VŠB–TU Ostrava, FEECS, Department of Computer Science
Czech Technical University in Prague, FEE, Dept. of Computer Science & Eng.
Charles University in Prague, MFF, Department of Software Engineering
Czech Society for Cybernetics and Informatics

Proceedings of the Dateso 2008 Workshop

DATESO 2008

Databases, Texts, Specifications, and Objects

http://www.cs.vsb.cz/dateso/2008/
http://www.ceur-ws.org/Vol-330/

April 16 – 18, 2008
Desná – Černá Říčka

DATESO 2008
© V. Snášel, K. Richta, J. Pokorný, editors

This work is subject to copyright. All rights reserved. Reproduction or publication of this material, even partial, is allowed only with the editors' permission.

Technical editor:
Pavel Moravec, pavel.moravec@vsb.cz

Page count:	84
Impression:	100
Edition:	1^{st}
First published:	2008

This proceedings was typeset by PDFLaTeX.
Cover design by Pavel Moravec (pavel.moravec@vsb.cz) and Tomáš Skopal.
Printed and bound in Ostrava, Czech Republic by TiskServis Jiří Pustina.

Published by VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Preface

DATESO 2008, the international workshop on current trends on Databases, Information Retrieval, Algebraic Specification and Object Oriented Programming, was held on April 16 – 18, 2008 in Desná – Černá Říčka. This was the 8^{th} annual workshop organized by VŠB-Technical University Ostrava, Department of Computer Science, FEL ČVUT Praha, Department of Computer Science and Engineering and MFF UK Praha, Department of Software Engineering. The DATESO aims for strengthening the connection between this various areas of informatics. The proceedings of DATESO 2008 are also available at DATESO Web site `http://www.cs.vsb.cz/dateso/2008/` and shortly after the workshop will appear at `http://www.ceur-ws.org/Vol-330/`.

The Program Committee selected 6 papers from 8 submissions, based on three independent reviews.

We wish to express our sincere thanks to all the authors who submitted papers, the members of the Program Committee, who reviewed them on the basis of originality, technical quality, and presentation. We are also thankful to the Organizing Committee and Amphora Research Group (ARG, `http://www.cs.vsb.cz/arg/`) for preparation of workshop and its proceedings.

March, 2008 V. Snášel, K. Richta, J. Pokorný (Eds.)

Steering Committee

Václav Snášel	VŠB-Technical University of Ostrava, Ostrava
Jaroslav Pokorný	Charles University, Prague
Karel Richta	Czech Technical University, Prague

Program Committee

Václav Snášel (chair)	VŠB-Technical University of Ostrava, Ostrava
Jaroslav Pokorný	Charles University, Prague
Karel Richta	Czech Technical University, Prague
Vojtěch Svátek	University of Economics, Prague
Peter Vojtáš	Charles University, Prague
Tomáš Skopal	Charles University, Prague
Dušan Húsek	Inst. of Computer Science, Academy of Sciences, Prague
Michal Kráký	VŠB-Technical University of Ostrava, Ostrava
Pavel Moravec	VŠB-Technical University of Ostrava, Ostrava
Irena Mlýnková	Charles University, Prague
Michal Valenta	Czech Technical University, Prague

Organizing Committee

Pavel Moravec	VŠB-Technical University of Ostrava
Jan Platoš	VŠB-Technical University of Ostrava
Yveta Geletičová	VŠB-Technical University of Ostrava

Table of Contents

Full Papers

Invited Lectures

Incox – A language for XML Integrity Constraints Description

Kateřina Opočenská, Michal Kopecký

Department of Software Engineering,
Faculty of Mathematics and Physics, Charles University, Prague
katerina.opocenska@matfyz.cz, michal.kopecky@mff.cuni.cz

Abstract. Presently, there is no specialized language for complex integrity constraints description in XML documents. In this paper we present a language that combines first-order logic together with *XPath* language to achieve needed expressive power. Standard quantifiers of first-order logic were extended to allow us to specify (either by count or by percentage) how many elements of the selected set must hold given constraint. The proposed language can be used in conjunction with any XML schema language. The *Incox* validator supports both plain-text and XML variants of constraint specification. While the first one is easily understandable for humans, the latter meets requirements of machine processing.

Keywords: integrity constraints, XML schema language, XML semantics, Incox

1 Introduction

Validation of an XML [1] document can be divided into two main parts – to validation of document structure (syntax validation) and to validation of element content and their correlation (semantics validation).

In the present there are many languages and tools for XML validation. Unfortunately, mostly all of them deal just with the syntax aspects and do not support complex content validation. Usually, only data types of elements and basic referential integrity are checked.

The structure of an XML document can be well described by *DTD* [1] or stronger languages like *XML Schema* [2] or *Relax NG* [3]. The specification of elements and data types of their attributes is also worked out well and so there is no need to do it again or in other way. On the other hand, description of relations among elements and/or attributes content is definitely worth of closer attention. Those relations can be often more complicated than solely uniqueness constraint.
There exist no well-established (e.g. W3C) standards for definition and validation of integrity constraints in XML documents. The only functionally similar (ISO)

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 1–12, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

standard is represented by the *Schematron* [4] language. Despite its unique approach and strength in comparison with above mentioned XML validation languages there still exist categories of constraints that can not be formulated in it. Such constraints typically describe some attachment among elements/attributes content that is not expressible in *XPath* [5]. Nevertheless, validation of such constraints can be useful in many information systems.

In this paper we present the *Incox*[1] language that is primarily designed for complex semantics constraints description in XML documents. We suppose that all processed XML documents were already successfully validated by one of schema language validators. In other words, we assume that processed documents are well-formed and that their structure matches the desired schema.

From this reason we did not design the *Incox* language to substitute the functionality of any of well-known schema languages. It represents just the next step in complex XML document validation.

First, we show some motivation examples of simple constraints that are not possible to easily validate by the classic schema languages. Next, we describe basic aspects of the *Incox* language and demonstrate how it can be used for validation of such constraints.
Persons concerned on the topic can find more examples and further details in [6][2]. Finally, we compare the strength and limits of *Incox* with two most similar languages *Schematron* [4] and *CliX* [7].

2 Motivation

Let us have an XML document describing a book that contains some `<chapter>` elements. Each chapter is identified by a unique value of its numeric attribute. The chapters do not need to be sorted by numbers. We want to check if the book is complete. It means that it contains each chapter from the first one to the chapter with the highest number. More precisely: to the chapter we assume to be the last. Each chapter must be of course present once and only once.

```
<book>
  <chapter no="3">...</chapter>
  <chapter no="4">...</chapter>
  <chapter no="6">...</chapter>
  <chapter no="1">...</chapter>
</book>
```

Common schema languages can define the uniqueness constraint (no more chapters with the same number) but they are not able to test whether some chapter is missing

[1] Integrity Constraints in XML

[2] The referential implementation of the Incox validator written in C# can be obtained from http://www.ms.mff.cuni.cz/~opock4am/bc.html (in Czech language)

or not. In the extreme we can imagine a little bit clumsy solution that combines computing of the total count of all `<chapter>` elements together with detection of the highest chapter number and the uniqueness testing. Anyway, if we generalize this constraint to the requirement of the occurrence of all elements from a given list, no schema language will be able to express it.

Another challenge brings to us a validation of documents in which we do not require the constraint to be held by all specified elements but only by a particular part of them. A simple example: we have an XML document containing a list of persons and we want to check, if there is approximately the same count of men and women. Particularly it means that neither men nor women make less than 45% or more than 55% of all persons registered within the document.

```
<people>
  <person sex="M">...</person>
  <person sex="F">...</person>
  <person sex="M">...</person>
          ...
</people>
```

This example can be further generalized to use of more specific ranges or absolute numbers of elements. For example "*Is there at least 10% of women listed in the document, but not more than 150 women at all?*" etc.

3 Simplified *Incox* language specification

The *Incox* language is based on first-order logic and uses quantifiers as its main expressive means. One constraint corresponds to one logical formula in prenex form. *XPath* 1.0 language is then used for navigation in the XML document and for the selection of tested elements.

The *Incox* language was inspired by *CliX*, but unlike this language the plain-text syntax of *Incox* reminds rather *XQuery* [8] or SQL. The intention is to let the constraints copy sentences of natural language to be easily expressible and writable for a human. In the cases when XML form of constraint specification is more suitable it is possible to use it as well. Detailed description of XML format can be found in [6].

The plain-text file with constraint definitions starts with the optional declaration section followed by a sequence of constraint sections. Constraints are evaluated independently one by one.

3.1 Declaration section

Four types of global declarations for constants, sets of constants, intervals and namespaces can appear in the declaration section in any count and order. Such declarations can be written in the following form:

```
CONST[:] constname = expr
ENUM[:] constname = (expr1, ... , exprN)
INTERVAL[:] constname = (start, end [, step])
NAMESPACE[:] px = "ns"
```

A number (either integer or real), string or *Xpath* [5] expression can be assigned to the identifier of a constant in the CONST declaration. *Incox* built-in conversion functions *str*(), *int*() and *real*() can appear in the *expr* statements – see section 3.2.4 for more details. If an *XPath* expression returns the node-set containing exactly one node, the result can be passed as an argument to the conversion function. The converted value is then assigned to the constant name. If we try to convert a node-set that contains more nodes (or no node), a run-time error is raised.

Examples

```
CONST pi = 3.14
CONST maxChap =
int('//book/chapter[not(../chapter/@no > @no)][1]/@no')
```

By *XPath* we can extract the set of the highest chapter numbers and then select only the first one to cover the case that there are more chapters with the same highest number. We assume there is at least one chapter in the book, so the set is never empty. Evaluated *XPath* expression is then converted by *int*() function and the result number is stored in the constant with identifier *maxChap*.

ENUM and INTERVAL declarations determine the sets whose identifiers can be used in constraints sections whenever a set is expected. If the *XPath* expression returns a node-set then the relevant constant is typed as a set and its identifier can occur in a constraint section at the place of set determination.

Examples

```
ENUM weekDays = ("mo","tu","we","th","fr","sa","su")
INTERVAL chapNums = (1, maxChap, 1)
```

The *chapNums* interval contains integers from 1 (inclusive) to the number that is stored in (previously evaluated) *maxChap* constant. The step of the interval, represented by the third argument, is of length 1. So the interval contains all the integers between the two mentioned.

3.2 Constraint section

The constraint section begins with the keyword CONSTRAINT followed by the name of the constraint in quotation marks. Except from the FORMULA some other optional blocks can occur here. They usually specify how the constraint is evaluated and some details for the output form. In this paper we present just the simplified version:

```
CONSTRAINT "Name of the constraint"
{
     FORMULA[:]
          select1
          …
          selectN
          ( predicate )
}
```

The logical formula after FORMULA keyword comprises of selections in form of FOR { ALL | AT LEAST | AT MOST } or EXISTS [!] quantifiers and the predicate section wrapped in round brackets. These selections and the predicate clause correspond to the logical formula in prenex form.

In selections we define names of (local) variables and determine their domains (sets of allowed values). In the predicate we bind these variables together and declare relations we want to be fulfilled by all (at least one, exactly one, given count, given percentage etc.) elements of the specified set.

3.2.1 Elements selection

The clauses for elements selection copy the use of either existential or universal quantifier in first-order logic. The last one is available also in the extended form.

The *x* variable acquires one–by-one individual values of elements in the specified set. The quantifiers differ only in the rate of how many of the elements in the set must satisfy the predicate to consider the constraint to be fulfilled.

FOR ALL x IN set All the elements in the set *set* must satisfy the predicate.

If the set is empty, the following predicate is always considered as satisfied; hence the constraint is evaluated as *true*.

FOR AT LEAST m [%], AT MOST n [%] x IN set By this clause we can specify more precisely how many elements in the set *set* must satisfy the predicate. It is not necessary to set the both AT LEAST and AT MOST boundaries. The selection can contain only one of them as well. The desired count can be expressed either by the absolute number or by the percentage of the whole set cardinality. It is allowed to combine absolute number and percentage within one selection.

EXISTS [!] x IN set At least one element or exactly one element (exclamation mark) in the set *set* must satisfy the predicate.

If the resulting set is empty, the following predicate is never satisfied, hence the constraint is evaluated as *false*.

3.2.2 Set Specification

For all types of selections a set of items can be defined either by an *XPath* expression or by a constant set declared as ENUM or INTERVAL. If the set is defined by the *XPath* expression, the expression must satisfy the following restrictions:

- The *XPath* expression returns set of elements not a value. For example it is not possible to use *XPath* expression *'count(//num)'*, because a number, not set, would be returned.
- Unless stated differently the root of the document is considered to be an implicit context.
- Referenced variable *var* from previous selection can be used in the *XPath* expression in form *$var*. If used, this variable must have already set its value (see example 4.3).
- The expression contains at most one referenced variable for context specification.

Any *XPath* expression used in the place of function parameter in the predicate section must fulfill all above mentioned restrictions except the first one. Such expressions can use *XPath* functions (version 1.0) and can return also numbers, Boolean values and strings.

3.2.3 Predicate

Each predicate is written in the form

$$(\texttt{boolval}_1\ \texttt{logop}\ \texttt{boolval}_2\ \texttt{logop}\ \ldots\ \texttt{logop}\ \texttt{boolval}_N)$$

Allowed logical operators are either OR or AND respectively operator `->` that represents a logical implication. The result of the predicate evaluation is a Boolean value. Individual operands can represent results of comparison of comparable expressions or values computed by some function.

If there is no function applied on the selection variable then it is considered to represent a node – a specified place in the document. If the selection variable is used as a parameter of some conversion function proposed in *Incox* language as *str*(), *int*() or *real*() – see section 3.2.4 – the validator tries to interpret the value of given XML node as the appropriate type.

The string value of the node is defined as a concatenation (in order of sequential reading) of all textual content of the node. For example the value of the node `<a><num>1</num><num>2</num></a>` is *'12'*. If the resulting type

of the used function differs from string, the value is further converted to appropriate type.

Because the variable contains at each time some element from given set, there can not arise the problem originating from conversion of set to value. Errors can still arise from unsuccessful conversion of string value to other type, i.e. number or Boolean.

3.2.4 Basic auxiliary functions

To allow adequate constraint validation, the *Incox* language introduces following auxiliary functions. Their parameters and return data types are written in C syntax to increase the comprehensibility. Data types are written in italics.

bool not (*bool* b)	Function negates any condition that can be evaluated as Boolean value.
string str (*expr* expr) *int* int (*expr* expr) *float* real (*expr* expr)	Listed functions convert given expression to string, integer, respectively float value. The expression can be either a constant name, variable, string, number or *XPath* expression. If the *XPath* returns set of values, this set must have exactly one element. In this case the result contains conversion of this element. In other cases the run-time error is raised.
int length (*string* s)	This function returns the length of given string.
string tolower (*string* s)	This function converts all upper case letters in the string to lower case.
string toupper (*string* s)	This function converts all lower case letters in the string to upper case.
string trim (*string* s)	It trims all white space characters from the beginning and the end of given string.
string trimall (*string* s)	It removes all white space characters from the given string.
bool match (*string* s, *string* regexp)	This function returns the information if the given string *s* matches to given regular expression.

4 Implementation of Examples

Having the formal apparatus, we can show the implementation of examples mentioned at the beginning of the paper.

4.1 Chapters in the book

```
CONST  maxChap =
int('/book/chapter[not(../chapter/@no > @no)][1]/@no')

INTERVAL chapNums = (1, maxChap, 1)

CONSTRAINT "Chapters in the book"
{
  FORMULA:
  FOR ALL chap IN chapNums
    EXISTS ! rec IN '/book/chapter'
        ( int('$rec/@no') = chap )
}
```

First, we store the highest number of the chapter found in the document in constant *maxChap*. Then we declare an interval *chapNums* containing all numbers from 1 to this maximal chapter number.

In the formula inside constraint *"Chapters in the book"* we go through all possible chapter numbers and check if there exists exactly one chapter *'/book/chapter'* whose attribute *no* converted to integer is equal to required value *chap* in the XML document.

Let suppose we will check the constraint against XML document shown in section 2. The highest number of the chapter is equal to six, but chapters number two and five are missing. The result of the referential implementation *Icval* (Integrity constraints validator) [6] invoked with options -c (counts) –f (fuzzy truth) –v (verbose) is displayed in the first column. If the option –x (XML) is added then the output is provided in XML format as it is shown in the second column.

The output informs us that corresponding elements were not found for two chapter numbers (two and five). I.e. the condition "*For each (chapter) number from one to six exists exactly one matching element*" is fulfilled for 66.7% of chapter numbers only.

Plain-text output:

```
CONSTRAINT: "Chapters in the book"
-------------------------------------------------
OVERALL RESULT : FALSE
Conversion errors resolved as INVALID
True/All for quantifier FOR ALL : 4/6
Fuzzy truth: 0,667
-------------------------------------------------
```

XML output:

```
<icval>
  <constraints>
    <constraint>
      <name>Chapters in the book</name>
      <overall_result>0</overall_result>
      <additional_info>
        <first_quantifier>FOR ALL
        </first_quantifier>
        <true_count>4</true_count>
        <all_count>6</all_count>
        <fuzzy_truth>0,667</fuzzy_truth>
      </additional_info>
    </constraint>
  </constraints>
</icval>
```

4.2 Approximately same number of men and women

The condition that checks if 45% to 55 % of persons registered in the document are men (women) can be written in form:

```
CONSTRAINT "Almost the same count"
{
  FORMULA:
  FOR AT LEAST 45%, AT MOST 55%
    x IN '/people/person/@sex'
        ( str(x) = "M" )
}
```

4.3 Referenced variable

The following example checks the document for fulfilling the condition "*There is exactly one employee having function 'boss' in each department*".

```
CONSTRAINT "One boss in each department"
{
  FORMULA:
  FOR ALL dep IN '//department'
    EXISTS ! emp IN '$dep/emplyoee'
        ( string('$emp/position') = "boss" )
}
```

This example shows the usage of the current node value for evaluation of nested conditions. In time of evaluation of expression *'$dep/emplyoee'* the value of variable *dep* is already set to particular node `<department>`. The expression

$dep/emplyoee' then selects element(s) `<employee>` belonging to the sub-tree specified by this node.

5 Comparison of *Incox* with Similar Languages

5.1 *Schematron* and *Incox*

In *Schematron* [4], the validation of each condition consists of three steps:

1. Selection of required set of nodes, specified by given *XPath* expression (the *context* attribute of the *rule* element)
2. Verification of the truthfulness of others *XPath* expressions (*test* attributes of the *report*/*assert* elements) in the context of selected node.
3. Output of given text. If the condition is met then the element *report* is written out. Else the output is defined by the *assert* element.

```
<pattern name="name">
    <rule context="context">
         <report test="test">
             Passed.
         </report>
    </rule>
</pattern>
```

Each condition defined in *Schematron* says: all nodes selected by the *XPath* expression fulfills the condition defined by the attribute *test* of the *report*/*assert* element. Thus, the condition in *Schematron* has fixed structure and the expression power of the language is based mainly on *XPath*.

Anyway, it is sufficient in most cases. *XPath* expressions can reference to the whole document and so elements and attributes from different parts of the document can be associated in the condition. It is possible to formulate lot of conditions even those that seems to be quite complicated as for example validation of heaps, search trees or consistency of insurance numbers (records can repeat inside the document, but whenever two persons have the same insurance number, they have to have also the same name).

Nevertheless, there exist complex constraints that are unfeasible or even impossible to express in *Schematron*. Typically complex semantic constraints used in business applications, where the *Schematron*'s author recommends using rather *CliX* [7] or *OASIS CAM* `[9].`
Among indefinable constraints belong those in the form „*Each element A has (at least) one sub-element B such that all its sub-elements C satisfy the condition P*". In this case the corresponding logical formula is too complex and it is not possible to write it down in *XPath*. In contrary it is not problem to write such a constraint in *Incox*.

```
CONSTRAINT "Constraint schema"
{
  FORMULA:
  FOR ALL a in '//a'
    EXISTS b in '$a/b'
      FOR ALL c IN '$b/c'
          ( constraint_p($c) )
}
```

5.2 *CliX* and *Incox*

The *Incox* language describes conditions similarly to *CliX* [7] and so it has at least the same expression power. In comparison with its competitor the *Incox* language offers further extensions that increase its power and simplify its usage.

Constants. Above mentioned constraint that checks for missing chapters is not expressible in *CliX*. The set of chapter numbers – set of integer numbers from one to *maxChap* – can not be defined in this language. In contrary to *Incox* the *CliX* can describe sets only by *XPath* expressions.

Constants can be used not only for higher effectiveness (we have not to select the same data from the document repeatedly), but together with ENUM and INTERVAL constructs also for validating conditions in form *„For each of defined values exists element / given number of elements that …”*. It is useful mainly in situations where the set is not defined somewhere in the XML document and/or the evaluation of needed expression would be impractical.

Extended quantifiers. The usage of extended quantifiers FOR AT LEAST, AT MOST allows us to validate data while tolerating some exceptions (a fraction of elements can fail to satisfy the condition). Thanks to the definable boundaries inside the quantifier we have the amount of abnormal elements under our control.

Built-in functions. Thanks to implemented built-in function for data type conversion and manipulation with strings we can easily express lot of quite complex conditions. For example, function *match*() compares given node value against given regular expression. That can often replace necessity to define complex data types. Following constraint tests if the value of all selected elements corresponds to a roman number.

```
CONSTRAINT "Roman numbers"
{
  FORMULA:
  FOR ALL r IN '//romnum'
  ( match( trim(str(r)),
 "^m*(d?c{0,3}|c[dm])(l?x{0,3}|x[lc])(v?i{0,3}|i[vx])$"
    )
  )
}
```

6 Conclusion

The *Incox* language represents simple yet powerful language for XML constraint validation that outperforms their current competitors *Schematron* and *CliX*. Its extended quantifiers can easily validate exact requirements as well as requirements allowing certain level of incorrectness. This feature together with the possibility to define constants, sets and intervals allows us to formulate and validate more complex constraints than existing languages.

The *Incox* language recognizes the constraint definition in two forms. The textual one is easily readable for human beings while the XML format is easily treatable by the computers. The same approach was chosen in case of output. The *Incox* validator can generate either plain text output or XML output that can be further processed by XML enabled programs and scripts.

Hence we believe that this language represents the way towards the future of XML content validation.

References

1. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. Extensible markup language (XML) 1.0 (third edition), W3C. February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/.
2. D. C. Fallside, P. Walmsley. XML Schema Part 0: Primer Second Edition, W3C. October 2004. http://www.w3.org/XML/Schema.
3. J. Clark, M. Murata. RELAX NG Specification, OASIS Committee Specification, December 2001. http://relaxng.org/spec-20011203.html.
4. International Organization for Standardization. Information Technology Document Schema Definition Languages (DSDL) Part 3: Rule-based Validation Schematron, ISO/IEC 19757-3. February 2005. http://www.schematron.com.
5. J. Clark, S. DeRose. XML Path Language (XPath) Version 1.0., W3C. November 1999. http://www.w3.org/TR/xpath.
6. K. Opočenská. Integrity Constraints in XML (bachelor thesis). MFF UK, Prague, September 2007. http://www.ms.mff.cuni.cz/~opock4am/incox.pdf.
7. M. Marconi, C. Nentwich. CLiX Language Specification Version 1.0. January 2004. http://www.clixml.org/clix/1.0/.
8. S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, J. Simeon. XQuery 1.0: An XML Query Language, W3C. January 2007. http://www.w3.org/TR/xquery/.
9. D. Webber, J. B. Clark. OASIS Content Assembly Mechanism Specification Version 1.1. February 2007. http://www.oasis-open.org/committees/cam/.

Conceptual Model Based Normalization of XML Views*

Martin Nečaský

Department of Software Engineering, Faculty of Mathematics and Physics,
Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic
martin.necasky@mff.cuni.cz

Abstract. As the popularity of XML as a format for data representation grows the need for storing XML data in an effective way grows as well. Recent research has provide us with effective solutions based on storing XML data into relational databases and with new technologies based on storing XML data in the native form. However, design of XML databases has not been studied sufficiently yet. In this paper, we suppose a set of XML schemes that describe XML representation of our data in several types of XML documents. We show that we can not usually store the data directly in this representation because it can contain redundancies. To design an optimal database schema we therefore need to locate these redundancies and eliminate them. We describe two types of redundancies in XML data in this paper and show how to utilize a conceptual schema of the XML schemes to locate such redundancies. We also show how to normalize the XML schemes to eliminate these redundancies.

Keywords: conceptual modeling, XML schema, normalization

1 Introduction

XML has become a popular format for data representation. Mainly it is because it is a variable format that is easy-to-use for a broad range of developers. Enterprises usually utilize several applications supporting different users for performing different business processes. Even though these applications share the same data (about customers, products, etc.), each of them requires the processed data to be represented in different forms suitable for the purposes of the application. XML proved itself as a suitable format for such various representations. For example, a sales reporting application for product managers represents customer's data in another type of XML documents than a web service for receiving and processing purchase orders from customers.

We need to store the data shared by the applications into a database and provide each application with the data represented in the required type or types

* This research was supported by the National programme of research (Information society project 1ET100300419) and by Grant Agency of Charles University (GAUK), grant number 204-10/257190

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 13–24, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

of XML documents. We therefore comprehend these types of XML documents as *XML views* on the data stored in the database. These XML views are described by XML schemes. Given a set of such XML schemes the problem is how to design an optimal schema of the shared database. Even though we can use a native XML database to store the data in an XML representation, we can not usually store it directly as represented by the XML views. It is because the XML views can contain redundancies which means that the same data can be duplicated. Such a duplication means not only inefficient storage space usage but also problems when manipulating the data. We therefore need to identify these redundancies and eliminate them. For example, we can have an XML schema of an XML view for purchase orders where data about one product can be repeated in different purchase orders. This is a redundancy that should be eliminated. After the identification and elimination of all redundancies we get a set of normalized XML schemes. The situation is demonstrated in Figure 1. The idea is same as in the case of designing a relational database schema where we eliminate redundancies by modifying our schemes to meet so called *normal forms* such as 2NF, 3NF, or 4NF. This process is called *normalization*.

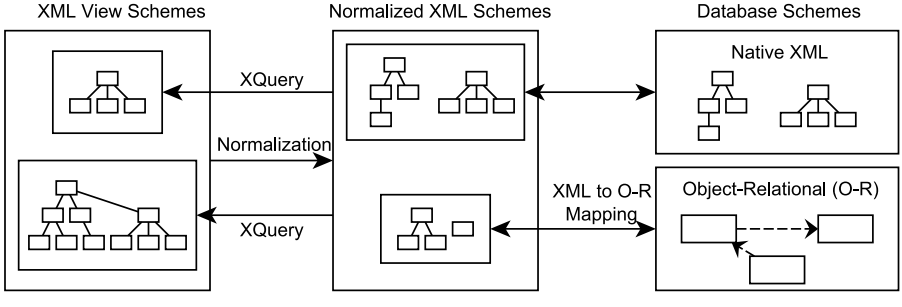


Fig. 1. Architecture

We can use the normalized XML schemes to design a schema of the database where we store the data shared by the applications. If our database is a native XML database (NXDB) we can directly use these normalized XML schemes as a NXDB schema. If our database is an object-relational database (ORDB) or a combination of ORDB and NXDB the normalized XML schemes are a good starting point to design the internal database schema. For example, we can map the normalized XML schemes into an ORDB schema. We can also combine both approaches and map structured parts of normalized XML schemes into an ORDB schema and use their unstructured parts or parts with a complex hierarchical structure as a schema of a NXDB. This situation is demonstrated in Figure 1.

In these cases, the database provides us the data in the form of XML documents with the structure given by the normalized XML schemes. However, we need to deliver the data to our applications in the form of the XML views. There-

fore, we moreover need a set of XQuery queries that transform the data between the normalized XML schemes representation and XML views. These queries can be derived automatically during the normalization process.

In this paper, we study normalization of a set of XML schemes as demonstrated in Figure 1. The other parts of the architecture displayed in the figure are out of the scope of this paper. Methods for mapping between XML and object-relational representations were studied for example in [5]. Derivation of the XQuery queries to reconstruct XML views is the subject of our further research. To normalize XML schemes we can apply the normal forms for relational data. Even though these normal forms should be considered when normalizing XML schemes we do not discuss them in this paper. We are interested in redundancies caused by hierarchical structure of XML schemes.

Related work. Several types of redundancies caused by hierarchical structure of XML schemes were also studied by other authors such as [1], [4]. Their results are based on functional dependencies in XML documents. In [3], authors show how to normalize XML schemes modeled in a richer model for XML data called ORA-SS. ORA-SS model is more a conceptual model then a logical XML model allowing to specify several integrity constraints for XML data. Authors show how to normalize XML schemes modeled as ORA-SS schemes using cardinality constraints. The advantage of this approach is that it is easier for the designer to specify cardinality constraints then discover functional dependencies in the hierarchical structure of the XML schema.

These approaches lead to good results when normalizing one XML schema. However, we need to normalize a set of XML schemes that can moreover represent the same data in different hierarchical structures. Discovering functional dependencies in such a set of XML schemes can be hard for the designer because he can be required to specify the same functional dependencies repeatedly for different XML schemes representing the same concept. Moreover, a concept represented in more different XML schemes also leads to redundancies as we show later in this paper. Such redundancies can not be identified and eliminated on the base of functional dependencies. Similar problems occur when we model XML schemes as ORA-SS schemes because each XML schema is modeled separately.

Contributions. In this paper we show how to normalize a set of XML schemes modeled at the conceptual level using a conceptual model for XML data called XSEM [6]. The advantage of this model is that the designer designs an overall non-hierarchical conceptual schema of the domain and derives the XML schemes of the required XML views from this overall conceptual schema. This allows to identify concepts that are represented in different XML views. Moreover, the designer is not required to specify functional dependencies or cardinality constraints repeatedly for different XML schemes. Instead, normalization is based on cardinality constraints specified in the overall conceptual schema where each cardinality constraint is specified only once.

Our approach can be applied in the systems where a large number of different XML views occurs and one or more databases to store the data in an effective way exits. In real systems, it is usually not required to fully normalize data

[2]. Following this requirement, it is not necessary to apply our approach to normalize the XML data fully to achieve better performance when reconstructing the views. The design of the normalized database schema is strongly influenced by the expected usage of data.

The paper is organized as follows. Section 1 is an introduction to the paper. In Section 2 we describe the XSEM model briefly. In Section 3 we describe two types of redundancies in XML data and we show how to eliminate them using the information from a conceptual schema. We conclude in Section 4.

2 XSEM Model

XSEM divides the conceptual modeling process to two levels. At the first level, we design an overall non-hierarchical conceptual schema of our domain using a part of XSEM called *XSEM-ER*. At the second level, we design hierarchical schemes as views on the XSEM-ER schema using a part of XSEM called *XSEM-H*. Each XSEM-H view schema describes an XML schema at the conceptual level. We briefly describe both parts of XSEM in this section. For a full and formal description of XSEM see [6].

2.1 XSEM-ER

XSEM-ER builds on an extension of the classical E-R model called HERM [8]. It allows to model real-world objects and relationships between them with entity types and relationship types and provides designers with extending constructs for modeling special XML features like irregular structure, ordering, and mixed content. In XSEM-ER, it is not important how the modeled data is organized in hierarchical XML documents. We show an example XSEM-ER schema modeling a small part of a business domain in Figure 2.

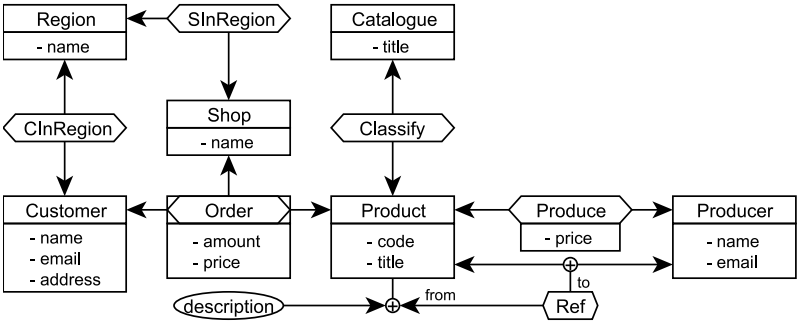


Fig. 2. XSEM-ER Schema for Business Company

The basic modeling constructs are *strong* and *weak entity type*, and *relationship type*. These constructs are known from the classical E-R model. Figure

2 shows strong entity types such as *Customer* and relationship types such as *Produce* with participants *Product* and *Producer*. It also shows weak entity types such as *Order* with determinants *Customer*, *Shop*, and *Product*.

There are two types of extending constructs. *Data node types* are used for modeling unstructured text parts of the data that can be mixed with structured parts. They are similar to attributes of entity or relationship types. However, they are not encapsulated directly in entity or relationship types but only assigned to them and grouped with another concepts in the schema. Data node types are displayed as ellipses. Figure 2 shows a data node type *description*. It models descriptions of products. We need to model that a description of a product can be mixed with references to other products and to producers. Therefore, we do not model description as an attribute of the entity type *Product* but as a data node type.

Cluster types are used for grouping different entity, relationship, and data node types. They are used to model irregular or mixed content at the conceptual level. We display cluster types as circles with inner +. There are *component* and *connection* cluster types. We use component cluster types for creating groups of two or more entity types. Such a group can then be assigned as a participant to a relationship type or determinant to a weak entity type. For example, there is a component cluster type composed of *Producer* and *Product*. This cluster type is assigned as a participant to a relationship type Ref. It models that references from products to other products and also producers.

Connection cluster types are used for creating groups of two or more concepts having the same entity type as a common participant or determinant. If there is a data node type in this group it models structured data mixed with unstructured data. For example, there is a connection cluster type composed of the data node type *description* and the relationship type Ref. It models that a description of a product is mixed with references to other products and producers.

An XSEM-ER schema does not specify how the data is organized in hierarchical XML documents. There is one or more possible hierarchical representations of each component of the schema. For example, we can represent instances of the weak entity type *Order* in the hierarchy where we have a list of orders and for each order we have the respective customer who made the order, ordered product, and shop where the order was made. We can also require another representation described as follows. We want a list of shops. For each shop we want a list of products ordered in the shop. For each such product we want a list of orders of the product made in the shop. Finally, for each such order we want the customer who made the order.

We need to describe such a hierarchical structure in a more formal way. For this we propose so called *hierarchical projections*. As an example, we show the following six hierarchical projections. The projections $(H1)$, $(H2)$, and $(H3)$ describe the former hierarchical representation of *Order*. The projections $(H4)$, $(H5)$, and $(H6)$ describe the other.

$$Order[Order \rightarrow Customer] \quad (H1)$$
$$Order[Order \rightarrow Product] \quad (H2)$$
$$Order[Order \rightarrow Shop] \quad (H3)$$
$$Order[Shop \rightarrow Product] \quad (H4)$$
$$Order^{Shop}[Product \rightarrow Order] \quad (H5)$$
$$Order^{Shop,Product}[Order \rightarrow Customer] \quad (H6)$$

Formally, a hierarchical projection h of an entity or relationship type T is an expression $T^{E^1,\ldots,E^k}[P \rightarrow Q]$ where $E^1, \ldots, E^k, P, Q$ are determinants or participants, respectively, of T. It specifies a hierarchy where P (called *parent*) is superior to Q (called *child*). The sequence $E^1, \ldots, E^k$ is called *context* and specifies the context in which the projection is considered. For example, $(H6)$ specifies a hierarchy where $Order$ is superior to $Customer$ in the context of $Shop$ and $Product$.

We also extend the notion of *cardinality constraints* for hierarchical projections. For the hierarchical projection h of T, we can specify a cardinality constraint for the parent or child, i.e. $card(h, P) = (m, n)$ or $card(h, Q) = (m, n)$, respectively. It means that for instances of the components from the context of h an instance of P (or Q, respectively) can appear in T with m up to n different instances of Q (or P, respectively). For example, a cardinality constraint $card(H6, Customer) = (0, *)$ specifies that for a given shop and product a customer can make an arbitrary number of orders of the product in the shop. A cardinality constraint $card(H6, Customer) = (0, 1)$ specifies that for a given shop and product a customer can make zero or one order of the product in the shop but not more.

2.2 XSEM-H

An XSEM-H schema models one type of XML documents. It is a view on a part of the XSEM-ER schema and specifies how the data described by this part of the XSEM-ER schema is represented in the modeled type of XML documents. It does not describe any further semantics of the data. We can derive several XSEM-H view schemes from the same part of the XSEM-ER schema. Therefore, they are not derived automatically but by the designer according to the required structure of the XML documents. Figure 3 shows three XSEM-H view schemes. For example, $CatalogueView$ describes the structure of XML documents with catalogue data.

An XSEM-H view schema is a set of trees with labeled oriented edges. Nodes in the view schema represent entity types, relationship types, and data node types. For clearness, we denote the nodes by $U_{T,n}$ where T is the type represented by the node and n is the counter for the nodes in the view schema representing T. For example, $OrderView$ contains a node U_{Order} representing the weak entity type $Order$ from the XSEM-ER schema. Edges in XSEM-H view schemes represent hierarchical projections of the types represented by the nodes. For example, the edge going from U_{Order} to $U_{Customer}$ in $OrderView$ represents the hierarchical projection $H1$, i.e. $Order[Order \rightarrow Customer]$. Nodes can have assigned labels displayed above the nodes. These labels are names of elements that are used to represent the nodes in the modeled type of XML documents. For example, U_{Order} in $OrderView$ has assigned a label *order*. It means that orders are represented in the modeled type of XML documents by elements `order`.

There are also several types of auxiliary nodes in XSEM-H view schemes. There are *cluster nodes* representing cluster types from the XSEM-ER schema.

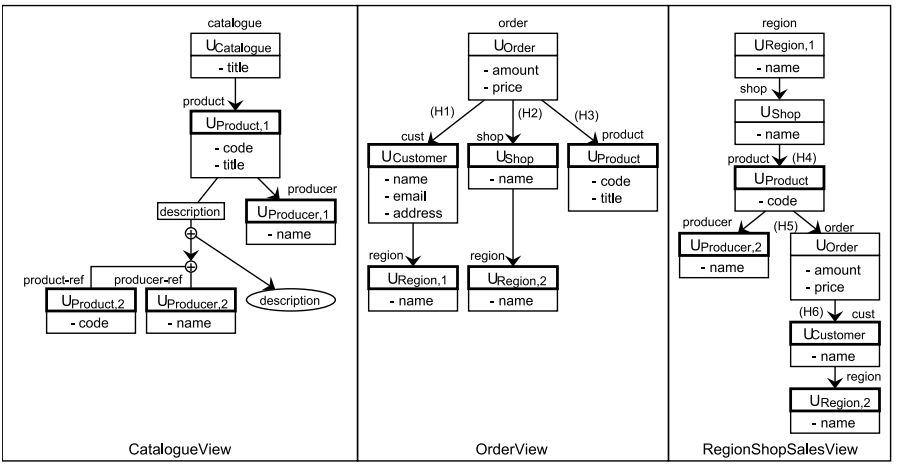


Fig. 3. XSEM-H view schemes for business company

They are displayed in the same way, i.e. as circles with an inner '+' symbol. Further there are so called *containers* that represent XML elements that group two or more different concepts but not have any equivalent at the conceptual level in the XSEM-ER schema. A container is displayed as a narrow rectangle with its name in the rectangle. For example, Figure 3 shows a container *description* assigned to the node $U_{Product,1}$. For a more detailed description of modeling constructs of XSEM-H, we refer to [6].

Each XSEM-H view schema models an XML schema at the conceptual level. This XML schema can be derived from the XSEM-H view schema automatically represented in a selected XML schema language. The derivation is straightforward. However, we do not discuss it in this paper because of the lack of the space.

3 Normalization

A set of XSEM-H view schemes can lead to redundancies when we represent our data in the respective types of XML documents. Normalization means to transform this set of XSEM-H view schemes to another set of XSEM-H view schemes that describe the same data but do not lead to redundancies. In this section, we show two types of redundancies and how to normalize XSEM view schemes that lead to such redundancies. Our goal is to modify the structure of the XSEM-H view schemes as little as possible during the normalization. We call the normalized XSEM-H view schemes *XSEM-H repository schemes* to distinguish them from the original ones.

Local redundancies. The first type of redundancies is caused by hierarchical projections with the maximal cardinality of their child greater then 1. In such case,

an instance of the child can be assigned to more different instances of the parent and therefore repeated in the respective hierarchical structure. Assume for example $SalesView$ in Figure 3. There is an edge going from $U_{Region,1}$ to U_{Shop} that represents a hierarchical projection $SInReg[Region \rightarrow Shop]$. The cardinality of $Shop$ in the projection is $(1, 1)$. Therefore, an instance of $Shop$ is assigned to one and only one instance of $Region$ and is therefore not repeated in the respective hierarchical structure. On the other hand, the edge going from U_{Shop} to $U_{Product}$ represents a hierarchical projection $Order[Shop \rightarrow Product]$ and the cardinality of $Product$ in this projection is $(0, *)$. It means that an instance of $Product$ can be repeated in zero or more instances of $Shop$.

On the base of this observation we define the first type of redundancies called *local redundancies*.

Definition 1. *Let U be a non-root node in an XSEM-H view schema. Let U represent a type P. Let e be an edge going to U and representing a hierarchical projection $T^{T_1,\ldots,T_{k-1}}[T_k \rightarrow P]$. Let the maximal cardinality of P in the hierarchical projection be greater than* 1. *If U represents one or more attributes of T or there is an edge going from U and representing a hierarchical projection with an empty context then we say that U* leads to local redundancies.

Assume that a node U, that represents a type P, leads to local redundancies. To eliminate these redundancies we normalize U by dividing it to two parts. The first part is called *context-independent* part of U and is composed of the attributes represented by U and edges going from U and representing hierarchical projections with an empty context. The second part is called *context-dependent* part of U and is composed of the edges going from U and representing hierarchical projections with a non-empty context. If an instance p of P is repeated at the location specified by U its content corresponding to the context-independent part of U is repeated as well. The content of p corresponding to the context-dependent part of U is different for each representation of p because it depends on the context. The normalization of U means to move its context-independent part to another node V that represents P as well but does not lead to local redundancies. The instance p of P is repeated at the location specified by V only once and its content corresponding to the context-independent part of U is therefore repeated only once as well. If such a node V does not exist we create a new XSEM-H repository schema and create V as its root node. The created node does not lead local redundancies because it is a root node. We call V *storage node* for P.

To reconstruct the original view we must be able to join U with its context-independent part moved to the storage node V. Joins are usually performed using keys and foreign keys. However, we did not show how to model keys in XSEM-ER schemes in this paper. We discussed this problem in [7]. The proposed keys can be used for modeling general XML keys that can be rather complex. This type of general keys is not however suitable for our purposes in this paper. Instead, we use a much simpler mechanism of artificial keys. We add an artificial key attribute *oid* to V and foreign key *oid* to U referencing *oid* in V. When reconstructing the original U, we join the normalized U with V using this pair.

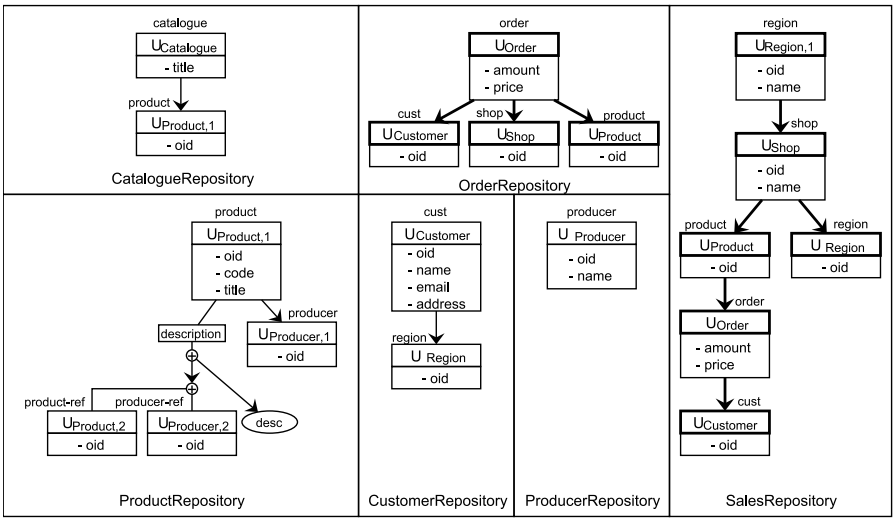


Fig. 4. XSEM-H repository schemes without nodes leading to local redundancies

Figure 3 shows nodes that lead to local redundancies in a bold line. Figure 4 shows the result of their elimination. For example, the node $U_{Product,1}$ in $CatalogueView$ leads to local redundancies. It is because the edge going to the node represents a hierarchical projection $Classify[Category \rightarrow Product]$ and the maximal cardinality of $Product$ in this projection is $*$. Normalization of the node means to move its context-independent content to the storage node for $Product$. However, all nodes in the XSEM-H view schemes that represent $Product$ lead to local redundancies and the storage node for $Product$ must be created. We therefore create a new XSEM-H repository schema $ProductRepository$ with a root node $U_{Product,1}$ representing $Product$. This node is a new storage node for $Product$. We move to this node all the attributes represented by $U_{Product,1}$ in $CatalogueView$ and all the edges representing hierarchical projections without a context, including clusters of edges and containers that contain these edges. Moreover, we add an artificial key attribute oid to $U_{Product,1}$ in $ProductRepository$ and foreign key attribute oid to $U_{Product,1}$ in $Catalogue-Repository$. The result of the normalization is that we store $Product$ instances according to $U_{Product,1}$ in $ProductRepository$. At the location specified by $U_{Product,1}$ in $CatalogueRepository$ we do not repeat whole $Product$ instances but only their values of the artificial foreign key oid. To reconstruct the original view we use this foreign key.

The other nodes representing $Product$ in the XSEM-H view schemes lead to local redundancies as well and are therefore normalized in the same way. We move the context-independent contents of these nodes to the previously created storage node $U_{Product,1}$ in $ProductRepository$. For the node $U_{Product}$ in $OrderView$ we move all its attributes. For the node $U_{Product}$ in $SalesView$

we move all its attributes and the edge going to $U_{Producer,2}$. The edge going to U_{Order} is in the context-dependent part of $U_{Product}$ and is therefore not moved.

Assume further U_{Shop} in $OrderView$ that also leads to local redundancies. To normalize it we do not need to create a storage node for *Shop* as in the previous case with *Product*. There is the node U_{Shop} in $SalesView$ that does not lead to local redundancies and each *Shop* instance is represented at this location. We can therefore use it as the storage node for *Shop* and we move here the context-independent content of U_{Shop} in $OrderView$.

Structural redundancies. The second type of redundancies in XSEM-H view schemes we discuss in this paper is caused by representing an entity or relationship type P at two or more different locations in XSEM-H view schemes. In such a case an instance of P can be repeated at two different locations in the respective XML representations. Assume for example the weak entity type *Order*. It is represented in $SalesView$ and $CustomerView$ as well. After the elimination of local redundancies we still have *Order* represented in two XSEM-H repository schemes $SalesRepository$ and $CustomerRepository$. It means that we represent an instance of *Order* twice in the respective XML representations. Once according to the former repository and once according to the other. We call this type of redundancy *structural redundancy*.

Definition 2. *We say that an entity or relationship type* leads to a structural redundancies *if it is represented at two or more different locations in XSEM-H view schemes.*

Assume that an entity or relationship type P leads to structural redundancies. To eliminate these redundancies we select one of its representations as *primary* and the others as *secondary*. We will use the primary representation for representing instances of P and the secondary representations will be reconstructed by XQuery queries. The selection of the primary representation is made by the designer. He can decide on the base of the usage of the representations. The most used representation should be selected as the primary one. The reader could argue that some more explicit guidelines to select the primary representation should be given. These guidelines could be based on statistics of the usage of the original views combined with the price of the reconstruction of the secondary representations. However, these guidelines overcome the scope of this paper.

Figure 4 shows XSEM-H repository schemes where nodes leading to local redundancies were normalized. However, there are several nodes that lead to structural redundancies and we need to normalize them. Figure 5 shows the resulting set of XSEM-H repository schemes after their normalization. Firstly, the relationship type $SInRegion$ is represented in $ShopRepository$ twice. The former representation is composed of the nodes $U_{Region,1}$ and U_{Shop} and the edge connecting them. The other representation is composed of U_{Shop} and $U_{Region,2}$ and the edge connecting them. It means that $SInRegion$ leads to structural redundancies. In other words each $SInRegion$ instance is represented in two different locations. To eliminate this structural redundancies we must select one

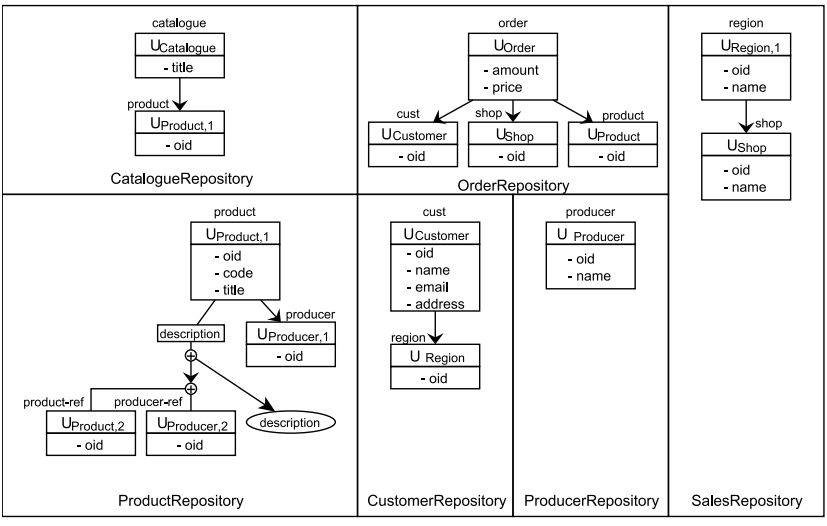


Fig. 5. XSEM-H repository schemes without nodes leading to local nor structural redundancies

of the representation as primary. We select the former representation as primary because it will be used more frequently then the other and its reconstruction would be therefore more expensive. The other representation is secondary and therefore not included in the resulting repository.

Another structural redundancy is the weak entity type *Order*. It is represented once in *OrderRepository* and once in *SalesRepository*. In both repositories the representation is composed of nodes U_{Order}, $U_{Customer}$, U_{Shop}, and $U_{Product}$. We select the representation in *OrderRepository* as primary. The other representation is not included in the resulting repository. The removed secondary representations of *SInRegion* and *Order* are not represented in the normalized XSEM-H repository schemes and must be therefore reconstructed from them by XQuery queries. These queries can be derived automatically. However, such derivation is out of the scope of this paper.

4 Conclusion

In this paper we showed how to model a set of XML schemes at the conceptual level using a conceptual model for XML data called XSEM. This model allows to model data at two levels. At the first level, an overall conceptual schema of the data is designed using a part of XSEM called XSEM-ER. At the second level, a conceptual schema modeling a given XML schema is derived from the XSEM-ER schema using a part of XSEM called XSEM-H. We further showed how to normalize a given set of XML schemes modeled by XSEM-H schemes. We described two types of redundancies caused by hierarchical structure of the XML

schemes, namely local and structural redundancies and showed how to eliminate these redundancies by normalization of the XSEM-H schemes. We also showed that these normalized XML schemes can be used to design a database schema suitable to store our data without redundancies.

References

1. Arenas, M., Libkin, L.: A Normal Form for XML Documents, *in* ACM Transactions on Database Systems (TODS), 29 (2004), pp. 195-232.
2. Balmin, A., Papakonstantinou, Y.: Storing and Querying XML Data Using Denormalized Relational Databases, *in* The VLDB Journal, 14(1), pp. 30-49, 2005.
3. Dobbie, G., Xiaoying, W., Ling, T.W., Lee, M.L.: ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. TR21/00, Department of Computer Science, National University of Singapore. December 2000.
4. Lee, M. L., Ling, T. W., Low, W. L.: Designing Functional Dependencies for XML, *in* Proceedings of the 8th Conference on Extending Database Technology (EDBT), Prague, March 2002, pp. 124-141.
5. Mlynkova, I., Pokorny, J.: XML in the World of (Object-)Relational Database Systems. *in* Proceedings of the 13th International Conference on Information Systems Development, Vilnius, Lithuania. Springer Science+Business Media, Inc., 2005. pp. 63-76,
6. Necasky, M.: XSEM - A Conceptual Model for XML. *in* Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling (APCCM2007), Ballarat, Australia. CRPIT 67. 2007. pp. 37-48.
7. Necasky, M., Pokorny, J. Extending E-R for Modelling XML Keys. *in* Proceedings of the 2nd International Conference on Digital Information Management. IEEE Computer Society. Lyon, France, 2007, pp. 236-241.
8. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer Verlag, Berlin, Germany. 2000.

Using taDOM Locking Protocol in a Functional XML Update Language

Pavel Strnad and Pavel Loupal

Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University
Karlovo náměstí 13, 121 35 Praha 2, Czech Republic
strnap1@fel.cvut.cz, loupalp@fel.cvut.cz

Abstract. In this paper we deal with a particular type of database systems – native XML database systems. For this category of systems we discuss potential application of the taDOM locking protocol implemented in a functional update language – XML-λ. By combination of these theoretical approaches we obtain a solution for querying and updating XML data that can be implemented in a native XML database system with transaction support. We present an LL(1) translation grammar for transformation of queries written in a functional language into sequence of Document Object Model API calls.

1 Motivation

Currently, our research group works on development of a native XML database system that uses XQuery and should also use the XML-λ query language. Therefore we are interested in properties and interconnections between these two artifacts. XQuery represents de-facto industrial standard in querying and XML-λ is a proposal of our group based on simply typed λ-calculus.

In this paper we submit a proposal for transformation of XML-λ statements into a list of DOM operations with ensured transaction isolation through the DOM-based locking protocol – taDOM. We plan to use this solution for extending our native XML database system in the future.

2 Introduction

The crucial property of modern database management systems (DBMS) [11] is concurrent user access. In this work we discuss it in context of a specific type of database systems – native XML database systems. Such systems are primarily used for storing XML data in their original form instead of mapping its structures into e.g. objects or relations.

We outline an existing locking protocol that was developed for XML data – taDOM [14] – and use it for a particular functional query and update language – XML-λ [19, 20]. It is a proposal of a functional framework for querying and

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 25–36, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

manipulating XML data. It is established on type system theory and utilizes simply typed λ-calculus as a base for specification of a query language. This project is currently in phase of development and in future we plan to include an XML-λ module into native XML database systems we currently work on – CellStore [25] and ExDB [1].

The approach we present in this paper is as follows: we transform basic update operations into standard DOM [24] operations that are supported by taDOM and so we can introduce transactional behavior into the language. Formally, we use an LL(1) translation grammar for converting XML-λ expressions to DOM API method calls.

The main contribution of this paper lies in combining the taDOM locking protocol and XML-λ query/update language. We offer a proposal how to interface referenced locking protocol and low-level update operations in given language using DOM operations. This work continues in the topic that we have opened in [17]. In this text we clarify more the update facility of the framework and design a translation grammar that provides the solution for transformation of XML-λ update statements into DOM operations.

The rest of the paper is structured as follows: Section 3 gives a short overview about concurrency and related issues in database systems, Section 4 then outlines the idea of the taDOM locking protocols family. Basic concept of XML-λ with its key update features is briefly introduced in Section 5. The main part of this work that describes our proposal for mapping of DOM operations to the query language is presented in Section 6. In Section 7 we gather a list of certain related papers available. Finally, we conclude with ideas for our future research in Section 8.

3 Concurrency in Native XML DBMS

Common requirement for a database management system is the concurrency control. There are four well-known properties for a transactional system known as *ACID* [11]. Transaction is generally a unit of work in a database. ACID properties are independent on a database (logical) model (i.e. it must be kept in all transactional database systems).

Isolation of transactions in a database system is usually ensured by a locking protocol. Direct application of a locking protocol used in relational databases does not provide high concurrency [15, 22] (i.e. transactions are waiting longer than it is necessary).

We show a huge difference between locking protocols for RDBMS and native XML database system on a small example. Let us have two lock modes: Shared mode and Exclusive mode. The granularity of exclusive lock in RDBMS is typically the row (or record) [6]. In a native XML database we have much more possibilities whether to lock a node or a whole subtree. Hence, we have more choices what to lock and for how long time. These protocols working on XML data extend the basic locking protocol. The basic protocol provides only two types of lock modes, but taDOM3+ has twenty lock modes. More lock modes

imply more complexity in a protocol algorithm. Also to prove whether the protocol is correct is a harder problem.

We suppose only well-formed transactions and serializable plan of update operations [6]. All protocols quoted in this paper satisfy these requirements. We call locking protocols for native XML databases simply *XML-locking protocols* in this paper. The most of these XML-locking protocols are based on the basic relational locking protocols. Hence, XML-locking protocols inherit most of the features, e.g. two-phase locking to ensure serializability.

To design a good locking protocol which minimizes the number of suspended transactions is a challenge because it is much more complex than in RDBMS. It requires new lock modes for individual elements and also for the axes in an XML document [22]. The locking mechanism depends on the query language used, concretely on the atomic operations of the query language and also on the context of these operations.

4 Locking Protocols

Actual research in the area of locking protocols is concentrated rather to DOM model and its methods of approaching individual parts of an XML document. Probably the most advanced research in this topic is carried out at the University of Kaiserslautern in Germany [16, 14, 15]. The researchers are working on XTC (XML Transaction Coordinator) Project [3] – a system which implements several different algorithms of transactional processing on XML data.

XTC project uses extended DOM model (it is called *taDOM*) as a basis for transactional processing.

4.1 taDOM Family of Locking Protocols

The first version of the protocol was denominated as taDOM2. Its improved version is then called taDOM2+. Both of these protocols work with DOM Level 2 operations (about 20 methods, see [23]). Next generation of taDOM locking protocols are taDOM3 and taDOM3+. As expected, these protocols correspond to DOM Level 3 model. The XTC project also provides detailed use cases for these protocols (36 use cases) which completely describe locking scenarios for each operation.

Each of taDOM locking protocols is specified by:

- Compatibility matrix
- Conversion matrix
- Use cases for DOM operations

The compatibility matrix is used when the transaction *t1* is requesting for lock *l1* on a node *n* and there is a lock *l2* of the transaction *t2*. The locking algorithm finds the row *l1* and column *l2* in the compatibility matrix and makes a decision whether to lock (+) or not (-). Table 1 describes Compatibility Matrix for edge locks (ER - edge read, EU - edge update and EX - edge exclusive).

	-	ER	EU	EX
ER	+	+	-	-
EU	+	+	-	-
EX	+	-	-	-

Table 1. The Compatibility Matrix of the Edge Locks

getNode(*nodeID*) **returns** *Node*					
Scenario 0-1 for taDOM3+ Lock Requests:					
Node	Lock	PSE	NSE	FCE	LCE
CN	NR	-	-	-	-

Table 2. Lock Scenario for DOM Operation *getNode*(*nodeID*)

The conversion matrix is used when the transaction *t1* is requesting for lock *l1* on a node *n* and there is a lock *l2* of the same transaction *t1*. Locking algorithm finds the row *l1* and column *l2* in the conversion matrix and converts the lock mode of the node. Hence, each transaction has at the most one lock on each node.

Use cases describe semantics of the locking protocol with regard to DOM operations. Table 2 contains description of the DOM operation *getNode*(*nodeID*). When *getNode*(*nodeID*) operation is called then the locking mechanism has to put the lock of type NodeRead (NR) on the context node(CN). PSE, NSE, FCE, LCE are abbreviations for previous sibling edge, next sibling edge, first child edge, last child edge. The *getNode*(*nodeID*) operation does not put locks on these virtual edges (-).

We consider only taDOM3+ in the next sections of this paper. This protocol is up-to-date nowadays, because it reflects today's needs and was formally checked[1]. The taDOM3+ locking protocol has also really low overhead (minimizes access to the storage) [14].

taDOM3+ protocol provides level 2.99 of isolation [4, 15]. It means that phantom reads[2] are not covered. Therefore it is necessary to do a small extension to these protocols by adding navigation edges to avoid existence of phantom reads. We need to define an additional mechanism – edge locks. To apply edge locks the authors had to extend the XML document model and added new edges between nodes – virtual edges. The compatibility matrix of these locks is discussed in more details in [15].

taDOM Model Structure The tree-like structure in taDOM is enriched by two new node types: *attributeRoot* and *string* [14]. This representational en-

[1] Valenta and Siirtola [21] made a formal proof of the protocol correctness. They verified the taDOM locking protocol using model-checking.

[2] Phantom read happens when new rows added by a transaction are visible from another transaction

hancement does not influence user operations and their semantics on the XML document, but is solely exploited by the lock manager to achieve certain kinds of optimization when an XML document is modified in a cooperative fashion [15].

- *attributeRoot* separates various attribute nodes from their element node. Instead of locking all attribute nodes separately they are locked all together by placing the lock to attributeRoot – concurrency of attribute processing is not allowed.
- A *string* node is attached to the respective text node and only contains the value of this node. It does not allow to block a transaction which only navigates across the node, although a concurrent transaction may have modified the text (content) and may still hold an exclusive lock on it.

Lock Modes The taDOM3+ protocol provides a set of lock modes for the nodes as well as for the edges. Edge locks are used to cover phantom reads in an XML document in order to allow desired level of concurrency. The lock modes together with their mutual relationships (expressed as compatibility matrices) provide concurrency and also preserve the expected ACID properties (especially the level of isolation).

5 Updating XML

There are various theoretical proposals, experimental implementations and defacto standards for XML update languages. As of the publication of the XML 1.0 standard [8] the efforts had been focused on querying such structured data. Approaches for updating have gained more attention in past few years.

Existing papers dealing with updating XML are mostly related to XQuery [7]. The need for introducing updates into XQuery is also considered as one of the most important topics in the further development of the language [10]. As a result, new specification of the XQuery Update Facility [9] was proposed.

In this paper we deal with our approach for querying and updating XML data – XML-λ [19, 20]. It is a framework for manipulating XML based on a type system and simply typed λ-calculus. It is a cornerstone of our long-term research that was invented not only for definition of an update language for XML but also for a broader exploitation, for example heterogeneous data integration or description of denotational and operational semantics of various languages. Indeed, in this paper we focus on using it as a query and update language for XML.

In following paragraphs we expect that reader is familiar with basic concept of the XML-λ Framework. Nevertheless, we repeat its basic concept for convenience.

5.1 Updates in General

Query execution has in general the following structure: (1) Declaration of variables, (2) Evaluation of variables and tree traversal and (3) Output construction.

For read-only systems and even parallel user access it is perfectly sufficient but for systems with update support it is necessary to use a different approach.

Usually, for particular update operations (in the world of native XML database systems) a structure called "pending update list" is utilized. This structure contains ordered list of fundamental modification operations to be carried out at the end of each transaction. Hence, the execution of an update operation (insert, delete, replace) then follows these two steps: (1) node locking and (2) appending an appropriate update operation to the list.

At the end of each transaction the pending update list is processed and all nodes locked by the transaction are unlocked at its end (both in case of commit or abort). This approach is applied both in XQuery and XML-λ[3]. In following sections we cover our solution based on XML-λ in detail.

5.2 XML-λ Framework Basics

XML-λ is a functional framework for manipulating XML. The original proposal [19, 20] defines its formal base and shows its usage primarily as a query language for XML but there is a consecutive work that introduces updates into the language available in [17].

In XML-λ there are three important components related to its type system: *element types*, *element objects* and *elements*. We can imagine these components as the data dictionary in relational database systems. Note also Figure 1 for relationships between basic terms of W3C standards and the XML-λ Framework.

Element types are derived from a particular DTD and in our scenario they cannot be changed – we do not allow any schema changes but only data modifications. For each element defined in the DTD there exists exactly one element type in the set of all available element types (called T_E).

Consequently, we denote E as a set of *abstract elements*. Set members are of element types.

Element objects are basically functions of type either $E \rightarrow String$ or $E \rightarrow (E \times \ldots \times E)$. Application of these functions to an *abstract element* allows access to element's content. *Elements* are, informally, values of *element objects*, i.e. of functions. For each $t \in T_E$ there exists a corresponding t-object.

For convenience, we add a "nullary function" (also known as 0-ary function) into our model. This function returns a set of all abstract elements of a given element type from an XML document.

Finally, we can say that in XML-λ the instance of an XML document is represented by a set E and set of respective t-objects.

Example. Let us consider an example DTD and a fragment of an XML instance shown in Figure 2. For given schema we derive element types as follows: $BIB : BOOK*$, $BOOK : (TITLE, AUTHOR+, PRICE)$, $AUTHOR : (LAST, FIRST)$, $LAST : String$, $FIRST : String$, $TITLE : String$, $PRICE :$

[3] Note that both XQuery and XML-λ are functional languages.

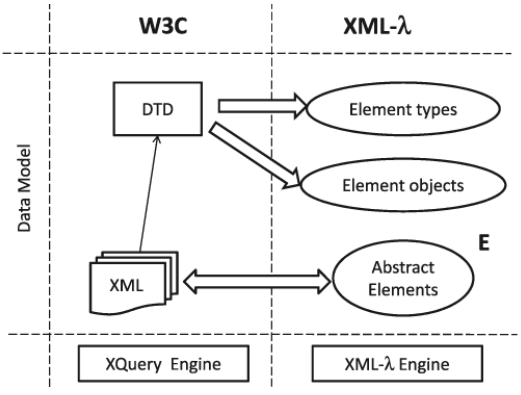


Fig. 1. The Relationship Between W3C and XML-λ Models

String.

Then, we define functional types – designated as t-objects: $BIB : E \to 2^E$, $BOOK : E \to (E \times 2^E \times E)$, $AUTHOR : E \to (E \times E)$, $TITLE : E \to String$, $LAST : E \to String$, $FIRST : E \to String$, $PRICE : E \to String$.

```
<!ELEMENT bib  (book* )>                |  <bib>
<!ELEMENT book  (title,  author+,       |    <book>
          price )>                      |      <title>TCP/IP Illustrated</titl
<!ELEMENT author  (last, first )>       |      <author>
<!ELEMENT title  (#PCDATA )>            |        <last>Stevens</last>
<!ELEMENT last  (#PCDATA )>             |        <first>W.</first>
<!ELEMENT first  (#PCDATA )>            |      </author>
<!ELEMENT price  (#PCDATA )>            |      <price>65.95</price>
                                        |    </book>
                                        |    ...
```

Fig. 2. Example DTD and Fragment of a Valid XML Instance

Having look at the Figure 2 we can see that there are 7 abstract elements (members of $E' \subset E$). Now, for instance, the *price*-object is defined exactly for one abstract element (the one obtained from `<price>65.95</price>` element) and for this abstract element it returns value "65.95".

Let us consider a query that returns all books with price higher than 100. This query is written in XML-λ as:

```
xmldata("bib.xml")
lambda b ( /book(b) and b/price > 100))
```

Here, we do not depict the query evaluation process in detail but it is described sufficiently in [19, 20].

5.3 Fundamentals of Updates

By introspecting the basics of the framework outlined in Section 5.2 – especially the idea of element objects we can see that by updating an XML document we modify actual domains of element objects (i.e. partial functions defined on E) and their ranges.

6 Locking Protocol Mappings

This section describes our solution for translation of XML-λ statements into DOM API calls through a top-down parser directed by an attributed LL(1) translation grammar.

For easier specification of transformation between XML-λ primitives and DOM operations we define new operation $\diamond$:

$$f^{+}(v) = \{f^{1}(v), f^{2}(v), f^{3}(v), \ldots\}$$

$$f^{+}(v) \diamond g() = \bigcup_{u=1}^{\infty} \{g(f^{u}(v))\}$$

This operation is defined on sets. We can say that the $g()$ function is applied on each element of a set.

6.1 A Pinch of Translation Theory

We solve the problem of mapping by translation from one language to another. The straightforward approach is based on construction of an attributed translation grammar [5]. Then all queries written in XML-λ can be translated into a sequence of DOM operations.

Here we refer shortly to definition related to translation grammars – note that we use an attributed translation grammar, i.e. a context-free grammar augmented with attributes, output symbols and semantic rules.

The attributed translation grammar is 4-tuple $APG =< PG, A, V, F >$, where PG is a basic translation grammar $PG =< N, \Sigma, D, R, S >$. N is set of non-terminal symbols, Σ is set of terminals, D is set of output symbols, R is a set of grammar rules $A => \alpha$, where $A \in N$, $\alpha \in (N \cup \Sigma \cup D)*$ and S is the start symbol.

Remaining symbols are related to APG and have the following meaning

A is a finite set of attributes. It is divided into two disjoint sets for synthesized (denoted S) and inherited (denoted I) attributes.
V is a mapping that assigns a set of attributes to each non-terminal symbol $X \in N$
F is a finite set of semantic rules

The example stated in the following section is based on this formalism.

6.2 XML-λ to DOM Translation Grammar

We use the standard formal translation directed by an LL(1) parser where the formal translation is described by translation grammar as follows:

$N = \{S, R_0, R_1, T\}$
$\Sigma = \{/, sL, var\}$
$D = \{$ Ⓢ,Ⓣ,Ⓒ $\}$
$R = \{\ S \rightarrow / \ R_0|varR_1,$
$R_0 \rightarrow sL$ Ⓢ $T\ R_1,$
$R_1 \rightarrow /$ Ⓒ $sL\ T\ R_1|\epsilon,$
$T \rightarrow$ Ⓣ $\}$

Note that terminal symbols are output tokens from a lexical analyzer.
We proposed necessary attributes for translation $A = \{name, string\}$, where $I(T) = \{name\}$, $I(Ⓣ) = \{name\}$, $S(sL) = \{string\}$. Attributes are used for storing tag names in the process of translation.

Syntax and semantics of the translation grammar is described in Table 3.

Syntax	**Semantics**
$S \rightarrow / \ R_0\|varR_1$	
$R_0 \rightarrow sL$ Ⓢ $T\ R_1$	T.name := sL.string
$R_1 \rightarrow /$ Ⓒ $sL\ T\ R_1\|\epsilon$	T.name := sL.string
$T \rightarrow$ Ⓣ	Ⓣ.name := T.name

Table 3. Syntax and Semantics Table

After translation the output symbols are rewritten in following way:

Ⓢ $\rightarrow doc \diamond getDocumentElement() \diamond getChildNodes()^+$
Ⓣ $\rightarrow \diamond getTagName($Ⓣ.name$)$
Ⓒ $\rightarrow \diamond getChildNodes()$

Following example shows how we can transform XML-λ queries to DOM operations. These operations implicitly use taDOM3+ locking protocol synchronization primitives.

6.3 XML-λ Query Evaluation Example

Let us have a look at an example of a delete operation in the XML-λ language. Following statement deletes all books specified by given title:

```
xmldata("bib.xml")
delete( lambda b ( /book(b) and
          b/title = "TCP/IP Unleashed"))
```

We translate the inner expression of the statement

```
(/book(b) and b/title = "TCP/IP Unleashed")
```

The translation is based on a top-down method using expansion operation $\Rightarrow$. Expansion rule depends on the top terminal of the processed input string. Then we can use a standard LL(1) parser. Translation then starts as follows:

$$S \Rightarrow / \; R_0 \overset{R_0}{\Rightarrow} / \; sL \; ⓢ \;\; T \; R_1 \overset{T}{\Rightarrow} \;\; / \; sL \; ⓢ \; ⓣ \; R_1 \overset{R_1}{\Rightarrow} \;\; / \; sL \; ⓢ \; ⓣ$$

By this derivation we have translated the first part of the expression – `/book(b)`. Then, we continue with the second part:

$$S \Rightarrow \;\; var \; R_1 \overset{R_1}{\Rightarrow} var/ \; ⓒ \; sLTR_1 \overset{T}{\Rightarrow} var/ \; ⓒ \; sL \; ⓣ \; R_1 \overset{R_1}{\Rightarrow} var/ \; ⓒ \; sL \; ⓣ$$

We get the translated string by omitting input symbols. We suppose that the semantic rules were applied during translation. In the input symbol `var` we saved the first part of the translation. The second part is concatenated with the first part through the variable `b`. The output of the translation is the following sequence of output symbols: ⓢ ⓣ ⓒ ⓣ.

We can rewrite these output symbols to taDOM operations and then we get:

$doc \diamond getDocumentElement() \diamond getChildNodes()^{+} \diamond getTagName(ⓣ.\text{name})$
$\diamond getChildNodes() \diamond getTagName(ⓣ.\text{name})$

The main part of the update statement is the path expression. Now we have to select nodes which satisfy condition – $title = "TCP/IP\ Unleashed"$. The string comparison operation is not a DOM operation, so for purpose of this paper is omitted here.

The translation grammar described above can be directly used to ensure isolation of transactions in the XML-λ language.

7 Related Work

First, considering query and update languages, there are many papers and proposals. The most important specifications in the context of this work are the XML Query Language 1.0 Specification [7] and a Working Draft of the XQuery Update Facility [9]. They form de-facto standard in the world of XML.

Another branch of papers is focused on a specific database system and describes usually a complete solution seen from a wider perspective. It is quite a common practice that database groups at technical universities and similar institutes develop their own database systems. In the area of XML we can mention eXist [2] or Natix [12]. Finally, we should also mention CellStore [25] – a

database system being developed at our department. Authors of these systems usually describe in detail the transformation of XQuery statements into their algebras.

Issues related to transactions and systems that support transactional behavior are covered as description of transaction protocols [6, 13, 14] or transactional benchmarking [18].

8 Conclusions and Future Work

We have shown an approach for introducing fundamental transactional operations into a functional query and update language for XML. Through the use of the taDOM protocol and the XML-λ Framework we have obtained a theoretical solution for native XML database systems that allows querying and updating XML data in a safe manner. We accomplish this by introducing an LL(1) attributed translation grammar for transformation of XML-λ statements into sequence of DOM API calls. This forms a solid base for our ongoing research – studying of mutual semantic transformations of queries written in one query language into another, e.g. conversion of XQuery queries into XML-λ and vice-versa.

Considering the fact that we have presented here only first sketch of the solution there is still a lot of clarification and experimental work ahead. In the near future we plan to design and implement a prototype of the XML-λ query engine into the CellStore database system. After that there are many open topics related to correctness and benchmarking of the prototype.

9 Acknowledgments

We would like to thank to Jiří Velebil for his valuable hints related to mathematical expressions we use in this paper to formulate formalisms in a (hopefully) clear way.

This work has been supported by the Ministry of Education, Youth, and Sports under Research Program MSM 6840770014.

References

1. ExDB Project Homepage. http://swing.felk.cvut.cz/~loupalp.
2. eXist Project Homepage. http://www.exist-db.org.
3. XTC Project. http://wwwdvs.informatik.uni-kl.de/agdbis/projects/xtc.
4. A. Adya, B. Liskov, and P. O'Neil. Generalized isolation level definitions. *ICDE*, 00:67, 2000.
5. A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling. II. Compiling*, volume II. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1973.
6. P. Bernstein and E. Newcomer. *Principles of Transaction Processing.* Morgan Kaufmann Publishers, 1st edition, 1997.

7. S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language, January 2007. http://www.w3.org/TR/xquery/.
8. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (XML) 1.0 (third edition), February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/.
9. D. Chamberlin, D. Florescu, and J. Robie. XQuery update facility. http://www.w3.org/TR/2006/WD-xqupdate-20060711/.
10. D. D. Chamberlin. XQuery: Where do we go from here? In *XIME-P*, 2006.
11. C. J. Date. *An Introduction to Database Systems, 6th Edition.* Addison-Wesley, 1995.
12. T. Fiebig, S. Helmer, C.-C. Kanne, G. Moerkotte, J. Neumann, R. Schiele, and T. Westmann. Anatomy of a native xml base management system. *VLDB Journal*, 11(4):292–314, 2002.
13. J. Gray and A. Reuter. *Transaction Processing : Concepts and Techniques.* Morgan Kaufmann Publishers, 1st edition, 1993.
14. M. P. Haustein and T. Härder. A synchronization concept for the DOM API. In H. Höpfner, G. Saake, and E. Schallehn, editors, *Grundlagen von Datenbanken*, pages 80–84. Fakultät für Informatik, Universität Magdeburg, 2003.
15. M. P. Haustein and T. Härder. An efficient infrastructure for native transactional XML processing. *Data Knowl. Eng.*, 61(3):500–523, 2007.
16. M. P. Haustein, T. Härder, C. Mathis, and M. W. 0002. Deweyids - the key to fine-grained management of xml documents. In C. A. Heuser, editor, *SBBD*, pages 85–99. UFU, 2005.
17. P. Loupal. Updating typed XML documents using a functional data model. In J. Pokorný, V. Snášel, and K. Richta, editors, *DATESO*, volume 235 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
18. M. Nicola, I. Kogan, and B. Schiefer. An xml transaction processing benchmark. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 937–948, New York, NY, USA, 2007. ACM Press.
19. J. Pokorný. XML functionally. In B. C. Desai, Y. Kioki, and M. Toyama, editors, *Proceedings of IDEAS2000*, pages 266–274. IEEE Comp. Society, 2000.
20. J. Pokorný. XML-λ: an extendible framework for manipulating XML data. In *Proceedings of BIS 2002*, pages 160–168, Poznan, 2002.
21. A. Siirtola and M. Valenta. Verifying parameterized taDOM+ lock managers. *SOFSEM 2008*, pages 460–472, 2008.
22. P. Strnad and M. Valenta. Object-oriented Implementation of Transaction Manager in CellStore Project. *Objekty 2006, Praha*, pages 273–283, 2006.
23. The W3C Consortium. W3C homepage. http://www.w3.org.
24. The W3C Consortium. Document Object Model (DOM), 2005. http://www.w3.org/DOM/.
25. J. Vraný. Cellstore - the vision of pure object database. In V. Snásel, K. Richta, and J. Pokorný, editors, *DATESO*, volume 176 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

Database Engineering from the Category Theory Viewpoint

David Toth

Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University
Karlovo náměstí 13, 121 35 Praha 2, Czech Republic
tothd1@fel.cvut.cz

Abstract. This paper gives an overview of XML formal models, summarizes database engineering practices, problems and their evolution. We focus on categorical aspects of XML formal models. Many formal models such as XML Data Model, XQuery Data Model or Algebra for XML can be described in terms of category theory. This kind of description allows to consider generic properties of these formalisms, e.g. expressive power, optimization, reduction or translation between them, among others. These properties are rather crucial to comparison of different XML formal models and to consequent decision which formal system should be used to solve a concrete problem. This work aim is to be the basis for further research in the area of XML formal models where category theory is applied.

1 Introduction

In this paper we will focus on some peculiarities from today's database world. Now, in spring 2008, we have many database technologies, many technical frameworks, many solutions for different and similar problems. What we do not have is a global point of view of databases (DB); theoretical approach stating theorems about database models and languages. This paper summarizes database technologies from higher perspective and introduces some of the terms from mathematical category theory (CT). These two aspects, databases and category theory, are put together in order to give new look at the database technologies, to give new way of data model and languages description; and to find new language in which we could ask and answer more generic questions, e.g. about expressive power of (query) languages of particular data models.

This paper deals with databases. More precisely we should say it treats problems which appear when we would like to know which database technology should be used in software project. There are generaly more requirements leading one to use DB, e.g. to make the data persistent, to assure concurrency, etc. More about database technology in general can be found in Date's *Introduction to Database Systems* [13]. There are many factors influencing the decision. In fact in praxis it is more subjective (personal or team) decision.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 37–48, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

From the software engineering point of view there are at least these kinds of factors; (1) *Human* factors, as e.g.: knowledge about particular DB product, concrete DB technology experience; individual or team, subjectivelly favourite / preferred DB technology; (2) *Technical* aspects: vendor influence, e.g. offered support, problem solving time, programming languages support, performance, accessibility, clustering, and many others. (3) *Problem definition*: teoretical aspects of problem, data itself, its nature (data character), i.e. (a) structuralization (no inner structure e.g. streams, files respectively, weak structure e.g. newspaper articles, strong structure any well structured forms, e.g. tax return form), (b) data contain metadata (typical for XML documents), data separated from metadata respectively (typical for tables—relations). (4) Possibly *other aspects*.

Some of these factors are summarized in SWEBOK [39]. In software engineering paper we would like to address especially the first two categories. In SIGSOFT [36] and especially in SEN [34] can be found more on these topics. But this text is intended to be considered as more database-oriented. Therefore we will focus more on the problem's aspects as the third factor mentioned above. Nevertheless all topics covered here are closely related to software engineering and even to database engineering which we deal with later. Next we will take a closer look at particular database technologies emphasizing the problem's aspect.

1.1 Relational Database Technology

Historically the first database approach which solved the inconsistencies, redundancy, concurrency and other problems was the relational model. C. J. Date in his *Introduction* [13] deals with the relational approach to represent data (i.e. relational data modeling and storing among other aspects). Other very deep insight into relational data model can be found in E. F. Codd's *Relational model for database management* [10]. One can imagine the main idea as data grasped via a relation in mathematical notion, i.e. all data can be viewed as relations, in other words sets with internal structure of its elements. Relations are interconnected together using values of particular set of elements which is usually called foregin key usage.

Can any data be represented using relational approach, i.e. can any data be stored as relations, tables respectively? We must consider the fact that the data could possibly change its structure, and even that we do not know the structure before we have the data physically. Can the changing data structure be modeled using relational approach? What other questions play a significant role when we consider expressive power of e.g. relational algebra, etc.? These and related questions will be considered in future works which will contain CT oriented features. In this paper data model description and related topics will be treated. Further we focus on object technologies.

1.2 Object Database Technology

Object and object-oriented databases arose out of the impedance mismatch between relational and object data models. The essence of this problem lies in a

different kind of data representation, i.e. once as relations or n-aries and once as objects. The problem inhere in data translation. In other words data must be mapped between classes of objects and relations of n-aries. More about the object relational mapping can be found in Fussel's *Foundations of Object Relational Mapping* (ORM) [20]. Another paper about ORM can be found in [1]; implementation issues are covered in persistence framework Hibernate [24].

To object and object-oriented databases and to object database management systems is dedicated the web site [32]. At this place many object and object-oriented technologies, especially database technologies, of course, and open source as well, can be found.

1.3 XML Databases

The XML Databases were born actually very shortly after the XML, the new language for semistructred data description, a W3C's standard respectively [17], emerged in 1998. More about XML evolution can be found at [46]. It did not took a long time and new term Native XML Database, often just NXD, came abroad [37]. We will use the term, as e.g. R. P. Bourret in [7] does. Why NXD appeared and what to expect from them is described in [23] or [6]. R. P. Bourret [8] also maintain fresh list of NXD products and even more wide XML products in general.

The main motivation for NXD usage resides in the impedance problem again, as in case of ODB as well. The typical situation where NXD are used is web portals and web applications communicating through web services. The web services standards are based on XML and related standards. Therefore it is evident that the need for XML document transformation should be avoided to speed up the performance of applications of this type. We have treated this yet earlier in [42].

The principle of NXD consists in XML data model as intrinsic data model of the database engine. R.P. Bourret is more specific about what XML-Enabled and what XML-native suppose to mean, e.g. in [7].

The paper is structured as follows. Section 2 deals with relationship of software engineering and database engineering. Section 2.2 refers problems relevant to appropriate database technology selection in the introduction above bearing in mind. Section 3 summarizes issues related to essences of particular data models. In section 4 there are mostly XML related standards and technical part of XML databases dealt with and section 5 treats formalisms developped for the purpose of XML Data model description. Section 6 introduce basic terms from category theory (CT) and gives formal background for cited formalisms. Section 7 summarizes exhibited XML formalism. Last section 8 reveals our future plans.

2 Software Engineering and Database Engineering

Terminological note on database engineering. Normally the term database engineering is used to describe an area of processes, methods and techniques,

formalisms and languages useful for database designing, and in general, useful for database application development. There are several conferences around database engineering topics. Clear definition of what is database engineering does not exist. What we mean under database engineering is specialization of software engineering practices for purposes of database application, i.e. application strongly related to data which makes persistent and which further operates with. Practically we mean specialization of all techniques where arbitrary database artifact, most typically it is database schema, is created, changed (most common case), or removed.

From waterfall to iterative development. Software engineering in past was understood as sequential processes equivalent to phases which must be performed in a serial way. The phases typically are: feasibility study, business analysis, requirements analysis, architecture analysis and design, logical design, GUI design, DB design, physical design, coding, testing, refactoring, installing, deploying, measuring, among others. The same can be said, as an analogy of course, about database engineering, i.e. database design, database tuning and administration among others. But in today's world when agile methodologies in sofware engineering become successfull and more and more widespread, it also seems to be inevitable to use agile or generally speaking iterative approaches in database community. It is a paintful step for every single database expert long time experienced sequential approach when starting use agile principles.

MDA—Model Driven Architecture. The MDA approach [30] seems to be contradictory in the context of agile methodologies. But not necessarilly. Iterative approaches allow to build software systems more focused on one particular problem, emphasizing one aim in time (during iteration). The basic imagination could be as very little waterfalls chaining every iteration stressing analysis or design or programming depending on current phase. We can figure out here the semantics of the word phase depends strictly on chosen methodology.

2.1 Database Engineering and Evolutionary Approach

From the point of view of the database engineering there is need to elaborate database design. Typically conceptual model is considered as a part of database modeling and as a part of a database design phase. In fact we would like to stress here that there is no need to create domain model as UML [31] class diagram during business analysis and also E-R diagram as a part of database modeling independently. It is possible to create or even generate E-R diagram or UML class diagram in Data Modeling Profile from domain model. It is typically expressed as UML class diagram which is done during business analysis. Actually some CASE tools offer this functionality in these days, e.g. the Enterprise Architect [15].

Database modeling, a part of database design, can be viewed as a transformation from domain model. And this does not mean that all the modeling must

be finished before normalization or tuning starts. The core of the evolutionary approach lies in doing the whole step by step in very small parts which have to be integrated. Continuous refinement is necessary. One of the biggest argument against iterative database development is the need for neverending reworking and refining of non-stabilized artifacts—which is possibly a great number.

As a resume here we would like to pinpoint the possibility to look at database evolution concurrently with regular software evolution. And therefore to see database engineering as a specialization of software engineering. The principles of MDA—model transformations are essentially the same in software and database engineering. This kind of abstraction should help us thinking in software engineering and database engineering in very similar way. Furthermore CT can help us when dealing with models, their properties and qualities, and transformations.

2.2 The Database Technology Selection

Which particular DB technology should we choose to use? What should lead us — help us? The discussion below involves these questions.

Relational Databases (RDB). From the historical perspective there is a big argument which says to use RDBs. It is deep insight into relational technology, strong mathematical background in form of data relational model and relational algebra. Many people made refinements of this technology for a long time. Shortly, RDBs are greatly elaborated in comparison to other (and younger) technologies.

Object Databases (ODB). In [32] we could find at least these important reasons why to select ODB instead of RDB or XDB: embedded DBMS application, complex data relationships, deep object structures, changing data structures, development team is using agile techniques, massive use of object oriented programming language, there are many objects including collections, data is accessed by navigation rather than query.

One of the most popular ODBMS in open source community is db4objects [14]. Another example could be the NeoDatis ODB [29] or GemStone/S [21].

XML Databases (XDB). With XDB, and NXD respectively, fine-grained reuse of content is possible; NXD allows sophisticated hypertext applications with mixture of stuctural and fulltext query. The most typically cited NXD benefits are flexibility and reuse.

We have treated of this issue in greater detail in [41]. Three distinct metrics, ρ, τ, and ξ, were proposed for different kinds of database technologies.

3 Essentials of Data Models

Does exist essential difference between different data models? In words of CT we could say: belong all categories of all data models into the same category (of categories)? We will focus a bit more on this in section 6 — The Category Theory Standpoint.

Now imagine not to use CT. The question if there exists any problem which cannot be solved using arbitrary technology would have to be proven hardly. We would have to prove that every single case of data expressed in one data model could also be expressed in every other data model.

Theoretically any data can be expressed in arbitrary format, i.e. (1) tables, nested tables respectively, (2) the web of objects or (3) hierarchy of elements if we found mappings between all data instances.

Mapping from XDB to RDB can be viewed so that any XML document can be stored (represented) in RDB in generic tables (ELEMENTS, ATTRIBUTES, DOCUMENTS, etc.). That objects can be stored as record in tables which can be seen e.g. in Object Relational Mapping (ORM) Pattern. The other way can be imaginated as direct overwriting of RDB data using wrapping method for column content and nesting in case of foreign keys (FK). FK can also be represented as ID and IDREF attributes in XML documents.

Mapping from XML documents to objects can be grasped in a way that XML data model will be grasped as a tree, object model would be accessed as a graph. A tree is also a kind of a graph. This idea is demonstrated e.g. in previous work [42], and it is implemented in java programming language in JAXB—Java API for XML Binding [38]. These mappings are typically based on DTDs or XML Schema or even RelaxNG. R. P. Bourret wrote general paper on XML document mapping between relational and object models [5].

The same could be obtained if we find all the mappings between one and another DB structure — technology (RDB, ODB, XDB). But a more convenient way would be to find out the way of general description and prove that these mappings have to exist or that it is impossible these mappings would exist. And not only convenient, we should consider all data models; even those which do not exist yet. It seems that different technologies fit for different kind of problems but they are essentially the same after all. Are they? Can we prove this using category theory? We would like to focus our future research on these questions.

And there are other interesting questions leading us to finding one framework only, CT, e.g. is it possible to store and effectively retrieve data with unknown and/or changing data structure in RDB, ODB and XDB?

4 XML Databases

XML standard [17] first released in 1998 and last updated in 2006 has initiated the great interest in XML Databases and NXDs.

R. P. Bourret summarizes and yet reconciles not only basic problems and principles of native XML databases in [7]. In his article R. P. Bourret says: "...

the problem is practical, not teoretical ..." about the problem of arbitrary data expressed in any data model. He also states "... in RDB there is an impractical number of joins ..." in [6].

Another resource concluding the benefits of NXD usage [23] tries to list not only the advantages but also the possible problems.

We will very shortly summarize here XML database technologies and in the next section we will cover formal models for technological standards and data models treated here. Three typical NXD representants are as follows: (1) One of the most common NXD's is eXist [16]. (2) Another very popular NXD from Apache is called Xindice [45]. Oracle Berkeley XML DB is described at [33].

XML formal models and languages from the point of view of XML standard are at least as follows. We could say the following list is an extension of XML Data Models according to R. P. Bourret [7]. The majority of all treated models are tree-based formal models and algebras.

- DOM — Document Object Model [40].
- SAX — Simple API for XML [28].
- Infoset [11].
- XPath [9].
- XQuery [18].
- XML-λ: functional approach to XML description [27].
- Most likely there are many other standard-based formalisms.

Next section reveals the formal background of stated standards and needed relationships.

5 XML Databases Formal Models

A Formal Data Model and Algebra for XML [3] is the name of the article suggesting a tree-based model as a formal data model for XML and as an algebra for XML, an algebra based on such trees, i.e. essentially same structure as DOM and the related.

XML Data Model as it is defined in XPath or XQuery is basically grasped as a forrest of trees of nodes representing elements and attributes and texts and other XML features mentioned in previous section. Many of the XML Data Model facets are explained in XML infoset [11].

As an XML data model could be grasped DOM. Basically the formalism is build on same terms as in case of XPath or XQuery. So from the CT point of view it would be grasped as one formalism.

Sengupta and Mohan summarized in [35] the formalisms used to describe data in XML format. They found these formalisms:

- Tree-based formalisms (XAlgebra, DOM and others).
- SAL — Semi-structured Algebra [4].
- The ENF — Element Normal Form concept: it is proved that attributes can be avoided in cases of general description because every XML document with attributes can be (without any information loss) transformed onto the XML document variant without attributes and vice versa.

- HNR — Heterogeneous Nested Relations — also arise from NF^2 (Non-first normal form).
- HNRC — HNR Calculus — analogously to relational calculus.
- HNRA — HNR Algebra — analogously to relational algebra.
- DSQL — Document SQL — as an analogy to SQL.

For all of these we would like to find the proper meta-formal way of description in terms of CT; and finally find out the properties valid among these categories.

XML Algebra based on monads is another interesting formalism [19]. But this approach, this XML Algebra, lacks references and dereferences. The algebra specified in [3] count on it and offer a way of how to solve this problem.

P. Wadler proposed several formal models. Especially formal semantics for XSL [43], and semantics for XPath [44].

Future challenges would be to describe formalisms used for metamodels—conceptual models and visualisations e.g. via UML.

Having in mind the extent of all this we will focus on just few factors from the previous list in CT. We introduce CT in the next section.

6 The Category Theory Standpoint

This section deals with an introduction to CT and categorical description of XML formal models defined above. Let us take a look at the word category itself.

6.1 Three semantics of the word Category

Categories originally arose in mathematics out of the need of a formalism to describe the transformation from one type of mathematical structure to another. Category represents a kind of mathematics. Barr and Wells [2] state then category as a mathematical workspace.

A category is also a mathematical structure. It is then a generalization of both ordered sets and monoids. Barr and Wells call it in this case category as a mathematical structure.

Category as a theory is the third recognized point of view. Category can be seen as a structure that formalizes a mathematican's description of a type of structure. Traditional way to do this in mathematics, in mathematical logic respectively, is to use formal languages with rules, terms, axioms and equations.

We now define the term category more precisely. We will use the notation and mathematical formalism used in [2] for the rest of this section.

6.2 Definition of a Category

Definition 1. A category $\mathcal{C}$ consists of **objects** (denoted by $A, B, C, ...$) and **morphisms** between them (denoted by $f : A \to B, g : B \to C, ...$). These data

are subject to obvious axioms expressing **composition**, its **associativity**, and existence of **identity** morphisms (units w.r.t. composition).

A paradigm category is the category $\mathcal{S}et$ of all sets and mappings. See [2] for more details.

6.3 Definition of CCC, Connections to λ-Calculus

We define now the concept of a cartesian closed category (CCC). It is proved in [26] that CCC's are *essentially the same thing* as simply typed λ-calculus. Altough the following definition is rather a technical, one may bear in mind that the category $\mathcal{S}et$ forms a paradigm example of a CCC.

Definition 2. A category $\mathcal{C}$ is called a cartesian closed category (CCC) if it satisfies the following:

(1) There is a terminal object 1.

(2) Each pair of objects A and B of C has a product $A \times B$ with projections

$$p_1 : A \times B \to A \text{ and } p_2 : A \times B \to B.$$

(3) For every pair of objects A and B, there is an object $[A \to B]$ and an arrow $eval : [A \to B] \times A \to B$ with the property that for any arrow $f : C \times A \to B$, there is a unique arrow $\lambda f : C \to [A \to B]$ such that the composite

$$C \times A \xrightarrow{\lambda f \times A} [A \to B] \times A \xrightarrow{eval} B$$

is f.

Note that we call an object 1 of a category $\mathcal{C}$ terminal iff there is exactly one arrow $A \to 1$ for each object A of $\mathcal{C}$.

λ-calculus is one of the formal description of what is usually called an XML data model. In [27], there is XML-λ approach to view XML. The typical point of view of XML is a tree. But this approach emphasizes the notion of a function. And there is a hypothesis that as functions or as trees we describe the same, and that both kinds of description are of the same power. We will try to prove this in future work. This proof will rely on what is stated above.

Related works involving Object Databases description using CT are [22] and [25]. Altough it is not about the XML data model the principles of formal description are very similar.

Because of the λ-calculus is one of the formal description or precise point of view of XML data model and because of what Lambek and Scott proved in their work [26], the XML data model can be described as CCC. We would like to find out if also other categories as descriptions of other formal models are also CCC. The main idea is to determine if all the models are also essentially the same in the sense of Lambek and Scott; which should be done in next work.

6.4 Proposed Descriptions based on Category Theory

The very first description we considered was the XML-λ approach. This approach is an instance of the λ-calculus theory which, grasped as a category, is CCC [26].

Let G be a graph. As a graph we mean special case of oriented graph with loops on nodes. The category $\mathcal{C}_{Graph}$ of such graphs is defined as follows: Collection of objects consists of all possible graphs G; Collection of arrows consists of all graph homomorphisms ϕ_G. Identity arrows are isomorphisms of objects. It is needed to be verified, that this mathematical structure is a category, but it is obvious; we let this to the kind reader. Furthermore this category is CCC, as is proved e.g. in [2]. This model, category $\mathcal{C}_{Graph}$, is actually a useful model for object databases [25] when other aspects than object visibility are ignored. Objects in this category can be grasped as objects from object programming. But as a model for XML databases it cannot be used because of the loops. When the arrow is interpreted as a relation of nesting, element in XML document cannot be nested into itself.

Let $\mathcal{C}_{Tree}$ be the category of trees (derived from the category above). Let objects be trees and arrows tree homomorphisms. Again that it is a category is needed to be verified as above. This category is not CCC. Because there would needed to exist the terminal object with loop node. But such an object cannot be interpreted as any XML document.

Let $\mathcal{C}_{HFS}$ be the category of hereditary finite sets. All these sets can be undrestood as ϵ-trees. This approach seems to be very promising and is currently under development.

There are many other approaches which will be in detail covered in subsequent works.

7 Conclusions

We have shown that the database technology selection is in praxis mostly subjective problem. There are few practical reasons which would lead us to develop theoretical framework for data modeling.

We have stressed the natural evolution in software engineering from waterfall to iterative database evolution approaches which still become more common.

We have discussed XML formal models and their properties.

The conclusions from CT applications are rather poor. But we tried to summarize the database problems, existing solutions and we tried to offer another, originial, approach.

Furthermore this work open the doors for further more specific research, using very strong mathematical background. Next section reveals our future plans.

8 Future Works

Subsequent work will be focused on the question of essentiality of the RDB, ODB and XDB models, their computational equivalence, expressive power of relative languages and similar aspects.

In the near future we will try to categorify every formal model for XML data which would be found.

In far future there is a huge space for using CT formalism to describe itself, i.e. use the notion of categories of categories. And according to Lambek and Scott [26] it seems to be possible use only one notation, one language and one formalism, we mean CT of course, for all (types of) data models. We would like to try to find out such a way of description of data models.

Next, in the future, not only XML databases and NXD will be described using CT. But we would like to try to give formal basis for all data models. Good example could be relational algebra and Crole's way of categorical description which should be further elaborated [12]; using categorical semantics. And there are many other similar examples as an inspiration for future research activities.

References

1. S. W. Ambler. Mapping Objects to Relational Databases: O/R Mapping In Detail. 2006. http://www.agiledata.org/essays/mappingObjects.html.
2. M. Barr and C. Wells. *Category Theory for Computing Science*. International Series in Computer Science. Prentice-Hall, 1990. Second edition, 1995.
3. D. Beech, A. Malhotra, and M. Rys. A formal data model and algebra for XML, 1999.
4. C. Beeri and Y. Tzaban. SAL: An algebra for semistructured data and XML. In *WebDB (Informal Proceedings)*, pages 37–42, 1999.
5. R. P. Bourret. Mapping DTDs to databases, 2001. http://www.xml.com/lpt/a/2001/05/09/dtdtodbs.html.
6. R. P. Bourret. Going native: Making the case for XML databases, 2005. http://www.xml.com/pub/a/2005/03/30/native.html.
7. R. P. Bourret. XML and databases, 2005. http://www.rpbourret.com/xml/XMLAndDatabases.htm.
8. R. P. Bourret. XML database products, 2007. http://www.rpbourret.com/xml/XMLDatabaseProds.htm.
9. D. Chamberlin, A. Berglund, and e. a. Scott Boag. XML Path Language (XPath) 2.0, September 2005. http://www.w3.org/TR/xpath20/.
10. E. F. Codd. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
11. J. Cowan and R. Tobin. XML information set (second edition), April 2004. http://www.w3.org/TR/2004/REC-xml-infoset-20040204/.
12. R. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. Cambridge University Press, 1993.
13. C. J. Date. *An Introduction to Database Systems*. Addison-Wesley Publishing Co., Inc., 2003. 8th ed.
14. db4objects — Open Source ODBMS. http://www.db4o.com.
15. Sparx's Systems Enterprise Architect UML CASE Tool. http://www.sparxsystems.com.
16. eXist — Open Source Native XML Database, Home Page. http://exist.sourceforge.net.
17. Extensible Markup Language (XML) 1.0 (Fourth Edition), 2006. http://www.w3.org/XML.

18. M. Fernández, A. Malhotra, J. Marsh, M. Nagy, and N. Walsh. XQuery 1.0 and XPath 2.0 Data Model, September 2005. http://www.w3.org/TR/xpath-datamodel/.
19. M. Fernandez, J. Simeon, and P. Wadler. A semi-monad for semi-structured data. *Lecture Notes in Computer Science*, 1973, 2001.
20. L. M. Fussel. Foundations of Object Relational Mapping. http://www.chimu.com.
21. GemStone/S ODB. http://www.gemstone.com/products/smalltalk.
22. J. Güttner. *Object Databases and the Semantic Web*. PhD thesis, 2004.
23. E. R. Harold. Managing XML data: Native XML databases, 2005. http://www.ibm.com/developerworks/xml/library/x-mxd4.html.
24. Hibernate — Java and .NET persistence framework. http://www.hibernate.org.
25. P. Kolenčík. *Categorical Framework for Object-Oriented Database Model*. PhD thesis, 1998.
26. J. Lambek and P. J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, March 1988.
27. P. Loupal. Querying XML with lambda calculi. In *Ph.D. Workshop, VLDB2006*, 2006.
28. D. Megginson. SAX – Simple API for XML, 2005. http://www.saxproject.org/.
29. NeoDatis ODB. http://wiki.neodatis.org.
30. Object Management Group (OMG). MDA — Model Driven Architecture, 2007. http://www.omg.org/mda.
31. Object Management Group (OMG). UML — Unified Modeling Language, 2007. http://www.uml.org.
32. ODBMS — Object And Object Oriented Database Management Systems. http://www.odbms.org.
33. Oracle Berkeley XML DB, home page. http://www.oracle.com/database/berkeley-db/xml/index.html.
34. SEN — Sigsoft Software Engineering Notes. http://www.sigsoft.org/SEN/surfing.html.
35. A. Sengupta and S. Mohan. Formal and conceptual models for xml structures - the past, present, and future, 2003.
36. SIGSOFT — ACM's Special Interest Group, dedicated to Software Engineering. http://www.sigsoft.org.
37. S. Staken. Introduction to Native XML Databases. 2001. http://www.xml.com/pub/a/2001/10/31/nativexmldb.html.
38. Sun Microsystems, Inc. Java architecture for XML binding (JAXB), 2003. http://java.sun.com/webservices/jaxb/.
39. SWEBOK — Software Engineering Body Of Knowledge. http://www.swebok.org.
40. The W3C Consortium. Document Object Model (DOM), 2005. http://www.w3.org/DOM/.
41. D. Toth and P. Loupal. Metrics analysis for relevant database technology selection. In *Objekty*, 2007.
42. D. Toth and M. Valenta. Using Object And Object-Oriented Technologies for XML-native Database Systems. In J. Pokorný, V. Snášel, and K. Richta, editors, *DATESO*, CEUR Workshop Proceedings. CEUR-WS.org, 2006.
43. P. Wadler. A formal model of pattern matching in XSL. Technical report, 1999.
44. P. Wadler. Two semantics for xpath, 1999.
45. Apache Xindice, Home Page. http://xml.apache.org/xindice/.
46. XML Main Page. http://www.w3.org/XML.

Tensor Decomposition for 3D Bars Problem

Jan Platoš, Jana Kočíbová, Pavel Krömer, Pavel Moravec, Václav Snášel

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic {jan.platos, jana.kocibova.st1, pavel.kromer.fei, pavel.moravec, vaclav.snasel}@vsb.cz

Abstract. In this paper, we compare performance of several dimension reduction techniques, namely SVD, NMF and SDD.The qualitative comparison is evaluated on a collection of bars. We compare the quality of these methods from on the base of the visual impact. We also compare dimension reduction techniques SVD and HO-SVD on tensors - 3D bars.

1 Introduction

In order to perform object recognition (no mater which one) it is necessary to learn representations of the underlying characteristic components. Such components correspond to object-parts, or features. These components can occur in different configurations to form many distinct images. Identifying the underlying components which are combined to form images is thus essential for learning the perceptual representations necessary for performing object recognition.

The application area of feature extraction on binary datasets addresses many problem areas, such as association rule mining, itemsets used for market basket analysis, discovery of regulation patterns in DNA microarray experiments, etc. For simplicity sake we used the well-known bars problem (see e.g. [2]), where we try to isolate separate horizontal and vertical bars from images containing their combinations.

In this paper we will concentrate on the last category – other feature extraction methods which use known dimension reduction techniques and clustering for automatic feature extraction.

In this paper we will use the bars collection as a benchmark collection. The bars problem (and its variations) is a benchmark task for the learning of independent image features (Föildiák [2]; Spratling [6];). In the standard version of the bars problem, as defined by Föildiák [2], training data consists of 8 by 8 pixel images in which each of the 16 possible (one-pixel wide) horizontal and vertical bars can be present with a probability of $\frac{1}{8}$. Typical examples of training images are shown in Figure 1.

One of the well-known methods of feature extraction is the *singular value decomposition* (*SVD*) which was already successfully used for automatic feature extraction.

We extended the bars problem to 3 dimensions, using planes instead of lines. The input cube contains several planes, which may or may not be parallel to x, y and z axes.

The straightforward approach to image indexing is to transform the 2D images into a single vector. This is often done by concatenating all the rows of an image into a single image vector [7] (although a more sophisticated method can be used). We will use similar approach for 3D bars and classic SVD, combining two dimensions into one,

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 49–60, ISBN 978-80-248-1746-0. VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

so that we can compare the original and reconstructed matrices based on the visual impact and Frobenius norm.

The rest of this paper is organized as follows. The second section explains dimension reduction methods, which were used for classic 2D bars problem. The third section mentions CubeSVD, which was originaly used for the 3D bars problem. Then in the fourth section we describe experimental results and finally in the section five we made some conclusions.

2 Dimension Reduction

We used four promising methods of dimension reduction for our comparison – *Singular Value Decomposition* (*SVD*), *Semi-Discrete Decomposition* (*SDD*) and *Non-negative Matrix Factorization* (*NMF*). All of them are briefly described bellow.

2.1 Singular Value Decomposition

SVD [1] is an algebraic extension of classical vector model. It is similar to the PCA method, which was originally used for the generation of eigenfaces in image retrieval. Informally, SVD discovers significant properties and represents the images as linear combinations of the base vectors. Moreover, the base vectors are ordered according to their significance for the reconstructed image, which allows us to consider only the first k base vectors as important (the remaining ones are interpreted as "noise" and discarded). Furthermore, SVD is often referred to as more successful in recall when compared to querying whole image vectors [1].

Formally, we decompose the matrix of images A by *singular value decomposition* (*SVD*), calculating singular values and singular vectors of A.

We have matrix A, which is an $n \times m$ rank-r matrix (where $m \geq n$ without loss of generality) and values $\sigma_1, \ldots, \sigma_r$ are calculated from eigenvalues of matrix AA^T as $\sigma_i = \sqrt{\lambda_i}$. Based on them, we can calculate column-orthonormal matrices $U = (u_1, \ldots, u_n)$ and $V = (v_1, \ldots, v_n)$, where $U^T U = I_n$ a $V^T V = I_m$, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, where $\sigma_i > 0$ for $i \leq r, \sigma_i \geq \sigma_{i+1}$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$.
The decomposition

$$A = U\Sigma V^T \tag{1}$$

is called *singular decomposition* of matrix A and the numbers $\sigma_1, \ldots, \sigma_r$ are *singular values* of the matrix A. Columns of U (or V) are called *left* (or *right*) singular vectors of matrix A.

Now we have a decomposition of the original matrix of images A. We get r nonzero singular numbers, where r is the rank of the original matrix A. Because the singular values usually fall quickly, we can take only k greatest singular values with the corresponding singular vector coordinates and create a *k-reduced singular decomposition* of A.

Let us have k $(0 < k < r)$ and singular value decomposition of A

$$A = U\Sigma V^T \approx A_k = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix} \tag{2}$$

We call $A_k = U_k \Sigma_k V_k^T$ a k-reduced singular value decomposition (rank-k SVD). Instead of the A_k matrix, a matrix of image vectors in reduced space $D_k = \Sigma_k V_k^T$ is used in SVD as the representation of image collection. The image vectors (columns in D_k) are now represented as points in k-dimensional space (the *feature-space*). represent the matrices U_k, Σ_k, V_k^T.

2.2 Semi-discrete Decomposition

The *SDD* is one of other LSI methods, proposed recently for text retrieval in [3]. As mentioned earlier, the rank-k SVD method (called *truncated SVD* by authors of semi-discrete decomposition) produces dense matrices U and V, so the resulting required storage may be even larger than the one needed by the original term-by-document matrix A.

To improve the required storage size and query time, the semi-discrete decomposition was defined as

$$A \approx A_k = X_k D_k Y_k^T, \tag{3}$$

where each coordinate of the matrices X_k and Y_k is constrained to have entries from the set $\varphi = \{-1, 0, 1\}$, and the matrix D_k is a diagonal matrix with positive coordinates.

The SDD does not reproduce A exactly, even if $k = n$, but it uses very little storage with respect to the observed accuracy of the approximation. A rank-k SDD (although from mathematical standpoint it is a sum on rank-1 matrices) requires the storage of $k(m + n)$ values from the set $\{-1, 0, 1\}$ and k scalars. The scalars need to be only single precision because the algorithm is self-correcting. The SDD approximation is formed iteratively.

The optimal choice of the triplets (x_i, d_i, y_i) for given k can be determined using greedy algorithm, based on the residual $R_k = A - A_{k-1}$ (where A_0 is a zero matrix).

2.3 Non-negative Matrix Factorization

The *NMF* [5] method calculates an approximation of the matrix A as a product of two matrices, W and H. The matrices are usually pre-filled with random values (or H is initialized to zero and W is randomly generated). During the calculation the values in W and H stay positive. The approximation of matrix A, matrix A_k, can be calculated as $A_k = WH$.

The original *NMF* method tries to minimize the Frobenius norm of the difference between A and A'_k using $\min\limits_{W,H} ||V - WH||_F^2$ as the criterion in the minimization problem.

Recently, a new method was proposed in [6], where the constrained least squares problem $\min\limits_{H_j}\{||V_j - WH_j||^2 - \lambda||H_j||_2^2\}$ is the criterion in the minimization problem. This approach is yields better results for sparse matrices.

Unlike in *SVD*, the base vectors are not ordered from the most general one and we have to calculate the decomposition for each value of k separately.

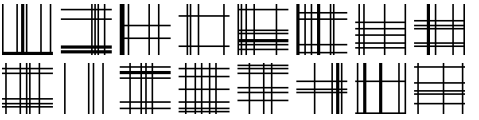


Fig. 1. Some more complex 2D bars

3 3D Theory

A tensor is a higher order generalization of a vector. Vector is a first order tensor and a matrix is a second order tensor. The order of a tensor $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is N. Elements of $\mathcal{A}$ is denoted as $a_{i_1 \ldots i_n \ldots i_N}$ where $1 \leq i_n \leq I_N$. Two basic operations are for calculation of CubeSVD: the unfolding of a tensor $\mathcal{A}\left(\mathcal{A}_{(n)}\right)$ and the mode-n product of a tensor $\mathcal{A}$ and matrix $\mathcal{M}\left(\mathcal{A} \times_{(n)} \mathcal{M}\right)$.

The operation unfolding unfolds the tensor $\mathcal{A}$ into matrix $\mathcal{A}_{(n)}$ along order N. Each column of tensor $\mathcal{A}_{(n)}$ is composed of $a_{i_1 \ldots i_n \ldots i_N}$ where i_n varies and the order indices are fixed. The operations unfolding are illustrated in Figure 2 for third order tensor. See [4] for details on operation unfolding of a tensor $\mathcal{A}$.

The n-mode product of a tensor $\mathcal{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $M \in R^{J_n \times I_n}$ is an $I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$-tensor of which the entries are given by

$$(\mathcal{A} \times_n M)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} m_{j_n i_n}$$

See [4] for details on operation mode-n product of a tensor $\mathcal{A}$ and matrix $\mathcal{M}$.

Matrix SVD can be rewritten as $A = \Sigma \times_1 V^{(1)} \times_2 V^{(2)}$ in terms of n-mode products. CubeSVD is a generalization of SVD and was described in [4]. Tensor $\mathcal{A}$ can be written as the n-mode product [4]

$$\mathcal{A} = \mathcal{S} \times_1 \mathcal{V}_1 \times_2 \mathcal{V}_2 \times_3 \cdots \times_N \mathcal{V}_N$$

as illustrated in Figure 3 for $N = 3$.

$\mathcal{S}$ is called core tensor. S is in general a full tensor, instead of being pseudodiagonal (this would mean that nonzero elements could only occur when the indices $i_1 = i_2 = \cdots = i_N$). $\mathcal{S}$ has the property of all-orthogonality [4]. $\mathcal{V}_\backslash$ contains the orthonormal vectors. They called n-mode singular vectors. The Frobenius-norms $\|\mathcal{S}_{in=i}\|$ are n-mode singular values of A. Their order is

$$\|\mathcal{S}_{in=1}\| \geq \|\mathcal{S}_{in=2}\| \geq \cdots \geq \|\mathcal{S}_{in=In}\| \geq 0$$

4 Experimental Results - 2D Bars

For testing of above mentioned methods, we used generic collection of 1600 32×32 black-and-white images containing different combinations of horizontal and vertical lines (bars). The probabilities of bars to occur in images were the same and equal to

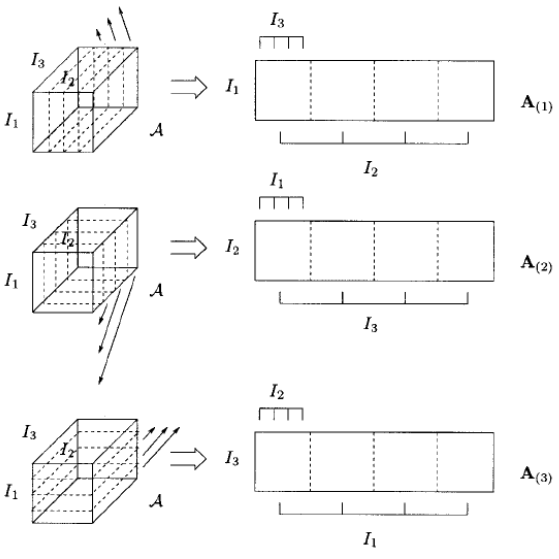


Fig. 2. Unfolding of third order of a tensor $\mathcal{A}$

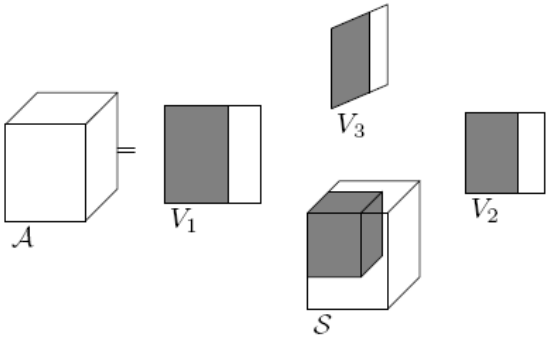


Fig. 3. Order-3 Singular Value Decomposition $\mathcal{A}$

$10/64$, i.e. images contain 10 bars in average. An example of several images from generated collection is shown in Figure 4.

Many of tested methods were able to generate a set of base images or factors, which should ideally record all possible bar positions. However, not all methods were truly successful in this.

With SVD, we obtain classic singular vectors, the most general being among the first. The first few are shown in Figure 5. We can se, that the bars are not separated and different shades of gray appear.

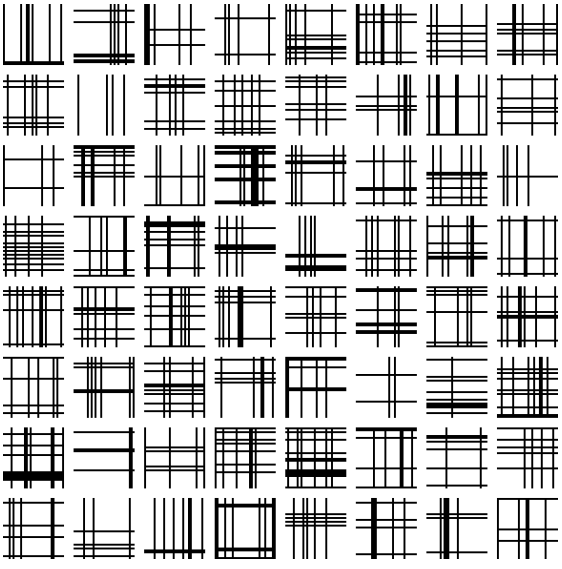


Fig. 4. Some generated images from bars collection

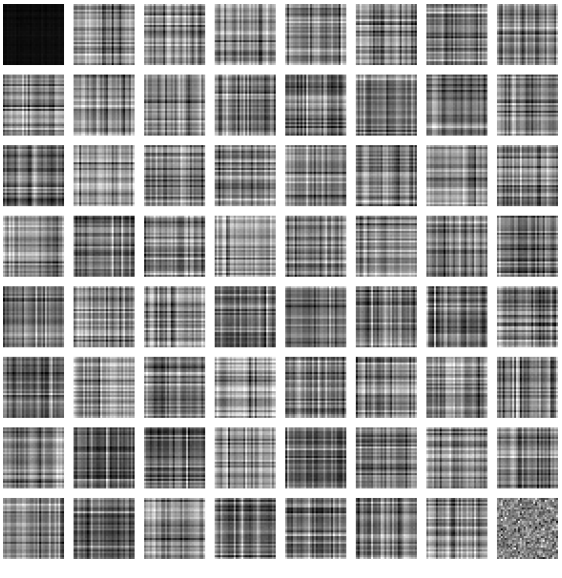


Fig. 5. First 64 base images – SVD method

The NMF methods yield different results. The original NMF method, based on the adjustment of random matrices W and H provides hardly-recognizable images even for

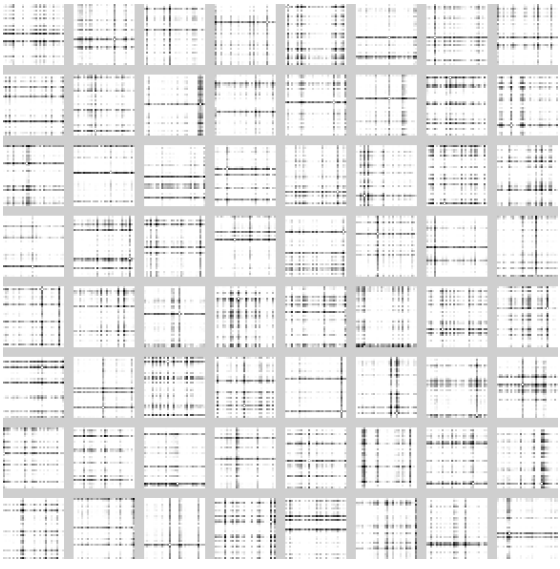


Fig. 6. First 64 factors – original NMF method

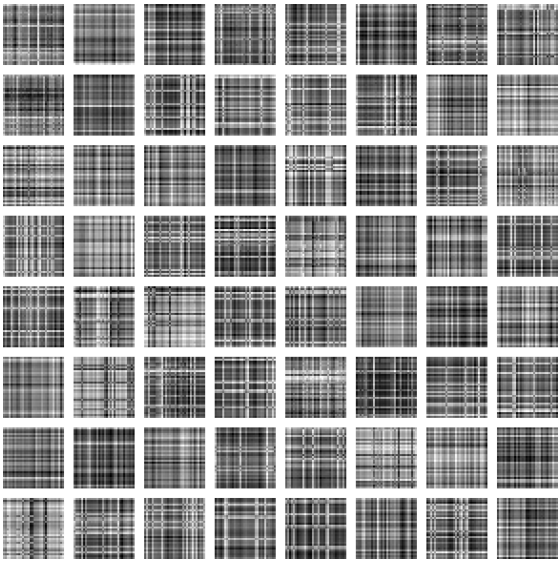


Fig. 7. First 64 factors for GD-CLS NMF method (0.01)

$k = 100$ and 1000 iterations (we used 100 iterations for other experiments). Moreover, these base images still contain significant salt and pepper noise and have a bad contrast.

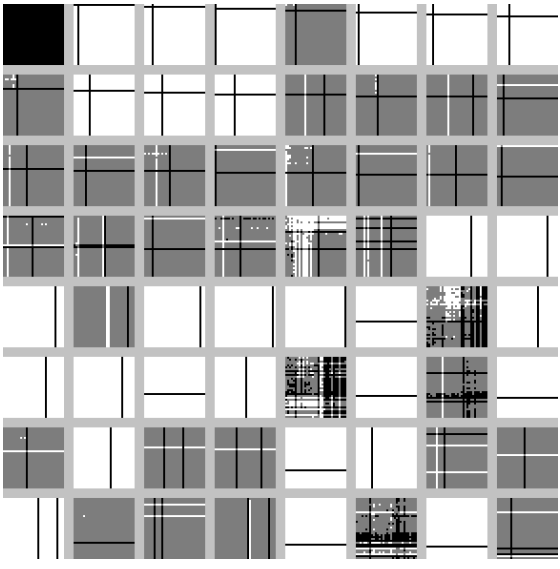


Fig. 8. First 64 base vectors – SDD method

The factors are shown in Figure 6. We must also note, that the NMF decomposition will yield slightly different results each time it is run, because the matrix(es) are pre-filled with random values.

The SDD method differs slightly from previous methods, since each factor contains only values $\{-1, 0, 1\}$. Gray in the factors shown in Figure 8 represents 0; -1 and 1 are represented with black and white respectively.

The base vectors in Figure 8 can be divided into three categories:

1. Base vectors containing only one bar.
2. Base vectors containing one horizontal and one vertical bar.
3. Other base vectors, containing several bars and in some cases even noise.

5 Experimental result - 3D Bars

For testing of CubeSVD method, we use several collections of $8 \times 8 \times 8$ 3-dimensional cubes. We create 2 types of test collections. The first type contains 2 collections with 15 cubes each which were used for local feature extraction (each cube was decomposed separately). The first collection contains cubes which are crossed by 2 perpendicular planes (Figure 9a). The second collection contains cubes crossed by 3 planes - 2 perpendicular and 1 skewed (Figure 9b). The resulting number of singular values was between 1 and 8 (full CubeSVD).

The second type contains 4 collections with 1000 cubes each which were used for collection-based feature extraction. The first collection contains cubes with one skewed

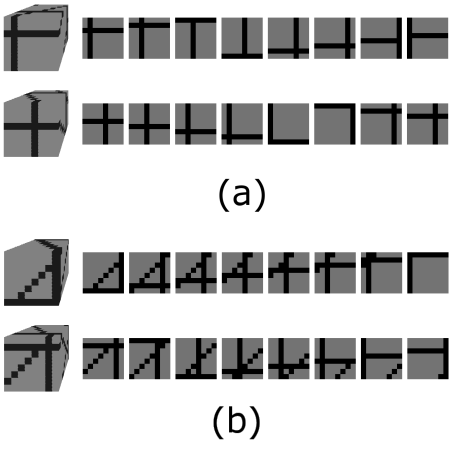


Fig. 9. Collections for local features extraction. (a) is example from first collection, (b) is example from second collection

plane, second collection contains cubes with 2 skewed planes and so on. Example cubes for each collection is depicted in Figure 10.

As a measure for comparing similarity between original and reduced cubes we used the Frobenius norm (without calculating the square root)

$$F^2(O, R) = \sum_i \sum_j \sum_k (O[i, j, k] - R[i, j, k])^2$$

5.1 Local feature extraction

In the first experiment we applied CubeSVD and SVD algorithm on the first two collections for extracting local features. Results for the first collection with 2 perpendicular planes are depicted in Figure 11 for 6 singular values and are depicted on Figure 12 for 2 singular values. Values of Frobenius norm are shown in tables 1 and 2.

It can be seen CubeSVD extracts all of the original bars in Figure 11, while the SVD with the same rank ignored one of the bars, while reconstructing the other two more sharply. This is even more visible in Figure 12, where all bars in CubeSVD are slightly recorded, but classic SVD reconstructs one of the bars perfectly, adding noise to other areas. We see, that this behavior leads to lower Frobenius norm in Tables 1 and 2 for th classic SVD, which satisfies the condition that the value for SVD should be minimal for given rank k.

6 Conclusion

Since the CubeSVD method provided only one singular value for planes parallel to the axes, which was to be expected, the experiments were done on planes both perpendicular and skewed.

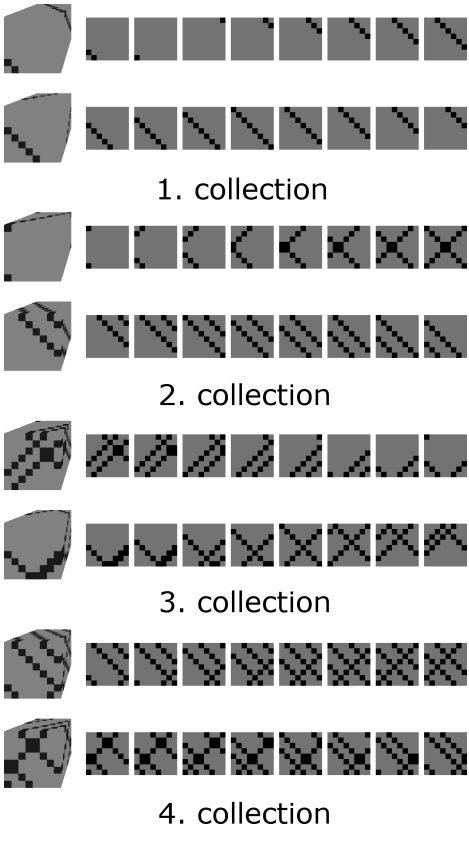


Fig. 10. Collections for collection-based features extraction

It seems, that the original SVD performed better than CubeSVD, based on the Frobenius norm, but the visual inspection of reduced tensors shows the reason – whilst the horizontal bars were reconstructed nearly perfectly, the vertical ones deteriorated more quickly. On the other hand, the CubeSVD tried to minimize the overall error.

We are currently studying the collection-based feature extraction of 3D bars, where the number of singular values ranges from 1 to 512 for $8 \times 8 \times 8$ cubes, compared to 8 singular values for high-order SVD. The classic SVD results for 1 to 8 singular values mentioned in previous section are not satisfactory.

We are currently extending our CubeSVD implementation to support tensors of 4 and mode dimensions and preparing to test the High-order SDD and NMF methods against their 2D counterparts.

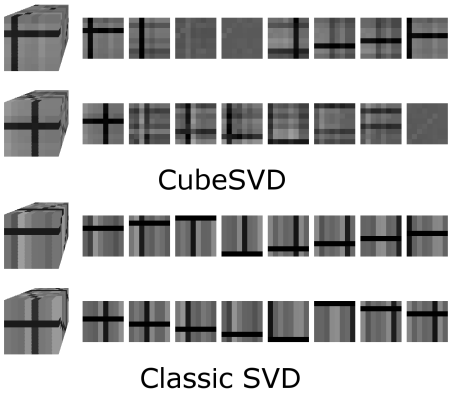


Fig. 11. Results for first collection and 6 singular values

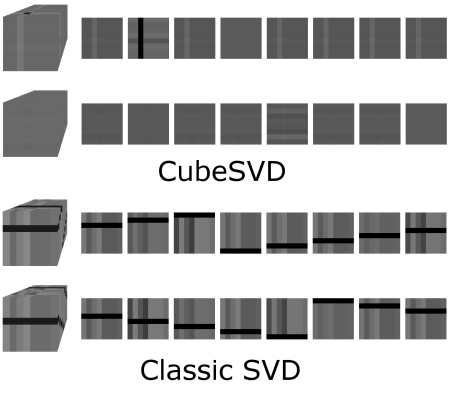


Fig. 12. Results for first collection and 2 singular values

References

1. M. Berry, S. Dumais, and T. Letsche. Computational Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, San Diego, California, USA, 1995.
2. P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological cybernetics, 64:22*, pages 165–170, 1990.
3. T. G. Kolda and D. P. O'Leary. Computation and uses of the semidiscrete matrix decomposition. In *ACM Transactions on Information Processing*, 2000.
4. L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.
5. F. Shahnaz, M. Berry, P. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. *Journal on Information Processing and Management*, 42:373–386, 2006.

Table 1. Cumulative Frobenius norm for first collection

Method	CubeSVD	SVD
K=1	1484.31	735
K=2	1356.21	630
K=3	1304.02	525
K=4	1169.81	420
K=5	976.62	315
K=6	708.54	210
K=7	379.29	105
K=8	0	0

Table 2. Cumulative Frobenius norm for second collection

Method	CubeSVD	SVD
K=1	1768.84	1045.91
K=2	1587.14	854.01
K=3	1489.88	677.87
K=4	1340.44	514.32
K=5	1091.12	364.01
K=6	813.81	228.41
K=7	438.31	102.49
K=8	0	0

6. M. W. Spratling. Learning Image Components for Object Recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.
7. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

Developing Genetic Algorithms for Boolean Matrix Factorization

Václav Snášel, Jan Platoš, Pavel Krömer

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{vaclav.snasel, jan.platos, pavel.kromer.fei}@vsb.cz

Abstract. Matrix factorization or factor analysis is an important task helpful in the analysis of high dimensional real world data. There are several well known methods and algorithms for factorization of real data but many application areas including information retrieval, pattern recognition and data mining require processing of binary rather than real data. Unfortunately, the methods used for real matrix factorization fail in the latter case. In this paper we introduce background and initial version of Genetic Algorithm for binary matrix factorization.

Keywords: binary matrix factorization, genetic algorithms

1 Introduction

Many applications in computer and system science involve analysis of large scale and often high dimensional data. When dealing with such extensive information collections, it is usually very computationally expensive to perform some operations on the raw form of the data. Therefore, suitable methods approximating the data in lower dimensions or with lower rank are needed. In the following, we focus on the factorization of two-dimensional binary data (matrices, second order tensors).

The paper is structured as follows: first, a brief introduction to matrix factorization is given. In the following section, the basics of Evolutionary and Genetic Algorithms are presented. The rest of the paper brings description of Genetic Binary Matrix Factorization and summarizes performed computer experiments and conclusions drawn from them.

2 Matrix Factorization

Matrix factorization (or matrix decomposition) is an important task in data analysis and processing. A matrix factorization is the right-side matrix product in

$$A \approx F_1 \cdot F_2 \cdot \ldots \cdot F_k \tag{1}$$

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 61–70, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

for the matrix A. The number of factor matrices depends usually on the requirements of given application area. Most often, $k = 2$ or $k = 3$. There are several matrix decomposition methods reducing data dimensions and simultaneously revealing structures hidden in the data. Such methods include Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF), which is our subject of interest in this research.

2.1 Non-negative matrix factorization

Non-negative matrix factorization (NMF) [1, 7] is recently very popular unsupervised learning algorithm for efficient factorization of real matrices implementing the non-negativity constraint. NMF approximates real $m\times\ n$ matrix A as a product of two non-negative matrices W and H of the dimensions $m \times r$ and $r \times n$ respectively. Moreover, it applies that $r << m$ and $r << n$.

$$A \approx W \cdot H \tag{2}$$

There are several algorithms for NMF computation based on iterative minimization of given cost function [7]. The original NMF algorithm involved minimization of the Frobenius norm [10] defined by formulae (3).

$$\|A - WH\|_F^2 = \sum_{ij} |A_{ij} - (WH)_{ij}|^2 \tag{3}$$

Other investigated cost measures include square of the Euclidean distance between V and its approximation (4) or Kullback-Leibler divergence D (5). For every cost function, there are update rules (multiplicative or additive) applied iteratively in order to reduce the distance between original matrix V and its model [7, 10].

$$\|A - WH\|^2 = \sum_{ij} (A_{ij} - (WH)_{ij}) \tag{4}$$

$$D(A \parallel WH) = \sum_{ij} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}) \tag{5}$$

Promising recent NMF algorithms are based on Gradient Descent Methods (GDM) or, extending the GDM, on Alternating Least Square computation [1, 10]. NMF was reported to give good results in extracting features or concepts from processed data. Unfortunately, the common NMF algorithms excelling in NMF computation for real valued matrices are unsuitable for efficient factorization of binary matrices.

2.2 Boolean matrix factorization

Boolean matrix factorization (BMF) or Boolean factor analysis is the factorization of data sets in binary $(1, 0)$ alphabet based on Boolean algebra. Boolean

factor analysis is extremely important in computer applications since the natural data representation for computerized processing is binary. Binary factorization finds its application in data mining, information retrieval, pattern recognition, image processing or data compression [5].

The BMF can be defined in a similar manner as NMF [11, 12, 14]. Consider binary[1] matrix A of the dimension $m \times n$ as a Boolean product of two binary matrices W and H of the dimensions $m \times r$ and $r \times n$ respectively. Let r be a subject to $r << m$ and $r << n$. Then, BMF is searching for best W and Hthat approximate A:

$$A \approx W \otimes H \tag{6}$$

where $\otimes$ stands for Boolean matrix multiplication.

The appeal of BMF lies in the fact that computerized data are binary in its essence and BMF is intensively investigated.

Keprt [5, 6] introduced BMF algorithms based on formal concepts and blind search. Meeds [8] et al. presented BMF model for factorization of dyadic data, however, Meeds' decomposition features one non-binary (integer) factor.

3 Evolutionary Algorithms

Evolutionary algorithms (EA) are family of iterative stochastic search and optimization methods based on mimicking successful optimization strategies observed in nature [2–4, 9]. The essence of EAs lies in the emulation of Darwinian evolution utilizing the concepts of Mendelian inheritance for the use in computer science and applications [2]. Together with fuzzy sets, neural networks and fractals, evolutionary algorithms are among the fundamental members of the class of soft computing methods.

EA operate with population (also known as pool) of artificial individuals (referred often as items or chromosomes) encoding possible problem solutions. Encoded individuals are evaluated using objective function which assigns a fitness value to each individual. Fitness value represents the quality (ranking) of each individual as solution of given problem. Competing individuals search the problem domain towards optimal solution [4].

For the purpose of EAs, a proper encoding representing solutions of given problem as encoded chromosomes suitable for evolutionary search process, is necessary. Finding proper encoding is non-trivial problem dependent task affecting the performance and results of evolutionary search while solving given problem. The solutions might be encoded into binary strings, real vectors or more complex, often tree-like, hierarchical structures, depending on the needs of particular application.

The iterative phase of evolutionary search process starts with an initial population of individuals that can be generated randomly or seeded with potentially good solutions. Artificial evolution consists of iterative application of genetic

[1] Matrix A is binary iff $\forall ij : [a]_{ij} = 0 \vee [a]_{ij} = 1$

operators (selection, crossover, mutation), introducing to the algorithm evolutionary principles such as inheritance, survival of the fittest and random perturbations. Current population of problem solutions is modified with the aim to form new and hopefully better population to be used in next generation. Iterative evolution of problem solutions ends after satisfying specified termination criteria and especially the criterion of finding optimal solution. After terminating the search process, evolution winner is decoded and presented as the most optimal solution found.

EAs are successful general adaptable concept with good results in many areas. The class of evolutionary techniques consists of more particular algorithms having numerous variants, forged and tuned for specific problem domains. The family of evolutionary algorithms consists of genetic algorithms, genetic programming, evolutionary strategies and evolutionary programming.

3.1 Genetic algorithms

Genetic algorithms (GA) introduced by John Holland and extended by David Goldberg are wide applied and highly successful EA variant. Basic workflow of originally proposed standard generational GA is:

I. Define objective function
II. Encode initial population of possible solutions as fixed length binary strings and evaluate chromosomes in initial population using objective function
III. Create new population (evolutionary search for better solutions):
    a. Select suitable chromosomes for reproduction (parents)
    b. Apply crossover operator on parents with respect to crossover probability to produce new chromosomes (offspring)
    c. Apply mutation operator on offspring chromosomes with respect to mutation probability. Add newly constituted chromosomes to new population
    d. Until the size of new population is smaller than size of current population go back to a.
    e. Replace current population by new population
IV. Evaluate current population using objective function
V. Check termination criteria; if not satisfied go back to III.

There are variants of standard generational GA. The differences are mostly in particular selection, crossover, mutation and replacement strategy [4].

4 Genetic Binary Matrix Factorization

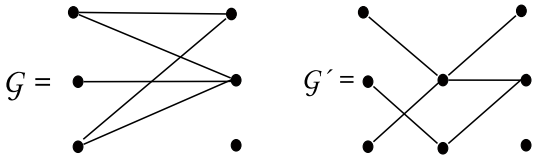
In this paper, we to propose a Genetic Algorithm for Binary Matrix Factorization (Genetic BMF - GBMF). For that, we first analzye the factors that are to be found by the algorithm and define an algorithm suggesting initial values of the factors.

4.1 Binary factors

The factors W and H found by NMF algorithm by Lee and Seung can be straightforwardly interpreted. Columns of W are basis vectors of column space of A and columns of H are weights associated with the base vectors. In order to find out interpretation of the matrix factorization task for GA which might be different, consider a graph-like representation of a matrix:

$$A = \begin{pmatrix} 1\,1\,0 \\ 0\,1\,0 \\ 1\,1\,0 \end{pmatrix} \equiv \mathcal{G} \approx WH = \begin{pmatrix} 1\,0 \\ 0\,1 \\ 0\,0 \end{pmatrix} \begin{pmatrix} 1\,1\,0 \\ 0\,1\,0 \end{pmatrix} \equiv \mathcal{G}' \tag{7}$$

Then, the factorization can be seen as a task of finding tripartite graph $\mathcal{G}'$ that will exclusively preserve the arcs between the pairs of 'edge' vertices from $\mathcal{G}$ in the form of a two step long paths through a 'middle layer'. Intuitively, the number of vertices in middle layer corresponds to r in NMF. The graphs $\mathcal{G}$ and $\mathcal{G}'$ from 7 can be depicted as follows:



When adopting this notion of Boolean matrix factorization, the interpretation of factors W and H slightly differs from the interpretation of NMF factors. The rows of H are base vectors of row space of A and rows of W are associated weights.

4.2 Constructive algorithm for suggesting base vectors of boolean matrix row space (CAS)

In order to provide the genetic algorithm with better than random initial population, a constructive algorithm for suggesting base vectors of matrix row space is defined:

I. Compute the cardinality (number of 1 elements) of the rows of A; divide rows of A to classes A^C per cardinality. Let W and H be 'empty'. Let $k = 0$.
II. Randomly pick up row A_i^C from the row class with lowest cardinality. Let $j = 0$.
III. For base row h_j in H :
   a. Check whether h_j is covered[2] by A_i^C. If not, attach A_i^C to H as h_k, and set $W[i,k] = 1$. Increase k and go back to II. In case h_j is covered by A_i^C, go to II.
   b. Let $W[i,j] = 1$. Let $A_i^C = A_i^C$ - w_j.
   c. If A_i^C is not zero vector, go back to II.

The rows of the matrix A can be then constructed using linear combinations of the rows of H.

[2] i.e. $h_j - A_i^C = o$ where $o = (0, 0, \ldots, 0)$ is zero vector.

4.3 Genetic algorithm for binary matrix factorization

We propose a genetic algorithm for BMF. It will exploit the initial factors constructed using the algorithm introduced in previous section. The objective function will be Hamming distance between reconstructed and original matrix. Crossover will aim to modify weights factor (matrix W) and mutation will primarily aim to alter basis vectors (matrix H). The algorithm can be summarized as follows:

I. Create initial population of N (WH) chromosomes and evaluate
II. Evolve population
   a. Select suitable chromosomes for reproduction (parents)
   b. Apply crossover on matrix W of selected parents.
   c. With very small probability, mutate W
   d. Mutate H
   e. Migrate offspring chromosomes to population
III. Evaluate current population using objective function
IV. Check termination criteria; if not satisfied go back to II.

The evaluation of chromosomes in population is implemented as comparison of original matrix V to the product of W and H. The fitness function is based on Euclidean distance between V and WH:

$$f = \frac{1}{\sqrt{\sum_i \sum_j (V[i,j] - WH[i,j])^2}} \tag{8}$$

The termination criteria were based on specified threshold defining minimum acceptance of evolved solution and maximum number of generations processed. The maximu number of generations was se to 1000 and the minimum acceptance 0.3.

In this way, the algorithm explores different combinations of base vectors (via crossover) and simultaneously adjust the base vector suggestions. Evolutionary principles will be applied and the factor interpretation maintained.

5 Experimental Algorithm Evaluation

This section provides summary on computer experiments conducted in order to verify proposed algorithms.

5.1 Evaluation of CAS algorithm

In 4.2 was defined a constructiove algorithm to suggest base vectors for initial matrix factors. We have compared the average error obtained after using random initial factors and initial factors suggested by CAS for several random square matrices. For every matrix dimension n, we have computed the average error (i.e. Hamming distance) between original matrix and its reconstruction. The

Table 1. Comparing average error of random initial factors and CAS suggested initial factors .

N	Random	CAS
32	513, 82	216
64	2047, 49	915
128	8195, 1	3809
256	32775, 3	15492
512	131070	63268

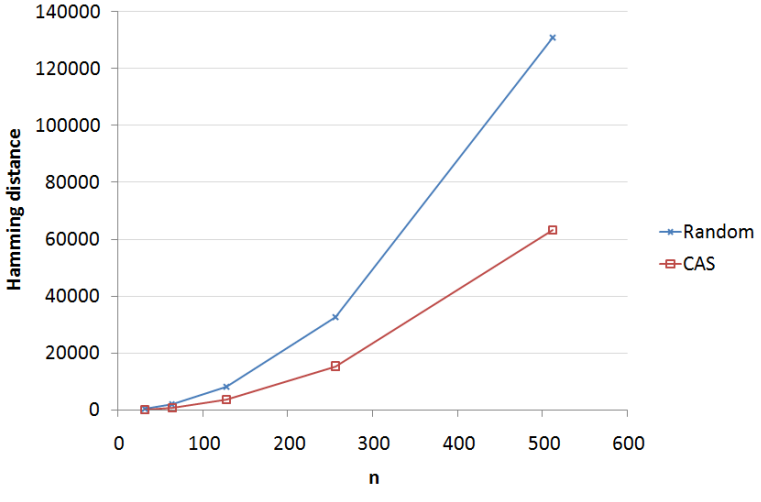


Fig. 1. The comparison of Random initial error and CAS initial error.

dimension was reduced to $\frac{n}{2}$ and the experiment was repeated 100 times for each n.

Obviously, the error produced by random initial factors is aproximately twice as big as the error obtained when using CAS initial factors.

5.2 Evaluation of GBMF algorithm

GBMF was implemented and run for some testing binary matrices. Black and white images were chosen as representation of input and output binary matrices for the ease of visual interpretatin of the results.

In all cases, GBMF was run with the following parameters: a population of 50 prospective factors, probability of crossover 0.9 and probability of mutation 0.2. GBMF was executed for 5000 generations.

The algorithm was first tested on a set of bar images. Testing bar images contained white background and black bars – vertical or horizontal lines gener-

ated with certain probability (0.2 for horizontal bar and 0.3 for vertical bar). The dimension of testing images was 15×25 pixels.

Initially, a small collection of 6 bar images (shown in figure 5.2) was processed by the algorithm to reduce its dimension to 3.

Fig. 2. Input bar images.

Fig. 3. Reconstructed bar images.

Fig. 4. Obtained base images.

Next, the progam was used to process in the same way a collection of 25 face images taken from the facial expression collection. The images were transformed from grayscale original to black and white (so they could be interpreted as binary matrices). The dimension of testing images was 19×19 pixels.

In both cases delivered the algorithm a set of images (i.e. matrices) which clearly share some elements with the original input. Albeit some base images were obtained, they did not contain distinguished features as for instance when using NMF for non-binary matrices. Moreover, the black and white images used for algorithm evaluation are not typical real world binary matrices. Also both binary and real-valued pseudorandom matrices do not contain features significant for matrices describing real world phenomena (i.e. in economics, physics etc.).

6 Conclusions and Future Work

In this paper, we have introduced initial version of a genetic algorithm for binary matrix factorization. In order to define Genetic Algorithm oriented approach to

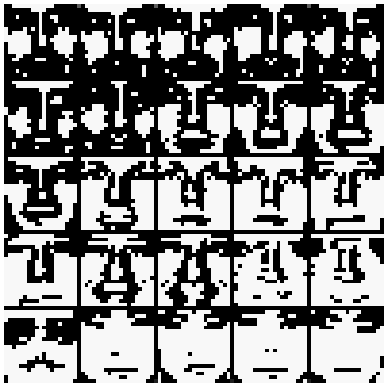


Fig. 5. Input face images.

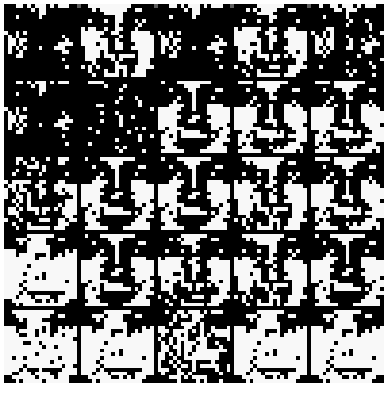


Fig. 6. Reconstructed face images.

Fig. 7. Obtained base faces.

BMF, an interpretation of binary factors was presented. Next, an algorithm for lossless BMF was used to create initial binary factors. Genetic BMF was defined, implemented and applied on first sample problems. In the future, we will focus on tuning the GBMF mplementation and evaluation of the algorithm on real world binary matrices. Some matrix properties such as sparsity might be exploited for algorithm modification.

References

1. Michael W. Berry, Murray Browne, Amy N. Langville, Paul V. Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization.
2. Ulrich Bodenhofer. *Genetic Algorithms: Theory and Applications*. Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter 2003/2004.
3. Mehrdad Dianati, Insop Song, and Mark Treiber. An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, N2L 3G1, Canada, July 2002.
4. Gareth Jones. Genetic and evolutionary algorithms. In Paul von Rague, editor, *Encyclopedia of Computational Chemistry*. John Wiley and Sons, 1998.
5. Ales Keprt. Using blind search and formal concepts for binary factor analysis. In Václav Snásel, Jaroslav Pokorný, and Karel Richta, editors, *DATESO*, volume 98 of *CEUR Workshop Proceedings*, pages 128–140. CEUR-WS.org, 2004.
6. Ales Keprt and Václav Snásel. Binary factor analysis with help of formal concepts. In Václav Snásel and Radim Belohlávek, editors, *CLA*, volume 110 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
7. Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.
8. Edward Meeds, Zoubin Ghahramani, Radford M. Neal, and Sam T. Roweis. Modeling dyadic data with binary latent factors. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 977–984. MIT Press, Cambridge, MA, 2007.
9. Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
10. Farial Shahnaz, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Inf. Process. Manage.*, 42(2):373–386, 2006.
11. Moravec, P., Snášel, V., Frolov, A. A., Húsek, D., Řezanková, H., Polyakov, P.: Image Analysis by Methods of Dimension Reduction. IEEE CISIM 2007: Elk, Poland, Pages 272–277
12. Snášel, V., Húsek, D., Frolov, A. A., Řezanková, H., Moravec, P., Polyakov P.: Bars Problem Solving - New Neural Network Method and Comparison. MICAI 2007: LNCS 4827 Springer 2007, Pages 671–682
13. Spellman, P.T., Sherlock, G., Zhang, M.Q., Anders, V.I.K., Eisen, M.B., Brown, P., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. In: Molecular Biology of the Cell 9. (1998) Pages 3273–3297
14. Zhang, Z., Tao Li, T., Ding, C., Zhang, Xiang-Sun: Binary Matrix Factorization with Applications. In Proceedings of 2007 IEEE International Conference on Data Mining (ICDM 2007),

Towards Cost-based Optimizations of Twig Content-based Queries

Michal Krátký and Radim Bača

Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
michal.kratky@vsb.cz

Extended Abstract

In recent years, many approaches to indexing XML data have appeared. These approaches attempt to process XML queries efficiently and sufficient query plans are built for this purpose. Some effort has been expended in the optimization of XML query processing [20].

There are not many works that take cost-based query optimizations into account. In work [20], we find some cost-based considerations, however, they work only with one type of structural join and one type of underlying index. There are works depicted two types of query processing as well [10, 17]. The first type applies an element-based index, the second type applies a navigation in a persistent DOM-like structure. In our work, we propose usage of two path-based indices that provide significant potential for a query optimization based on a cost-based join selection.

We can identify some classes of approaches for efficient processing of XML queries. The first class includes approaches based on *shredding* [18] (storing an XML document in many relations, where each element name has its own relation). These approaches work well only on specific documents with a defined XML schema. The second class of approaches [15, 21, 1] provides an *element-based* decomposition of an XML document. These methods usually work with a labeling scheme and they differ mainly in various join algorithms.

We identify two main types of join algorithms. The first join type works with sorted node sets merged by a type of holistic join [3, 4]. This kind of structural join is optimal when a small number of nodes is rejected during the structural join. In this case, nodes are retrieved in a very efficient way with a small number of I/O. In our article, this kind of join operation is called a *merge join*. Another type of join algorithm is based on context nodes [9], where each location step is processed using context nodes from the previous step. We say that this approach utilizes a previously processed location step and uses the nodes for an efficient search of nodes in the next step. If there is a large number of rejected nodes in the join operation, this kind of join is efficient. In our article, this kind of join operation is called a *progressive join*.

There are approaches that decompose XML documents according to the node path, as opposed to the node name. Handling the paths during a query processing

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 71–73, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

provides some advantages. If the `a/b/c//e/f/g` simple-path query is considered, then the query is evaluated by five structural joins. The evaluation may be very inefficient compared to *path-based* approaches [16, 19, 6, 14, 13, 11] and summarizing approaches [8, 7], even when an additional optimization of structural joins is used [4, 3, 12]. The path-based approaches perform these queries by finding matched labelled paths – a relatively simple task – and then finding relevant nodes with a single index search [16]. Since there is a significantly lower number of labelled paths than nodes in an XML document, the search can be performed very quickly. Supposing there is a set of matched labelled paths, we finish the simple-path query process by finding nodes corresponding to those labelled paths in an inverted list. In work [5], we find a comparison of different decomposition approaches and a labelled-path approach (Prefix-Path streaming in this case) has good experimental results.

In the case of path-based approaches, we can observe the same issue as in the case of element-based approaches. When we join two path results, we can perform it with two different types of joins. One join is based on an inverted index and the second one utilizes the previous query path result.

Chen et al. in [6] compares two path-based approaches to processing twig queries. Whereas the first index, *ROOTPATHS* index, is able to process twig queries only with the merge join, the second index, *DATAPATHS* index, is able to process a query path by utilizing previous query path results. Consequently, DATAPATHS index applies a progressive join algorithm. These indices can outperform each other depending on the query. However, there is no general proposal that can help to decide that the index should be used to achieve the best query evaluation performance. In works [13, 14], we have introduced an index to provide these join operations.

In work [2], we introduce a simple, cost-based, optimization technique for a join selection during a query evaluation. This technique joins the advantages of a simple path query processing based on inverted lists and usage of previous results for a twig query processing. We show that the knowledge of the result size can help to choose a good query evaluation strategy. We utilize advantages of existing, state-of-the-art, path-based approaches, such as [16, 19, 6], to achieve an optimal query performance.

References

1. S. Al-Khalifa, H. V. Jagadish, and N. Koudas. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proceedings of International Conference on Data Engineering, ICDE 2002*. IEEE Computer Society, 2002.
2. R. Bača and M. Krátký. A Cost-based Join Selection for XML Twig Content-based Queries. In *Proceedings of the Third International Workshop on Database Technologies for Handling XML Information on the Web, DataX 2008, EDBT, Nantes, France*. Accepted, to appear in ACM DL, 2008.
3. N. Bruno, D. Srivastava, and N. Koudas. Holistic Twig Joins: Optimal XML Pattern Matching. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2002*, pages 310–321. ACM Press, 2002.

4. S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig2Stack: Bottom-up Processing of Generalized-tree-pattern Queries Over XML documents. In *Proceedings of International Conference on Very Large Databases, VLDB 2006*, pages 283–294. VLDB Endowment, 2006.
5. T. Chen, J. Lu, and T. Ling. On Boosting Holism in XML Twig Pattern Matching Using Structural Indexing Techniques. *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2005*, pages 455–466, 2005.
6. Z. Chen, G. Korn, F. Koudas, N. Shanmugasundaram, and J. Srivastava. Index Structures for Matching XML Twigs Using Relational Query Processors. In *Proceedings of ICDE 2005*, pages 1273–1273. IEEE Computer Society, 2005.
7. C.-W. Chung, J.-K. Min, and K. Shim. APEX: an Adaptive Path Index for XML Data. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2002*, pages 121–132, New York, NY, USA, 2002. ACM Press.
8. B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A Fast Index for Semistructured Data. In *Proceedings of the 27th International Conference on Very Large Databases, VLDB 2001*, pages 341–350, 2001.
9. T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach a Relational DBMS to Watch Its (Axis) Steps. In *Proceedings of the 29th, International Conference on Very Large Databases, VLDB 2003*, pages 524–535. VLDB Endowment, 2003.
10. A. Halverson and et al. Mixed Mode XML Query Processing. In *Proceedings of VLDB 2003*, pages 225–236. VLDB Endowment, 2003.
11. W. H. Hanyu Li, Mong Li Lee. A Path-Based Labeling Scheme for Efficient Structural Join. In *Proceedings of XSym 2005*, pages 34 – 48. Springer–Verlag, 2005.
12. H. Jiang, W. Wang, H. Lu, and J. Yu. Holistic twig joins on indexed XML documents. *Proceedings of VLDB 2003*, pages 273–284, 2003.
13. M. Krátký, R. Bača, and V. Snášel. Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In *Proceedings of DEXA 2007*, volume LNCS 4653/2007. Springer–Verlag, 2007.
14. M. Krátký, J. Pokorný, and V. Snášel. Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In *Current Trends in Database Technology, EDBT 2004*, volume LNCS 3268/2004. Springer–Verlag, 2004.
15. Q. Li and B. Moon. Indexing and Querying XML Data for Regular Path Expressions. In *Proceedings of VLDB 2001*, 2001.
16. T. S. M. Yoshikawa, T.Amagasa and S. Uemura. XRel: a Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases. *ACM Trans. Inter. Tech.*, 1(1):110–141, 2001.
17. N. May, M. Brantner, A. Böhm, C.-C. Kanne, and G. Moerkotte. Index vs. Navigation in XPath Evaluation. In *Proceedings of Database and XML Technologies, XSym 2006*, volume LNCS 4156/2006, pages 16–30. Springer–Verlag, 2006.
18. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proceedings of the 25th International Conference on Very Large Databases, VLDB 1999. Edinburgh, Scotland, UK*, pages 302–314. Morgan Kaufmann, 1999.
19. S. S.Prakas and S.Madria. SUCXENT: An Efficient Path-Based Approach to Store and Query XML Documents. In *Proceedings of DEXA 2004*, volume LNCS 3180/2004, pages 285–295. Springer-Verlag, 2004.
20. Y. Wu, J. M. Patel, and H. Jagadish. Structural Join Order Selection for XML Query Optimization. In *Proceedings of ICDE 2003*. IEEE CS, 2003.
21. C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of ACM SIGMOD 2001*, pages 425–436, New York, USA, 2001. ACM Press.

Vector-Oriented Retrieval in XML Data Collections

Jaroslav Pokorný

Faculty of Mathematics and Physics, Charles University,
Malostranské náměstí 25, Prague, Czech Republic
pokorny@ksi.mff.cuni.cz

Many modern applications produce and process XML data, which is queried in its both structural and textual component. This is especially useful if we consider a casual user who looks for information in web-based database systems or intranets containing XML data, like online shops, airline reservations, digital libraries catalogues or any other, and does not expect an exact answer. Many websites are built from document-centric XML documents [3]. A remarkable characteristic of such XML data collections is that they are mostly heterogeneous, i.e. they contain domain-focused data, possibly valid w.r.t. various DTDs or XML schemes. XML documents can come from various sources. These collections can be managed as XML databases [5] as well as collections, providing an approximate way for users to search their contents. To ensure such functionality, it is required to approach these collections with both database and information retrieval (IR) methods.

Current XML query languages like XPath and XQuery are applicable rather for data-centric than for document-centric XML data. Moreover, XML schemes are often necessary for their use. In other words, the languages are not longer appropriate for searching in such environments because they can not cope with the diversity of data. Hence, a research of integration of database querying and IR in context of XML is undoubtedly interesting and promising trend. Despite of the fact that a variety of systems that support such methods have been proposed, conventional IR techniques [2], e.g. vector space model, can be employed only restrictedly. The reason for it is that two types of queries should be dealt with: content-only (CO) queries, i.e. the traditional ones in IR, and content-and-structure (CAS) queries.

A number of techniques to extend the vector space model have been designed, e.g. [6], [7], [8], [9], [11], and [12]. A usual critique of the mentioned approaches is that they not sufficiently reflect the structure of XML documents. A more advanced, two-phase evaluation schema is proposed in [1]. First, a modified vector space model is employed to obtain similarity scores for the textual nodes of XML trees. Then, the scores are propagated upward in the XML-trees with a possible modification and possibly new scores of other nodes are generated.

In [13] we described a matrix model based on an extension of the vector space model for XML data. A document D in a collection of XML documents C is represented by a matrix $\boldsymbol{D}$, whose each row vector w_t associated with a term t contains the weights of t for each path occurring in C. A query Q considered also as an XML tree is expressed as a matrix $\boldsymbol{Q}$. The matrix model proposes to evaluate the degree of similarity of D with regard to the Q as the correlation between the matrices $\boldsymbol{D}$ and $\boldsymbol{Q}$.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 74–78, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

Experiments have shown that it is not possible to rely only on this score. Instead we adjust the matrix ***D*** by an additional data structure, so called a path transform matrix, which reflects relationships among paths. The same is done for the matrix ***Q***. Then, the resulted transformed matrices ***TD*** and ***TQ*** are used for query processing. First experiments have been done with the well-known collection of Shakespeare's plays [4] and synthetic data generated by a widely used database benchmark XBench.

In next development of the matrix model we found its critical points and proposed its new version based on the approach [7]. In experimental implementation (called MAMEX in [14]) we used INEX collection [10] as input data. We have compared vector model and renewed matrix model and explored cases in which precision of results are comparable and cases where the latter model wins. The experiments confirmed that the matrix model is mostly not worse than vector model and is significantly better in the cases of queries with more terms. This can be of an importantance for Web querying where a page is a query unit and a collection of pages is relatively stable.

References

1. Anh, V.N., Moffat, A.: Compression and an IR Approach to XML Retrieval. In: Proc. of the First Workshop of INEX, Dagstuhl, Germany, December 2002, pp. 99-104.
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. NY: ACM Press, 1999.
3. Barbosa, D., Mignet, L., Veltri, P.: Studying the XML Web: Gathering Statistics from an XML Sample. World Wide Web 8(4): 413-438, Springer Business + Media; 2005.
4. Bosak, J.: Shakespeare 2.00. Los Altos, California, http://www.ibiblio.org/bosak/, 1999.
5. Bourret, R.: XML and Databases, http://www.rpbourret.com/xml/XMLAndDatabases.htm.
6. Bremer, J.-M., Gertz, M.: XQuery/IR: Integrating XML Document and Data Retrieval In: Proc. of the 5th Int. Workshop on the Web and Databases (WebDB), June 2002, pp. 1-6.
7. Carmel, D., Efraty, N., Landau, G.M., Maarek, Y., Mass, Y.: An Extension of the Vector Space Model for Querying XML Documents via XML Fragments. In: Proc. of XML and Information Retrieval (Workshop) Tampere, 2002, pp. 14-25.
8. Crouch, C.J., Apte, S., Bapat, H.: Using the Extended Vector Model for XML Retrieval. In: Proc. of the 1st INEX 2002 Workshop, Dagstuhl, December 2002, pp. 95-98.
9. Fuhr, N., Großjohann, K.: XIRQL: A Query Language for Information Retrieval. In: Proc. of ACM-SIGIR, New Orleans, 2001, pp. 172-180.
10. Gövert, N., Kazai, G.: Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. In: Proc. of the first Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, 2002, pp. 1-17.
11. Grabs, T., Schek, H.: Generating vector spaces on-the-fly for flexible XML retrieval. In: Proc. of XML and Information Retrieval (Workshop), Tampere, ACM Press, 2002, pp. 4-13.
12. Kakade, V., Raghavan, P.: Encoding XML in vector spaces. In: Proc. of the 27th European Conf. in Information Retrieval (EPIC). LNCS 3408. Springer, NY, 2005, pp. 96-111.
13. Pokorný, J., Rejlek, V.: A Matrix Model for XML Data. Chap. in: Databases and Information Systems, Volume 118 Frontiers in Artificial Intelligence and Applications, Eds. J. Barzdins and A. Caplinskas, IOS Press, 2005, pp. 53-64.
14. Vávra, J.: Matrix model in context of XML IR methods. Master Thesis, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic, 2005. (in Czech)

Decathlon, Conflicting Objectives and User Preference Querying

Peter Vojtáš

Faculty of Mathematics and Physics, Charles University,
Malostranské náměstí 25, Prague, Czech Republic
Peter.Vojtas@mff.cuni.cz

Extended Abstract

First motivation of our approach is Decathlon as an athletic discipline and its development in the last century (http://www.decathlon.ee). During this period, motivated by a desire of a fair competition, a development in scoring tables occurred. Today we can say it is stabilized. It shows single disciplines ordered (and scored) in direction of better (harder) achievement. Comparison with single discipline world records shows that current decathlon world record holder (Roman Sebrle, CZ) was able to achieve about 65-92% of physical measurement and about 59-88% of point achievements of single discipline world records. Of course the point of decathlon is that all disciplines should be done by a single athlete in two consecutive days. Total achievement in decathlon is evaluated by the sum of point in single disciplines.

Another motivation comes from the paper IHAB F. ILYAS, GEORGE BESKALES and MOHAMED A. SOLIMAN. A Survey of Top-k Query Processing Techniques in Relational Database Systems, To appear in ACM Computing Surveys. A person looking for a house evaluates market offers by an aggregation of house price and tuition price in a school near to location. Nevertheless the price for house and tuition price cannot be simply added, the aggregation is, as in following SQL query

```
SELECT h.id, s.id
FROM House h; School s
WHERE h.location=s.location
ORDER BY h.price + 10 * s.tuition
LIMIT 5
```

Last motivation comes from multicriterial decision (Source [RTC] R.T.Clemen. Making hard decisions. Brooks/Cole Publ. Comp. 1996). Here, several examples are used, especially for conflicting objectives (like price and durability of a car). The solution is, that single objectives are represented by a objective function $U_i(x_i)$ and aggregation (either linear or nonlinear) is here called utility function, gives a ordering of products (decisions) by total score equal to $U(x_1, \ldots, x_m) = k_1U_1(x_1) + \cdots + k_mU_m(x_m)$.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 76–78, ISBN 978-80-248-1746-0.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2008.

Main goal of this talk is to point to similarities in all of these applications: IAAF – "combined events", Decision analysis and rank aware Querying.

Similarities can be characterized as follows: Incomparable disciplines (attributes) are mapped to points, score, ... - hence comparable values. Best, top k – preserves ordering if better in all axes (disciplines, attributes). There is some monotone aggregation, combination (e.g. (weighted) sum)

In what follows a class of aggregation functions is discussed - Sum and/or average, Weighted average, ... general. Common characteristics are: Monotone in attribute score, ideal point given by application, user dependent, implicit learning (in athletics took about 100 years), user similarity - collaborative. Our contribution reflects explicit learning, adaptation during the query cycles. A possibility was discussed: to fix aggregation and tune attribute scoring or attribute score fixed - tuning aggregation.

We have presented a unifying approach: (local) attribute preference represented by a scoring (fuzzy) function $f : DA \rightarrow [0, 1]$, Combination @ : $[0, 1]^n \rightarrow [0, 1]$, (global) score $score(o) = @(f_1(o.A_1), \ldots, f_m(o.A_m))$ We have a model-theoretic semantics based on – fuzzy logic, fixpoint semantics – fuzzy Data(/Pro)log Proof-theoretic semantics – best, top-k, heuristics.

We discussed also a form of data (in one table, several tables, locally or distributed, on the Web, frequently changing versus rather stable data, in relational, XML, HTML, text, ...). Further issues are preprocessing, indexes, Query optimization, Top-k versus table scan (experiments) Fagin instance optimal TA/NRA algorithm.

Different models were considered for User (One, many, different, ... User profile, Group decision, Changing intention during querying, Query formulation – clicking conjunctive query or sample evaluation).

We have a procedure how to learn users "Decathlon principle aggregation (combination). One for learning $score(o) = @(f_1(o.A_1), \ldots, f_m(o.A_m))$ either with Fixed @, tuning f_i (like IAAF @=+), or fixed f_i, learning @ - fuzzy ILP. Learning both -local preferences (user can have different order of preference, e.g. close-far) global preferences. For this new inductive task – ordinal classification we can either compare orderings or generalize precision, recall.

Special focus in different objectives for different user, e.g. (as a variation of above example)

```
SELECT h.id, s.id
FROM House h; School s
WHERE h.location=s.location
ORDER BY MAX h.price + 10 * s.tuition
LIMIT 5

SELECT h.id, s.id
FROM House h; School s
WHERE h.location=s.location
ORDER BY MIN h.price + 10 * s.tuition
LIMIT 5
```

```
SELECT h.id, s.id
FROM House h; School s
WHERE h.location=s.location
ORDER BY h.price + 60 * s.tuition
LIMIT 5

SELECT h.id, s.id
FROM House h; School s
WHERE h.location=s.location
ORDER BY @(f(h.price), g(s.tuition))
LIMIT 5
```

We have concluded with open problems and future work. Unified framework does not cover nominal data, multidimensional data, hierarchical data, ...

Author Index