A Journey towards More Efficient Processing of XML Data in (O)RDBMS

Irena Mlynkova

irena.mlynkova@mff.cuni.cz

Charles University

Faculty of Mathematics and Physics

Department of Software Engineering

Prague, Czech Republic

Introduction

- XML = a standard for data representation and manipulation
  - Growing demand for efficient managing and processing of XML data

⇒ A natural alternative: To exploit tools and functions of (object-)relational database management systems ((O)RDBMS)

  (–) XML trees vs. flat relations ⇒ inefficiency

  (+) Long theoretical and practical history, mature technology

⇒ The techniques should be further enhanced

Goals of This Presentation

Proposal of improvement of XML processing based on (O)RDBMS

- **Overview and classification of existing approaches**
- **Motivation for improvements**
- **Proposal of improvement of user-driven methods**
- **Experiments**
- **Conclusion**

Content

1. **Overview of existing approaches**
2. Motivation for improvements
3. Proposed improvement
4. Experiments
5. Conclusion

Database-Based XML Processing Methods (1)

Key concern: Choice of the most efficient XML-to-relational mapping strategy

- **Fixed – predefined set of mapping rules and heuristics**
  - **Generic vs. schema-driven**
- **Adaptive – adapt the target schema to intended usage**
  - **Cost-driven**
- **User-involving – storage decisions in hands of users**
  - **User-defined vs. user-driven**

Database-Based XML Processing Methods (2)

- **Generic** vs. **schema-driven** – omitting / exploiting XML schema
  - Straightforward mapping
- **Cost-driven** – search a space of possible mappings and choose the one which conforms the target application most = the least "expensive"
  - Application: sample XML data, XML queries
- **User-defined** vs. **user-driven** – the amount of user involvement
  - User-driven = a type of adaptivity
    - Schema is adapted to the annotations

User-Driven Methods: Shortcomings and Improvements

- Default mapping strategy is always fixed
  - Systems are able to store schema fragments in various ways $\Rightarrow$ adaptive enhancing is natural
- Weak exploitation of user-given information
  - Annotations are just directly applied
  - Idea: Annotations = "hints" how a user wants to store particular XML patterns $\Rightarrow$
    - We search for similar fragments
    - We use the knowledge in adaptive enhancing

$\Rightarrow$ General idea: Emphasis on user-given information

Mlynkova, Pokorny: Adaptability of Methods for Processing XML Data using Relational Databases - the State of the Art and Open Problems. IJCSA, volume 4, issue 2, 2007. ISSN 0972-9038.

Content

Why User-Given Information? (Example 1)

- Situation: Documents with XHTML fragments
- Problem: Shredding into tables = inefficient fragment reconstructions
  - XHTML DTD contains complete graphs on up to 10 nodes
- What if the real complexity is much simpler?
  - Statistical analysis: Yes, it is <u>much</u> simpler!

$\Rightarrow$ Simpler storage strategy (CLOB)

Mlynkova, Toman, Pokorny: Statistical Analysis of Real XML Data Collections. COMAD 2006. Tata McGraw-Hill Publishing Co. Ltd., 2006. ISBN 0-07-063374-6.

Why User-Given Information? (Example 2)

- **Situation: Updatability of data vs. fast query evaluation**
- **Problem: Amount of mutual relationships information**
  - **Fast querying $\Rightarrow$ additional indices, numbering schemes**
  - **Fast updates $\Rightarrow$ the simplest information of mutual relationships**
  - **Fast querying, fast updates $\Rightarrow$ compromise**

Why User-Given Information? (Example 3)

- Situation: Data redundancy
- Question: Is it always necessary to strictly follow the rules of normal forms?
  - No, it is not.
  - Optimal XML-to-relational storage strategy = 4NF
    - No null values, no redundancy

⇒ In all the cases we need additional information given by <u>a user</u>
  - XML data, XML queries, annotations…

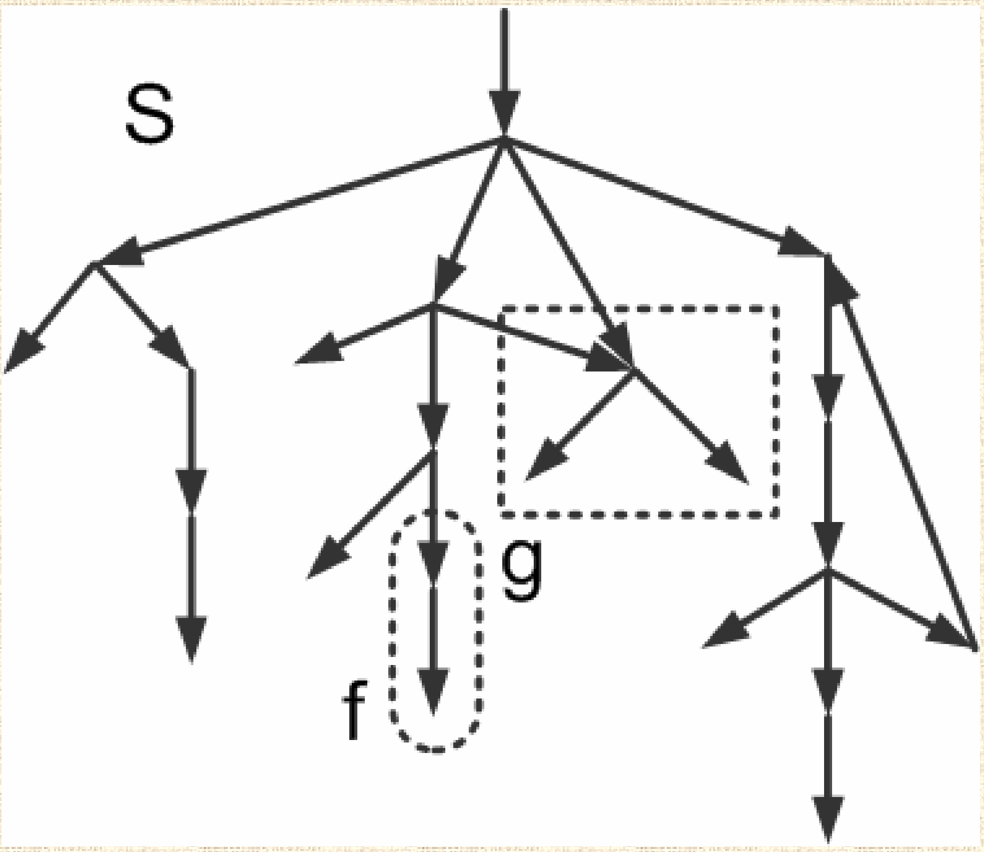
Content

1. Overview of existing approaches
2. Motivation for improvements
3. **Proposed improvement**
4. Experiments
5. Conclusion

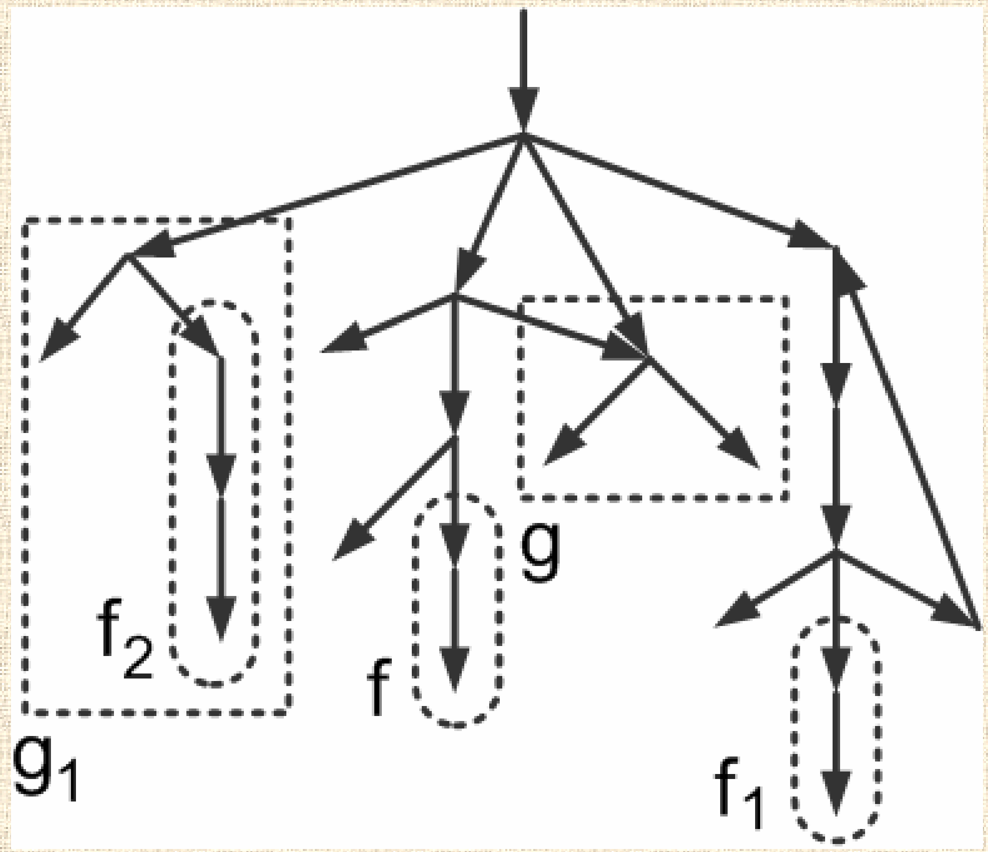
Basic Ideas

- Searching for similar fragments in the not annotated schema parts
  - The user is not forced to annotate all schema fragments
  - The system can reveal new structural similarities
- Searching for optimal mapping strategy for the remaining schema fragments
  - Adaptive strategy

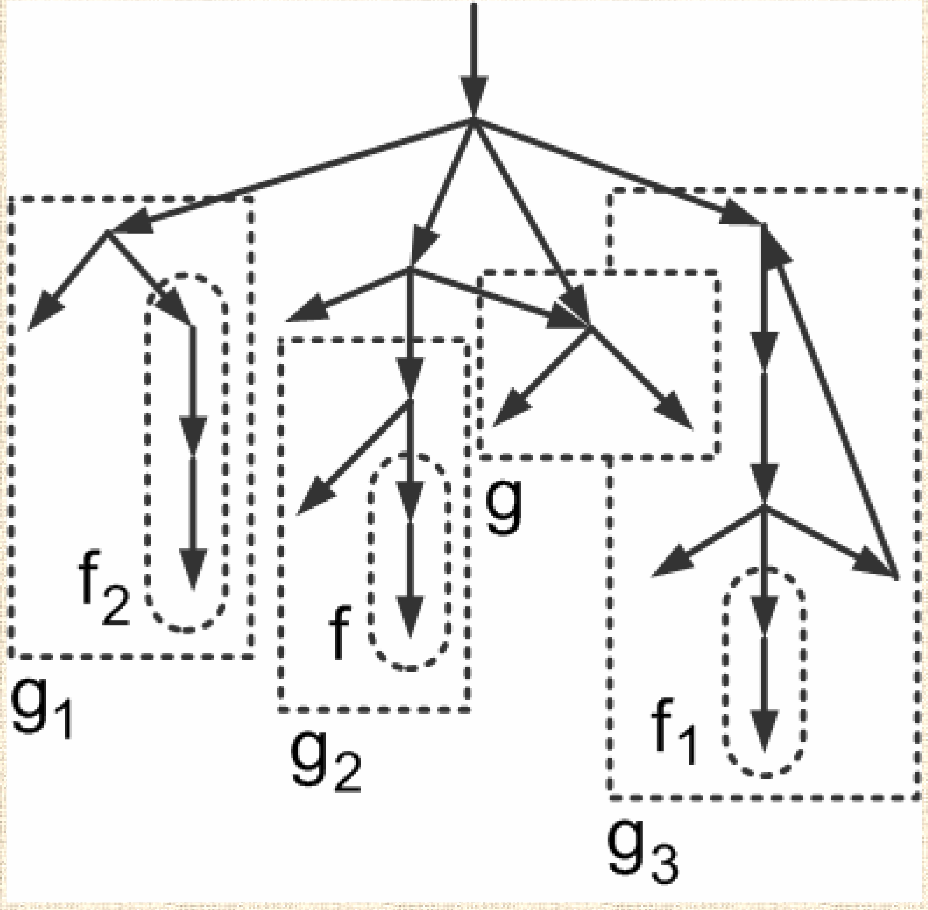
Step 1. Annotated Schema



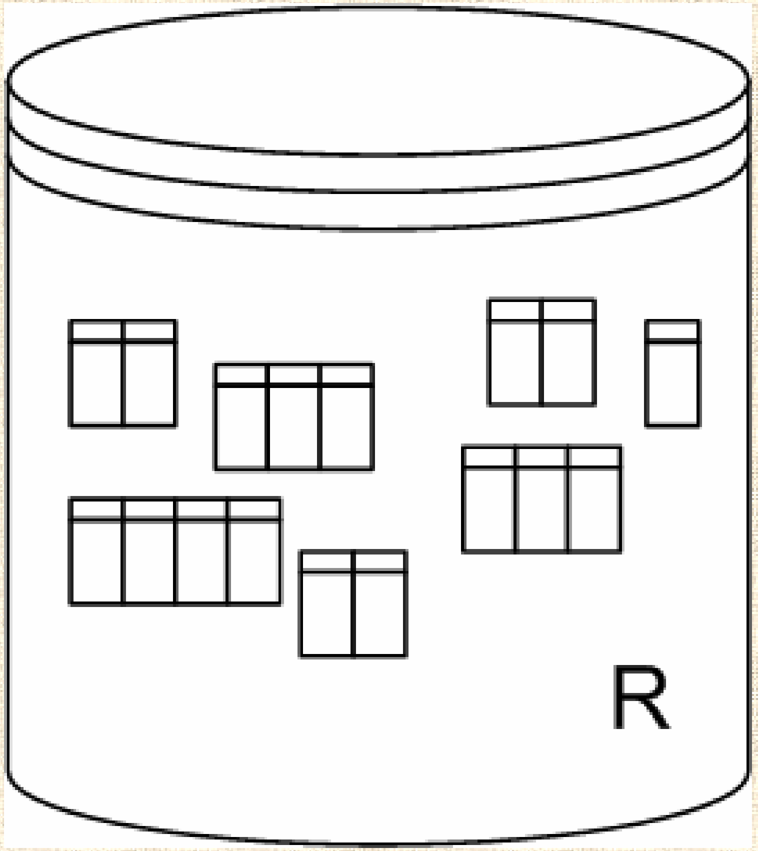
Step 2. Searching for Similar Fragments



Step 3. Adaptive Strategy



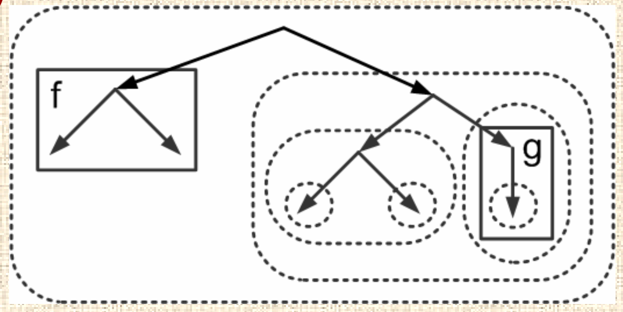
Step 4. Mapping to Relations



Open Issues

- **What similarity function is used? Can we optimize the exhaustive search for similar fragments?**
- **What types of annotations, i.e. fixed mapping strategies, are supported? How are they combined?**
- **Should we use a classical adaptive strategy?**

Similarity Function and Search Algorithm (1)



- Closely related
  - No knowledge of function
    $\Rightarrow$ Few ways how to avoid exhaustive search
    - Clustering (expensive preprocessing)
- Idea: Exploit knowledge of the similarity function
- Modification of a classical approach:
  - A set of matchers = partial similarity functions
    - Similarity of a particular feature
      - e.g. depth, number of nodes, complexity of content…
  - A composite similarity function = aggregates the results
    - Weighted sum

Similarity Function and Search Algorithm (2)

- Features of matchers $\Rightarrow$ a bottom-up strategy
  - Knowledge for child nodes $\Rightarrow$ knowledge for parent node
    - e.g. depth, number of nodes
- Idea: Searching can terminate if the similarity function reaches its optimum
  - Hard to define a function having a single optimum
- Heuristics: Searching can terminate if reasonable amount of matchers exceed their <u>single optimum</u>
  - e.g. similarity of depths
    - With growing number of nodes the depth grows until it reaches the depth of the searched schema fragment

Mlynkova, Pokorny: Similarity of XML Schema Fragments Based on XML Data Statistics. Innovations 2007, Dubai, United Arab Emirates, November 2007. IEEE Computer Society Press.

Types of Annotations

- Particular mapping methods influence versatility of implementation
  - CLOB, shredding to tables, indices, numbering schemes…
- Key aspect: Intersection of annotated fragments
  - Redundant - both methods are applied on intersection
    - XHTML fragments ⇒ shredding + CLOB
  - Overriding - only one of the methods is applied
    - Classical user-driven strategies
  - Influencing - both methods are combined into one storage strategy
    - Shredding + indices/numbering schemes

Mlynkova, Pokorny: UserMap - an Adaptive Enhancing of User-Driven XML-to-Relational Mapping Strategies. ADC 2008, Wollongong, Australia, January 2008. Australian Computer Society, Inc.

Adaptive Strategy (1)

- **Classical approach: Target DB schema is adapted to sample XML data and queries**
  - **+ annotations = too much information**
- **Idea:**
  - **Queries = How the data are typically manipulated**
  - **Data = How complex are XML documents**

⇒ How to store the data

  - **Annotations = How particular schema fragments should be stored**

⇒ Annotations can be reused ⇒ no need for additional information

Adaptive Strategy (2)

- **Key operations:**
  - **Contraction = replaces each annotated fragment with an auxiliary node**
  - **Expansion = all auxiliary nodes are expanded to original schema fragments**
- **Algorithm:**
  1. **The searching for similar fragments and operation contraction repeats until there are no identified candidates for annotating**
  2. **The resulting schema is expanded**
- **Intersection of original and new annotations: Newly defined are overridden**

Content

1. Overview of existing approaches
2. Motivation for improvements
3. Proposed improvement
4. **Experiments**
5. Conclusion

Experiments

- Experimental implementation *UserMap*
- First idea: Analysis of efficiency of the resulting mapping? Useless.
  - Basic idea: user assigns a mapping strategy to a schema fragment suitable for the actual application
    - Can be highly inefficient in the general case
- Aim: Analysis of behaviour of the strategy on real-world data
  - Can we find similar schema fragments?
  - Can we find any in contracted graphs?
  - How many contractions can be applied, if any?
- Assumption: Reliable and tuned similarity function

Mlynkova, Pokorny: Similarity of XML Schema Fragments Based on XML Data Statistics. Innovations 2007, Dubai, United Arab Emirates, November 2007. IEEE Computer Society Press.

Real-World Data

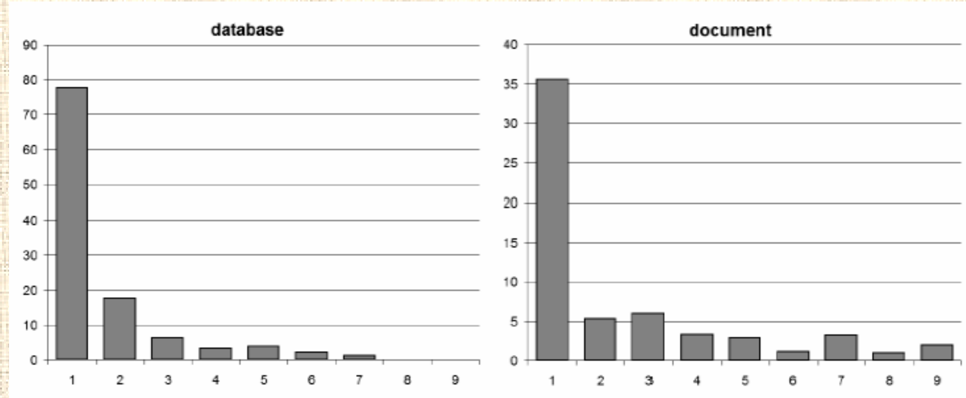
- 99 real-world XML schemes used in statistical analysis
  - Divided into 5 categories - database, document, exchange, report, research
    - Classical data and document centric + finer division
- Testing set of real-world schema fragments
  - 5 data-centric, 5 document-centric, 3 relational, 3 DNA
- Modification of algorithm: Testing set represents annotations, used for all 5 categories of schemes

Mlynkova, Toman, Pokorny: Statistical Analysis of Real XML Data Collections. COMAD 2006. Tata McGraw-Hill Publishing Co. Ltd., 2006. ISBN 0-07-063374-6.

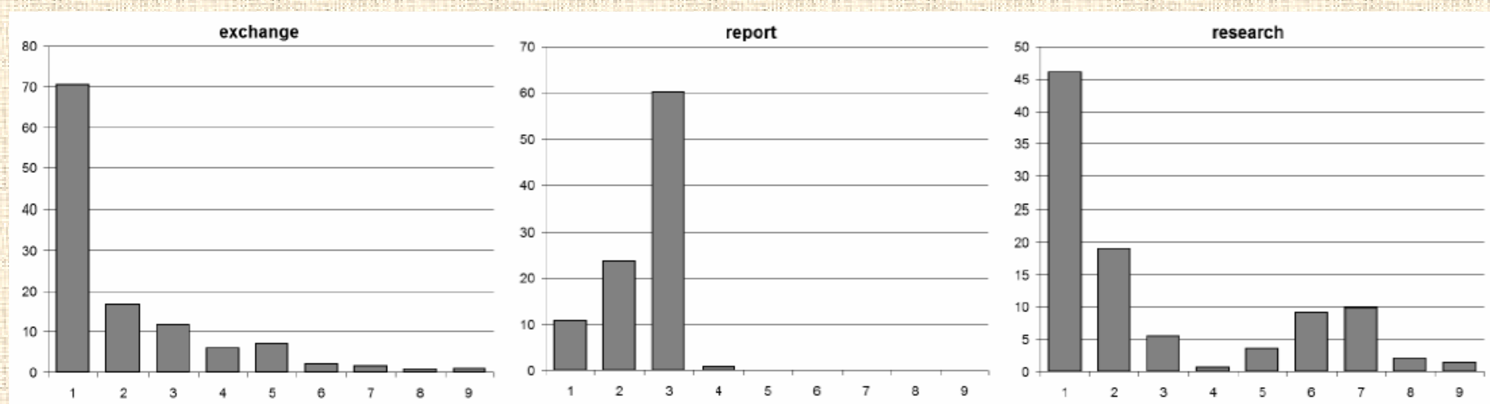
Results (1)

Characteristic	dat	doc	ex	rep	res
Average number of iterations	2.7	3.9	2.9	4.1	4.3
Average % of not annotated nodes	2.1	53.4	13.5	25.6	31.1
% of fully contracted schemes	93.7	22.2	81.1	0.0	28.6

- **Number of iterations is reasonable**
  - **Multiple contractions are performed**
  - **No extreme values**
    - **Results correspond to depth of documents**
- **Schemes are usually not fully contracted**
  - **Default mapping is necessary**
  - **More information should be used**



Results (2)



- **The percentage of annotated nodes is usually highest in the first iteration**
  - **Exception: Report, research - low number of sample data, irregular structure**
- **No relation between types of schema fragments and iterations $\Rightarrow$ no degenerated schemes**

Content

1. Overview of existing approaches
2. Motivation for improvements
3. Proposed improvement
4. Experiments
5. **Conclusion**

Conclusions and Future Work

- Experiments $\Rightarrow$ user-given information can and should be further exploited
- Current and future research
  - Combination with classical adaptive methods
    - Exploitation of sample data and queries
    - Too many input data vs. fully contracted schemes and better results
  - Efficient querying over the resulting schema
    - Query plans vs. intersecting annotations

Thank you