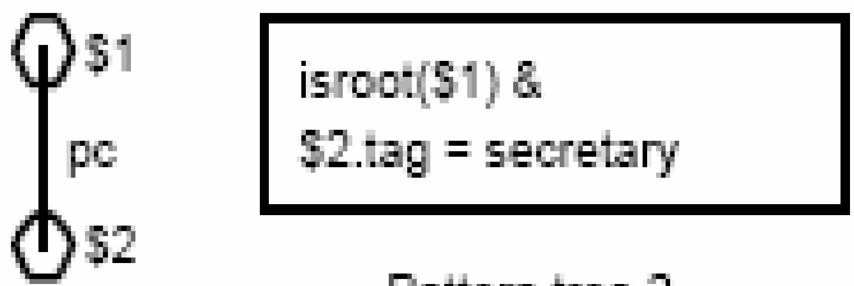
- XML Query in XQuery
- Parsed into algebraic operator tree by Query Parser
- Query Optimizer reorganizes the tree
- Resulting query plan evaluated by Query Evaluator
- Query Evaluator calls Data- /Index- Manager which in turn call shore

# Query Evaluation (1)

- Pattern Tree Reuse
  - Same pattern trees often used more than once
  - Computationally profligate to re-evaluate

  - Persistence of matches accomplished using PIDWIDs
    - PID Pattern tree identifier
    - WID witnes node identifier

# Query Evaluation (2)




Pattern tree 2

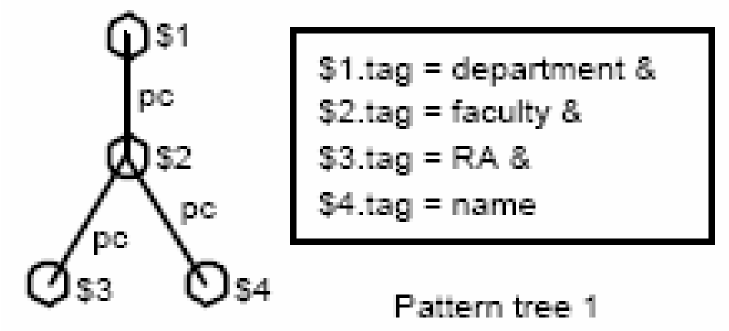
Occuring Problems:

- Operation change the date Structure
- Example:
  1. Pattern tree 2 on Database
  2. First aply select which returns all faculties and their child nodes.

     Now aply pattern tree 2: Will return every secretay in the database

# Query Evaluation (3)

- Node Materialization
  - We mentioned equivalents for RIDs in relational databases
  - Physical algebra has own materialization operator
  - What does it mean to materialize „one node“
  - Example:
  - The name of each faculty member
    that has an RA




Pattern tree 1

# Query Evaluation (4)

- Structural Join
  - Used to specify parent-child / Ancestor-descendant relationships
  - Each edge in a pattern tree represents a structural relationship
  - 2 Step processing
    - Finding candidate nodes (with the help of indices)
    - Bringing them into relationship (structural join)

# Query Evaluation (5)

- Structural Join example / intuition:
  - Pattern to be matched:

    Parent node with tag faculty

    Child node with tag secretary
  - Nagivational plan:

    Find one node from the pattern and navigate from there on
  - Structural Join Plan

    Create lists of matches for each individual node in the pattern

    Then perform structural join to find pairs

# Summary - Part I

- We saw two different approaches to storing XML in a Database. A relational one and a native one
- Insight into native XML DBMS „Timber“
  - Overall Architecture
  - Tree Algebra: Pattern Tree and Witness Trees
  - Loading the database and Query processing

# Transition to Part II „Benchmark“

- Apart from Timber there are many other proposals for a XML DBS
- Question:

  „How to assess the different XML DBS from the user‘s point of view?“

# Main goals of our talk Part II

- Part I : developers‘ point of view
- Part II: users‘ point of view
  - Show how and why to benchmark XML Database Systems
  - XMark Benchmark

# Motivation (1)

- **What does „benchmark“ mean?**
  - Scale to assess and compare new techniques and system components
  - Simple: „Comparing pros & cons of different systems“
- **Why „XML“ benchmarking?**
  - XML DBMS grow in complexity & capacity
  - Many suggestions of different ways to store XML data
  - ⇨ Variably different query characteristics of the data
  - Need of benchmarks from the economic view:

    Help to determine success or failure of implemented XML-based solutions

# Motivation (2)

- **Groups profiting by Benchmarks**
- Database vendors
  - Verify and refine their query processors
- Customers
  - Which product? Costs? Which system fits best my needs?
- Researchers
  - Tailor exitsting technology for use in XML
  - Refinement or design of algorithms

# Motivation (3)

- **What this talk will show**
- No conclusive methodology for assessing differences of different storage schemes available to date

⇨ Talk gives desiderata for general purpose benchmark for XML databases

# Important pre-considerations

„What operations on an XML document are conceivable and reasonable?“

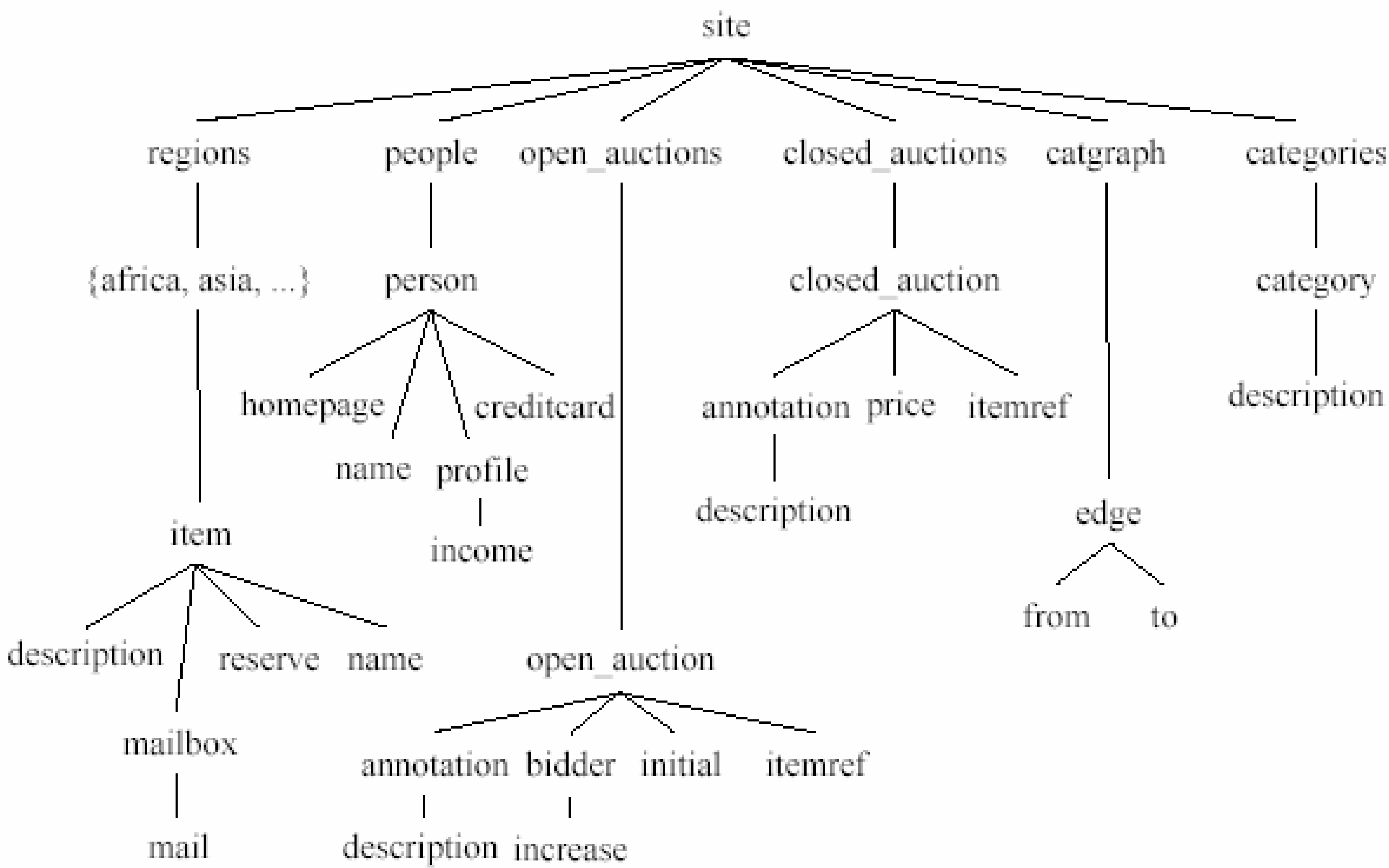
# XML Benchmark requirements

- 1. Step:
  - **10 basic challenges for comprehensive analysis covering all performance critical aspects of processing XML**
  - order, type problem, hierachical order, loose schema
- 2. Step:
  - Development of XMark taking those 10 challenges into account

# XMark Benchmark

- Evaluates retrieval performance of XML stores and query processors
- Framework to assess the abilities of an XML database to cope with broad range of different query types
  - Each set of queries challenge a particular aspect of query processor
  - Use of XQuery

# XMark Database



# 14 Queries‘ concepts (1)

- Exact Match
  - `Return the name of the person with ID ‚person0‘.`
  - Ability to handle simple string lookups with a fully specified path.
  - Gives a performance baseline.

# 14 Queries‘ concepts (2)

- Reconstruction aka Round Tripping
  - Reconstruction of the original XML document from its broken-down representation.
  - XML enables extensive reuse of material.
  - `List the the names of items registered in Australia along with their descriptions.`

# 14 Queries' concepts (3)

- Path Traversals
  - `Print the keywords in emphasis in annotations of closed auctions.`
  - `Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.`
  - Quantify costs of long path traversals

# 14 Queries‘ concepts (4)

- Missing Elements
  - `Which persons don't have a homepage?`
  - Queries test how well query processor know how to deal with semi-structured aspect of XML data, especially elements that are declared optional in the DTD.

# 14 Queries‘ concepts (5)

- Casting
  - Strings are the generic data type
  - `How many sold items cost more than 40?`
  - Queries challeng the DBMS in terms of the casting primitives it provides.

# 14 Queries‘ concepts (6)

- Chasing References
  - Queries define horizontal traversals with increasing complexity.
  - List the names of persons and the numbers of items they bought.
  - List the names of persons an the names of the items they bought in Europe.

# 14 Queries‘ concepts (7) – in short

- Full Text
  - Conduct full-text search in the form of keyword search.
- Ordered Access
  - Insight how DBMS cope order of XML documents
  - Insight how efficiently the DBMS handle queries with order constraints.

# 14 Queries‘ concepts (8) – in short

- Regular Path Expressions
  - Insight how well query processor can optimize path expressions and prune traversals of irrelevant parts of trees.
- Construction of Complex Results
- Joins on Values
  - Queries tests database‘s ability to handle large (intermediate) results.

# 14 Queries‘ concepts (9) – in short

- Function Application
  - User defined functions (UDF)
- Sorting
- Aggregation

# What have we seen?

- Part I:
  - XML and DBMS
  - Native XML Database „Timber“
- Part II - Benchmark:
  - Why benchmarking?
  - How to benchmark?
  - XMark
- Questions?