



Advanced databases and data models:

Research topics: Some examples

Lena Strömbäck



June 17, 2009

1

Content

- How hard is XML querying?
- Making data accessible
 - Keyword search
 - Presentation and visualisation
- A web of concepts
- Alternative DB models
- Discussions about lab3



How complex is XML?

XML used for many web applications
Common to have incomplete information

Pablo Barcelo, Leonid Libkin, Antonella Poggi, Cristina Sirangelo XML with Incomplete Information: Models, Properties, and Query Answering. Proceedings of PODS 09, June 29-July 2, 2009, Providence, Rhode Island, USA.

Which kind of incompleteness can there be?

What does this mean for processing and querying?


Relational databases

NULL, Closed-world assumption vs open world assumption

Name	Phone	Address
Ludvig	12345	x
Lena	23456	x

Implications on query efficiency!

- Querying (conjunctive):
 - PTIME (Codd) NPComplete (with vars)
- Relational algebra:
 - coNPcomplete (CWA) undecidable (OWA)



Incompleteness of XML

- (a) node ids (they can be replaced by node variables);
- (b) node labels (they can be replaced by wildcards);
- (c) precise vertical relationship between nodes (we can use descendant edges in addition to child edges);
- (d) precise horizontal relationship between nodes (using younger-sibling edges instead of next-sibling).

Trees vs DOM-trees

Problems:

- Consistency
- Membership
- Querying

Results:

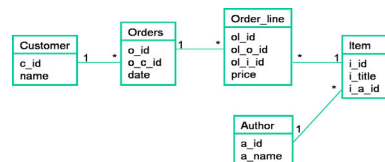
- Presence of DTDs makes consistency and query answering intractable
- Lack of complete structure leads to intractability
- For tractability with nulls:
 - Only unions of conjunctive queries
 - Rigid incomplete trees (further structural restrictions.)

Keyword searches over relational databases

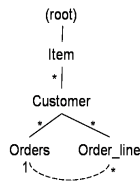
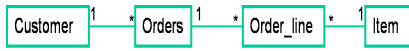
Wang, Tatemura, Sawires, et al.
Hierarchical Result Views for Keyword Queries over Relational Databases
KEYS 2009, pages 3-8.

Keyword queries easy to interpret, but how to interpret results.

Query: Jeff, database



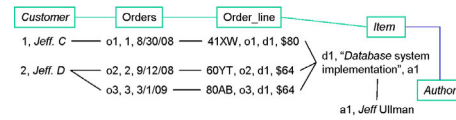
Example idea:



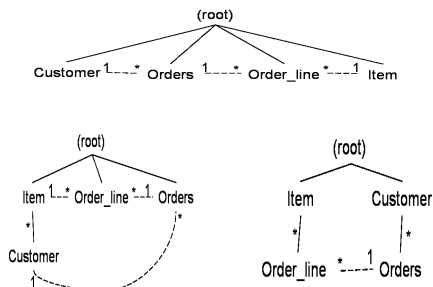
Item: Database system implementation	
Customer: Jeff. C	
Orders:	8/30/08, id=1
Order_line:	41XW, \$80, id=1
Customer: Jeff. D	
Orders:	9/12/08, id=2
Orders:	3/1/09, id=3
Order_line:	60YT, \$64, id=2
Order_line:	80AB, \$64, id=3

Algorithm

1. Matching keywords to tuples
2. Identifying joining tuples
3. Enumerate all possible join queries



How to compute the most efficient tree



A First Study on Strategies for Generating Workflow Snippets.

Tommy Eilkvist, Lena Strömbäck,
 Linköpings universitet, Sweden
 Lauro Didier Lins and Juliana Freire
 University of Utah, USA

Scientific workflows

- Scientific workflows are increasingly being used to represent computational processes.

Workflow snippets – state of the art

- Emphasis on meta-data
- Low quality when information is insufficient or absent

June 17, 2009

Our aim

Besides meta-data, leverage the information present in workflow descriptions

Module
Port
Parameter
Connection
Annotations
Workflow collection
Query keywords
Result set

Problems: Graphs can be large, how to select important information, graph processing is expensive.

Requirements for snippets

- Self-contained
 - A snippet should contain the context of a keyword
 - If a keyword matches a module, its parameters or annotation then that module should be included in the snippets.
- Representative
 - The user should be able to grasp the essence of the result from its snippet.
 - Include the modules representing the most prominent features of a workflow and include them in the snippet.
- Distinguishable
 - The snippet should make the corresponding query result distinguishable from other results
 - Find and display the structural differences among the workflows
- Small
 - A snippet should be small so that it is easy to browse several results
 - We show maximum g modules

Huang, Liu and Chen (2008) Query biased snippet generation in XML search. SIGMOD 2008.

Summarizing workflow structure

Basis for deciding which parts of a workflow that should be displayed or hidden, i.e. which substructures are important.

Intuition 1: Modules that are present in a workflow but unusual in the collection is distinguishing.

Identified by IDF (Baeza-Yates and Ribeiro-Neto 1999):

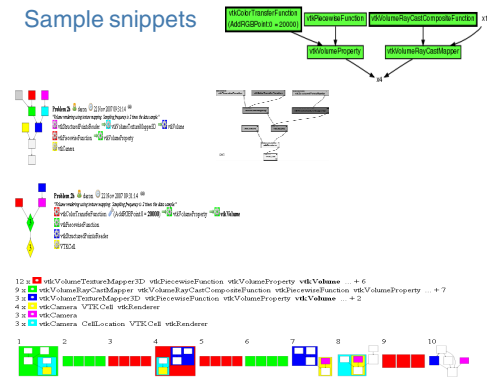
$$idf(m) = \frac{\log N}{N(m)}$$

Intuition 2: Semantic entities can be determined by modules often occurring together.

Can be identified by the Jaccard distance:

$$dist(M_a, M_b) = \frac{P(M_a \cap M_b)}{P(M_a) + P(M_b) - P(M_a \cap M_b)}$$

Sample snippets



Snippet presentation

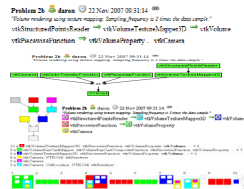
•Independent of selection strategy there are several options for presentation

-Text-based

-Dynamic image

-Legend

-Highlighting differences



Evaluation: Qualitative user feedback

- Query keyword and IDF strategies lead to better snippets.
- Labeling split the structure and content of modules
- Inconvenient with groupings without labels
- Optimal when presented structure similar to the original workflow

Presentation and visualisation

Applications such as:

<http://www.nytimes.com/interactive/2009/06/25/arts/0625-jackson-graphic.html>
<http://www.babynamewizard.com/voyager>
<http://vis.berkeley.edu/papers/sense.us/>
<http://manveyes.alphaworks.ibm.com/manveyes/>
<http://mrtaggy.com/>

Tools such as: <http://www.qlikview.com/>
<http://www.marklogic.com> in particular
<http://www.marklogic.com/product/application-services.html>

Yahoo!: A web of concepts

- Concept: Things of interest to the users of the web.
- Concept represented as:
 - Id
 - meta-data (attributes with values)
- Goals: Concept centric data organization
- What to support:
 - Nested structure?
 - Provenance, versions uncertainty?
 - Relations between concepts?

What do we search for?

Individual concepts: 60-70% Sets of concepts 10-20%

Attributes of a concept:

Rather small correlation (restaurant menu 3%)

Aggregation: 59% of users click on more than one URL.

Concepts vs. Browsing: Follow paths of how user browsed.
Easy to find patterns of what users commonly visit.

Goal:

Create a web of concepts by:

Extracting structured data from documents to find concepts

Create obtained links.

Analyse documents to attach meta-data.

Recent work: Neo

Neo4j is a **graph database**. It is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables.

Linköping related company.

Interesting for semi-structured data.

The Neo Persistence Engine

Primitives:

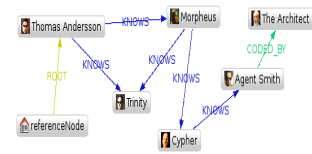
nodes,
relationships
properties

Features

ACID transaction
Durable persistence
Transaction recovery

Implementation

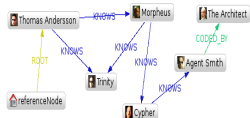
Java



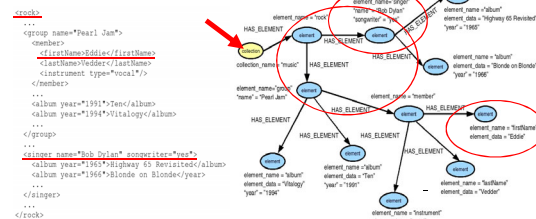
Querying: Traverser framework

order: in which order to traverse
stop evaluator: when to stop
returnable evaluator: which nodes to return
relationship type: which relationship types to follow
direction: in what direction to follow relationships

```
private void printFriends(Node person)
{
    Traverser
    traverser = person.traverse(
        Order.BREADTH_FIRST,
        StopEvaluator.END_OF_NETWORK,
        ReturnableEvaluator.ALL_BUT_START_NODE,
        MyRelationshipTypes.KNOWS,
        Direction.OUTGOING);
    for (Node friend : traverser) {
        System.out.println(friend.getProperty("name"));
    }
}
```



Representing XML in Neo: Basic solution





Representing XML in Neo: Customizations

Nodes:

- Resource type
- Resource

Edges:

- Customized edges

Remove unnecessary attributes

