ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING 186

Tadeusz Morzy
Theo Härder
Robert Wrembel (Eds.)

Advances in Databases and Information Systems

Springer

Tadeusz Morzy, Theo Härder,
and Robert Wrembel (Eds.)

Advances in Databases and Information Systems

Springer

Editors

Prof. Tadeusz Morzy
Institute of Computing Science
Poznan University of Technology
Poznan
Poland

Prof. Robert Wrembel
Institute of Computing Science
Poznan University of Technology
Poznan
Poland

Prof. Dr. Theo Härder
Department of Computer Science
University of Kaiserslautern
Kaiserslautern
Germany

ISSN 2194-5357 e-ISSN 2194-5365
ISBN 978-3-642-32740-7 e-ISBN978-3-642-32741-4
DOI 10.1007/978-3-642-32741-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012944708

Preface

These proceedings contain 27 contributed papers presented at the 16th East-European Conference on Advances in Databases and Information Systems (ADBIS 2012), held on September 18–21, 2012, in Poznań, Poland.

The ADBIS 2012 conference continues the series of ADBIS conferences organized every year in different countries of Eastern and Central Europe, beginning in St. Petersburg (Russia, 1997), Poznań (Poland, 1998), Maribor (Slovenia, 1999), Prague (Czech Republic, as a joint ADBIS-DASFAA conference, 2000), Vilnius (Lithuania, 2001), Bratislava (Slovakia, 2002), Dresden (Germany, 2003), Budapest (Hungary, 2004), Tallinn (Estonia, 2005), Thessaloniki (Greece, 2006), Varna (Bulgaria, 2007), Pori (Finland, 2008), Riga (Latvia, 2009), Novi Sad (Serbia, 2010), and Vienna (Austria, 2011). The main objective of the series of ADBIS conferences is to provide a forum for the dissemination of research achievements as well as to promote interaction and collaboration between the database and information systems research communities from Central and East European countries and the rest of the world. The conferences are initiated and supervised by an international Steering Committee, which consists of representatives from Armenia, Austria, Bulgaria, Czech Republic, Greece, Estonia, Finland, Germany, Hungary, Israel, Latvia, Lithuania, Poland, Russia, Serbia, Slovakia, Slovenia, Ukraine, and Italy.

The ADBIS 2012 conference attracted 122 paper submissions from Algeria, Argentina, Belgium, Bosnia and Herzegovina, Brazil, Colombia, Czech Republic, Egypt, Estonia, Finland, France, FYR Macedonia, Germany, Greece, Hungary, India, Iran, Italy, Japan, Latvia, Poland, Romania, Russia, Slovakia, Spain, Sweden, Sultanate of Oman, The Netherlands, Tunisia, UK, and USA. In a rigorous reviewing process the international program committe of 74 members from 31 countries selected 32 contributions for publishing in a separate volume of proceedings by Springer-Verlag in the Lecture Notes in Computer Science series and 27 contributions for publishing in this volume. Topically, the selected papers cover a wide spectrum of topics in the database and information systems field, including database theory, database architectures, query languages, query processing and optimization, design methods, data integration, view selection, nearest neighbor searching, analytical query processing, indexing and caching, concurrency control, distributed systems, data mining, data streams, ontology engineering, social networks, multi-agent systems, business processes modeling, knowledge management, and application oriented topics like RFID, XML, and data on the web.

Additionally, ADBIS 2012 aimed to create conditions for experienced researchers to impart their knowledge and experience to the young researchers participating at the Doctoral Consortium organized in association with the ADBIS 2012 conference. Moreover, this year, the 6 following workshops associated with the conference were organized: GPUs in Databases (GID), Mining Complex and Stream Data (MCSD), OAIS: Ontologies meet Advanced Information Systems, Second Workshop on Modeling Multi-commodity Trade: Data models and processing (MMT), Social Data Processing, Social and Algorithmic Issues in Business Support. The accepted workshop papers were published in a separate volume by Springer-Verlag in the Advances in Intelligent Systems and Computing series.

We would like to express our thanks to all people who contributed to the success of ADBIS 2012. We thank the authors, who submitted papers to the conference, the program committee members and external reviewers for ensuring the quality of the scientific program. We thank the colleagues of our universities for their help in the conference and workshops organization, all members of the local organizing team in Poznań for giving their time and expertise to ensure the success of the conference. We express our gratitude to Alfred Hofmann, from Springer-Verlag, for accepting these proceedings for the LNCS series and to Professor Janusz Kacprzyk, from the Polish Academy of Sciences, for accepting the short contributions and workshops proceedings in the Advances in Intelligent Systems and Computing series. We thank the Steering Committee and, in particular, its chair, Leonid Kalinichenko, for their help and guidance. Last but not least, we thank the Sponsors and Suporters of our conference, including: **Allegro Group**, **the City of Poznań**, **IBM**, **Roche**, **Microsoft**, **Targit**, **Samsung**, **Edge Solutions**, **ITelligence**. Without their financial support the high quality of the conference proceedings and events would not be possible to achieve.

September 2012

Tadeusz Morzy
Theo Härder
Robert Wrembel

Conference Organization

General Chair

Tadeusz Morzy	Poznan University of Technology, Institute of Computing Science, Poland

Program Committee Co-chairs

Theo Härder	University of Kaiserslautern, Computer Science Department, Germany
Robert Wrembel	Poznan University of Technology, Institute of Computing Science, Poland

Workshop Co-chairs

Mykola Pechenizkiy	Eindhoven University of Technology, Computer Science Department, Netherlands
Marek Wojciechowski	Poznan University of Technology, Institute of Computing Science, Poland

PhD Consortium Co-chairs

Mikołaj Morzy	Poznan University of Technology, Institute of Computing Science, Poland
Alexandros Nanopoulos	University of Hildesheim, Institute of Computer Science, Germany

Program Committee

Divyakant Agrawal	University of California at Santa Barbara, USA
Costin Badica	University of Craiova, Romania
Ladjel Bellatreche	Ecole Nationale Supérieure de Mécanique et d’Aérotechnique, France
Andras Benczur	Hungarian Academy of Science, Hungary

Maria Bielikova	Slovak University of Technology, Slovakia
Albertas Caplinskas	Vilnius University, Lithuania
Barbara Catania	University of Genova, Italy
Wojciech Cellary	Poznan University of Economics, Poland
Ricardo Rodrigues Ciferri	Universidade Federal de São Carlos, Brazil
Alfredo Cuzzocrea	ICAR-CNR and University of Calabria, Italy
Todd Eavis	Concordia University, Canada
Johann Eder	Klagenfurt University, Austria
Pedro Furtado	University of Coimbra, Portugal
Johann Gamper	Bolzano University, Italy
Matjaz Gams	Jozef Stefan Institute, Slovenia
Minos Garofalakis	Technical University of Crete, Grece
Goetz Graefe	HP Labs, USA
Janis Grundspenkis	Riga Technical University, Latvia
Jarek Gryz	York University, Canada and Warsaw University of Technology, Poland
Adam Grzech	Wroclaw University of Technology, Poland
Hele-Mai Haav	Tallinn University of Technology, Estonia
Mirjana Ivanovic	University of Novi Sad, Serbia
Hannu Jaakkola	Tampere University of Technology, Finland
Manfred A. Jeusfeld	Tilburg University, Netherlands
Leonid Kalinichenko	Russian Acaedmy of Science, Russia
Ahto Kalja	Küberneetika Instituut, Estonia
Alfons Kemper	Technische Universität München, Germany
Martin Kersten	Centrum Wiskunde & Informatica, Netherlands
Maurice van Keulen	University of Twente, Netherlands
Marite Kirikova	Riga Technical University, Latvia
Attila Kiss	Eötvös Lorand University, Hungary
Margita Kon-Popovska	Ss Cyril and Methodius University, Macedonia
Christian Koncilia	Klagenfurt University, Austria
Stanisław Kozielski	Silesian University of Technology, Poland
Sergei Kuznetsov	University of Colorado, USA
Wolfgang Lehner	Technical University Dresden, Germany
Yannis Manolopoulos	Aristotle University, Greece
Rainer Manthey	Universität Bonn, Germany
Bernhard Mitschang	Universität Stuttgart, Germany
Felix Naumann	Hasso-Plattner-Institut für Softwaresystemtechnik, Germany
Pavol Navrat	Slovak University of Technoloy, Slovakia
Anisoara Nica	Sybase, SAP, Canada
Mykola Nikitchenko	National Taras Shevchenko University of Kyiv, Ukraine
Kjetil Norvag	Norwegian University of Science and Technology, Norway
Boris Novikov	University of St. Petersburg, Russia

Gultekin Ozsoyoglu	Case Western Reserve University, USA
Torben Bach Pedersen	Aalborg University, Denmark
Dana Petcu	West University of Timisoara, Romania
Evaggelia Pitoura	University of Ioannina, Grece
Jaroslav Pokorny	Charles University in Prague, Czech Republic
Peter Revesz	University of Nebraska-Lincoln, USA
Tore Risch	Uppsala University, Sweden
Stefano Rizzi	University of Bologna, Italy
Henryk Rybiński	Warsaw University of Technology, Poland
Klaus-Dieter Schewe	Software Competence Center, Austria
Holger Schwarz	Universität Stuttgart, Germany
Timos Sellis	Research Center “Athena" and National Technical University of Athen, Greece
Vaclav Snasel	VSB-Technical University of Ostrava, Czech Republic
José Neuman de Souza	Federal University of Ceara, Brasil
Bela Stantic	Griffith University, Australia
Janis Stirna	Royal Institute of Technology, Sweden
Bernhard Thalheim	Christian-Albrechts-University Kiel, Germany
Goce Trajcevski	Northwestern University, USA
Olegas Vasilecas	Vilnius Gediminas Technical University, Lithuania
Krishnamurthy Vidyasankar	Memorial University of Newfoundland, Canada
Peter Vojtas	Charles University in Prague, Czech Republic
Gottfried Vossen	Universität Münster, Germany
Florian Waas	Greenplum/EMC, USA
Xin Wang	Columbia University, Canada
Gerhard Weikum	Max-Planck-Institut für Informatik, Germany
Tatjana Welzer	University of Maribor, Slovenia
Limsoon Wong	National University of Singapore
Shuigeng Zhou	Fudan University, China
Esteban Zimanyi	Université Libre de Bruxelles, Belgium

ADBIS Steering Committee Chair

Leonid Kalinichenko	Russian Academy of Science, Russia

ADBIS Steering Committee

Paolo Atzeni, Italy
Andras Benczur, Hungary
Albertas Caplinskas, Lithuania
Barbara Catania, Italy
Johann Eder, Austria
Marite Kirikova, Latvia
Hele-Mai Haav, Estonia
Mirjana Ivanovic, Serbia
Hannu Jaakkola, Finland
Mikhail Kogalovsky, Russia

Organizing Committee

Publicity Chair

Krzysztof Jankiewicz	Poznan University of Technology, Institute of Computing Science, Poland

Proceedings Chair

Witold Andrzejewski	Poznan University of Technology, Institute of Computing Science, Poland

Treasurer

Robert Wrembel	Poznan University of Technology, Institute of Computing Science, Poland

Local Organizing Committee Co-Chairs

Piotr Krzyżagórski	Poznan University of Technology, Institute of Computing Science, Poland
Stanisław Woźniak	Poznan University of Technology, Institute of Computing Science, Poland

Local Organizing Committee

Bartosz Będbel	Poznan University of Technology, Institute of Computing Science, Poland
Paweł Boiński	Poznan University of Technology, Institute of Computing Science, Poland
Dariusz Brzeziński	Poznan University of Technology, Institute of Computing Science, Poland
Anna Leśniewska	Poznan University of Technology, Institute of Computing Science, Poland
Maciej Piernik	Poznan University of Technology, Institute of Computing Science, Poland

Partners

allegro group

POZnan*
* Miasto know-how

IBM

Microsoft®

Roche

TARGIT®
business intelligence

SAMSUNG

EDGE SOLUTIONS

itelligence

Contents

Reduction Relaxation in Privacy Preserving Association Rules Mining 1
Piotr Andruszkiewicz

Decomposition of Non-deterministic Services for the Purpose of Replication 9
Marcin Bazydło, Szymon Francuzik, Cezary Sobaniec, Dariusz Wawrzyniak

Partitioning Approach to Collocation Pattern Mining in Limited Memory Environment Using Materialized iCPI-Trees 19
Pawel Boinski, Maciej Zakrzewicz

Towards the Automated Business Model-Driven Conceptual Database Design 31
Drazen Brdjanin, Slavko Maric

Aligning Business Process Models and Domain Knowledge: A Meta-modeling Approach 45
Samira Si-Said Cherfi, Sarah Ayad, Isabelle Comyn-Wattiau

Optimization of Object-Oriented Queries through Pushing Selections 57
Marcin Drozd, Michał Bleja, Krzysztof Stencel, Kazimierz Subieta

On Parallel Sorting of Data Streams 69
Zbyněk Falt, Jan Bulánek, Jakub Yaghob

Mining Association Rules from Database Tables with the Instances of Simpson's Paradox 79
Wojciech Froelich

Synchronization Modeling in Stream Processing 91
Marcin Gorawski, Aleksander Chrószcz

Towards Automated Enterprise Architecture Documentation: Data Quality Aspects of SAP PI 103
Sebastian Grunow, Florian Matthes, Sascha Roth

Two-Stage Stochastic View Selection for Data-Analysis Queries 115
Rong Huang, Rada Chirkova, Yahya Fathi

Choosing Values for Text Fields in Web Forms 125
Gustavo Zanini Kantorski, Tiago Guimaraes Moraes, Viviane Pereira Moreira, Carlos Alberto Heuser

A Relative Feature Selection Algorithm for Graph Classification 137
Yaser Keneshloo, Sasan Yazdani

Generic Conceptual Framework for Handling Schema Diversity during Source Integration: Semantic Database Case Study 149
Selma Khouri, Ladjel Bellatreche, Nabila Berkani

Adaptation of Asymptotic Trust Algorithm: Reputation on Internet Forums 161
Konrad Leszczyński, Maciej Zakrzewicz

PropScale: An Update Propagator for Joint Scalable Storage 173
Paweł Leszczyński, Krzysztof Stencel

An Indexing Structure for Dynamic Multidimensional Data in Vector Space 185
Elena Mikhaylova, Boris Novikov, Anton Volokhov

Social Network Analysis on Highly Aggregated Data: What Can We Find? 195
Mikołaj Morzy, Kamil Forenc

An Adaptive Navigation Method for Semi-structured Data 207
Rim Zghal Rebai, Corinne Amel Zayani, Ikram Amous

Relational Schema Summarization: A Context-Oriented Approach 217
Marcus Sampaio, Jefferson Quesado, Samarony Barros

Semantic Approach to Cluster Validity Notion 229
Elena Sivogolovko, Bernhard Thalheim

An Automated Approach of Designing Multiplex PCR Primers for the Amplification of Exons 241
Adam Skowron, Rafal Pokrzywa

Functional Dependencies on Symbol Strings Generated by Extended Context Free Languages 253
Gyula I. Szabó, András Benczúr

Extending HQL with Plain Recursive Facilities 265
Aneta Szumowska, Marta Burzańska, Piotr Wiśniewski, Krzysztof Stencel

Trust in RDF Graphs 273
Dominik Tomaszuk, Karol Pąk, Henryk Rybiński

Inference of XML Integrity Constraints 285
Matej Vitásek, Irena Mlýnková

Optimizing the Resource Allocation for Approximate Query Processing 297
Anna Yarygina, Boris Novikov

Author Index 309

Reduction Relaxation in Privacy Preserving Association Rules Mining

Piotr Andruszkiewicz

Abstract. In Privacy Preserving Association Rules Mining, when frequent sets are discovered, the relaxation can be used to decrease the false negative error component and, in consequence, to decrease the number of true frequent itemsets that are missed. We introduce the new type of relaxation - the reduction relaxation that enable a miner to decrease and control the false negative error for different lengths of frequent itemsets.

1 Introduction

One task in Privacy Preserving Association Rules Mining is to discover associations which are present in an original database based on a distorted database. In this task parameters of the distortion procedure used to modify the original database as well as the distorted database are known, however, a miner does not have an access to the original database.

In order to find frequent sets based on a distorted centralised database, the MASK [4] scheme can be used. This scheme estimates the original support of candidate itemsets based on a counted support of itemsets or itemsubsets and parameters of a distortion procedure. Unfortunately, in the results provided by the MASK scheme there are itemsets which were assumed to be frequent in an original database but their are not and itemsets which were assumed to be infrequent in an original database but they really are. The first set is called false positive and the second is false negative component. Even with the usage of MMASK [3], which is the optimisation for MASK that eliminates exponential complexity in estimating a support of an itemset with respect to its cardinality and improves the accuracy of the results, itemsets incorrectly assumed to be frequent exist.

Piotr Andruszkiewicz
Institute of Computer Science, Warsaw University of Technology, Poland
e-mail: P.Andruszkiewicz@ii.pw.edu.pl

One method of decreasing of the false negative component in MASK is the relaxation proposed in [4], which marginally relaxes a minimum support threshold at the beginning of the process of frequent itemsets mining. In this paper we apply this relaxation to the MMASK scheme and propose the new type of relaxation, i.e., the retention relaxation. Moreover, we combine these two types of relaxations for the MMASK scheme.

The proposed retention relaxation combined with the relaxation enable a miner to decrease and control the false negative component for different lengths of frequent itemsets discovered during the process of association rules mining.

2 Related Work

A framework for mining association rules from a centralised distorted database was proposed in [4]. A scheme called MASK attempts to simultaneously provide a high degree of privacy to a user and retain a high degree of accuracy in the mining results. To address efficiency, several optimisations for MASK were originally proposed in [4]. In [3] we proposed MMASK optimisation for MASK, which breaks exponential complexity in estimating a support of an itemset with respect to its cardinality. Not only does this optimisation allow attributes to be randomised using different randomisation factors, based on their privacy levels, but also allow attributes to have different randomisation factors for each value; that is, when an item is present in an original database and when it is not.

3 Relaxation

As the false negative error causes the mining process to miss some frequent itemsets, it is possible to use an effect of marginally relaxing *minimumSupport* to reduce this problem. The relaxation proposed in [4] can decrease the false negative error component and, in consequence, decrease the number of true frequent itemsets that are missed. As a result of the relaxation the false positive error components goes up, inevitably attracts not frequent sets. The relaxation can be also applied to the MMASK scheme.

Let *relax* be the parameter of the relaxation, for instance, $relax = 0.02 = 2\%$. The relaxed minimum support ($minimumSupport'$) is calculated as follows:

$$minimumSupport' = \frac{minimumSupport}{1 + relax} = \frac{minimumSupport}{1 + 0.02}.$$

As the relaxation enables a miner to decrease the false negative component for all levels; that is, the length of the itemsets without the differentiation for a particular level, we propose to introduce the *reduction relaxation*. This approach bases on that the relaxation is repeated every time the reduction of the distorted transaction set is performed and allow a miner to control the false negative for different lengths of itemsets.

The *reduction relaxation* has the *rrelax* parameter, which is used to relax the minimum support every time the reduction in MMASK is performed. The modified minimum support (*minimumSupport*′) in the reduction relaxation is calculated as follows:

$$minimumSupport' = \frac{minimumSupport}{1 + rrelax}.$$

Moreover, the *rrelax* parameter can have different values for each length of itemsets or can be changed in a systematic way, e.g., by increasing it by a given parameter for each length of itemsets.

These two relaxations can be combined at the same time in the MMASK scheme, we will call this scheme rMMASK. For combined relaxations, the modified minimum support can be calculate as follows:

$$minimumSupport' = \frac{minimumSupport}{1 + relax + k \cdot rrelax},$$

where k is equal to the number of times the reduction was performed.

We present the application of the both relaxations on the example of the PPApriori-rMMASK algorithm, which utilises the rMMASK scheme in estimating itemsets support during generation of frequent itemsets candidates in Apriori fashion manner (please refer to Algorithms 1, 2, 3).

We use the following notation to present PPApriori-rMMASK.

- $\mathscr{X}_m$ denotes candidate m-itemsets, which are potentially frequent.
- $\mathscr{F}_m$ are frequent m-itemsets based on estimations of original supports of itemsets.
- $X[i]$ is the i-th item in the itemset X.
- $X[1] \cdot X[2] \cdot X[3] \cdot \ldots \cdot X[m]$ denotes m-itemset, which consists of $X[1]$,$X[2]$, $X[3],\ldots,X[m]$.
- $\mathscr{T}$ is the original data set.
- $\mathscr{D}$ is the data set distorted according to the MASK scheme and each item i is distorted according to the matrix $\mathbf{M}_i$.
- $X.R$ is the reduced itemset of X.
- $X.R.\mathbf{C}^D$ is the support vector field of the reduced itemset $X.R$ in the distorted data set $\mathscr{D}$.
- $X.R.\mathbf{C}^T$ is the support vector field of the reduced itemset $X.R$ in the true data set.
- $X.R.\mathbf{C}^T_j$ is the j-th element of the vector $X.R.\mathbf{C}^T$.
- $X.R.\mathbf{C}^D_j$ is the j-th element of the vector $X.R.\mathbf{C}^D$.
- $X.R.\mathbf{M}$ is a $\mathbf{M}$ matrix for the reduced itemset $X.R$.
- $X.R.\mathbf{M}^{-1}$ is an inverted $\mathbf{M}$ matrix for the reduced itemset $X.R$.
- $X.R_F$ is the lexicographically first ancestor of the reduced itemset $X.R$.
- $X.\mathscr{D}_R$ is the subset of transactions from the database $\mathscr{D}$ which support $X.R$ with a high probability.

The Privacy Preserved Apriori-rMMASK (PPApriori-rMMASK) algorithm for mining frequent itemsets which uses rMMASK scheme (Algorithms 1, 2, 3) estimates

Algorithm 1. The PPApriori-rMMASK algorithm, the apriori algorithm modified to use the rMMASK scheme (i.e., MMASK and relaxation)

```
input: $minimumSupport$
input: $\mathscr{D}$    // binary distorted data set
input: $rThreshold$, $rThreshold > 0$
input: $relax$
input: $rrelax$
$\mathscr{F}_1$ ={1-itemsets which are frequent based on estimated original support of singletons;
that is, have estimated support greater than or equal to $\frac{minimumSupport}{1+relax}$ }
for all $X \in \mathscr{F}_1$ do begin
  $X.\mathscr{D}_R = \mathscr{D}$
  $X.R = X$
end
$index = 1$
$currentRelax = relax$        // relaxation
for $(m = 2; \mathscr{F}_{m-1} \neq \emptyset; m{+}{+})$ do begin
  $index{+}{+}$
  $\mathscr{X}_m$ = $aprioriGen(\mathscr{F}_{m-1})$    //generate new candidates
  if $index > rThreshold$ then begin
    for all $X \in \mathscr{X}_m$ do begin    //choose the subset of transactions
      $X.\mathscr{D}_R = \{O \in X.\mathscr{D}_R | O$ supports $X.R_F$ with a high probability in the true
        database}
    end
  end
  $supportCount(\mathscr{X}_m)$
  $\mathscr{F}_m = \{X \in \mathscr{X}_m | X.R.\mathbf{C}^T_{2^m-1} \geq \frac{minimumSupport}{1+currentRelax}\}$
  if $index > rThreshold$ then begin
    $index = 1$
    $currentRelax = currentRelax + rrelax$        // reduction relaxation
  end
end
return $\bigcup_m \mathscr{F}_m$
```

original supports of candidate m-itemsets like the Apriori algorithm modified to use MASK [3] and checks the minimal support condition with the relaxation equal to the parameter $rrelax$, until m is less than or equal to $rThreshold$. When m is greater than $rThreshold$, the relaxation is increased by the parameter $rrelax$ and the support of the m-itemset X is determined by estimating the support of a reduced itemset R in the set $\mathscr{D}_R$ of distorted transactions which support $X \setminus R$, $|X \setminus R|$ = $rThreshold$, in the true database with a high probability.

The true supports for itemsets with higher length are estimated based on the transaction set $\mathscr{D}_R \subset \mathscr{D}$ until the length of a reduced itemset exceeds $rThreshold$. Then a subset of transactions $\mathscr{D}_{R'}$ is chosen (by means of the CTS algorithm, for details please refer to [3]) from the subset $\mathscr{D}_R$. The chosen transactions from the subset $\mathscr{D}_{R'}$ support the subset $X' \setminus R'$ of the itemset candidate X', $|X' \setminus R'|$ = $rThreshold$, $X \subset X'$, with a high probability. The true support of the itemset X' is estimated as the support of an itemset R' in the subset of transactions $\mathscr{D}_{R'}$. When the length of a

Algorithm 2. The candidate generation algorithm for rMMASK

```
function $aprioriGen$(var $\mathscr{F}_m$)
  for all $Y, Z \in \mathscr{F}_m$ do begin
    if $Y[1] = Z[1] \wedge \ldots \wedge Y[k-1] = Z[k-1] \wedge Y[k] < Z[k]$ then begin
      $X = Y[1] \cdot Y[2] \cdot Y[3] \cdot \ldots \cdot Y[k-1] \cdot Y[k] \cdot Z[k]$
      $X.\mathscr{D}_R = Y.\mathscr{D}_R$
      if index $> rThreshold$ then begin
        $X.R_F = Y.R$
        $X.R = Z[k]$
      end else begin
        $X.R = Y.R \cup Z[k]$
      end
      add $X$ to $\mathscr{X}_{m+1}$
    end
  end
  for all $X \in \mathscr{X}_{m+1}$ do begin
    for all $m$-itemsets $Z \subset X$ do begin
    if $Z \notin \mathscr{F}_m$ then delete $X$ from $\mathscr{X}_{m+1}$
    end
  end
  return $\mathscr{X}_{m+1}$
end
```

reduced itemset exceeds *rThreshold*, the reduction relaxation is performed; that is, the current relaxation is increased by the *rrelax* parameter.

The true supports for longer candidate itemsets are estimated based on the transaction set $\mathscr{D}_{R'}$ until the length of a reduced candidate itemset exceeds *rThreshold*. Then the subset of transactions $\mathscr{D}_{R''}$ is chosen (by means of the CTS algorithm [3]) and the reduction relaxation is performed once again.

Algorithm 3. The support count algorithm for rMMASK

```
procedure $supportCount$(var $\mathscr{X}_m$)
  for all transactions $T \in \mathscr{D}$ do begin
    for all candidates $X \in \mathscr{X}_m$ do begin
      if $T \in X.\mathscr{D}_R$ then $X.\mathbf{C}_j^D$++       //$j$ is the number which has a binary form
                                        // (in $m$ digits) of $X.R$ in the transaction $T$
    end
  end
    for all candidates $X \in \mathscr{X}_m$ do begin
      $X.R.\mathbf{C}^T = X.R.\mathbf{M}^{-1} X.R.\mathbf{C}^D$
    end
end
```

The PPApriori-rMMASK algorithm generates candidates for frequent sets with a given length in the Apriori-like fashion. In every iteration of candidates generation when the reduced itemset used to estimate an original support of a candidate in the true database based on a support counted in distorted transactions exceeds *rThreshold*, the reduction of the set of transactions which is used to estimate an original support of the candidate is performed. Then, having estimated the original supports of candidates, the relaxed minimum support condition is checked and candidates which are not frequent are removed. Hence, the PPApriori-rMMASK algorithm finds the itemsets with the estimated support greater than or equal to the relaxed *minimumSupport*.

To sum up, the PPApriori-rMMASK algorithm finds frequent sets with relaxation equal to the parameter *relax*, at the beginning. Then, if the length of the reduced itemset is greater than *rThreshold*, the reduction is performed and the relaxation is increased by the parameter *rrelax* (please refer to Algorithm 1). As the result of the PPApriori-rMMASK algorithm, the itemsets with the estimated support greater than or equal to relaxed *minimumSupport* are provided.

4 Experimental Evaluation

In this section, we present the results of the experiments conducted to check the influence of the relaxation on the accuracy of the MMASK scheme. We also compare the results obtained for MASK. For the definitions of the used measures please refer to [4] and [3].

We present the results of the experiments carried out on the chosen database (the remaining results with similar patterns were not presented): a synthetic database, T10I8D100kN100, generated from the IBM Almaden generator [1]. The data set was created with parameters T=10 (the average size of the transactions), I=8 (the average size of the maximal potentially frequent itemsets), D=100k (the number of transactions), N=100 (the number of items). More information on a generator and naming convention can be found in [1]). The data set contains about 100000 tuples with each customer purchasing about ten items on average.

In our experiments we performed the comparison of the results for MASK and MMASK without the relaxation, with the relaxation, the retention relaxation and both relaxations applied simultaneously.

The first experiment was conducted on the synthetic set T10I8D100kN100 with the distortion parameters of $p = 0.5$ and $q = 0.87$, and no relaxation to present the results for the case without the relaxation, which we treat as a baseline. Basic Privacy [4, 2] for the given values of p and q is equal to 81%. We experimentally chose *rThreshold* to be equal to 3 and *minimumSupport* to 0.005. The results of this experiment are shown in Table 1. The level, which corresponds to the consecutive iterations in Apriori-like algorithms, indicates the length of frequent itemsets, $|F_0|$ indicates the number of frequent itemsets at a given level, $|F_r|$ ($|F_{rm}|$) shows the number of mined frequent itemsets from the distorted database using MASK (MMASK). The other columns are the measures defined in [4] and [3].

As shown in Table 1, MASK and MMASK lead to the high false negative error. In order to reduce it, the relaxation can be used.

For rMASK and rMMASK compared to the case without the relaxation the support error is similar. The false negative error was smaller than without the relaxation.

Table 1 The results of mining the frequent sets in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, rThreshold = 3, minimumSupport = 0.005

Level	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2704	20.4	10.8	18.0	2250	2704	20.4	10.8	18.0	2250
3	10930	16780	37.5	25.9	79.5	8094	16780	37.5	25.9	79.5	8094
4	10185	22411	62.4	40.1	160.2	6098	7787	16.4	36.3	12.7	6490
5	2021	5810	115.9	57.9	245.4	851	1129	16.6	57.3	13.2	862
6	24	200	318.5	79.2	812.5	5	28	6.8	70.8	87.5	7

Table 2 The results of mining the frequent sets with both relaxations in the set T10I8D100kN100 with parameters p = 0.5, q = 0.87, relax = 0.01, rrelax = 0.02, rThreshold = 3, minimumSupport = 0.005

Level	$\|Fo\|$	$\|Fr\|$	ρ_r	$\sigma-_r$	$\sigma+_r$	f_r	$\|Frm\|$	ρ_{rm}	$\sigma-_{rm}$	$\sigma+_{rm}$	f_{rm}
1	98	98	5.4	1.0	1.0	97	98	5.4	1.0	1.0	97
2	2522	2718	20.4	10.7	18.4	2253	2718	20.4	10.7	18.4	2253
3	10930	17000	37.5	25.7	81.2	8123	17000	37.5	25.7	81.2	8123
4	10185	22816	62.3	39.9	163.9	6123	8340	16.5	33.7	15.6	6756
5	2021	5925	115.3	57.6	250.8	857	1304	17.0	53.6	18.2	937
6	24	208	318.5	79.2	845.8	5	35	8.0	66.7	112.5	8

As a negative result of the relaxation, the false positive error was higher. Lower minimum support causes the number of discovered frequent sets (true frequent sets also) to grow.

The support error and identity errors for 5% reduction relaxation were quite similar compared to 5% relaxation in our experiments. f measure for levels 1-3 was lower because the reduction relaxation works for levels higher than $rThreshold$. However, for levels 4-6 f measure was almost the same for both types of the relaxation. Second difference was that the reduction relaxation have not influenced the original MASK scheme.

Both relaxations can be combined. The results with 1% relaxation and 2% reduction relaxation are shown in Table 2. This combined relaxation is not as strong as the 5% relaxations we have tested - the number of correctly discovered frequent itemsets was lower compared with both 5% relaxation applied separately for MMASK and levels 4-6.

By combining relaxations, we can control the number of discovered frequent itemsets of particular length. The higher relaxation results in more discovered frequent itemsets for all lengths of itemsets. The reduction relaxation applied on a particular level makes the number of discovered frequent itemsets higher for this and higher levels.

Summarising, the relaxation can be used to reduce the false negative error and boost f measure. Furthermore, the reduction relaxation enable a miner to control the false negative error for different lengths of frequent itemsets.

5 Conclusions and Future Work

We investigated the problem of the reduction of the false negative error and the boost of f measure in frequent set discovery by means of the MASK and MMASK scheme.

In this paper, we proposed the reduction relaxation as the solution to this problem, combined the proposed reduction relaxation with the relaxation for MMASK scheme and applied this solution in PPApriori-rMMASK algorithm for discovering frequent sets.

We have tested both relaxations and their combination. The obtained results have shown that the relaxations can be used to reduce the false negative error and boost f measure. Moreover, combining these relaxations a miner is able to control the decrease of the false negative error for different lengths of discovered frequent itemsets. The relaxation can be also used for data distorted with different randomisation factors for 0's and 1's; that is, when an item is not present and is present in an original database. Furthermore, the presented solution can be applied in the case when different items have different probabilities of retaining original values.

In future work, we plan to investigate the possibility of extension of MMASK scheme and the relaxations to quantitative [6] and generalised [5] association rules. We also plan to determine what are the best values for the relaxation and the reduction relaxation and the $rThreshold$ and to find a rule to help a miner to choose the best value for these parameters for a given set.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of 20th International Conference on Very Large Data Bases, VLDB 1994, Santiago de Chile, Chile, September 12-15, pp. 487–499. Morgan Kaufmann (1994)
2. Agrawal, S., Krishnan, V., Haritsa, J.R.: On addressing efficiency concerns in privacy preserving data mining. CoRR cs.DB/0310038 (2003)
3. Andruszkiewicz, P.: Optimization for MASK Scheme in Privacy Preserving Data Mining for Association Rules. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 465–474. Springer, Heidelberg (2007)
4. Rizvi, S.J., Haritsa, J.R.: Maintaining data privacy in association rule mining. In: VLDB 2002: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 682–693. VLDB Endowment (2002)
5. Srikant, R., Agrawal, R.: Mining generalized association rules. In: Dayal, U., Gray, P.M.D., Nishio, S. (eds.) VLDB, pp. 407–419. Morgan Kaufmann (1995)
6. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. In: Jagadish, H.V., Mumick, I.S. (eds.) SIGMOD Conference, pp. 1–12. ACM Press (1996)

Decomposition of Non-deterministic Services for the Purpose of Replication

Marcin Bazydło, Szymon Francuzik,
Cezary Sobaniec, and Dariusz Wawrzyniak

Abstract. Replication can improve performance, availability and reliability of services. However, its application raises several design and implementation problems. One of them is non-determinism of processing on replicas. We propose a "design pattern" for structuring the service so that it is possible to overcome the problem.

1 Introduction

Replication is a form of redundancy in which multiple copies of the same logical data are available/created in a system. Depending on the context or, more formally, system model the replicated data may be represented by files, objects, pages, tuples, tables etc. — they are generalized as resources. Nowadays, replication is fundamental to improving dependability of distributed systems. It is also of considerable importance to scalability, thereby efficiency. This is due to the possibility of simultaneous access handling and potentially reduced distance between a replica and a requesting site. All these properties are crucial to distributed services, even more to their clients, hence they are highly desirable. However, replication mechanism raises several design and implementation problems.

Replication can be achieved in a variety of ways depending on the approach to different aspects of its implementation or application. As for the general architecture, replication can be deeply embedded in the service itself, e.g. tightly integrated with the service implementation. Alternatively, it can be set up as an external mechanism intercepting interaction between clients and a service, in particular a service not designed to replicate itself. We will

Marcin Bazydło · Szymon Francuzik · Cezary Sobaniec · Dariusz Wawrzyniak
Institute of Computing Science, Poznań University of Technology, Poland

concentrate on the latter case, i.e. we assume the service is instantiated at several servers and the work of the instances is somehow coordinated by a replication mechanism in order to provide consistent view of the service state. The service may cooperate with the replication mechanism or may be even unaware of its existence. Thus, the process of replication may be transparent to both the clients and services.

One of the main challenges of replication is keeping the replicas consistent. Every modification submitted to one replica makes all others outdated. The inconsistency may be observed by clients if they switch from one replica to another or when the request is routed to another server. In order to keep the consistency, the replication infrastructure must disseminate all updates among all servers. There are two general methods of propagating updates between replicas: *state transfer* and *operation transfer* [5]. State transfer consists in tracking changes caused by modifying requests and propagating the changes (or the whole states) to other replicas. Operation transfer consists in propagation of operations and subsequent re-execution on every replica. Tracking changes of internal data structures of a service and its external storage is usually computationally expensive or even impossible to implement without tight integration with the service itself. Similarly, it is hard to integrate the whole state transferred between replicas in the case of concurrent conflicting modifications. Besides, this approach may cause significant communication overhead. Consequently, transparent (from the service point of view) replication must rest on operation transfer.

Re-execution of operations on replicas, even in the same order, does not necessary cause the same state changes, unless the service is deterministic. A service is deterministic when it produces the same results after executing the same sequence of operations with the same arguments starting from the same initial state. Replication based on operation transfer has been formulated as *state machine replication* [6] , where all requests are sent to all replicas in the same order and then deterministically processed. The model of the replication mechanism considered in this paper is similar, but it does not enforce any propagation strategy. We assume that the requests are eventually disseminated between servers, ordered, and executed. The system should guarantee *eventual* convergence of the states of replicas if the replicas are deterministic.

The main contribution of this paper is twofold. Firstly, we analyze possible sources of non-determinism that prevent application of state machine replication to different types of services (Sect. 2). Secondly, we propose decomposition of the service that will allow its replication despite non-determinism (Sect. 3). The decomposition assumes some reconstruction of the service, but we believe that this method is still simpler than embedding the replication mechanism into the service, which requires its reimplementation. Newly designed services following the proposed "design pattern" can benefit from possible optional replication without any additional costs.

In this paper we discuss services but the essence of the decomposition idea is equally applicable to other distributed environments. In fact, we assume a distributed system following the client-server style of interactions where all interactions between clients and servers can be intercepted and somehow transformed (i.e. supplemented, duplicated). It means that we expect some visibility of interactions between communicating sides, and a uniform interface to reuse the replication infrastructure. RESTful web services fulfill this assumptions so that we will use them as an example.

2 Sources of Non-determinism

In the context of replication, certain characteristics of services are particularly important. We have identified four characteristics of services that influence replication: *state*, *internal non-determinism*, *external requests*, and *spontaneous events*. To our knowledge, this is the first such review of service characteristics. Presented insights are useful for understanding issues of service replication. Identification of presented characteristics in a service is a necessary step for proper service decomposition introduced in the next section.

Let us consider a service without any state or which state is immutable. This kind of service — when replicated — is always consistent. It is very scalable as there is no need for propagation of modifications. It is also highly available as each replica can perform all tasks without communication with other replicas. Additionally, service recovery or addition of new service replicas is extremely easy. The service may generate even non-deterministic results: the clients are prepared for that, and the replicas still can work in parallel without any coordination. Unfortunately, applicability of stateless services is very limited.

Replication becomes problematic in case of stateful services. The services maintain some data that may be updated by clients. Updates submitted at one replica are not available at other replicas unless some synchronization mechanism is activated. The state may be of different type. The *service state* influences subsequent interactions of all clients. The *communication state* influences interactions of one particular client. Sometimes the communication state is called a *session*. All updates of the service state — regardless of its characteristics — must be propagated to other replicas to keep the replication transparent. In the case of the communication state it is often not viable to synchronize it with other replicas because this type of state is not shared between clients. The cost may be avoided by means of *session affinity*, i.e. by binding the client to one replica for the whole processing within a session. Another possibility consists in removing the session state at all. In this approach the servers are often called stateless but it is a shortcut of

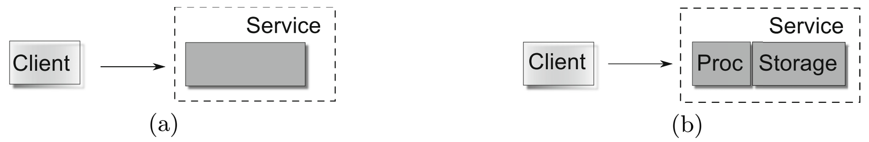


Fig. 1 Service decomposition into processing part and storage

saying that the servers do not maintain information concerning processing of individual clients. Stateless communication assumes that all information necessary for processing requests is attached to every request, which may result in high communication overhead. However, it allows to arbitrarily choose a replica for request processing as long as its state is up-to-date. The REST architectural style [4] assumes stateless communication model.

One simple approach to replication (quite popular in web servers) is to replicate only those parts of the service that are stateless. It means that it is necessary to separate from the original service (see Fig. 1a) those parts that maintain state (Fig. 1b). The stateless part, called *processing* part and denoted by "Proc" in the figures, can be next replicated. The whole state of the service is maintained at one server and shared between all replicas, thus this approach is called *shared storage* (see Fig. 2a). It is worth noting that in this approach possible non-determinism of processing parts is not an issue, because state changes generated by it are shared between all replicas. The storage may become a bottleneck in some situations which may be alleviated in some applications by introducing data partitioning (see Fig. 2b) or some form of the storage replication. The shared storage approach, however, has two important disadvantages. Firstly, it is assumed that the storage is well connected to all replica servers so that the remote access does not introduce substantial latency. Secondly, this architecture indeed improves performance but does not improve availability of the service because replicas depend on the shared storage availability.

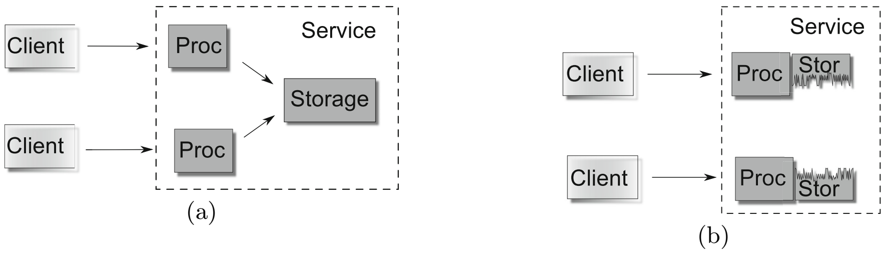


Fig. 2 Replication using shared storage

2.1 Internal Non-determinism

In deterministic service all information used for request processing come from sources which are shared by all replicas (e.g. state of the service, request content). It is easy to unintentionally make service non-deterministic by using data accessible only on local computer like local clock or counter. Nonetheless, replication system may extend the request with additional data which come from non-deterministic sources. It is perfectly valid for replication system to use such data, as long as exactly the same extended request will be disseminated to all replicas.

2.2 External Requests

Some sort of non-determinism is also introduced by external requests issued by a replica. Let us consider two main types of external requests: reads and modifications. Reads do not change the state of an external service, so they may be repeated arbitrary number of times. Still, if the request is sent by each replica individually it can lead to inconsistency between replicas. The replicas may receive different results because they have sent the requests at different moments in time. The results for particular requests may differ, due to communication or service errors, even if the external service is stateless and deterministic. As a result, the external requests should be considered as non-deterministic sources of data. The problem is even more aggravated when modifications are considered. Modifications change the state of the external service thus repeated invocations may cause accumulation of updates.

The problem of external requests may be solved by sending all outgoing traffic through a centralized proxy server which returns the same response for all repeated requests without reissuing them. However, this solution leads to centralization of processing (which impairs availability). Additionally, this solution may not work properly for optimistically replicated services. In such services, the external requests generated by the same operation on two replicas may differ due to different order of execution of operations between replicas. In such a situation it would be necessary to reissue the request.

2.3 Spontaneous Events

Most services are implemented in the client-server model. It is usually assumed that all actions performed by a service are initiated by a prior client request. It is not always the case. Services may need to perform some tasks spontaneously (e.g. periodically). The problem with spontaneous actions is that they are usually not perceived and not controlled by the replication system, thus they are not consistently scheduled. Such actions may lead to temporal inconsistencies among replicas, like in the case of garbage collection, which is perfectly acceptable for the replication system ensuring eventual consistency. However, it is also possible that they may lead to permanent inconsistency among replicas. The only uniform solution for spontaneous actions is forcing them to pass through the same channel as the requests received from clients. This way spontaneous actions become indistinguishable from regular requests of any client. This solution also improves visibility of interactions in the service.

3 Service Decomposition

In the view of previous insights we propose to classify services according to two orthogonal characteristics: state handling and determinism of processing (see Table 1). As stated in the previous section it is trivial to replicate transparently a deterministic stateless service (lower left cell in the table). Actually, replication of any stateless service (also non-deterministic) is relatively simple task as there are no consistency issues. When a stateless service is replicated it is enough to execute each request only in one replica without any further communication. Similarly, transparent replication of a stateful deterministic service (upper left cell) is relatively easy as this problem has been deeply investigated in many papers. For example state machine replication [6] assumes a system model which complies with the stateful deterministic service. However, in case of stateful non-deterministic services (upper right cell) transparent replication is hard, or even impossible to achieve.

To cope with the problem of replication of stateful non-deterministic services we propose *service decomposition* — a general "design pattern" facilitating maintaining copies of the service. It allows transparent replication of stateful non-deterministic services by improving visibility of interactions within the service. Service decomposition separates the state management from non-deterministic processing (compare arrows in Table 1). As a result of the decomposition we obtain two services instead of one. The first service, called storage, is stateful and deterministic. The second one, called agent,

Table 1 Service characteristics influencing replication

	deterministic	non-deterministic
stateful	known solutions	$\leftarrow$ difficult $\downarrow$
stateless	trivial	easy

is stateless and non-deterministic. Both those subservices fall into categories which allow transparent replication.

The idea of decomposition is illustrated in the Fig. 3. The monolithic processing part of the service is decomposed into two parts. One of the parts is still very close to the data — it is called storage. The other one accesses data through the first service, which is exactly the same way as clients access the service. The whole service is now composed of two interacting parts.

Service decomposition improves visibility of interactions. An intermediary placed before the storage will not only observe modifications issued by clients, but also those issued by the agent. This allows the intermediary to observe state modifications resulting from non-deterministic processing (including spontaneous events). For example, when the agent obtains some information from an external service and based on that modifies certain objects, the intermediary will observe incoming modification. The improvement of visibility of interactions between agent and storage not only makes transparent replication possible but also facilitates other fields of service management like monitoring[2] or state recovery[3].

Service decomposition allows for relatively straightforward replication of the storage service. Increasing number of storage replicas improves availability, fault tolerance and performance of the service. As clients access only storage its response time directly influences clients perception of the service performance.

Replication of agents, while not necessary for fault tolerance (agent may be simply restarted in case of a crash), might be useful for improving availability or performance. However, replication of agents may result in processing of the same request by each agent replica. For example if the agent sends an external request as part of its processing, the request may be sent multiple times. For the most cases it is unintended behavior. In order to prevent this, some kind of coordination between agent replicas is required. Fortunately in service decomposition scheme all agents access the same stateful deterministic service. It is relatively easy to obtain required coordination by using static (e.g. hashing) or dynamic (e.g. locking) load distribution.

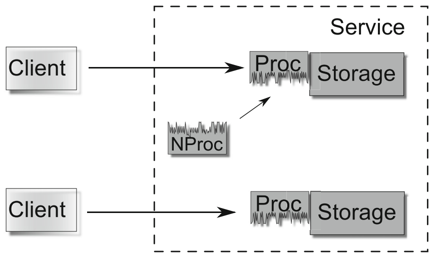


Fig. 3 The decomposed service architecture

Service decomposition has also its drawbacks. Firstly, as the agent accesses state by the uniform channel, there is an increased communication cost for obtaining information. Secondly, the agent starts its processing spontaneously, so non-deterministic requests will have higher latency comparing to the same processing in the non-decomposed service. Despite mentioned drawbacks the service decomposition proved its usefulness in our experimental implementation, which due to space limits of this paper couldn't be described in detail [1].

It is worth noting that the communication model of the decomposed service bears resemblance to queuing systems. In a queuing system all processing is done asynchronously. Each request is sent to the queue and then obtained from it by a processing module. Processing of non-deterministic requests in our scheme is very similar: the client first issues a request to the storage and then the agent asynchronously obtains data necessary for its processing and saves the results back to the storage. In our scheme we do not define whether the client request should be synchronous (and wait for the agent to finish processing) or asynchronous. However, synchronous processing may lead to deadlock if the number of connections to the storage part is limited and there is no reserved channel for the agent.

Observations similar to ours are also made in [6] and there is also a short presentation of a solution which assumes the use of a monitor for deterministic processing. The solution is somehow similar, but our solution is targeting services therefore we have deepened analysis of sources of non-determinism, and consequences of the decomposition.

Example

Let us consider replication of a lottery service. The lottery service has to receive bets from users and at certain moment it must randomly choose a winner. Receiving bets is an easy to replicate functionality as it is completely deterministic. Any number of replicas may receive bets in order to improve availability, performance and fault tolerance. Conversely, the functionality of choosing the winner is difficult to replicate. This procedure must use some non-deterministic source of information for random number generation. If each replica were to start this procedure independently, each would decide on a different winner.

To overcome this problem we decompose our service into two services following the proposed approach. In effect we obtain two services, each handling part of the functionality. Receiving bets is a deterministic stateful functionality handled by the storage. Whereas deciding winner is a non-deterministic stateless functionality provided by the agent. In the case of this service the non-determinism comes from spontaneous and random decision that must be taken uniformly for the whole service. As this decision is taken only once there is no need for agent replication. However, storage replication will be beneficial for the service as it improves availability, fault tolerance and performance.

The replication of a storage part may be easily achieved for example with state machine replication.

4 Conclusions

Replication of a deterministic service is much easier then in the case of non-deterministic one as they may be replicated transparently. Similarly, stateless services are usually trivial to replicate, whereas stateful services require carefully chosen replication algorithms to keep them consistent.

This paper discussed characteristics of a web services from the replication point of view. We have distinguished the following traits of services: *state, internal non-determinism, external requests,* and *spontaneous events.* We have observed that replication becomes difficult in the case of services which are both stateful and have some kind of non-deterministic processing. As a solution for this problem we have proposed decomposition of the service. Such decomposition separates state handling part of the service (storage) from non-deterministic part of the service (agent). Decomposition leads to improved overall visibility of interactions in the system, and allows transparent replication.

References

1. Bazydło, M.: RESTmail – Design and Implementation of E-Mail System as a RESTful Web Service. Master's thesis, Institute of Computing Science, Poznań University of Technology (September 2009)
2. Brodecki, B., Brzeziński, J., Dwornikowski, D., Kobusiński, J., Sajkowski, M., Sasak, P., Szychowiak, M.: Selected aspects of management in SOA. In: Ambroszkiewicz, S., Brzeziński, J., Cellary, W., Grzech, A., Zieliński, K. (eds.) SOA Infrastructure Tools: Concepts and Methods. UEP (2010)
3. Danilecki, A., Hołenko, M., Kobusińska, A., Szychowiak, M., Zierhoffer, P.: ReServE Service: An Approach to Increase Reliability in Service Oriented Systems. In: Malyshkin, V. (ed.) PaCT 2011. LNCS, vol. 6873, pp. 244–256. Springer, Heidelberg (2011)
4. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
5. Saito, Y., Shapiro, M.: Optimistic replication. ACM Computing Surveys 37(1), 42–81 (2005)
6. Schneider, F.: Implementing fault tolerant services using the state machine approach: A tutorial. ACM Computing Surveys 22(4), 299–319 (1990)

Partitioning Approach to Collocation Pattern Mining in Limited Memory Environment Using Materialized iCPI-Trees

Pawel Boinski and Maciej Zakrzewicz

Abstract. Collocation pattern mining is one of the latest data mining techniques applied in Spatial Knowledge Discovery. We consider the problem of executing collocation pattern queries in a limited memory environment. In this paper we introduce a new method based on iCPI-tree materialization and a spatial partitioning to efficiently discover collocation patterns. We have implemented this new solution and conducted series of experiments. The results show a significant improvement in processing times both on synthetic and real world datasets.

1 Introduction

Spatial data mining [6] is a research field that aims at discovery of regularities hidden in huge spatial datasets. One of the possible types of such regularities is called a *spatial collocation pattern* (a *collocation* in short). A collocation is defined as a subset of spatial features (e.g. police stations, schools, hospitals) whose instances are frequently located together in a spatial neighborhood. Each spatial feature attribute has a boolean nature that provides information about occurrence of this feature in a particular location in space.

The process of searching for collocation patterns is referred as the *collocation discovery problem* and has been formalized by *Shekhar and Huang* [6]. The authors proposed definitions of new measures of interest called *participation index* and *participation ratio* together with a novel algorithm *Co-location Miner*. The method used a computationally demanding *join-based* strategy to generate instances of candidate collocations. A disparate (*join-less*) approach [12] to identify collocation

Pawel Boinski · Maciej Zakrzewicz
Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 2, 60-965 Poznan, Poland
e-mail: {pawel.boinski,maciej.zakrzewicz}@cs.put.poznan.pl

instances consists in introduction of an additional data structure, called *star neighborhoods*, which serves as a data source for instances generation. In comparison with the *join-based* strategy, the number of potential collocation instances is dramatically reduced, however, some of them can form a star pattern instead of a clique, and therefore should be removed from the result set. In [7], the authors proposed a method that stores star neighborhoods in a *tree structure*. Modified procedure of generating candidates results in construction of proper collocation instances only.

Due to the complexity of the spatial data, internal data structures used by collocation discovery algorithms [4, 6, 7, 9, 11, 12] can necessitate large amounts of the system memory. *Nanopoulos et al.* consider in their work [5] real world database systems, where OLTP and data mining queries for association rules discovery are executed simultaneously. In [10], the authors noticed evolution of data mining environments towards their full integration with DBMS. This trend can impose even higher requirements for hardware resources.

The problem of limited memory has been addressed in [3] where an efficient method for performing the *join-less* algorithm has been proposed. In [2], disk materialization and specially designed search procedure for a tree (containing star neighborhoods) has been introduced. In this paper, we introduce a new method for collocation pattern mining based on the state of art algorithm improved with our novel partitioning and materialization techniques.

The structure of this paper is as follows. Section 2 contains basic definitions for the collocation discovery problem and brief description of the existing algorithm. In section 3 and 4 we present our new materialization schema and new method based on partitioning respectively. Section 5 covers conducted experiments and their results.

2 Definitions and Problem Formulation

2.1 Basic Definitions

Definition 1 (instance). Let f be a spatial feature. An object x is an **instance** of the feature f, if x is a type of f and is described by a location and unique identifier.

Definition 2 (collocation). Let F be a set of spatial features and S be a set of their instances. Given a neighbor relation R, we say that the **collocation** C is a subset of spatial features $C \subseteq F$ whose instances $I \subseteq S$ form a clique with respect to the relation R.

Definition 3 (participation ratio). The **participation ratio** $Pr(C, f_i)$ of a feature f_i in the collocation $C = \{f_1, f_2, \ldots, f_k\}$ is a fraction of objects representing the feature f_i in the neighborhood of instances of collocation $C - \{f_i\}$.

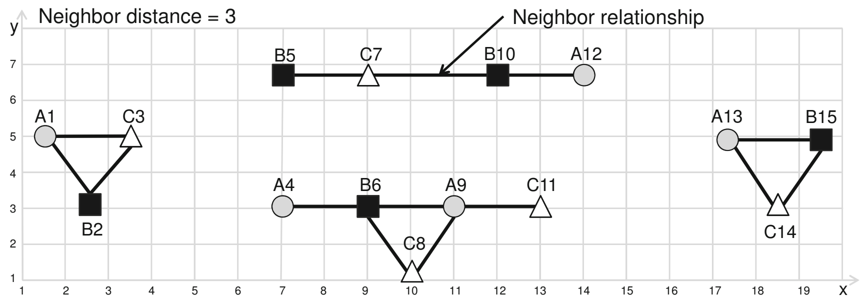


Fig. 1 Sample 2-dimensional dataset

Definition 4 (participation index). The **participation index** $Pi(C)$ of a collocation $C = \{f_1, f_2, \dots, f_k\}$ is defined as $Pi(C) = \min_{f_i \in C}\{Pr(C, f_i)\}$.

Example. Figure 1 presents an example of 2-dimensional dataset. There are three spatial features: A, B and C. The neighbor relationship is the Euclidean distance with given threshold ($d = 3$). For clarity, the graph contains edges connecting all pairs of neighbors. Each spatial feature has 5 instances. Consider a candidate collocation $c = \{B, C\}$ with the following instances: $\{B2, C3\}$, $\{B5, C7\}$, $\{B6, C8\}$, $\{B10, C7\}$, $\{B15, C14\}$. The participation ratio of the feature B in the collocation c is equal to 1 because all five instances of the feature B take part in instances of c. The participation ratio of the feature C is equal to $\frac{4}{5}$. Finally, $Pi(c) = \frac{4}{5}$.

Lemma 1. *The participation ratio and participation index are monotonically non-increasing with increases in the collocation size.*

Definition 5 (star neighborhood). Given a spatial object $o_i \in S$ whose feature type is $f_i \in F$ the **star neighborhood** of o_i is defined as a set of spatial objects:

$$T = \{o_j \in S \parallel o_i = o_j \vee (f_i < f_j \wedge R(o_i, o_j))\}$$

where $f_j \in F$ is the feature type of o_j and R is a neighbor relationship.

Definition 6 (star instance). Let $I = \{o_1, \dots, o_k\} \subseteq S$ be a set of spatial objects whose feature types $\{f_1, f_2, \dots, f_k\}$ are different. If all objects in I are neighbors to the first object o_1, I is called a **star instance** of collocation $C = \{f_1, f_2, \dots, f_k\}$.

2.2 *Problem Formulation*

The collocation pattern mining is defined as follows. Given (a) a set of spatial features $F = \{f_1, f_2, \dots, f_n\}$ and a set of their instances $S = S_1 \cup S_2 \cup \dots \cup S_n$ where $S_i (1 \le i \le n)$ is a set of instances of feature $f_i \in F$ and each instance that belongs to

S contains information about its feature type, instance id and location; (b) a neighbor relationship R over locations; (c) a minimum prevalence threshold (*min_prev*) and minimum conditional probability threshold (*min_cond_prob*); (d) a size of the available memory, find efficiently (with respect to the memory constraint) a correct and complete set of collocation rules with participation index $\geq$ *min_prev* and conditional probability $\geq$ *min_cond_prob*. We assume that relation R is a distance metric based neighbor relationship with a symmetric property and spatial dataset is a point dataset.

2.3 iCPI-Tree Algorithm

In [8], *Wang et al.* proposed a tree structure to store neighbor relationships and identify collocation instances recursively from it. Compared with the *join-less* method [12], this approach eliminates an additional, computationally demanding filtering step required to check whether a particular instance holds the clique definition. However, this algorithm gives up the *Apriori*-like [1] strategy for generating new candidates, which in many cases can significantly reduce the number of candidates and therefore increase the overall performance. This problem has been addressed in [7]. The authors introduced a method based on a new structure called *iCPI-tree*. This method leverages both the *Apriori*-like strategy and the concept of a tree structure. A short description of this method is as follows.

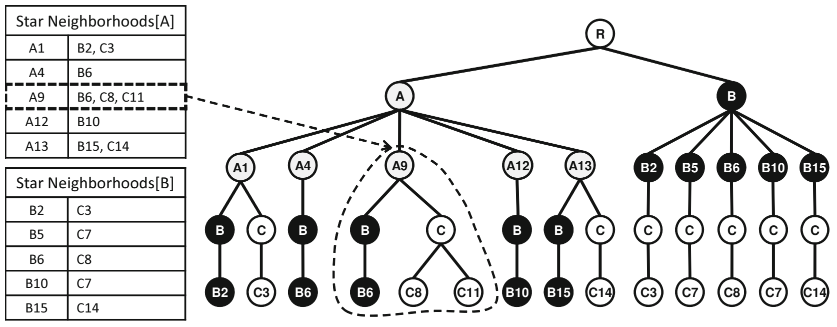
Step 1. *Convert a spatial dataset to a set of spatial ordered neighbor relationships between instances* - in this step, star neighborhoods are created. Each star has a central object, i.e. an object which has a neighbor relationship with all other objects. After sorting (with respect to the spatial feature and object identifier), star neighborhoods form a set of ordered spatial neighbor relationships between instances.

Example. Consider 2-dimensional sample data presented in Fig. 1. For each data point, star neighborhood is created (Fig. 2), e.g. $\{A9, C11, B6, C8\}$ for point $A9$, and then sorted with respect to the spatial feature and instance id. The final set $\{A9, B6, C8, C11\}$ constitutes an ordered neighbor relationship between the objects.

Step 2. *Generate the iCPI-tree of the set of spatial ordered neighbor relationships* - from the set of ordered spatial neighbor relationships, the *iCPI-tree* is iteratively created.

Example. Consider the aforementioned ordered star neighborhood instance in Fig. 2 $\{A9, B6, C8, C11\}$. The first element of this set has neighbor relationship with the rest of the elements. It is represented as a branch of the *iCPI-tree* under the node with feature A. Sub-branches B and C determine neighbors with the appropriate spatial feature ($B6$ and $C8, C11$ respectively). See [7] for the *iCPI-tree* construction details.

Step 3. *Iteratively mining collocation patterns* - this step consists of three tasks: generation of candidate collocations (using the *Apriori* strategy), searching for their



Star Neighborhoods[A]	
A1	B2, C3
A4	B6
A9	B6, C8, C11
A12	B10
A13	B15, C14

Star Neighborhoods[B]	
B2	C3
B5	C7
B6	C8
B10	C7
B15	C14

Fig. 2 Sample structure of the *iCPI-tree* (*Improved Candidate Pattern Instance Tree*)

instances in the *iCPI-tree* and filtering candidates with respect to the minimal prevalence threshold. At the beginning, all one-element candidates are considered as collocations with prevalences equal to 1. Iterations start with $k = 2$ size candidates. This step lasts as long as there is a possibility to generate new candidates using the *Apriori* strategy.

Example. Given a candidate collocation $\{A,B,C\}$ in Fig. 2, its instances can be expanded from the appropriate size $k-1$ instances discovered in the previous iteration. For example, instance $\{A1,B2\}$ can be expanded to $\{A1,B2,C3\}$ as a result of $C3$ being a common neighbor of $A1$ and $B2$. However, the instance $\{A4,B6\}$ cannot be expanded with $C8$ since $C8$ is not a neighbor of $A4$.

Step 4. *Generate collocation rules from discovered collocation patterns* - this step can be executed after each iteration in the third step or at the end of the algorithm.

3 iCPI-Tree Materialization

In [3] the *iCPI-tree* algorithm has been compared with two other algorithms based on the joinless strategy. Performed experiments measured the efficiency of the collocation mining task in a limited memory environment. Although the *iCPI-tree* method turned out to be the most effective one, there was a dramatic increase of processing times with lower values of the available memory. The cause of the poor performance lied in the additional I/O in the form of operating system paging and swapping.

Additionally, the tests have shown that *iCPI-tree* can reach size bigger than the amount of the available memory. The *iCPI-tree* structure is based on the star neighborhoods, which can take up more space than the underlying database containing spatial objects. Moreover, the *iCPI-tree* requires huge number of pointers to maintain easy access to neighbors. We proposed a materialization of tree structure (*MiCPI-tree - Materialized iCPI-tree*) and an optimized tree search method [3]. In

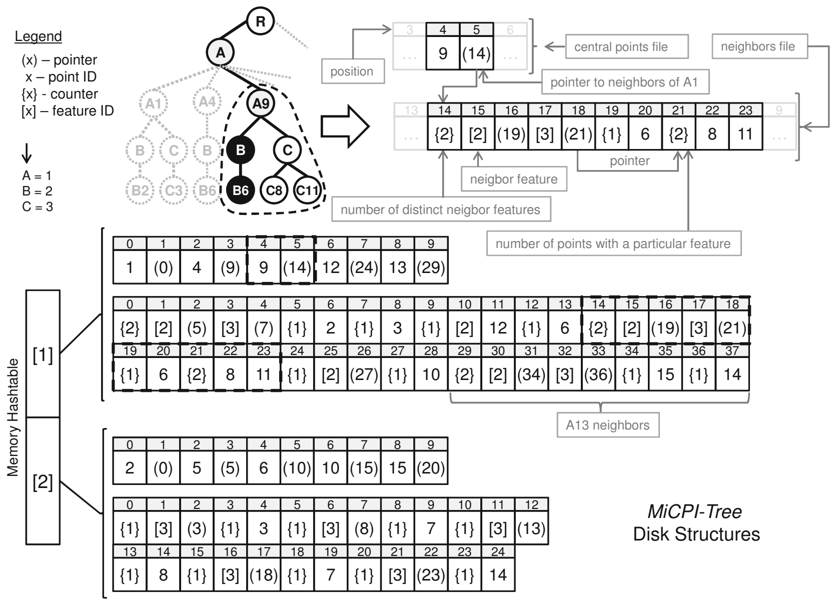


Fig. 3 The physical organization of the (*materialized iCPI-tree*)

this work we employ this strategy to store trees on disk, however we introduce a supporting structure designed for fast retrieval of tree nodes from the disk.

Figure 3 presents the physical representation of the *iCPI-tree* from Fig. 2. For each spatial feature f (except the last one) there are two data files that can be accessed using a hash table:

1. *Central points file* - contains information about first elements (denoted by o_x) from *ordered neighbor relationship sets* of instances with the feature f, i.e. about "central" points off star neighborhoods for the feature f. The file consists of pairs $\{o_{x_{id}}, pn_{id}\}$ sorted according to the increasing identifier. By $o_{x_{id}}$ we understand the identifier of the element o_x and pn_{id} denotes a pointer to the position in the appropriate neighbors file.
2. *Neighbors file* - contains information about neighbors of elements stored in the central points file. Let $N_x = \{o_1, o_2, \ldots, o_n\}$ denote the set of n neighbors of the element o_x, let elements from N_x have spatial features from set $F_x = \{f_1, f_2, \ldots, f_k\}$ and let $I_{f=y}$ denote instances with feature $y \in F$ sorted according to the increasing identifier. The structure of the file entry for N_x is as follows: $\{|F_x|, f_1, p_1, \ldots, f_k, p_k, |I_{f=1}|, I_{f=1_{id}}, \ldots, |I_{f=k}|, I_{f=k_{id}}\}$ where p_j is a pointer to the position in the current file where neighbors with feature $j \in F_x$ are stored.

Example. Given the ordered neighbor relationship set $\{A9, B6, C8, C11\}$ according to the aforementioned denotations $o_x = A9$, $N_x = \{B6, C8, C11\}$, $F_x = \{B, C\}$, $I_B = \{B6\}$, $I_C = \{C8, C11\}$. Assuming that feature B is represented by key 2 and

feature by key 3 , a new entry in the neighbors file would have the following form $\{|\{B,C\}|,B,p_B,C,p_C,|\{B6\}|,\{B6\}_{id},|\{C8,C11\}|,\{C8,C11\}_{id}\}=$ $\{2,2,p_B,3,p_C,1,6,2,8,11\}$
where values of pointers p_B and p_C result from the current positions in the data file. In Fig. 3 p_B is equal to 19 and p_C is equal to 21.

4 Partitioning Space

In this section we present our new algorithm for the collocation discovery problem. This algorithm can be used for efficient execution of collocation mining tasks in environment with a limited memory.

The original *MiCPI-tree* algorithm utilizes cache memory (with *LRU* strategy) to store neighbor entries. In the ideal circumstances, the cache memory can hold all objects and no I/O transfer is required. When the memory size is not sufficient, there is necessity to materialize some nodes on the disk, to ensure that they can be retrieved if needed. In [3] an optimized search procedure has been proposed for increasing the cache hit ratio. In general, we can say that the main goal of *MiCPI-tree* algorithm is to minimize disk I/O transfer. Conversely, a new *PMiCPI-trees* (*Partitioned MiCPI-trees*) method aims at the minimization of the processing time by reducing the number of disk I/O operations, although the total disk I/O transfer is less relevant. In *PMiCPI-trees* algorithm, when the memory is limited and we are not able to keep all tree nodes in memory at the same time, the whole process is being split into multiple phases. Each phase consists in loading and processing a part of the *MiCPI-tree*. We propose to partition the *MiCPI-tree* into a set of sub-trees in such a way that each sub-tree fits in memory. The collocation mining task is then executed iteratively on each sub-tree and results are merged. The key component of our approach is to preserve all of the neighborhood information during the partitioning step. With this end in view, the partial sub-trees can share a part of the set of ordered neighbor relationships. In the next paragraphs we present the method for building the aforementioned sub-trees set with *plane sweep* strategy and the *PMiCPI-trees* algorithm.

PMiCPI-Trees Construction

The step of creating a set of *PMiCPI-trees* (pseudocode in Alg. 1) consists in reading the data file, detecting ordered neighbor relationship sets and splitting a tree when the given memory limit is reached. Our solution is based on the well-known *plane sweep* strategy. In general, a plane sweeps across the space, only to stop at data points (lines 18, 20). An ordered buffer, denoted by B_1, is maintained to store points lying in the strip (called window) of given width (neighbor relationship criterion). In Fig. 4, the sliding window is at the position $x = 18.5$ and the buffer B_1 extends from 15.5 to 18.5.

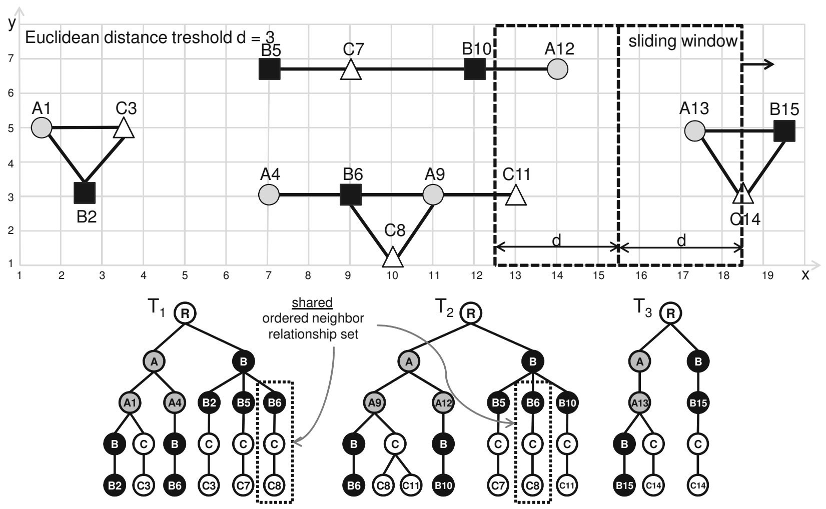


Fig. 4 The partitioning of the *MiCPI-tree*

Ordered neighbor relationship sets are created in the buffer B_1 (line 28) and are stored in the tree when their central points are out of the $d - size$ sliding window. This operation is referred as the pruning step (line 21). We extended this basic approach by adding a second buffer B_2 (area from 12.5 to 15.5 in the example) with the same size and placed directly after the original B_1 buffer. The elements from buffer B_1 are moved to the buffer B_2 in the pruning step (line 21). When the available memory size reaches zero (pruning step returns "false"), the next tree is created (line 24) with the initial set of shared, ordered neighbor relationships from the buffer B_2 (line 25). The idea behind this two-level buffering is to preserve, in one tree, all ordered neighbor relationship sets that are required to form particular collocation instance. For example, consider trees T_1 and T_2 in Fig. 4. Given a candidate collocation $\{A, B, C\}$, an instance $\{A9, B6, C8\}$ can be constructed from T_2 although the relationship $\{B6, C8\}$ belongs to T_1. Nevertheless, due to the two-level buffering, the relationship $\{B6, C8\}$ is shared between T_1 and T_2 and therefore no information about neighbors is lost. Finally, when central points of elements stored in B_2 are out of the $2d - size$ sliding window, they are simply removed from B_2 (line 27).

The Main PMiCPI-Trees Algorithm

The pseudocode for the main algorithm has been shown in Alg. 1. *PMiCPI-trees* method consists of six steps:

Step 1. *Generate a set of PMiCPI-trees for the input dataset (line 3)* - this procedure has been described in the previous paragraph.

Algorithm 1. The PMiCPI-trees algorithm with the plane sweep strategy

```
    Variables:
        $F$ - set of spatial features
        $S$ - set of instances
        $dist$ - max. neighbor distance
        $MEMS$ - available memory
        $min\_prev$ - min. prevalence
        $P_k$ - set of collocations
        $C_k$ - set of candidate collocations
        $SI_k$ - set of instances
        $Inst_k$ - set of distinct instances
        $T$ - set of PMiCPI-trees
    Method:
 1: procedure PMiCPI($F$, $S$, $dist$, $min\_prev$, $MEMS$)
 2:     $k = 1$
 3:     $T = \text{genTrees}(F, S, dist, MEMS)$
 4:     $P_k = \text{genOneElementCollocations}(F, S)$
 5:     while ($P_k \neq \emptyset$) do
 6:         $C_{k+1} = \text{genCandidates}(P_k)$
 7:         for each $t \in T$
 8:             $SI_{k+1} = \text{genInstances}(t, Inst_k, C_{k+1})$
 9:             $Inst_{k+1} = \text{merge}(Inst_{k+1}, SI_{k+1})$
10:         end
11:         $P_{k+1} = \text{getPreval}(C_{k+1}, Inst_{k+1}, min\_prev)$
12:         $k = k + 1$
13:     end while
14:     return $\bigcup(P_1, P_2, ..., P_{k-1})$
15: end procedure

    Variables:
        $F$ - set of spatial features
        $S$ - set of instances
        $dist$ - max. neighbor distance
        $MEMS$ - available memory
        $D$ - set of sorted instances
        $T_j$ - PMiCPI tree
        $T$ - set of PMiCPI trees
        $B_1$ - buffer 1st level
        $B_2$ - buffer 2nd level
    Method:
16: procedure GENTREES($F$, $S$, $dist$, $MEMS$)
17:     $j = 1; B_1 = \emptyset; B_2 = \emptyset$
18:     $D = \text{sortData}(F, S)$
19:     $T_j = newTree$
20:     for each $p \in D$
21:         if prune($p$, $B_1$, $dist$, $MEMS$) = $false$
22:             $T = T \cup \{T_j\}$
23:             $j = j + 1$
24:             $T_j = newTree$
25:             insert($T_j$, $B_2$)
26:         end if
27:         prune($p$, $B_2$, $dist$)
28:         insert($B_1$, $p$)
29:     end
30:     $T = T \cup \{T_j\}$
31:     return $T$
32: end procedure
```

Step 2. *Generate one element collocations (line 4)* - all spatial features are collocations with the prevalence equal to 1 (by the definition of the participation index); $k = 1$.

Step 3. *Generate* $k+1$ *element candidate collocations (line 6)* - $k+1$ size candidate collocations can be generated from prevalent k size collocations by applying the *apriori_gen* method [1].

Step 4. *Iteratively search for candidate collocations instances (lines 7-10)* - for each *PMiCPI-tree* instances of given candidate collocations are constructed. Due to the existence of shared branches in the set of *PMiCPI-trees*, there is a possibility that for a given candidate collocation, the same instances can occur multiple times (constructed from different trees). There is a necessity to merge those repetitious instances into a distinct instance. For better efficiency of the described step, for each candidate collocation instance, the identifier of the source tree should be stored. Therefore, in the next iterations, it will be possible to process only instances derived from a particular tree.

Step 5. *Selection of the prevalent collocations (line 11)* - for each candidate collocation C_{k+1} the participation index is calculated from the set of instances I_{k+1}.

Step 6. *Iteratively mine collocation patterns (lines 5-13)* - While the set of prevalent collocations is not empty (therefore a new candidate set can be generated), repeat steps 3 - 5. The value of k is increased by one after each iteration.

5 Experimental Evaluation

We conducted a series of experiments using both synthetic and real world[1] datasets. Hardware: Linux workstation equipped with 1800MHz Athlon CPU, 2GB RAM and 120GB 7200RPM HDD. Parameters for experiments on the synthetic data: the number of spatial features [15-50], the minimum prevalence threshold: [25%-40%], and the neighbor distance [10-20]. Parameters for experiments on the real world data: number of spatial features 20; distance threshold [2-8] units; prevalence threshold [0.25-0.55].

We have examined the performance of *MiCPI-tree* and *PMiCPI-trees* algorithms. In the first series of the experiments (Fig. 5(a)), we present averages of results from multiple runs executed on the synthetic dataset. The average size of the available memory was equal to 70% of the required size. Additionally, we present results from the original *iCPI-tree* method. With no special structures to handle limited resources, the performance of the *iCPI-tree* algorithm decreased substantially. Our new solution performs better than *MiCPI-tree* for all tested sizes of input datasets.

In the second series of experiments we executed collocation mining tasks on the real world dataset. Figure 5(b) (logarithmic scale) presents how the performance of the algorithm changes with the limited memory. One can notice, that even with quite large amount of the available memory (more than 90% of the required one), the performance drops rapidly. With limited memory, the *MiCPI-tree* algorithm has to store some nodes on disk. Therefore the search procedure has to look up the cache memory. If the particular element is not in the cache, disk must be accessed. Moreover, if wanted element does not exist (i.e. there is no neighbor relationship), the disk is still searched. In contrast, the *PMiCPI-trees* method perform searches only in memory. This results in 100% cache hit ratio, while the hit ratio drops significantly for the *MiCPI-tree* method with decreasing size of the available memory (Fig. 5(c)).

Figure 5(d) and 5(e) present how disk is accessed when there is a lack of memory to store whole neighborhoods. The high number of disk I/O calls (Fig. 5(d)) for the *MiCPI-tree* algorithm arises from multiple searches performed on disk (also for non-existing elements). The disk I/O transfer (Fig. 5(e)) is similar for both methods however it increases more rapidly with decreasing cache size in the *MiCPI-tree* method.

Finally, in Fig. 5(f), we present how the performance changes with the density of the data. With increasing threshold for the neighborhood size (1 unit is approx. 111 meters), the number of ordered neighbor relationship sets rises dramatically. Presented results were gathered for two memory thresholds: 40% and 80% of the required size. The *PMiCPI-trees* method is more efficient than competing algorithm and is much less sensitive to the size of the required memory.

The conducted experiments show that in environments with a limited memory our new method performs much better than the original solutions. The performance gap increases with the decreasing the size of the available memory.

[1] Spatial data acquired from the *OpenStreetMap Project* (http://www.openstreetmap.org).

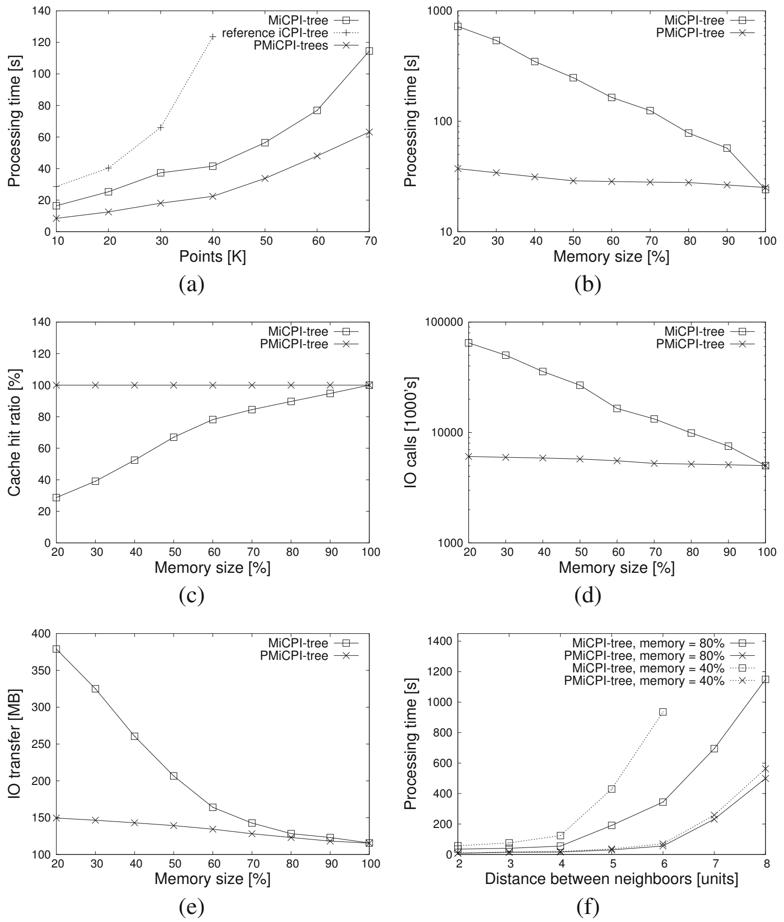


Fig. 5 The results of experiments over synthetic (a) and real world (b, c, d, e, f) datasets

6 Summary

We have addressed the problem of spatial collocation mining in a limited memory environment. The popular *iCPI-tree* method requires multiple searches over a tree structure whose size can exceed the size of the available memory (especially when a data mining task is executed along with other OLTP or OLAP queries). In this paper, we have presented a new method, called *PMiCPI-trees*, designed to avoid expensive disk searches by partitioning the tree into a set of overlapping sub-trees in such a way that each sub-tree fits in memory. The results of the conducted experiments, both on synthetic and real world datasets, show that our method performs much better than the existing solutions.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
2. Boinski, P., Zakrzewicz, M.: Hash Join Based Spatial Collocation Pattern Mining. Foundations of Computing and Decision Sciences 36(1), 3–15 (2011)
3. Boinski, P., Zakrzewicz, M.: Collocation Pattern Mining in Limited Memory Environment Using Materialized iCPI-tree. In: Cuzzocrea, A., Dayal, U. (eds.) Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2012). LNCS. Springer (accepted to publish, 2012)
4. Huang, Y., Shekhar, S., Xiong, H.: Discovering Co-location Patterns from Spatial Datasets: A General Approach. IEEE Transactions on Knowledge and Data Engineering 16, 472–485 (2004)
5. Nanopoulos, A., Manolopoulos, Y.: Memory-Adative Association Rules Mining. Inf. Systems 29(5), 365–384 (2004)
6. Shekhar, S., Huang, Y.: Discovering Spatial Co-location Patterns: A Summary of Results. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 236–256. Springer, Heidelberg (2001)
7. Wang, L., Bao, Y., Lu, J.: Efficient Discovery of Spatial Co-Location Patterns Using the iCPI-tree. The Open Information Systems Journal 3(2), 69–80 (2009)
8. Wang, L., Bao, Y., Lu, J., Yip, J.: A New Join-less Approach for Co-location Pattern Mining. In: Wu, Q., He, X., Nguyen, Q.V., Jia, W., Huang, M.L. (eds.) CIT, pp. 197–202. IEEE, Sydney (2008)
9. Wang, L., Zhou, L., Lu, J., Yip, J.: An Order-clique-based Approach for Mining Maximal Co-locations. Inf. Sci. 179(19), 3370–3382 (2009), doi:10.1016/j.ins.2009.05.023
10. Wojciechowski, M., Zakrzewicz, M.: Data mining query scheduling for apriori common counting. In: Barzdins, J. (ed.) Proceedings of the Sixth International Baltic Conference on Databases and Information Systems (DB&IS 2004). Scientific Papers. University of Latvia, vol. 672, pp. 270–281. University of Latvia (2004)
11. Yoo, J.S., Shekhar, S.: A Partial Join Approach for Mining Co-location Patterns. In: Pfoser, D., Cruz, I.F., Ronthaler, M. (eds.) GIS, pp. 241–249. ACM (2004)
12. Yoo, J.S., Shekhar, S., Celik, M.: A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results. In: Proceedings of the IEEE International Conference on Data Mining, pp. 813–816. IEEE Computer Society, Washington (2005)

Towards the Automated Business Model-Driven Conceptual Database Design

Drazen Brdjanin and Slavko Maric

Abstract. The paper presents an UML-business-model-driven approach to the automated design of the initial conceptual database model. The proposed approach is based on: (i) extraction of characteristic concepts from the business model consisting of a finite set of UML activity diagrams representing the business processes of the whole enterprise, and (ii) automated generation of the UML class diagram representing the target conceptual model. The implemented two-phase automatic generator iterates through the source business model, processes all activity diagrams and generates the target class diagram. The classes are being generated in the first phase, while the class associations are being added in the second phase. The application of the implemented generator is illustrated on the simplified real business model.

Keywords: Business Model, Activity Diagram, Class Diagram, Conceptual Model, UML, ADBdesign, Topcased.

1 Introduction

The automatization of *conceptual database model* (CDM) design has been the subject of research for many years. Starting with Chen's eleven heuristic rules [5] for the translation of information requirements specified in a natural language into an E-R diagram, a lot of research has been done in the field of *natural language processing* (NLP) on extracting knowledge from requirements specifications and automating CDM design. At present, a natural language is commonly used for requirements

Drazen Brdjanin · Slavko Maric
University of Banja Luka, Faculty of Electrical Engineering,
Patre 5, 78000 Banja Luka, Bosnia and Herzegovina
e-mail: {bdrazen,ms}@etfbl.net

specifications and the majority of approaches to automated CDM design are NLP-based. The effectiveness and limitations of these approaches are usually closely related to the source language since they depend on the size and complexity of the grammar used and the scope of lexicon. Consequently, the utilization of NLP-based approaches is questionable for languages with complex morphology.

Currently, there are several non-NLP-based alternatives taking *business process model* (BPM) [6, 11, 1, 10, 4] as the starting basis, but with modest achievements in automated CDM generation due to: (i) different and/or non-harmonized notations for business process modeling and database design, that usually don't conform to the same or common metamodel, (ii) insufficiently identified semantic capacity of BPM for automated CDM design, and (iii) lack of formal transformation rules for automated CDM design based on BPM. Additionally, all previous papers consider only one single BPM, although the enterprise model contains a finite set of BPMs representing all business processes in the enterprise. A more detailed overview of the related work has been provided in our recent paper [3].

Inspired by [6] and some subsequent proposals [1, 9, 10, 4, 2], the semantic capacity of typical business process patterns in BPM represented by the UML activity diagram (AD) has been identified and the formal rules for automated design of the initial CDM represented by the UML class diagram (CD) have been defined in [3]. The proposed approach is based on: (i) automatic extraction of business entities (*participants* and *objects*) from the source AD, and (ii) automatic generation of corresponding classes and their associations (*participant-object* and *object-object*) in the target CD. In this paper we extend the scope of the generator (named ADBdesign) and cover the whole business model containing a finite set of ADs.

The rest of the paper is structured as follows. The second section provides necessary assumptions and definitions related to the source business model. In the third section we present the formal rules and process of automated CDM generation. The fourth section presents experimental results. The fifth section concludes the paper.

2 Source Business Model

In this paper we assume that the source business model, denoted by *BM*, is a collection that contains a finite number of detailed activity diagrams

$$BM = \{DAD_1, \ldots, DAD_i, \ldots, DAD_n\},$$

where each *detailed activity diagram*, denoted by *DAD*, represents corresponding business process at *complete* level (related excerpt from the UML superstructure [8] is shown in Fig. 1), i.e.

- Each step in the realization of the given business process is represented by a corresponding *action node* (`OpaqueAction`) and will be shortly referred to as *action* in the rest of the paper.
- Each action is performed by a particular business process participant (shortly referred to as *participant*) that plays some business role modeled by a corresponding

activity partition (usually called *swimlane*). Let P_i and A_i be sets of *participants* and *actions* in business process represented by DAD_i, respectively. The fact that an action $a \in A_i$ is performed by some participant $p \in P_i$ is represented in both corresponding DAD_i elements, i.e. the `inPartition` attribute of the corresponding action contains the swimlane's identifier, while the `node` attribute of the given swimlane contains the identifiers of all actions performed by the given participant. More formally:

$$\forall p \in P_i, \forall a \in A_i \mid a \in node(p) \quad \Rightarrow \quad inPartition(a) = \{p\}.$$

- Each action may have a number of inputs and/or outputs represented by *object nodes* (`CentralBufferNode`) that can be in different *states* in the given business process. In the rest of the paper they are referred to as *input objects* and *output objects*, respectively. Let O_i be a set of *objects* in business process represented by DAD_i, where $O_i^I \subseteq O_i$ and $O_i^O \subseteq O_i$ constitute sets of input and output objects in the given business process, respectively.
- Objects and actions are connected with *object flows* (`ObjectFlow`). An object flow is a kind of *activity edge* which is directed from an input object toward the corresponding action (*input object flow*) or from an action toward the corresponding output object (*output object flow*). Let F_i be a set of *object flows* in business process represented by DAD_i, where $F_i^I \subseteq F_i$ and $F_i^O \subseteq F_i$ constitute sets of input and output object flows in the given business process, respectively. The fact that an action $a \in A_i$ has an input object $io \in O_i^I \subseteq O_i$ is represented in all three corresponding DAD_i elements, i.e. the `source` and `target` attributes of the given input object flow contain the identifiers of the given object and action, respectively, while the `outgoing` attribute of the object and the `incoming` attribute of the action contain the identifier of the given input object flow. The fact that an action $a \in A_i$ has an output object $oo \in O_i^O \subseteq O_i$ is similarly represented. More formally:

$$\forall io \in O_i^I, \forall a \in A_i, \forall if \in F_i^I \mid if \in outgoing(io) \land if \in incoming(a)$$
$$\Rightarrow \quad source(if) = io \land target(if) = a$$

  and

$$\forall a \in A_i, \forall oo \in O_i^O, \forall of \in F_i^O \mid of \in outgoing(a) \land of \in incoming(oo)$$
$$\Rightarrow \quad source(of) = a \land target(of) = oo.$$

- Each object flow has a *weight* attribute, whose value is by default one. In UML semantics, the object flow weight represents the minimum number of *tokens* that must traverse the edge at the same time. We assume that the weight of an object flow represents the total number of objects required for an action if they are input objects, or the total number of objects created in an action if they are output objects. An unlimited weight $(*)$ is used if the number of input/output objects is not a constant.

Figure 2 depicts the sample source business model containing four simplified DADs representing the four members-related business processes in (university) library. Although simplified, it is sufficiently illustrative to cover the most important concepts and basic rules for automated CDM design.

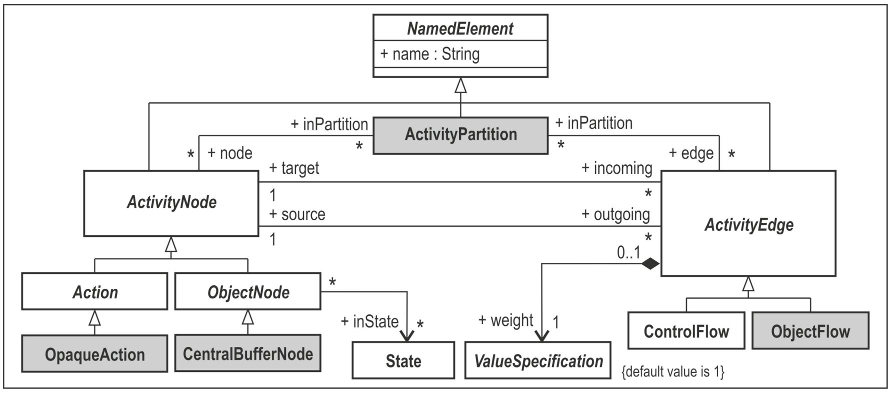


Fig. 1 UML metamodel [8] excerpt used for DAD representation

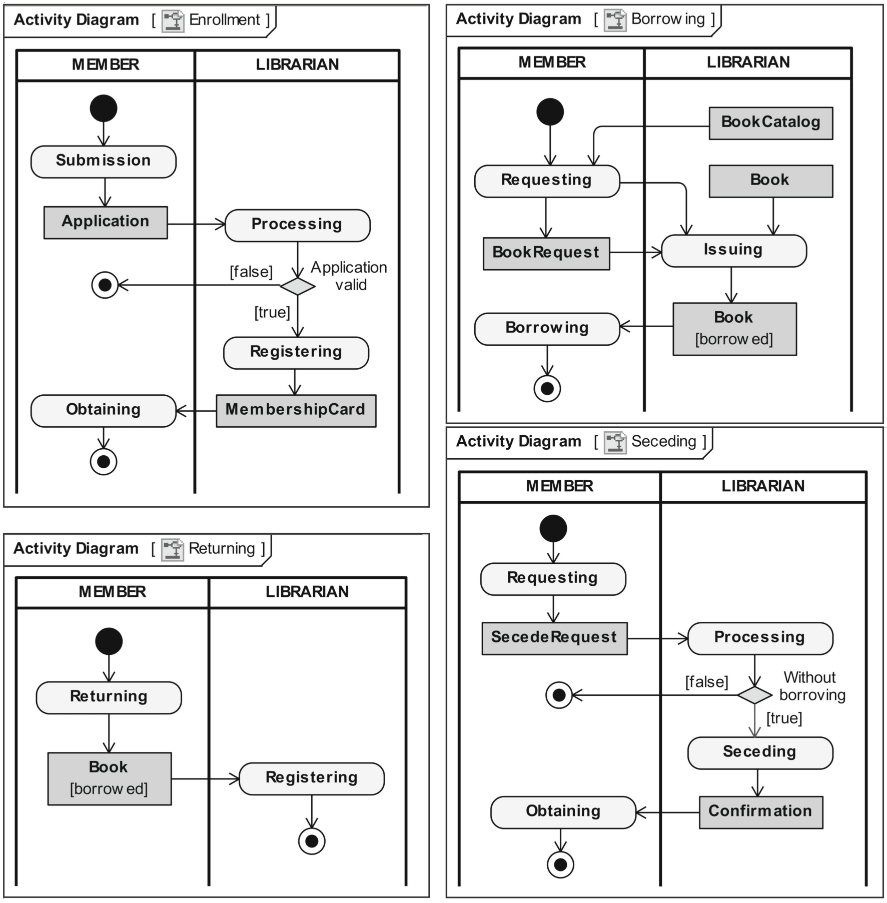


Fig. 2 Sample source business model

3 CDM Generator

We use the UML CD to represent the CDM (related excerpt from the UML infrastructure [7] is shown in Fig. 3). Let E and R be sets of *classes* (*entity types*) and their *associations* (*relationships*) in the target CDM. Since we are currently focused on automated generation of proper structure of the target model: (i) each generated class $e \in E$ will (if necessary) contain only one `ownedAttribute` named `id`, which represents a primary key, and (ii) each generated association $r \in R$ will be a binary association, whose two `memberEnd` attributes represent `source` and `target` association ends with the appropriate multiplicities.

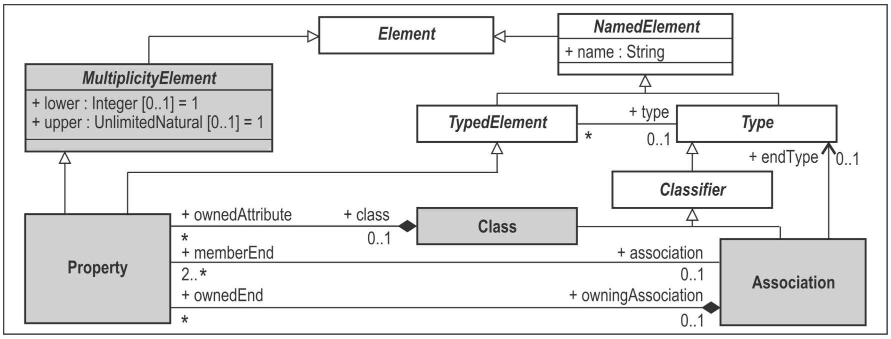


Fig. 3 UML metamodel [7] excerpt for CDM representation

3.1 Automated Generation of Classes

There are three important bases [3] for automated generation of classes in the target CDM: (i) *participants*, (ii) *objects*, and (iii) *activations of existing objects*.

Participants. Each business role in some business process is represented by a corresponding swimlane in the respective DAD. The same participant also may participate in different processes in the business system, i.e. the swimlanes of the same name may occur in several DADs. The total set of different types of participants in the whole system is given with $P = \bigcup P_i$, where P_i is the set of different types of participants in business process represented by DAD_i.

Total set E^P of classes generated for all $p \in P$ (for all participants in the source business model) is as follows:

$$E^P = \left\{ e_P \in E \,\middle|\, e_P = T_P(p),\ p \in P \right\},$$

where the T_P rule, that maps participant $p \in P$ into class $e_P \in E^P$, is:

$$T_P\big(p\big) \overset{def}{=} e_P \,\big|\, \big(name(e_P) = name(p)\big).$$

For the sample business model, $E^P = \{\texttt{MEMBER}, \texttt{LIBRARIAN}\}$.

Objects. During the execution of a business process, participants perform actions. Each action may have a number of input and/or output objects that can be in different states. The same type of objects may occur in several DADs. For example, in the sample business model, objects of the `BOOK` type are used in the `Borrowing` and `Returning` process, as well. The total set of different types of objects in the whole system is given with $O = \bigcup O_i$, where O_i is the set of different types of objects in business process represented by DAD_i.

Total set E^O of classes generated for all $o \in O$ is as follows:

$$E^O = \Big\{ e_O \in E \;\big|\; e_O = T_O(o),\ o \in O \Big\},$$

where the T_O rule, that maps object $o \in O$ into class $e_O \in E^O$, is:

$$T_O(o) \overset{def}{=} e_O \;\big|\; \big(name(e_O) = name(o)\big).$$

For the sample business model, $E^O = \{$`Application`,`MembershipCard`, `BookRequest`,`BookCatalog`,`Book`,`SecedeRequest`,`Confirmation`$\}$.

Activations of existing objects. According to [3], the *activation* represents the fact that some *existing*[1] object(s) constitute(s) the input in some action that changes its/their state. For example, in the sample model, the `Issuing` action represents the activation of the `Book` object(s). The librarian issues the required book and changes its state into `borrowed`. Such activated existing object may constitute the input object in some other action in the same and/or some other business process(es) in the business system. In the sample business model, the activated `Book` objects constitute the inputs in the `Borrowing` action after the activation in the `Borrowing` process, but also occur in the `Returning` process as the output of `Returning` action, as well as the input in `Registering` action. The naming of the activation classes, as suggested in [3] (concatenation of object name and action name), is not suitable if activated objects are being used in different processes. It is more suitable that the name of activated object contains the state name instead of action name.

Let X_i^A be a set of *activations* $\langle p,a,o\rangle$ in a business process represented by DAD_i, where $p \in P_i$ is participant performing action $a \in A_i$ on existing object $o \in O_i$. Total set of activations in the whole system is given with $X^A = \bigcup X_i^A$. Total set E^A of classes generated for all activations $\langle p,a,o\rangle \in X^A$ is:

$$E^A = \Big\{ e_A \in E \;\big|\; e_A = T_A(\langle p,a,o\rangle),\ \langle p,a,o\rangle \in X^A \Big\},$$

where the T_A rule, that maps activation $\langle p,a,o\rangle \in X^A$ into class $e_A \in E^A$, is:

$$T_A\big(\langle p,a,o\rangle\big) \overset{def}{=} e_A \;\big|\; \big(name(e_A) = concat(name(o), state(o))\big).$$

For the sample model, $E^A = \{$`Book_borrowed`$\}$.

Finally, total set E of classes in the target CDM is $E = E^P \cup E^O \cup E^A$.

[1] Existing objects are objects that are not created in the given business process, but in some other process. For example, in the sample business model, objects of the `Book` type are existing objects in the `Borrowing` process (each object of the `Book` type is generated in the `BookPurchasing` process, but not in the `Borrowing` process).

3.2 Automated Generation of Associations

We distinguish two different kind of class associations in the target CDM: (i) *participant-object* associations (associations between classes representing *participants* and *objects*), and (ii) *object-object* associations (associations between classes representing *objects*).

***Participant-object* associations.** There are several typical bases for automated generation of *participant-object* associations [3] that are related to: (i) *creation and subsequent usage of generated objects*, and (ii) *activation of existing objects and subsequent usage of activated objects*.

Creation and subsequent usage of generated objects. Let G_i^C be a set of triplets $\langle p,a,o\rangle$, where triplet $\langle p,a,o\rangle \in G_i^C$ represents the fact that action $a \in A_i$, performed by participant $p \in P_i$, creates object $o \in O_i$ in the business process represented by DAD_i. Total set G^C for the whole system is $G^C = \bigcup G_i^C$.

Let G_i^U be a set of triplets $\langle p,a,o\rangle$, where triplet $\langle p,a,o\rangle \in G_i^U$ represents the fact that generated object $o \in O_i$ constitutes the input object in action $a \in A_i$, performed by participant $p \in P_i$, in the business process represented by DAD_i. Total set G^U for the whole system is $G^U = \bigcup G_i^U$.

Total set G of triplets $\langle p,a,o\rangle$ representing the facts of creation and subsequent usages of generated objects for the whole system is $G = G^C \cup G^U$, and total set R^{PG} of *participant-object* associations representing facts of creation and subsequent usages of generated objects for the whole system is:

$$R^{PG} = \Big\{ r_{PG} \in R \;\Big|\; r_{PG} = T_{PG}(\langle p,a,o\rangle),\ \langle p,a,o\rangle \in G \Big\},$$

where the T_{PG} rule, that maps triplet $\langle p,a,o\rangle \in G$ into association $r_{PG} \in R^{PG} \subseteq R$ between classes e_P and e_G corresponding to the given participant and generated object, respectively, is:

$$\begin{aligned} T_{PG}\big(\langle p,a,o\rangle\big) \overset{def}{=} r_{PG} \;\Big|\; \Big(& name(r_{PG}) = name(a) \wedge \\ & \big(memberEnd(r_{PG}) = \{source, target\} \mid \\ & \quad type(source) = e_P \wedge multiplicity(source) = 1 \wedge \\ & \quad type(target) = e_G \wedge multiplicity(target) = *\big)\Big). \end{aligned}$$

Activation and subsequent usage of activated objects. Total set X^A of activations in the whole system has already been defined.

Let X_i^U be a set of triplets $\langle p,a,o\rangle$, where triplet $\langle p,a,o\rangle \in X_i^U$ represents the fact that activated existing object $o \in O$ is used in action $a \in A_i$, performed by participant $p \in P_i$, in the business process represented by DAD_i. Total set X^U of all usages of activated existing objects in the whole system is $X^U = \bigcup X_i^U$.

Total set X of triplets $\langle p,a,o\rangle$ representing the facts of activation and subsequent usages of activated existing objects for the whole system is $X = X^A \cup X^U$, and total set R^{PA} of *participant-object* associations representing facts of activation and subsequent usages of activated objects for the whole system is:

$$R^{PA} = \Big\{ r_{PA} \in R \;\big|\; r_{PA} = T_{PA}(\langle p,a,o\rangle),\ \langle p,a,o\rangle \in X \Big\},$$

where the T_{PA} rule, that maps triplet $\langle p,a,o\rangle \in X$ into association $r_{PA} \in R^{PA} \subseteq R$ between classes e_P (participant) and e_A (activation) is:

$$\begin{aligned} T_{PA}\big(\langle p,a,o\rangle\big) \overset{def}{=} r_{PA} \;\big|\; \Big(& name(r_{PA}) = name(a) \wedge \\ & \big(memberEnd(r_{PA}) = \{source, target\} \mid \\ & \quad type(source) = e_P \wedge multiplicity(source) = 1 \wedge \\ & \quad type(target) = e_A \wedge multiplicity(target) = *\big)\Big). \end{aligned}$$

***Object-object* associations.** There are two typical bases for automated generation of *object-object* associations [3] that are related to: (i) *activation of existing objects*, and (ii) *actions having input and output objects*.

Activation of existing objects. Besides the association between classes corresponding to the participant and activation of existing object, one more association is to be generated for each activation (between classes corresponding to the existing object and its activation). Total set R^{EA} of *object-object* associations between classes corresponding to the existing objects and their activations for the whole system is:

$$R^{EA} = \Big\{ r_{EA} \in R \;\big|\; r_{EA} = T_{EA}(\langle p,a,o\rangle),\ \langle p,a,o\rangle \in X^A \Big\},$$

where the T_{EA} rule, that maps triplet $\langle p,a,o\rangle \in X^A$ into association $r_{EA} \in R^{EA} \subseteq R$ between classes e_E (existing object) and e_A (activation) is:

$$\begin{aligned} T_{EA}\big(\langle p,a,o\rangle\big) \overset{def}{=} r_{EA} \;\big|\; \Big(& name(r_{EA}) = name(a) \wedge \\ & \big(memberEnd(r_{EA}) = \{source, target\} \mid \\ & \quad type(source) = e_E \wedge multiplicity(source) = 1 \wedge \\ & \quad type(target) = e_A \wedge multiplicity(target) = *\big)\Big). \end{aligned}$$

For the sample model, R^{EA} is to contain one association named `Issuing` between classes `Book` and `Book_borrowed`.

Actions having input and output objects. According to [3], each action $a \in A_i$ having $p \in \mathbb{N}$ different types of input objects $io_1,...,io_p \in O_i^I$ and $q \in \mathbb{N}$ different types of output objects $oo_1,...,oo_q \in O_i^O$ can be considered as a set $M(a) = \{\langle io_j, if_j, a, of_k, oo_k\rangle, 1 \le j \le p, 1 \le k \le q\}$ of $p \times q$ SISO (single input - single output) tuples, where $if_1,...,if_p \in F_i^I$ and $of_1,...,of_q \in F_i^O$ constitute the corresponding input and output object flows, respectively. Total set $R^{OO}(a)$ of *object-object* associations for the given action $a \in A_i$ is:

$$R^{OO}(a) = \bigcup_{j,k} R^{OO}_{j,k}(a).$$

The $R^{OO}_{j,k}(a)$ set containing exactly $w \in \mathbb{N}$ associations between classes e_{IO} and e_{OO} corresponding to the input and output objects io_j and io_k, respectively, is:

$$R^{OO}_{j,k}(a) = \Big\{ r^{(l)}_{OO} \in R \;\big|\; r^{(l)}_{OO} = T_{OO}\big(\langle io_j, if_j, a, of_k, oo_k\rangle\big), l = 1,\ldots,w \Big\},$$

where the basic T_{OO} rule, that maps a SISO tuple $\langle io, if, a, of, oo\rangle$ into binary association r_{OO} between corresponding classes, is given with:

$$T_{OO}\big(\langle io, if, a, of, oo\rangle\big) \overset{def}{=} r_{OO} \,\Big|\, \Big(name(r_{OO}) = name(a) \wedge \big(memberEnd(r_{OO}) = \{source, target\} \,|\, type(source) = e_{IO} \wedge multiplicity(source) = m_s \wedge type(target) = e_{OO} \wedge multiplicity(target) = m_t\big)\Big).$$

The corresponding source and target classes e_{IO} and e_{OO} are given with:

$$e_{IO} = \begin{cases} e_G,\ io \in O^G \\ e_X,\ io \in O^{Xn} \\ e_A,\ io \in O^{Xa} \end{cases} \qquad e_{OO} = \begin{cases} e_G,\ oo \in O^G \\ e_A,\ oo \in O^{Xa} \end{cases},$$

where $O^G \subseteq O^I$ represents a set of generated input objects, $O^{Xa} \subseteq O^I$ represents a set of activated existing input objects, while $O^{Xn} \subseteq O^I$ represents a set of existing input objects that are not activated. The corresponding source and target association end multiplicities and the total number of associations are:

$$m_s = \begin{cases} *,\ w_{if} = * \\ 1,\ otherwise \end{cases} \quad m_t = \begin{cases} *,\ w_{of} \neq 1 \vee io \in O^{Xn} \\ 1,\ otherwise \end{cases} \quad w = \begin{cases} 1,\ w_{if} \in \{1, *\} \\ w_{if},\ otherwise \end{cases},$$

where w_{if}/w_{of} represents the weight of the input/output object flow. In the sample model, all object flows have the same weight (1).

Total set R^{OO} of *object-object* associations, generated for the actions having input and output objects, for the whole system is:

$$R^{OO} = \bigcup_i \bigcup_{a \in A_i} R^{OO}(a).$$

For the sample model, R^{OO} is to contain two associations: (i) association named `Requesting` between classes `BookCatalog` and `BookRequest`, and (ii) association named `Issuing` between classes `BookRequest` and `Book_borrowed`.

Finally, total set R of associations is $R = R^{PG} \cup R^{PA} \cup R^{EA} \cup R^{OO}$.

3.3 Implementation

The CDM generator is implemented as an improved release of Eclipse-Topcased [12] plugin named **ADBdesign**, whose prototype has been presented in [4, 3].

The process of the automated CDM generation is determined by the mutual dependence of the rules. By following the recommendation [3] that the rules aimed at automated generating of classes are to be applied before the rules for automated generation of associations, the implemented generator generates the target CDM in two phases: (i) classes are being generated in the first phase, and (ii) class associations are being added in the second phase. The process of CDM design is expressed by the high-level algorithm presented in Fig. 4.

```
1: $E \leftarrow \emptyset$; $R \leftarrow \emptyset$
2: for all $DAD_i \in BM$ do
3:     for all $p \in P_i$ do                              /* participants */
4:         $E \leftarrow E \cup \{T_P(p)\}$
5:     end for
6:     for all $o \in O_i$ do                              /* objects */
7:         $E \leftarrow E \cup \{T_O(o)\}$
8:     end for
9:     for all $\langle p,a,o\rangle \in X_i^A$ do         /* activations */
10:        $E \leftarrow E \cup \{T_A(\langle p,a,o\rangle)\}$
11:    end for
12: end for
13: for all $DAD_i \in BM$ do
14:    for all $\langle p,a,o\rangle \in G_i^C \cup G_i^U$ do      /* participant - generated object */
15:        $R \leftarrow R \cup \{T_{PG}(\langle p,a,o\rangle)\}$
16:    end for
17:    for all $\langle p,a,o\rangle \in X_i^A \cup X_i^U$ do      /* participant - activated object */
18:        $R \leftarrow R \cup \{T_{PA}(\langle p,a,o\rangle)\}$
19:    end for
20:    for all $\langle p,a,o\rangle \in X_i^A$ do                /* existing object - activation */
21:        $R \leftarrow R \cup \{T_{EA}(\langle p,a,o\rangle)\}$
22:    end for
23:    for all $a \in A_i$ do                                      /* input object - output object */
24:        for all $\langle io,if,a,of,oo\rangle \in M(a)$ do      /* SISO tuples */
25:            $R \leftarrow R \cup \{T_{OO}(\langle io,if,a,of,oo\rangle)\}$
26:        end for
27:    end for
28: end for
```

Fig. 4 High-level algorithm for automated CDM generation

4 Experimental Results

The implemented generator has been applied to the sample business model (Fig. 2). The visualization result of the automatically created CDM is shown in Fig. 5.

The generator has created all classes and associations, as was expected during the analysis of the sample model. From the database designer's point of view:

- All generated classes are suitable and could be retained in the CDM without any change. There are no classes that could be considered as surplus.
- Only the `Returning` association (association between classes `MEMBER` and `Book_borrowed`) could be considered as surplus in the given business system (there is no need for two associations – borrowing and returning of some book are related to the same member).
- Only one generated association has partly incorrect multiplicities – the multiplicity of the `Book_borrowed` class is to be '0..1' in the `Issuing` association with the `BookRequest` class (incorrect multiplicity is the consequence of the source model simplification).

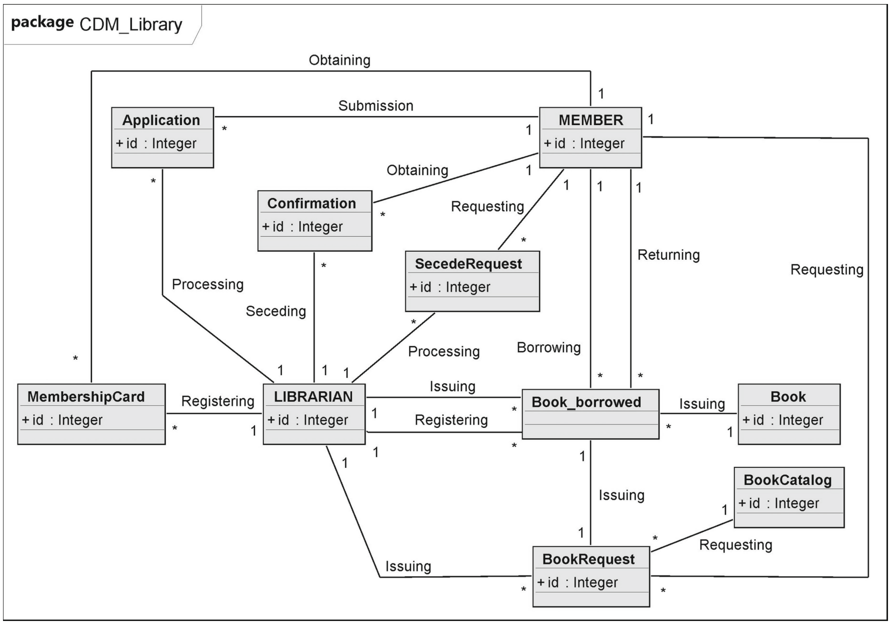


Fig. 5 Automatically generated CDM based on sample business model

Since the source business model is very simplified and does not cover the whole business system, the automatically generated CDM is not representative enough for an objective quantitative evaluation of the approach, particularly for the estimation of the *recall* measure (percentage of all concepts in the target CDM that are automatically generated). However, *precision* as a measure of correctness of the automatically generated CDM, which is defined as:

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \cdot 100\%,$$

can be calculated based on the previous short qualitative evaluation. Basic metrics and calculated precision for the sample CDM are given in Table 1. A very high precision of the implemented generator (above 90%) confirms the previously obtained results [3].

Table 1 Quantitative evaluation based on sample CDM

Concepts	Metrics				*Precision* [%]
	$N_{generated}$	$N_{correct}$	$N_{incorrect}$	$N_{surplus}$	
Classes	10	10	0	0	**100**
Associations	17	16	1	1	**94**

5 Conclusion

In this paper we have presented an approach to the automated CDM design, that is based on UML business model containing a finite set of DADs representing the business processes of the whole enterprise. We have identified the semantic capacity of typical business process patterns and defined formal rules for automated CDM design. Based on those formal rules we have implemented two-phase automatic generator that iterates through the source model, processes all DADs and generates the target CD. The classes are being generated in the first phase, while the class associations are being added in the second phase.

The application of the generator is illustrated on the simplified real business model. The evaluation of automatically generated CDM implies that the generator is able to generate a very high percentage of the target CDM with a very high precision (over 90% of all generated concepts are correct). An extensive and objectified evaluation of the approach, based on statistically reliable number of models, will be part of future work.

References

1. Brdjanin, D., Maric, S.: An example of use-case-driven conceptual design of relational database. In: Proc. of Eurocon 2007, pp. 538–545. IEEE (2007)
2. Brdjanin, D., Maric, S.: On Automated Generation of Associations in Conceptual Database Model. In: De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H. (eds.) ER Workshops 2011. LNCS, vol. 6999, pp. 292–301. Springer, Heidelberg (2011)
3. Brdjanin, D., Maric, S.: An Approach to Automated Conceptual Database Design Based on the UML Activity Diagram. ComSIS 9(1), 249–283 (2012)
4. Brdjanin, D., Maric, S., Gunjic, D.: ADBdesign: An Approach to Automated Initial Conceptual Database Design Based on Business Activity Diagrams. In: Catania, B., Ivanović, M., Thalheim, B. (eds.) ADBIS 2010. LNCS, vol. 6295, pp. 117–131. Springer, Heidelberg (2010)
5. Chen, P.: English sentence structure and entity-relationship diagrams. Information Sciences 29(2-3), 127–149 (1983)
6. García Molina, J., José Ortín, M., Moros, B., Nicolás, J., Toval, A.: Towards Use Case and Conceptual Models through Business Modeling. In: Laender, A.H.F., Liddle, S.W., Storey, V.C. (eds.) ER 2000. LNCS, vol. 1920, pp. 281–294. Springer, Heidelberg (2000)
7. OMG: Unified Modeling Language: Infrastructure, v2.4.1. OMG (2011)
8. OMG: Unified Modeling Language: Superstructure, v2.4.1. OMG (2011)
9. Rodríguez, A., Fernández-Medina, E., Piattini, M.: Towards Obtaining Analysis-Level Class and Use Case Diagrams from Business Process Models. In: Song, I.-Y., Piattini, M., Chen, Y.-P.P., Hartmann, S., Grandi, F., Trujillo, J., Opdahl, A.L., Ferri, F., Grifoni, P., Caschera, M.C., Rolland, C., Woo, C., Salinesi, C., Zimányi, E., Claramunt, C., Frasincar, F., Houben, G.-J., Thiran, P. (eds.) ER Workshops 2008. LNCS, vol. 5232, pp. 103–112. Springer, Heidelberg (2008)

10. Rodriguez, A., Garcia-Rodriguez de Guzman, I., Fernandez-Medina, E., Piattini, M.: Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach. Information and Software Technology 52(9), 945–971 (2010)
11. Rungworawut, W., Senivongse, T.: From business world to software world: Deriving class diagrams from business process models. In: Proc. of the 5th WSEAS Int. Conf. on Aplied Informatics and Communications, pp. 233–238. WSEAS (2005)
12. TOPCASED Project: Toolkit in OPen-source for Critical Application & Systems Development, v5.1.0, http://www.topcased.org

Aligning Business Process Models and Domain Knowledge: A Meta-modeling Approach

Samira Si-Said Cherfi, Sarah Ayad, and Isabelle Comyn-Wattiau

Abstract. In recent years the problems related to modeling and improving business processes have been of growing interest. Indeed, companies are realizing the undeniable impact of a better understanding and management of business processes (BP) on the effectiveness, consistency, and transparency of their business operations. BP modeling aims at a better understanding of processes, allowing deciders to achieve strategic goals of the company. However, inexperienced systems analysts often lack domain knowledge leading and this affects the quality of models they produce. In this paper we propose to support this modeling effort with an approach that uses domain knowledge to improve the semantic quality of BP models. This approach relies on domain ontologies as a mean to capture domain knowledge and on meta-modeling techniques. The main contribution of this paper is threefold: 1) the meta-models describing both a domain ontology and a BP model are described, 2) the alignment between the concepts of both meta-models is defined and illustrated, 3) a set of OCL mapping rules is provided. A simple case study illustrates the process.

Keywords: Domain knowledge, Domain ontology, Semantic quality, Business process modeling, Quality improvement.

1 Introduction

Modeling is the intellectual activity of creating abstract and comprehensive representation of a system necessary to understand its existing or planned behavior. In practice, conceptual models have been recognized as playing an

Samira Si-Said Cherfi · Sarah Ayad
CEDRIC-CNAM, 292 Rue Saint Martin, F-75141 Paris Cedex03
e-mail: samira.cherfi@cnam.fr,ayad.sarah87@gmail.com

Isabelle Comyn-Wattiau
CEDRIC-CNAM and ESSEC Business School
e-mail: wattiau@cnam.fr

important role in communication and understanding among various stakeholders within a project. Business Process models are conceptual models supposed to give a complete description of the underlying business processes. Consequently, companies are today aware of the undeniable impact of a better tuning of business processes (BP) on the effectiveness, consistency and transparency of their business operations. This tuning requires a better understanding and an effective management of BP. However, to achieve the expected benefits it is necessary to rethink the approach of designing these processes. BP modeling is a prerequisite. It is now considered as an engineering activity aiming at providing the actors with a better understanding of the processes in which they are involved. But BP modeling is difficult. It is an expert task that needs to be performed by trained experts. And, what about quality? Quality can be defined as the total of properties and characteristics of a product or service that are relevant for satisfying specific and obvious requirements [1]. The business process modeling approaches share many similarities with conceptual modeling activities, but are much more complex [19]. Indeed, a business process model captures a dynamic vision of the system through activities descriptions, generally done at a low level of abstraction; with a difficult issue of ending with a high level description for which a good acquaintance and understanding of domain knowledge is necessary. This is why the activity of modeling BP requires a high degree of pragmatic expertise generally referred to as empirical rules and heuristics difficult to formalize and to share. Commercial tools for business process modeling activities mainly focus on the accuracy of models based on a set of syntactic criteria imposed by the notation and provide little or no guide to guarantee the quality of produced models. We propose to assist the modeling activity with a quality centered approach that aims to exploit the domain knowledge. The domain knowledge in Information Systems discipline refers to knowledge provided by both methods and application domain [12]. In our approach we propose to exploit domain ontologies knowledge with alignment rules to identify similarities between BP models and domain ontologies elements. The aim is to improve the semantic completeness and expressiveness of BP models according to domain knowledge contained in the ontologies. This paper is organized as follows. State of the art is described briefly in Section 2. The overall approach of our semantic is broadly described in the third section. The meta-models structuring both BP models and domain ontologies are described in detail in Section 4. Section 5 is dedicated to alignment rules. Finally Section 6 concludes and describes future research.

2 State of the Art

A Business Process (BP) is a set of related activities that transform an input to create an output with added values [10]. Experts in information systems and professionals agree that the success of a company depends particularly of a good understanding of business processes [4]. To make a business process model understandable, reliable, and reusable it is important to ensure its quality. Several approaches that work in this direction exist in the literature. We have classified them into three categories:

1. Approaches focused on improving BP methods of analysis and design: improving the process development improves the quality of products. we can mention [5] where the authors propose a set of guides to improve clarity, comprehensibility. Other authors focus on improving the comprehensibility of models [15].
2. Process quality measurement: considers the quality level of business processes and their execution. We categorize the research on simulation and control of process as in [3]. In [9], the authors present and discuss several techniques for the analysis of processes during execution such as verification.
3. Process model quality measurement: Our focus is in this category that addresses the quality from the point of view of its evaluation and improvement. In [2], the authors mention the most important five measures: coupling, cohesion, complexity, modularity, and finally the size. [20] propose an approach based on GQM method (Goal-Question-Metric). One of the characteristics t hat has been the subject of several proposals is the complexity [8, 6]. However, these studies are based primarily on structural characteristics of processes and their models.

In conclusion, our analysis of the state of the art leads us to argue that the quality of BP model is mainly addressed in terms of structural and syntactic and rarely in terms of semantics. In the remainder of this paper, we present our approach which aims to go a step forward into a semantic quality based approach of BP model.

3 The Overall Approach for Semantic Quality Improvement

Modeling activity in general and BP modeling in particular are creative activities conducted by modelers using a given notation or modeling language. The result is of course highly dependent on the modeler experience in the notation practice, on his/her interpretation of the reality, and on the decision he/she makes regarding the choice of concepts and details to be modeled. This explains the fact that several correct but different models can usually be generated from the same reality. However, these models are supposed to be faithful representations of the reality. Thus the definition of quality requirements for these models is, in fact, a mean to evaluate this modeling activity and ensure a better result. Many factors may be defined to characterize this quality. The semantic quality measures the degree of correspondence between the model and the domain. The semantic quality is related to both completeness and validity of the models; here the BP models [13]. To improve the quality of models produced, several approaches are possible: assistance in the development process phase by generic methodological guides from experience, measurement of the specifications quality, reusing approved specifications fragments etc. In this paper, we propose to exploit knowledge of field, which are supposed to reflect the knowledge shared by a community of actors, in order to improve the quality of process models. Our approach relies on the process having as input point the business process model to be evaluated and a domain ontology representing business knowledge and rules of the underlying problem domain. The steps of the process are the following:

- Discovering similarities between input BP model and domain ontology: this is based on a set of alignment rules at both syntactic and semantic levels.
- Evaluating semantic quality includes measuring a value of quality according to quality metrics.
- Improving semantic quality: An originality of our work is to integrate the quality improvement within the proposed approach.

3.1 Identifying Model-Ontology similarities

In the first step, the approach consists in discovering the mappings between business process model elements the domain ontology elements. To make these alignment rules generic and independent of both the BP modeling notation and the ontology implementation language, we have defined two meta-models namely a BP meta-model and an ontology meta-model presented in detail in Section 4. The alignment rules aim to identify similarities between the process model elements and the domain ontology concepts. Once these similarities identified they serve as input for both semantic quality evaluation and improvements activities. In this paper we mainly focus on this alignment activity.

3.2 Evaluating Semantic Quality

Semantic quality expresses the degree of correspondence between the information expressed within a model and the domain that is modeled. In order to evaluate the semantic quality we have identified a set of what we call quality deficiencies such as incompleteness and ambiguity. These deficiencies result from modeling choices producing models that do not cover the intended requirements or with low expressiveness. Such models lead to inadequate systems due to incompleteness or to misunderstanding during their implementation. Once a similarity has been identified between a BP model element, let it be bpmi and an element from the domain ontology doi, our approach exploits the knowledge from the domain ontology related to doi to detect and measure semantic quality deficiencies according to quality metrics we have defined. For space reason, this part of the work is not presented here.

3.3 Quality Improvement

The quality improvement activity consists in suggesting to the analyst or the quality expert a set of improvement guidelines to improve the quality of their models. Again, this step uses the domain knowledge to generate improvement actions. This means that the completeness and even the relevance of these guides rely partly on the quality of the domain ontology but this aspect is out of the scope of our approach. For example, if the approach identified a similarity between bpmi a BP model element- and doi - an element from the domain ontology- and the domain ontology describes a relationship between doi and doj (an other element from the ontology), then our approach will propose an enrichment action on the BP model based on the relationship between doi and doj.

This article focuses on discovering similarities between BP models and domain ontologies. We will however provide some examples of improvements without detailing the mechanism leading to generate them in Section 5, dedicated to the illustration of the approach.

4 Ontology and Process Model Meta-models

In order to identify similarities between knowledge contained in the ontology and the one represented by the BP model, our approach relies on alignment. To ensure the generality of these rules, we have chosen to define them at a meta-modeling level. Hence, the first contribution is the construction of meta-models representing ontologies and BP models.

4.1 Business Process Meta-model

There are several advantages of defining such a meta-model. First, the meta-model provides a synthetic vision of concepts used independently of specific notations helping in the understandability of models. Second, instead of defining mapping rules for each couple of BP modeling notation and ontology language we define the rules only at the meta-model level. Finally, since we consider that domain knowledge contains also knowledge embedded in methods and consequently in notations, we will use meta-models to integrate completeness, validation and corectness rules defined by BP notations to enrich our actual vision of domain knowledge. The meta-model defined in this section was constructed as a synthesis of a selection of concepts proposed by several authors and according to several notations and more specifically the work presented in [6, 14]. A business process model is composed of flows of objects and connectors. A flow object can be an event, an activity or a gateway [16]. An event that occurs is a fact and impacts the progress of a process. Our events can be of three types: initial, intermediate and final. An activity can be an atomic task if it is not decomposable or a process if it is complex and has a visible structure. A gateway is a mechanism that can manage the convergence or divergence of activities flow. A connecting element can be an association, a sequence or a message flow. An association is used as a simple link between two concepts. The sequence flow defines an execution order of activities. A message flow is used to represent exchange of information between two participants in the process. Activities refer to resources. A resource is a concept which includes abstract concepts such as the human agent responsible for execution of the activity and information produced or consumed by it. The exact role of the resource in the process is explained by the concept of role. Figure 1 shows an example of BP model from a "Mission order" case study. The example uses the Eriksson and Penker notation for Business process modeling [6]. An employee who has to travel for his/her job must first obtain the authorization of his/her boss. If he/she gets it, he fill a form called mission order and takes care of other formalities (book a ticket, a hotel, etc.) and sends the mission order to the financial service. When he/she comes back after his/her travel, he/she provides the financial service with expense accounts. The financial service may then reimburse him/her. A peopleResource -indicated by stereotype People at Figure 1 - "Employee" responsible of two processes "Request authorization" that requests an information resource input "Mission information", and "Establish MO" with a physical resource output "Mission order (MO)".

A sequence of processes "Request authorization" and "Analyze request" led to a process divergence. The two processes "Establish MO" and "Carry out mission formalities" may be executed in parallel.

4.2 Ontology Meta-model

The ontology meta-model allows representing domain ontologies using the same concepts independently of the language for their implementation. There are several contributions in literature concerning ontology meta-modeling. The authors in [18] introduced simple concepts and constructors (negation, conjunction, disjunction) to define complex concepts. They also defined several relationships including inheritance links, instantiation and constraints. In [11] five types of concepts have been proposed to represent the functional requirements (function, object, and environment) and non-functional requirements (constraints, quality). In our approach, we consider an ontology as a

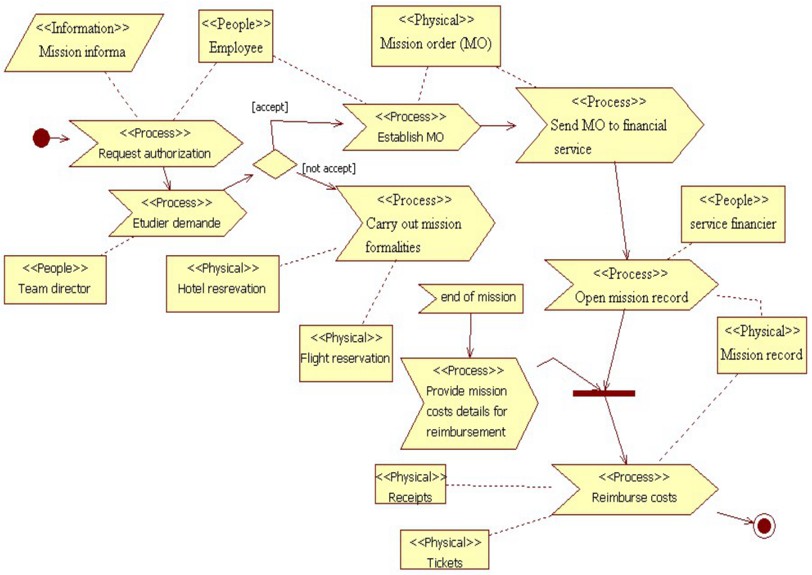

<<Information>> Mission informa
<<People>> Employee
<<Physical>> Mission order (MO)
<<Process>> Request authorization
[accept]
<<Process>> Establish MO
<<Process>> Send MO to financial service
[not accept]
<<Process>> Etudier demande
<<Process>> Carry out mission formalities
<<People>> service financier
<<People>> Team director
<<Physical>> Hotel resrevation
<<Process>> Open mission record
end of mission
<<Physical>> Flight reservation
<<Process>> Provide mission costs details for reimbursement
<<Physical>> Mission record
<<Physical>> Receipts
<<Process>> Reimburse costs
<<Physical>> Tickets

Fig. 1 Business process model for the Mission Order example

set of classes and relationships. This vision is largely adopted. We distinguish between three types of concepts of type class: actor, action and artifact.

- An actor is an independent entity, able to perform actions.
- An action represents the execution of an action.
- An artifact is an inanimate object incapable of performing an action.
- An artifact may represent information or an abstract concept.

However, most of meta-models take into account two kinds of relationships, namely inheritance and structural relationships. For the needs of our approach we adapted the classification of relationships proposed by [17], which has been initially defined to analyze semantics of relationships within a relational database. This classification offers several types of relationships allowing us to characterize precisely the nature of links between concepts.

Relations are first decomposed into three categories:

- Status: represents relationships that may be structural (inheritance, composition, instantiation, etc.), influence (own, control, creation, destroy, etc.), or temporal (follow, require, etc.).
- Change of status: reveals the occurrence of remarkable events. This type of relationship is primarily used to express the interdependence of status in the life cycle of an entity.

- Interaction: represents short-term relationships between entities. Several semantic relations are defined for interactions such as communication, observation, execution, etc.

Figure 2 shows an example of a domain ontology. This is an extract from the ontology "mission plan".

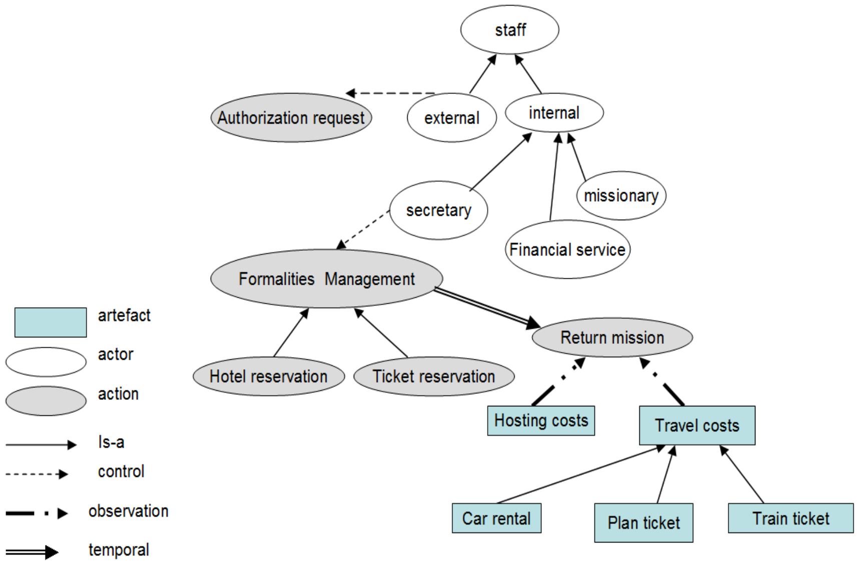


Fig. 2 Instantiation of the Ontology meta-model

The actor external" is linked to actor staff by an is-a relation of type structural (status). Also actors secretary and missionary are related to the actor "internal staff" by an is-a relation. In addition the actor missionary is related to the action Formalities management by a control relation of type influence (status), similarly to the actor "external" and the action authorization request. Moreover, "Formalities management" action is related to "return mission" action with a temporal relationship indicating that managing formalities precedes a return mission. The example indicates also that "Hosting costs" and "Travel costs" abstract concepts are linked to "Return mission" action by observation relationships, meaning that the action performs no changes on the abstract concept.

5 Mapping Process Model and Ontology Meta-models

Thanks to the precise categorization of concepts in both ontology and process model meta-models we are likely able to predefine some concepts correspondences allowing the mapping of the domain ontology concepts with the PM concepts. We have defined two kinds of mapping, namely type-based mapping and semantics-based mapping.

5.1 Type-Based Mapping Rules

This mapping involves the types of concepts in order to establish correspondences between the concepts at the meta-level. These correspondences allow reconciliation based on the types of concepts independently of their meaning. These rules are still essential to avoid typing errors. An extract of predefined meta-model concepts mappings is given in Table 1.

Table 1 Concept alignment

BP model meta-model concept	Domain Ontology meta-model concept
People resource	Actor
Abstract resource	Abstract
Information resource	Knowledge
Process / activity	Action

Similarly, we have established mappings between meta-model relations of BPM and those of the ontology meta-model. The result is given in Table 2.

Table 2 Relation alignment

BP model meta-model connectors	Domain Ontology meta-model relations
Sequence Flow	Temporal
Message Flow	Communication Transfer
Role	Execution Manipulation Observation Influence

The second type of mapping, presented in the following section, is richer, being based on the semantics of concepts.

5.2 Semantics Based Mapping Rules

Based on meta-models presented above, we developed a set of matching rules, allowing the mapping of the ontology field with the concepts of process models. These rules are written in Object Constraint Language (OCL) (OMG, 2010). There are four classes of matching rules. The rules are all defined as functions having as input one or several BP model concepts and returning one or several concepts from the domain ontology.

Equivalence: returns the ontology concept which is syntactically equivalent (they have the same names) to the BP model concept.

Synonymy: returns a set of ontology concepts that are synonyms of the BP model concept. The synonymy value is calculated by comparing the existence of common names or synonyms based on Wordnet [7].

More general: returns the ontology concepts having a superiority relationship (also called hyperonymy or IS-A relationship) with a concept from the ontology already detected as synonym or equivalent to the BP model concept.

More Specific: returns the ontology concepts having an inferiority relationship with a concept from the ontology detected as synonym or equivalent to the BP model concept.

These classes of rules are instantiated for each of the concepts of the BP meta-model.

5.3 Application to the Example

To illustrate the alignment activity of our approach, we consider the example of mission order process. The input is the business process model under construction represented at Figure 1 and an excerpt from the domain ontology mission plan from Figure 2. As mentioned above, all the semantic rules are based on type mapping rules i.e. equivalence is between concepts that are not only syntactically equivalent but also type mapped. By applying the equivalence rule on the actor financial service, it will return the actors of the ontology that have the same name. Equivalence rules are not fired since the concepts from the ontology do not have the same names as concepts from the BP model of the example. Also we can catch the actions synonym of the process formalities management by applying the synonymy rule: synonym(carry out mission formalities) returns formalities management and synonym(employee) returns {staff, internal staff, external staff, missionary, secretary}. By querying the ontology and firing the semantic/type mapping rules, we can map the ontology concepts to the BPM concepts. Thus we elicit the equivalent, synonyms, different abstraction levels of concepts. But finally, it's up to the analyst to validate these mappings. The step forward builds on these mappings to detect and correct quality defects. For example the ontology provides several kinds of costs such as "car rental costs" or the distinction between "train" and "flight" tickets that are not considered within the process in hand. This could express an incompleteness of the BP model that could be corrected by replacing "ticket" by "travel costs tickets" if the BP is the same for all the travel costs or it can be redesigned if not. Another enrichment that could be provided thanks to the ontology is the distinction between internal and external staff for which the reimbursement process can be different.

6 Conclusion

The quality of business process models is a hot topic both for researchers and practitioners. Many studies demonstrated that the quality of produced models depends highly on the degree of expertise of the modelers. Moreover, modeling activities are practiced by a significant number of non experts including IT professionals. One of the reasons impacting the quality of produced models is the lack of domain knowledge covering both knowledge about the methods and notations used as well as application domain knowledge. In this paper, we tried to propose a solution aiming to exploit application domain knowledge in the improvement of BP models. Our approach considers domain ontologies that are produced in several disciplines (web services, health care, administrative processes etc.) to improve the semantic quality of BP models. We have defined BP model and ontology meta-models in order to provide a uniform description of both process models and domain ontologies. We have then defined an alignment process using both type-based and semantics-based mappings to detect similarities between concepts from the BP models and the domain ontologies. The results serve as an input for the semantic quality evaluation and improvement processes. As regards the alignment process, the future research aims to validate the approach on real case studies based on domain ontologies widely agreed and accepted by practitioners and/or researchers. The rules need also to be completed to cover all the kinds of concepts and relationships semantics. To help in achieving these objectives, we are currently developing a prototype for the definition and execution of mapping rules.

References

1. The international standards organisation iso
2. van der Aalst, W.M.P.: Challenges in business process analysis (2007)
3. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
4. Aguilar-Savén, R.S.: Business process modelling: Review and framework. International Journal of Production Economics 90(2), 129–149 (2004), http://dx.doi.org/10.1016/S0925-5273(03)00102-6, doi:10.1016/S0925-5273(03)00102-6
5. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000), http://www.springerlink.com/index/DRWFDXGUUPX7EVXB.pdf
6. Eriksson, H.E., Penker, M.: Business Modeling With UML: Business Patterns at Work, 1 edn. Wiley (2000), http://www.amazon.com/gp/redirect.html%3FASIN=0471295515%26tag=ws%26lcode=xm2%26cID=2025%26ccmID=165953%26location=/o/ASIN/0471295515%253FSubscriptionId=13CT5CVB80YFWJEPWS02

7. Fellbaum, C.: WordNet: An Electronic Lexical Database. Language, Speech, and Communication. MIT Press (1998), http://books.google.fr/books?id=Rehu800zMIMC
8. Ghani, A., Muketha, K., Wen, W.: Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM). International Journal of Computer Science and Network Security 8, 219–225+ (2008)
9. Jansen-vullers, M.H., Netjes, M.: Business process simulation - a tool survey. In: Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN (2006)
10. Johansson, H. (ed.): Business process reengineering, reprinted edn. Wiley, Chichester (1994), http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+124207413&sourceid=fbw_bibsonomy
11. Kaiya, H., Saeki, M.: Ontology based requirements analysis: Lightweight semantic processing approach. In: International Conference on Quality Software, pp. 223–230 (2005), doi: http://doi.ieeecomputersociety.org/10.1109/QSIC.2005.46
12. Khatri, V., Vessey, I., Ramesh, V., Clay, P., Park, S.J.: Understanding conceptual schemas: Exploring the role of application and is domain knowledge. Information Systems Research 17(1), 81–99 (2006)
13. Krogstie, J., Lindland, O.I., Sindre, G.: Defining quality aspects for conceptual models. In: Proceedings of the IFIP International Working Conference on Information System Concepts: Towards a Consolidation of Views, pp. 216–231. Chapman & Hall, Ltd., London (1995), http://dl.acm.org/citation.cfm?id=645618.660960
14. List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, pp. 1532–1539. ACM, New York (2007), http://doi.acm.org/10.1145/1141277.1141633, doi:10.1145/1141277.1141633
15. Mendling, J., Recker, J.C., Reijers, H.A.: On the usage of labels and icons in business process modeling. International Journal of Information System Modeling and Design 1(2), 40–58 (2010), http://eprints.qut.edu.au/32288/
16. Lopes de Oliveira, J., Graciano Neto, V.V., Larissa da Costa, S.: A business process metamodel for enterprise information systems automatic generation. In: Anais do I Congresso Brasileiro de Software: Teoria e Prtica - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos, pp. 45–52. UFBA, Salvador (2010)
17. Purao, S., Storey, V.C.: A multi-layered ontology for comparing relationship semantics in conceptual models of databases. Appl. Ontol. 1(1), 117–139 (2005), http://dl.acm.org/citation.cfm?id=1412350.1412360
18. Tziviskou, C., Keet, C.M.: A Meta-model for Ontologies with ORM2. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 624–633. Springer, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1780909.1781012
19. Vanderfeesten, I., Cardoso, J., Reijers, H.A., Van Der Aalst, W.: Quality Metrics for Business Process Models (2006)
20. Vanderfeesten, I., Cardoso, J., Reijers, H.A., Van Der Aalst, W.: Quality Metrics for Business Process Models (2007), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.95.7331

Optimization of Object-Oriented Queries through Pushing Selections

Marcin Drozd, Michał Bleja, Krzysztof Stencel, and Kazimierz Subieta

Abstract. We propose a new optimization method for object-oriented queries. The method enables pushing selection conditions before structure constructors, joins and quantifiers. A less general variant of this method is known from relational systems and SQL as pushing a selection before a join. If a query involves a selection which predicates refer to proper arguments of a structure constructor or a join or a quantifier that occurs at the left-hand subquery of this query then it can be rewritten. The rewriting consists in pushing such predicates down before a proper operator. The approach follows the stack-based approach (SBA) and its query language SBQL (Stack-Based Query Language). The assumed strong typing based on the compile time simulation of run-time actions gives the possibility to solve this optimization problem in its full generality. The paper presents examples how the optimization method works. General features of the implemented algorithm are also presented.

1 Introduction

Currently we notice a new wave of interests to object-oriented databases. Our two prototypes of OODBMS, ODRA [1, 7, 8] and VIDE [16], are examples of many similar projects and products. ODRA implements the object-oriented query and

Marcin Drozd · Michał Bleja
Faculty of Mathematics and Computer Science, Łódź University, Banacha 22, 90-238 Łódź, Poland
e-mail: {mdrozd,blejam}@math.uni.lodz.pl

Krzysztof Stencel
Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland
e-mail: stencel@mimuw.edu.pl

Kazimierz Subieta
Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland
e-mail: subieta@pjwstk.edu.pl

programming language SBQL [13, 14, 15] and VIDE implements the query language OCL [17], an OMG standard. Among other proposals we can mention db4o system and the LINQ [9] project developed by Microsoft Research. There are also many database projects focused on storing parsed XML, with XQuery as a query language, which could be considered simplified object-oriented databases.

When we started to implement SBQL we learned carefully query optimization methods that were developed for SQL, as well as proposals in this respect for object-oriented databases. We adopted and generalized the SQL methods, but actually abandoned the state-of-the-art concerning optimization of object-oriented query languages, considering it as not enough matured. We decided to develop and implement our own optimization methods, such as factoring out independent subqueries [10, 15] factoring out weakly dependent subqueries [3], processing views by query modification [15], methods based on indices [6], and others. All these methods are implemented in the ODRA system. We hope that our research into query optimization will be relevant not only to SBQL and OCL, but to a wide class of object-oriented or XML-oriented query languages.

In this paper we focus on a query optimization method that belongs to an important group of methods based on query rewriting. The presented method is known from SQL in a much less general variant known as "pushing a selection before a join", see [4, 5]. We generalize it to an object-oriented query language and involve a dependent join, a query operator unknown in the SQL versions before SQL99. To explain it, we consider the following example in SBQL. Consider a database consisting of objects *Emp* (employee), *Person* and *Comp* (company). Let pointer links *employs* lead from a *Comp* object to *Emp* objects and pointer links *friend* lead from an *Emp* object to *Person* objects. The query should return companies together with their employees and the number of their friends. Companies should have the budget greater than 1000000, employees should earn between 2000 and 1% of the budget of their company and should have more than 10 friends:

((*Comp* **as** *c* **join** *c.employs.Emp as* **e**) **join** **count**(*e.friend.Person*) **as** *f*) **where** *c.budget* >1000000 **and** *e.sal* >2000 **and** *e.sal* <0.01 * *c.budget* **and** *f* >10

The query involves two dependent join operators. The first line of the query implies sequential lookup of companies, then sequential lookup of their employees, then calculating the number of their friends. For large collections of *Comp*, *Emp* and *Person* objects this could be expensive. Clearly, the boolean expressions (predicates) after the *where* operator could be pushed before joins and in this way the performance of the entire query will be much improved:

(((((*Comp* **as** *c* **where** *c.budget* >1000000) **join** (*c.employs.Emp* **as** *e* **where** *e.sal* >2000)) **where** *e.sal* <0.01 * *c.budget*)) **join** **count**(*e.friend.Person*) **as** *f*) **where** *f* >10

Such a rewriting is possible due to an important property of the *where* and *join* operators that we define as "distributivity w.r.t. processed collections".

In SBQL there are two structure generators: a dependent join and a usual structure constructor (in SQL referred to as "cartesian product"). Hence we have at least two optimization patterns:

$$(q_1 \textbf{ join } q_2) \textbf{ where } p_1 \textbf{ and } p_2 \textbf{ and } \ldots\, p_m$$

and

$$(q_1, q_2,\ldots,q_n) \textbf{ where } p_1 \textbf{ and } p_2 \textbf{ and } \ldots\, p_m$$

where q_i are some queries and p_j are predicates. These patterns can be combined. Similar patterns are relevant not only for the *where* operator, but also for quantifiers.

Providing some conditions are satisfied, we can push some particular predicate p_i just after some query q_j. As follows from our example, some predicate p_i can be stuck not only with a single q_j, but also with two or more queries joined by the *join* or comma operators. The problem is: if and how such a query optimization method can be defined and designed in its full generality? In this paper we show that it is possible. Pushing selections is a lossless optimization method: in all cases, after application of the method, the performance cannot be worse than before. For very large databases the gain from this optimization method can be several orders of magnitude shorter query response time.

The rest of the paper is organized as follows. In Section 2 we briefly present SBA and SBQL. Section 3 describes our optimization methods and the corresponding algorithm. Section 4 concludes.

2 Main Concepts of the Stack-Based Approach (SBA)

The Stack-Based Approach (SBA) and its Stack-Based Query Language (SBQL) [13, 14, 15] are the result of investigations into a uniform conceptual platform for an integrated query and programming languages for object-oriented databases. One of the most important concept of SBA is *environment stack* (ENVS), known also as *call stack*. The stack is responsible for binding names, scope control, procedure and method calls, parameter passing, inheritance mechanism, polymorphism and other features of object-oriented query and programming languages. In SBA the stack has also a new role: it participates in evaluation of non-algebraic operators.

SBA respects the *full internal identification* principle: each run-time entity that can be separately retrieved or updated must possess a unique internal identifier. Objects on any hierarchy level have the same formal properties and are treated uniformly; this is known as the *object relativity* principle. SBA assumes no differences in defining types and queries addressing transient and persistent data (this is known as *orthogonal persistence* [2]).

In SBA classes are understood as prototypes, which means that they are objects, but their role is different. A class object stores invariants (e.g. methods) of the objects that are instances of that class. A special relationship - instantiation - between a class and its instances is introduced. Inheritance between classes is supported. To present SBA object store we assume the class diagram presented in Fig.1. *Person* is the superclass of the classes *Student* and *Emp*. The classes *Course*, *Student*, *Emp* and *Dept* model students attending courses, which are supervised by employees working in departments. Names of classes (attributes, links, etc.) are followed by cardinality numbers. Cardinality [1..1] is dropped.

SBQL is described in detail in [13, 15]. The language has several implementations, in particular for the ODRA system [1, 7, 8]. Semantics of SBQL queries follows the *compositionality* principle, which means that the semantics of a query is a function of the semantics of its components, recursively, down to atomic queries (names and literals). This feature much simplifies implementation and optimization.

SBQL operators are subdivided into *algebraic* and *non-algebraic*. The main difference between them is whether they modify the state of ENVS during query processing or not. If an operator does not modify ENVS then it is algebraic. Algebraic operators include string and numerical operators and comparisons, Boolean *and*, *or*, *not*, aggregate functions, bag and sequence operators and comparisons, structure constructors, etc. A very useful algebraic operators are *as* and *group as* which name query results. Operator *as* names each element in a bag or sequence returned by a query, while *group as* names the entire query result.

Evaluation of non-algebraic operators is more complicated and requires further notions. If a query $q_1\ \theta\ q_2$ involves a non-algebraic operator θ, then q_2 is evaluated in the context of q_1. The context is determined by a new section (sections, depending

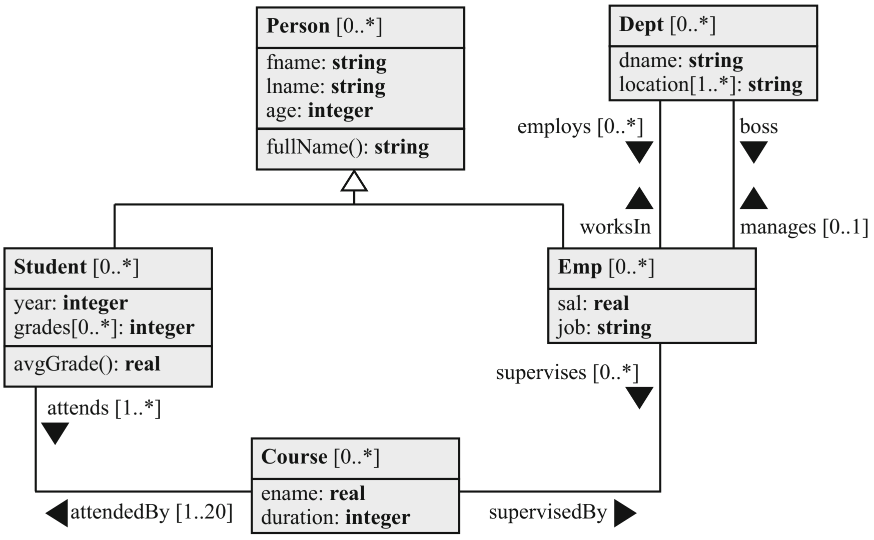


Fig. 1 A schema (class diagram) of an example database

on a store model [15]) opened by the θ operator on ENVS for an element of q_1. A new stack section(s) pushed onto ENVS are constructed and returned by a special function *nested*. The order of evaluation of subqueries q_1 and q_2 is important, they cannot be processed independently. The non-algebraic operators include: selection (q_1 **where** q_2), projection/navigation ($q_1.q_2$), dependent join (q_1 **join** q_2), quantifiers ($q_1 \exists (q_2)$) and ($q_1 \forall (q_2)$), ordering and transitive closures.

A property of *where*, *join* and structure operators that is fundamental for the optimization method presented in this paper is called distributivity w.r.t. processed collection. Let query q_1 return the collection **bag**$\{ e_1, e_2,..., e_k \}$ and let the notation **bag**$\{ e_1, e_2,..., e_k \}$ θ q_2 denote the result of processing this collection by a non-algebraic operator θ and q_2. The operator is distributive if it satisfies in general the following condition ($\cup$ stands for sum of bags):

$$\mathbf{bag}\{ e_1, e_2,...,e_k \}\ \theta\ q_2 = \mathbf{bag}\{ e_1 \}\ \theta\ q_2 \cup \mathbf{bag}\{ e_2 \}\ \theta\ q_2 \cup ... \cup \mathbf{bag}\{ e_k \}\ \theta\ q_2$$

Operators *where*, dot and *join* are distributive what stems directly from their definitions. The distributivity property makes it possible to move conditions along operators without changing the semantics of a query, providing types and name bindings are not violated.

3 Optimization Methods

In this paper we consider optimization methods based on query rewriting. Rewriting means transforming a query q_1 into a semantically equivalent query q_2 ensuring much better performance. It consists in locating parts of a query matching some pattern. These parts are to be replaced according to the rewriting rule by other parts. Such optimizations are compile-time actions entirely performed before a query is executed. It requires performing a special phase called *static analysis* [11, 15] that is a function of a strong type checker [12]. The strong typing mechanism of SBQL assumes simulation of run time actions during the compile time.

The static analysis uses special data structures: a static environment stack S_ENVS and a static query result stack S_QRES. These structures are compile-time equivalents of run-time structures: an object store, an environment stack ENVS and a query result stack QRES, correspondingly. S_ENVS models bindings (in particular opening new scopes and binding names) that are performed on ENVS. The process of accumulating intermediate and final query result on QRES is modeled by S_QRES. The main component of the metabase is a *schema graph* that is compiled from a database schema. It contains nodes representing database entities (objects, attributes, classes, links, etc.) and interfaces of methods. The edges represent relationships between nodes. The graph nodes are identified by internal identifiers that are processed on static stacks.

3.1 Pushing Selections before Structure Constructors

Consider the following query pattern:

$$(q_1, q_2,...,q_n) \textbf{ where } p_1 \textbf{ and } p_2 \textbf{ and } ... \ p_m$$

where q_i are some queries and p_j are predicates. Due to the distributivity property of the *where* operator and the structure constructor predicates p_1, p_2,...,p_m can be stuck (using the operator *where*) with some q_i . For instance, p_2 can be stuck with q_1 and in this way we obtain a semantically equivalent query:

$$((q_1 \textbf{ where } p_2), q_2,...,q_n) \textbf{ where } p_1 \textbf{ and } p_3 \textbf{ and } ... \ p_m$$

The same we can do with other predicates p_j, sticking them with proper q_i. If all the predicates after *where* are moved before the structure constructor, the final *where* operator can be removed.

The question is what is a criterion for matching some q_i with some p_j. The criterion can be easily established on the ground of the static analysis of the query. During the analysis, the signature s_i of the result returned by each q_i is calculated. Then, the *where* operator uses this signature as an argument of the *static_nested* function, which returns at the top of S_ENVS the signature of the environment e_i that is implied by the result of q_i. The environment e_i contains all the static binders (with proper names) that are the contribution of this q_i to the entire environment. In effect, we obtain the following situation on S_ENVS, Fig.2:

Now we can easily establish the criterion which governs sticking a particular p_j with a particular q_i:

- p_j contains at least one name equal to the name of some static binder within e_i;
- p_j does not contain a name of some static binder within e_1,...,e_{i-1}, e_{i+1},...,e_n;

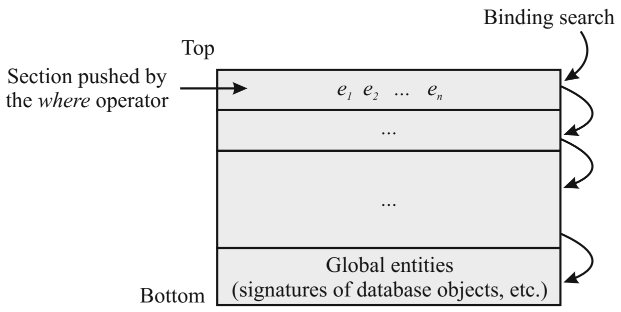

Binding search
Top
Section pushed by the *where* operator
e_1 e_2 ... e_n
...
...
Global entities (signatures of database objects, etc.)
Bottom

Fig. 2 S_ENVS stack during static analysis of $p_1, p_2, \ldots, p_m$

- p_j may contain other names that are bound in some sections of S_ENVS different from the section pushed by the *where* operator. Names can be bound below this section or (if p_j contains a non-algebraic operator) above this section.

In more friendly words, p_j can be stuck with q_i if and only if in the same source code context in which the entire query pattern is recognized the query q_i *where* p_j is typologically correct. p_j has no dangling names that cannot be bound and p_j is not independent from the *where* operator: at least one name in p_j must be bound at the section of S_ENVS pushed by this operator.

Example 1 (c.f. Fig.1). Assume (a bit contrived) query: *Get persons together with departments and their bosses. A persons should have last name the same as the department name and should be older than 50. A department should have 3 locations.*

$$\begin{gathered}(\mathit{Person}, \mathit{Dept}, \mathit{Emp}\ \mathbf{as}\ e)\ \mathbf{where}\ \mathit{lname} = \mathit{dname}\\ \mathbf{and}\ \mathit{age} > 50\ \mathbf{and}\ \mathbf{count}(\mathit{location}) = 3\ \mathbf{and}\ \mathit{boss.Emp} = e\end{gathered} \tag{1}$$

The structure constructor has three argument subqueries, *Person*, *Dept* and *Emp as e*. The counted signatures for them will be following (*metanode* is the identifier of a node in the metabase):

- For *Person*: $metanode_{Person}$
- For *Dept*: $metanode_{Dept}$
- *Emp as e*: static binder $e(metanode_{Emp})$

These signatures will be processed by the *where* operator, which for particular subqueries will return at the top of S_ENVS the following static binders:

- For *Person*: $static\_nested(metanode_{Person}) = fname(metanode_{fname})$, $lname(metanode_{lname})$, $age(metanode_{age})$, $fullName(metanode_{fullName})$
- For *Dept*: $static\_nested(metanode_{Dept}) = dname(metanode_{dname})$, $location(metanode_{location})$, $employs(metanode_{employs})$, $boss(metanode_{boss})$
- For *Emp as e*: $static\_nested(e(metanode_{Emp})) = e(metanode_{Emp})$ (for binders function *nested* returns them without changes)

Now we can see that according to our criterion the first predicate cannot be moved, the second one can be moved to the first subquery (name *age*), the third one can be moved to the second subquery (name *location*) and the fourth one cannot be moved too. Hence the optimized version of the query according to the criteria is the following:

$$\begin{gathered}((\mathit{Person}\ \mathbf{where}\ \mathit{age} > 50), (\mathit{Dept}\ \mathbf{where}\ \mathbf{count}(\mathit{location}) = 3), \mathit{Emp}\ \mathbf{as}\ e)\\ \mathbf{where}\ \mathit{lname} = \mathit{dname}\ \mathbf{and}\ \mathit{boss.Emp} = e\end{gathered} \tag{2}$$

Now the algorithm of pushing selections before the structure constructors is clear and can be easily developed. However, our example shows next opportunities for this kind of optimization. According to the same reasoning we can check neighboring pairs of arguments of the structure constructor and try to stick the rest of predicates with the pair. We see that the predicate *lname* = *dname* can be stuck with the first and second subqueries. Name *lname* is bound in the section supplied by *Person* and name *dname* is bound in the section supplied by *Dept*. In this way we obtain:

$$\begin{gathered}(((\mathit{Person}\ \textbf{where}\ \mathit{age} > 50),\ (\mathit{Dept}\ \textbf{where}\ \textbf{count}(\mathit{location}) = 3)) \\ \textbf{where}\ \mathit{lname} = \mathit{dname}),\ \mathit{Emp}\ \textbf{as}\ e)\ \textbf{where}\ \mathit{boss.Emp} = e\end{gathered} \tag{3}$$

According to the same reasoning we can check neighboring triples of subqueries, neighboring quadruples of subqueries, etc., till we push all $p_j - s$ or conclude that some $p_j - s$ cannot be pushed at all. Below we present a sketch of such an algorithm:

```
1. Traverse recursively AST looking for the pattern.
   $(q_1, q_2,...,q_n)$ where $p_1$ and $p_2$ and ... $p_m$
   If the pattern is not found then stop; else
2. For each $p_j, j = 1,2,...,m$, for each $q_i, i = 1,2,...,n$ check if pushing selection
   criteria are satisfied. If they are satisfied for some $j$ and $i$, then

   a. reorganize AST by pushing predicate $p_j$ to $q_i$;
   b. goto 1;

   else for each $p_j, j = 1,2,...,m$, for each $q_i, q_{i+1}, i = 1,2,...,n-1$ check if pushing
   selection criteria are satisfied. If they are satisfied for some $j$ and $i$, then

   a. reorganize AST by pushing predicate $p_j$ to the pair $(q_i, q_{i+1})$;
   b. goto 1;

   else for each $p_j, j = 1,2,...,m$, for each $q_i, q_{i+1}, q_{i+2}, i = 1,2,...,n-2$ check if
   pushing selection criteria are satisfied. If they are satisfied for some $j$ and $i$, then

   a. reorganize AST by pushing predicate $p_j$ to the triple $(q_i, q_{i+1}, q_{i+2})$;
   b. goto 1;

   else ... (check for quadruples of $q_i$, etc. till the number of neighboring $q_i - s$ is
   equal to $n-1$)
```

Note that in each cycle of the algorithm we push one predicate (through reorganizing AST). Because after the reorganization the pattern is different hence it makes no sense to continue. We break further actions and start the algorithm from the beginning. The algorithm is naturally finished when the required pattern is not found or the number of required neighboring subqueries achieves *n*.

The above algorithm does not exhaust optimization opportunities. Note that in Example 1 we can also move the predicate *boss.Emp* = *e* to the pair of second and third subquery, obtaining in this way the following optimized query:

$$\begin{gathered}((\mathit{Person}\ \textbf{where}\ \mathit{age} > 50), \\ (((\mathit{Dept}\ \textbf{where}\ \textbf{count}(\mathit{location}) = 3),\ \mathit{Emp}\ \textbf{as}\ e)\ \textbf{where}\ \mathit{boss.Emp} = e)) \\ \textbf{where}\ \mathit{lname} = \mathit{dname}\end{gathered} \tag{4}$$

Hence we obtain some choice: when we push *lname* = *dname* we cannot push *boss.Emp* = *e*, and v/v. We can optimize this choice by some additional criterion (e.g. higher predicate selectivity ratio), but this could lead to very sophisticated reasoning with no guarantee for the success. Additionally, we observe that a predicate can be stuck with two or more $q_i - s$ that are not neighboring. Hence the presented algorithm anyway requires some alteration to take into account these opportunities.

These alterations require some new features of the AST mechanism. Consider Example 1. We can introduce to AST an additional artificial subquery *dummy(i)*. This is an internal optimization trick, the programmer need not to be aware of it. The subquery means copying the value of the *i*-th element of a structure. Below we present the query (3) after this modification:

$$\begin{gathered}(\mathit{Person}, \mathit{Dept}, \mathit{dummy}(2), \mathit{Emp}\ \textbf{as}\ e)\ \textbf{where}\\ \mathit{lname} = \mathit{dname}\ \textbf{and}\ \mathit{age} > 50\ \textbf{and}\ \textbf{count}(\mathit{location}) = 3\ \textbf{and}\ \mathit{boss.Emp} = e\end{gathered} \tag{5}$$

Now we can push all predicates:

$$\begin{gathered}(((((\mathit{Person}\ \textbf{where}\ \mathit{age} > 50), (\mathit{Dept}\ \textbf{where}\ \textbf{count}(\mathit{location}) = 3))\ \textbf{where}\\ \mathit{lname} = \mathit{dname}), (\mathit{dummy}(2),(\mathit{Emp}\ \textbf{as}\ e))\ \textbf{where}\ \mathit{boss.Emp} = e))\end{gathered} \tag{6}$$

The same trick we can use for the case when a predicate is to be stuck with subqueries that are not neighboring. At the end of run time query processing all dummy elements of structures are to be removed.

There is apparently a pitfall in our optimization method. Assume that *Emp* and *Dept* objects have a numeric attribute *rank*. Consider the query: Get pairs of employees and departments for which the summary rank is greater than 10:

$$(\mathit{Emp}, \mathit{Dept})\ \textbf{where}\ \textbf{sum}(\mathit{rank}) > 10 \tag{7}$$

According to the above criteria, the predicate $sum(rank) > 10$ will be pushed after the subquery *Emp* or, alternatively, after *Dept*. However in both cases pushing will cause changing the query semantics. Providing an employee and a department have ranks equal to 6, the original predicate will be true, but after pushing the selection the predicate will be false. Should we provide a method variant to deal with such cases?

We avoid to do that. We have concluded that this query should be treated as illegal and should be rejected by the strong typing system. In general, our strong type checker should disallow "false collections", i.e. situations when two or more components of the structure constructors deliver binders with the same name to the same environment. The programmer should reformulate this query to avoid false collections, e.g. by using auxiliary names to one or more arguments of the structure constructor. For instance, the equivalent query (8) is typologically correct, but it creates no opportunity for pushing the selection:

$$(\mathit{Emp}\ \textbf{as}\ e, \mathit{Dept})\ \textbf{where}\ e.\mathit{rank} + \mathit{rank} > 10 \tag{8}$$

3.2 Pushing Selections before Dependent Join

The dependent join is a binary operator hence there is no so many optimization variants as for the structure constructor. Consider the following query pattern:

$$(q_1 \textbf{ join } q_2) \textbf{ where } p_1 \textbf{ and } p_2 \textbf{ and } \dots\ p_m$$

According to the method described in the previous section, the static analysis of the query returns the signature s_1 of q_1 and the signature s_2 of q_2. Then, the optimizer calculates $static\_nested(s_1)$ and $static\_nested(s_2)$. The criterion of sticking a particular p_j with a particular q_i is exactly the same as for structure constructors. We are sticking p_j with q_1 if there is a name within p_j that can be resolved by a binder delivered by $static\_nested(s_1)$ and there is no name within p_j that is resolved by a binder delivered by $static\_nested(s_2)$. Similarly for q_2. The action can be recursively continued if q_1 or q_2 also contain joins.

Example 2. *Get departments (named d) together with their employees, with the courses supervised by them (named c) and with students attending these courses (named s). Department boss lname should be different from lname of an employee, employees should be older than 50, the department should have 3 locations and average grade of a student should be greater than 4.5.*

$$\begin{gathered}(((Dept \textbf{ as } d \textbf{ join } (d.employs.Emp)) \textbf{ join } (supervises.Course \textbf{ as } c)) \textbf{ join}\\ (c.attendedBy.Student \textbf{ as } s)) \textbf{ where } d.boss.Emp.lname <> lname \textbf{ and}\\ age > 50 \textbf{ and count}(d.location) = 3 \textbf{ and } avgGrade > 4.5\end{gathered} \tag{9}$$

Now we have a join with two arguments: the first line presents the first and the second one (before *where*) presents the second. The first, second and third predicate we can stick with the first argument (according to the criterion presented above) and the fourth predicate we can stick with the second argument:

$$\begin{gathered}(((((Dept \textbf{ as } d \textbf{ join } (d.employs.Emp)) \textbf{ join } (supervises.Course \textbf{ as } c)))\\ \textbf{where } d.boss.Emp.lname <> lname \textbf{ and } age > 50 \textbf{ and count}(d.location) = 3\\ \textbf{join } ((c.attendedBy.Student \textbf{ as } s) \textbf{ where } avgGrade > 4.5)\end{gathered} \tag{10}$$

Then, we can recursively repeat this action with parts of the entire query.

Another pattern with a dependent join is the following:

$$q_1 \textbf{ join } (q_2 \textbf{ where } p_1 \textbf{ and } p_2 \textbf{ and} \dots p_m)$$

In this case we can consider sticking some $p_j - s$ with q_1. The case does not imply anything new to our method: static analysis calculates the signature s_1 of q_1, then calculates $static\_nested(s_1)$ and then applies the described above criteria to all $p_j - s$.

$$Emp \textbf{ join } (worksIn.Dept \textbf{ where } sal > 1000 \textbf{ and } \text{"Rome"} \textbf{ in } location) \tag{11}$$

Query (12) presents an optimized variant of (11).

$$(\mathit{Emp}\ \textbf{where}\ \mathit{sal} > 1000)\ \textbf{join}\ (\mathit{worksIn.Dept}\ \textbf{where}\ \text{"Rome"}\ \textbf{in}\ \mathit{location}) \tag{12}$$

3.3 Pushing Selections before Quantifiers

For a query $\exists q(p)$ with an existential quantifier it is quite easy to prove that it is semantically equivalent to $\exists(q\ \textbf{where}\ p)(\mathit{true})$. Hence the patterns

$$\exists(q_1, q_2,...,q_n)\ (p_1\ \textbf{and}\ p_2\ \textbf{and}\ ...\ p_m)$$
$$\exists(q_1\ \textbf{join}\ q_2)\ (p_1\ \textbf{and}\ p_2\ \textbf{and}\ ...\ p_m)$$

can be rewritten as:

$$\exists((q_1, q_2,...,q_n)\ \textbf{where}\ p_1\ \textbf{and}\ p_2\ \textbf{and}\ ...\ p_m)(\mathit{true})$$
$$\exists((q_1\ \textbf{join}\ q_2)\ \textbf{where}\ p_1\ \textbf{and}\ p_2\ \textbf{and}\ ...\ p_m)(\mathit{true})$$

and in this way we have reduced the problem of pushing selection before an existential quantifier to the previously considered method. If some predicate p_j cannot be pushed before a structure constructor or a join, then we can move it back to a second quantifier argument.

Example 3. *Is it true that there is a clerk from the "Trade" department earning more than his/her boss?*

$$\begin{gathered}\exists\ (\mathit{Emp}\ \textbf{as}\ e\ \textbf{join}\ \mathit{e.worksIn.Dept}\ \textbf{as}\ d\ \textbf{join}\ \mathit{d.boss.Emp}\ \textbf{as}\ b)\\ (\mathit{e.job} = \text{"clerk"}\ \textbf{and}\ \mathit{d.dname} = \text{"Trade"}\ \textbf{and}\ \mathit{e.sal} > \mathit{b.sal})\end{gathered} \tag{13}$$

After rewriting:

$$\begin{gathered}\exists\ ((\mathit{Emp}\ \textbf{as}\ e\ \textbf{where}\ \mathit{e.job} = \text{"clerk"})\ \textbf{join}\ (\mathit{e.worksIn.Dept}\ \textbf{as}\ d\ \textbf{where}\\ \mathit{d.dname} = \text{"Trade"})\ \textbf{join}\ \mathit{d.boss.Emp}\ \textbf{as}\ b)\ (\mathit{e.sal} > \mathit{b.sal})\end{gathered} \tag{14}$$

For the universal quantifier, the de Morgan's laws can be applied to the above patterns. We present the rule for the pattern $\forall(q_1,q_2,...,q_n)$ $(p_1$ **or** p_2 **or**$...p_m)$; for patterns involving joins the rules can be developed by analogy. According to the laws, the pattern $\forall(q_1,q_2,...,q_n)$ $(p_1$ **or** p_2 **or**$...p_m)$ is equivalent to **not** $\exists(q_1,q_2,...,q_n)$ (**not** p_1 **and not** p_2 **and**...**not** p_m). In this way we again have reduced the problem to the previous cases.

4 Conclusions

We have presented the optimization method which was aimed at performing a selection as early as possible. Although this optimization is well-recognized for relational query languages, we generalize it for the object-oriented queries and for richer set of query patterns. We focus on pushing selection predicates being the part of the right-hand subquery of a *where* operator or a quantifier before structure constructors or join operators. Pushing selection before these operators is a lossless optimization which guarantees not worse and in majority of cases much better performance.

The proposed rewriting rules are developed in full generality. They hold for any data model (assuming it will be expressed in terms of SBA), including the relational and XML models. The rule makes also no assumption concerning the complexity of selection predicates and their location. Although the idea of rewriting rules seems to be clear, the resulting algorithm is quite sophisticated. The algorithm is repeated to detect and push down all the possible selections and ends when no optimization action is possible.

References

1. Adamus, R., et al.: Overview of the Project ODRA. In: Proc. 1st ICOODB Conf., pp. 179–197 (2008) ISBN 078-7399-412-9
2. Atkinson, M., Morrison, R.: Orthogonally Persistent Object Systems. The VLDB Journal 4(3), 319–401 (1995)
3. Bleja, M., Kowalski, T., Subieta, K.: Optimization of Object-Oriented Queries through Rewriting Compound Weakly Dependent Subqueries. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 323–330. Springer, Heidelberg (2010)
4. Ioannidis, Y.E.: Query Optimization. Computing Surveys 28(1), 121–123 (1996)
5. Jarke, M., Koch, J.: Query Optimization in Database Systems. ACM Computing Surveys 16(2), 111–152 (1984)
6. Kowalski, T., et al.: Optimization by Indices in ODRA. In: Proc. 1st ICOODB Conf., pp. 97–117 (2008)
7. Lentner, M., Subieta, K.: ODRA: A Next Generation Object-Oriented Environment for Rapid Database Application Development. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 130–140. Springer, Heidelberg (2007)
8. Subieta K.: ODRA (Object Database for Rapid Application development) Description and Programmer Manual (2008), http://www.sbql.pl/various/ODRA/ODRA_manual.html
9. Official Microsoft LINQ Project, http://msdn.microsoft.com/en-us/netframework/aa904594.aspx
10. Plodzien, J., Kraken, A.: Object Query Optimization through Detecting Independent Subqueries. Information Systems 25(8), 467–490 (2000)
11. Plodzien, J., Subieta, K.: Static Analysis of Queries as a Tool for Static Optimization. In: Proc. IDEAS, pp. 117–122 (2001)
12. Stencel, K.: Semi-strong Type Checking in Database Programming Languages. Editors of the Polish-Japanese Institute of Information Technology, Warsaw (2006)
13. Subieta, K.: Stack-Based Approach (SBA) and Stack-Based Query Language, SBQL (2008), http://www.sbql.pl
14. Subieta, K.: Stack-based Query Language. In: Encyclopedia of Database Systems, pp. 2771–2772. Springer US (2009)
15. Subieta, K.: Theory and Construction of Object Query Languages. Editors of the Polish-Japanese Institute of Information Technology, Warsaw (2004)
16. VIDE: Visualize All Model Driven Programming. European Commission 6th Framework Programme IST 033606 STP, http://www.vide-ist.eu
17. Warmer, J., Kleppe, A.: Object Constraint Language: Getting Your Models Ready for MDA. Addison Wesley (2003)

On Parallel Sorting of Data Streams

Zbyněk Falt, Jan Bulánek, and Jakub Yaghob

Abstract. Since the development of applications for parallel architectures is complicated and error-prone, many frameworks were created to simplify this task. One promising approach which is applicable especially for the development of parallel databases is expressing algorithms as stream programs, i.e. inputs and outputs of procedures are data streams and these procedures are connected so that they form an oriented graph. In this paper, we introduce highly scalable sorting algorithm which is suitable for streaming systems. We achieve this mainly by introducing multiway merge algorithm which is able to merge multiple independent sorted streams in parallel.

1 Introduction

There are many streaming systems and streaming languages for many architectures [19, 15, 2, 14]. These systems operate with data streams, which are basically flows of some data elements. Streams are processed by operators which may have multiple inputs and outputs. They transform input streams into output streams by performing their built-in operations. The program for streaming systems typically describes an oriented graph the nodes of which are operators on the streams and edges determine data flows. We denote this graph as an execution plan in the rest of the paper.

Streaming systems naturally introduce three types of parallelism [11] – task parallelism, when independent streams are processed in parallel, data parallelism, when independent parts of one stream are processed in parallel, and pipeline parallelism,

Zbyně Falt · Jakub Yaghob
Department of Software Engineering of the Charles University in Prague
e-mail: falt@ksi.mff.cuni.cz,yaghob@ksi.mff.cuni.cz

Jan Bulánek
Department of Theoretical Computer Science and Mathematical Logic of the Charles University in Prague
e-mail: bulda@math.cas.cz

when the producer of a stream runs in parallel with its consumer. Every efficient and scalable algorithm for these systems should take these types of parallelism into account; however, it might be sometimes difficult to exploit all of them together.

The sorting is fundamental algorithm and is used by many applications. Its streaming version (i.e. the input is a stream of tuples, instead of an array with all the tuples) is very important since there are many algorithms (such as the evaluation of query execution plan), which produce their output sequentially.

Although this is a basic operation, up to our knowledge, no specialized sorting algorithms for streaming systems in the multiprocessors with shared memory were proposed. The problem is that highly optimized algorithms for multicores and GPUs (e.g. [5, 18, 16, 17]) do not exploit pipeline parallelism. Similarly, it is hard to use results for the theoretical models as PRAM [6], because we have only limited number of processors available. Additionally, the communication between the processing units and synchronization is inappropriate for the streaming systems.

The work [12] introduces complex overview of sorting techniques used in databases but it focus mainly on non-parallel techniques and there is just brief overview of problems connected to parallelisms in parallel sorting.

The work [4] is very close to this work, since it also researches sorting in data stream environment – Auto-Pipe [10]. However, the algorithm fully corresponds to our SplitSortMergeNet, which has limited scalability according to our experiments (see Section 4).

The PMCC algorithm [13] uses multiway merge algorithm for merging of sorted substreams as we do; however, the algorithm is basically the same as SplitSortMergeNet (see Fig. 1) but with a net of multiway merges instead of the net of 2-way merges. Thus, the last merge limits the scalability of the sorting algorithm.

A work on parallel merging of data is already published [7], but it focuses only on merging of two sequences, which is unusable for our purposes.

Our contributions are:

- Introduction of scalable algorithm for the merge of multiple sorted streams in parallel. Moreover, the algorithm is usable in other areas than streaming systems.
- Introduction of parallel sorting algorithm suitable for streaming systems which is highly scalable and resistant to the skewness of input data.

The rest of this paper is organized as follows: In Section 2, we describe briefly the Bobox system, which we used for development and testing of sorting algorithm. In Section 3 we analyse and describe the algorithm used for sorting. The experimental results of the implementations are shown in Section 4. In Section 5 we summarize our results.

2 Bobox

The Bobox system [1, 8, 3] is an implementation of the streaming system. The system is responsible for evaluation of execution plans in multiprocessor system with shared memory. One of its main aims is to make the development of parallel databases easier.

The most important properties of the system are:

- Streams are split into packets, each packet contains a set of tuples with data elements.
- The communication between operators in the execution plan is possible only via packets. Since all operators reside in one address space, only shared pointers to packets are send. This also allows multiple operators to share data of packets and it significantly affects the performance of the whole system.
- The execution of one operator is always single-threaded. This makes the implementation of the operators much easier, since their developer does not have to take synchronization into account.
- The overall efficiency of the evaluation depends critically on the efficiency of the system. Therefore, its implementation is highly optimized and takes many hardware factors such as cache memories, system architecture etc. into account [9].

As we performed all benchmarks in the Bobox system, we adopted its terminology in the rest of the paper. Therefore, we denote operators as boxes, packets as envelopes and data tuples in packets as rows.

3 Parallel Sorting Algorithm

In this section by N we denote the total number of rows to be sorted and by L the number of rows which fit into one envelope.

In order to achieve task parallelism we use a traditional approach – we split the input stream into multiple independent substreams, sort them in parallel with some single-threaded algorithm and after that we merge them into the resulting stream.

For substream sorting we used algorithm, which sorts all envelopes of a substream independently on each other by `std::sort` function in C++ STL library. After each envelope of the substream is received and sorted, it starts to merge all received envelopes by simple single-threaded algorithm for multiway merge. This algorithm gives according to our experiments best results.

Parallel sorting algorithms used in streaming systems differ in the implementation of split and merge operation. Basically, there are two approaches:

1. Split the streams to several (preferably equally sized) substreams, so that all values in the $(i-1)$-th substream are lower than values in the i-th substream. In this case, the merge operation is simple, since it is only a concatenation of the substreams. The main problem is selecting samples which are used for the stream splitting.
2. Split the streams randomly to several equally sized substreams. The splitting is easy in this case in opposite to merge, which must merge all sorted substreams together.

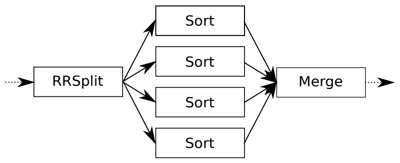


Fig. 1 The SplitSortMerge algorithm

Since we focus on skew-resistant algorithm, we chose the second approach. First of all, we developed very straightforward implementation of this algorithm, which we denoted as *SplitSortMerge* (Fig. 1). This algorithm splits streams in round robin manner into substreams, sort them concurrently and merge them together with multi-way merge.

However, the SplitSortMerge algorithm has a serial bottleneck, because only one box is responsible for the substream merging. Therefore, we tried to remove this bottleneck. First modification, we implemented, was the replacement of the single merge box by net of two-way merge boxes. We denote this algorithm as *SplitSort-MergeNet* (Fig. 2). The algorithm increased the scalability, since some merge boxes might work in parallel; however, the last merge box is still a bottleneck. We rule out this bottleneck in the next Section.

3.1 Parallelization of Stream Merging

To implement the parallel merging without any bottleneck, we introduce the *PMerge* box. Each PMerge box has K input streams (one from each sort box) and produces a predefined subsequence of the hypothetical stream S obtained by merging all K input streams S_j $(1 \leq j \leq K)$. More precisely, we number all K PMerge boxes from 1 to K and i-th PMerge box emits i-th, $(i+K)$-th, $i+2K$-th $\ldots$ envelopes of S. Since the PMerge boxes do not change input streams, they are obviously independent and they can be processed in parallel. With the PMerge boxes we might create the plan depicted in Fig. 3. We denote this algorithm as *SplitSortPMerge*. In this algorithm, the envelopes from the PMerge boxes are aggregated into one output stream by the *RRJoin* box. This box takes the envelopes from the PMerge boxes one by one in the round robin manner and produces the resulting stream S.

In this algorithm, the operations RRSplit and RRJoin manipulate only with the shared pointers to the envelopes and do not access their data. Each envelope contains typically thousands of rows [9], therefore, these operations are several orders

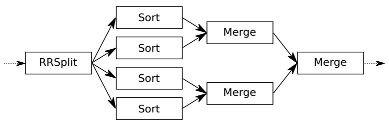


Fig. 2 The SplitSortMergeNet algorithm

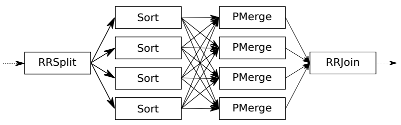


Fig. 3 The SplitSortPMerge algorithm

of magnitude faster than any other box in the algorithm and they do not limit the scalability significantly. The measurements prove this (see Section 4).

3.2 Implementation of the PMerge Box

Let us first focus on the classic algorithm for multiway merge of K sorted sequences. It keeps one pointer for each sequence. Each pointer is initially set to the beginning of its sequence. During one step of the algorithm it selects a pointer which points to the smallest value, puts this value to the output and increments the pointer. When the corresponding sequence becomes completely processed, its pointer is omitted from further calculations. The algorithm performs this step repeatedly until all sequences are completely processed.

The problem is that the algorithm is sequential in the sense that if we want to emit the i-th envelope, we must know what the positions of the pointers are after the creation of the $(i-1)$-th envelope. We show that these values may be computed much faster than by a straightforward simulation of the algorithm.

The key idea is to introduce an algorithm which moves the pointer by B positions instead of 1 after removing the first item of any stream. Let P_j^B be the position of the j-th pointer after $\lceil (i-1) * L/B \rceil$ steps. First notice, that for $B = 1$, P_j^B denotes the position of the j-th pointer in the stream, where the original algorithm would finish after creating the i-th envelope. Then it is easy to see that for arbitrary B and each j it holds that $|P_j^B - P_j^1| \leq B - 1$.

Consider that we performed the algorithm for some B and we got the positions of the pointers P_j^B. Now we want to increase the precision (i.e. find the pointers which are closer to P_j^1); therefore, we perform the algorithm for some $\hat{B} < B$ to get $P_j^{\hat{B}}$. If B is divisible by $\hat{B}$, the algorithm does not have to start from the beginning of the sequences. It is enough to start from the positions which are equal to $max(0, P_j^B - B)$ for each possible j. The number of steps which the algorithm performs to get $P_j^{\hat{B}}$ from P_j^B is $O(\frac{KB}{\hat{B}})$, since it is equal to

$$\left\lceil \frac{i \cdot L}{\hat{B}} \right\rceil - \left(\left\lceil \frac{i \cdot L}{B} \right\rceil - O(K) \right) \cdot \frac{B}{\hat{B}}$$

where $\frac{i \cdot L}{\hat{B}}$ is the number of steps which would have had to be done if the algorithm for $\hat{B}$ had started from beginning, $\frac{i \cdot L}{B}$ is the number of steps which were performed by algorithm for B when we start from the beginning, $O(K)$ is number of "steps back" and $\frac{B}{\hat{B}}$ is the number of steps for $\hat{B}$ which corresponds to one step of algorithm for B. This is equal to:

$$\left\lceil \frac{i \cdot L}{\hat{B}} \right\rceil - \left\lceil \frac{i \cdot L}{\hat{B}} \right\rceil - O\left(\frac{B}{\hat{B}}\right) + O\left(K\frac{B}{\hat{B}}\right) = O\left(\frac{KB}{\hat{B}}\right).$$

The whole algorithm for getting positions of the pointers after merging envelope i works is shown in Algorithm 1.

Algorithm 1. Algorithm for getting positions of pointers after finishing the i-th envelope

```
$B \leftarrow 2^{\lceil \log_2 i \cdot L \rceil}$
$pos \leftarrow 0$ {position in the output stream}
for $j = 1$ to $K$ do
    $P_j \leftarrow 0$
end for
while $B \geq 1$ do
    {steps back}
    for $j = 1$ to $K$ do
        if $P_j > 0$ then
            $P_j \leftarrow P_j - B$
            $pos \leftarrow pos - B$
        end if
    end for
    {improving of $P_j$}
    while $pos + B \leq i \cdot L$ do
        choose $j$ so that $S_j[P_j]$ is minimal
        $P_j \leftarrow P_j + B$
        $pos \leftarrow pos + B$
    end while
    $B \leftarrow \lfloor B/2 \rfloor$
end while
```

The time complexity of Algorithm 1 is $O(K \log(i \cdot L) \log K)$ because there are $O(\log(i \cdot L)$ phases and each phase takes $O(K \log K)$. During each phase, $O(\frac{K \cdot B}{B/2}) = O(K)$ steps for improving the pointers positions are performed and each step of this algorithm takes $O(\log K)$ since it must find a minimum of K values. This can be done easily for example with a heap data structure.

The implementation of the PMerge box is now trivial. Each PMerge box first calculates the positions in the input streams using the Algorithm 1 for the envelope it should produce. Then it merges the sequences, creates the output envelope and sends it. This is repeated until all envelopes are sent.

The only drawback of Algorithm 1 is, that its time complexity grows with the sequential number of the envelope which is being produced. However, it is easy to see, that after merging the current envelope, we can omit already used rows of input sequences in next calculations. With this modification, the time complexity of the algorithm is $O(K \log(L \cdot K) \log K)$ and does not depend on the sequence number of the produced envelope.

4 Results

To measure the results, we used the execution plan shown in Fig. 4. The first box generates pseudorandom input data. Subsequently, on the even positions there are the sort boxes and on odd positions there are *work* boxes. They only simulate some

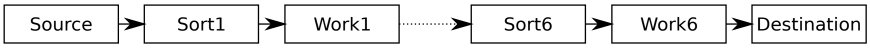


Fig. 4 The test execution plan

work on incoming envelope; in this case they multiple each row by a random number. This operation is advantageous, since it randomize data for the consecutive sorting box. There are six sorting and working boxes in the execution plan.

We measured the total time needed for the evaluation of the whole execution plan. We used 10^8 random 32b numbers. For the evaluation of the plan, we used the system Bobox which was described in the related paper [9] on a computer with four Intel Xeon E7540, which ran at 2.00GHz with 18MB shared L2 cache. Each processor has 6 cores with Hyper-Threading technology, thus the system has up to 48 worker threads available. The size of operating memory is 128 GB. The operating system is Red Hat Enterprise Linux Server 6.1 and we used g++ 4.4.6 with -O2 switch.

To show the throughput and the scalability of the sorting algorithm, we used multiple parallel work boxes (see Fig. 5) to randomize rows in parallel and to avoid the fact, that the work boxes are bottlenecks of the execution plan.

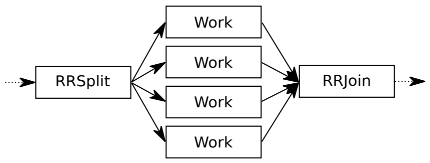


Fig. 5 The parallel work boxes

We performed 6 measurements. They differ by the level of parallelization of the sorting algorithm. As the level of parallelization we denote the number of parallel sort boxes used in the sorting algorithm (the number of auxiliary merge boxes used in the sorting algorithm corresponds to the level of parallelization). The results are depicted in Fig. 6.

From the measurement it follows that in the most parallelized measurement, the SplitSortPMerge algorithm is more than two times faster than the SplitSortMergeNet algorithm. In that measurements, the last merge box in the SplitSortMergeNet

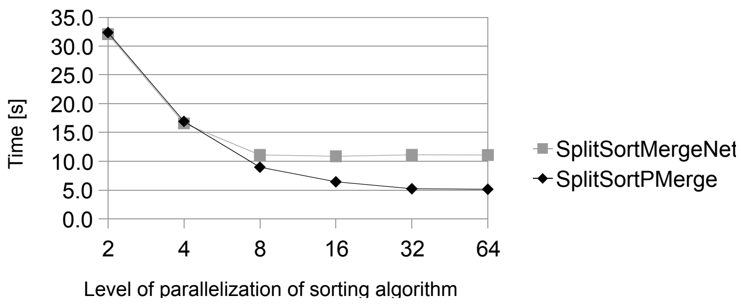


Fig. 6 The results with parallel work box

becomes the bottleneck, whereas the SplitSortPMerge has no obvious bottleneck. Therefore, its performance may grow further as the the level of parallelization increases.

However, the performance of the algorithm does not grow linearly with the increasing level of parallelization. There are three reasons – firstly, the number of utilized worker threads is greater than the level of parallelization because of pipeline parallelism. Secondly, the more boxes the execution plan has, the more overhead the Bobox system has with their management and their mutual communication. And thirdly, the system has only 24 physical cores with 48 logical processor thanks to Hyper-Threading technology. Therefore, the overall performance of the system does not grow linear with the number of utilized worker threads.

5 Conclusion

In this paper, we introduced a very scalable implementation of sort algorithm which is appropriate to be used in streaming systems. A very important part of the algorithm is the scalable parallel algorithm for multiway merge of sorted streams; however, the use of this algorithm is not limited to streaming systems. For example, this algorithm may easily replace the multiway merge algorithm (in fact the same as SplitSortMergeNet) in the work [5], which claims to present the fastest sorting performance for modern computer architectures at that time and further increase its scalability.

Acknowledgements. The authors would like to thank the Grant Agency of Charles University (GAUK), project no. 277911 and project no. 344711, which supported this paper. This work was also supported by the grant SVV-2012-265312.

References

1. Bednárek, D., Dokulil, J., Yaghob, J., Zavoral, F.: Using Methods of Parallel Semi-structured Data Processing for Semantic Web. In: Third International Conference on Advances in Semantic Processing, SEMAPRO 2009, pp. 44–49. IEEE (2009)
2. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P.: Brook for GPUs: Stream Computing on Graphics Hardware. ACM Transactions on Graphics 23, 777–786 (2004)
3. Čermak, M., Falt, Z., Dokulil, J., Zavoral, F.: Sparql query processing using bobox framework. In: The Fifth International Conference on Advances in Semantic Processing, SEMAPRO 2011, pp. 104–109 (2011)
4. Chamberlain, R., Galloway, G., Franklin, M.: Sorting as a streaming application executing on chip multiprocessors. Dept. of Computer Science and Engineering, Washington University, Tech. Rep. WUCSE-2010-21 (2010)
5. Chhugani, J., Nguyen, A., Lee, V., Macy, W., Hagog, M., Chen, Y., Baransi, A., Kumar, S., Dubey, P.: Efficient implementation of sorting on multi-core SIMD CPU architecture. Proceedings of the VLDB Endowment 1(2), 1313–1324 (2008)

6. Cole, R.: Parallel merge sort. In: 27th Annual Symposium on Foundations of Computer Science, pp. 511–516. IEEE (1988)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press (2009)
8. Falt, Z., Bednárek, D., Čermak, M., Zavoral, F.: On Parallel Evaluation of SPARQL Queries. In: The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2012, pp. 97–102 (2012)
9. Falt, Z., Yaghob, J.: Task scheduling in data stream processing. In: Proceedings of the Dateso 2011 Workshop, pp. 85–96. Citeseer (2011)
10. Franklin, M., Tyson, E., Buckley, J., Crowley, P., Maschmeyer, J.: Auto-pipe and the X language: A pipeline design tool and description language. In: 20th International Parallel and Distributed Processing Symposium, IPDPS 2006, pp. 1–10. IEEE (2006)
11. Gordon, M.I., Thies, W., Amarasinghe, S.: Exploiting coarse-grained task, data, and pipeline parallelism in stream programs. In: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS-XII, pp. 151–162. ACM, New York (2006)
12. Graefe, G.: Implementing sorting in database systems. ACM Comput. Surv. 38(3) (2006), http://doi.acm.org/10.1145/1132960.1132964, doi:10.1145/1132960.1132964
13. Hao, S., Du, Z., Bader, D., Ye, Y.: A partition-merge based cache-conscious parallel sorting algorithm for cmp with shared cache. In: International Conference on Parallel Processing, ICPP 2009, pp. 396–403. IEEE (2009)
14. Kapasi, U.J., Dally, W.J., Rixner, S., Owens, J.D., Khailany, B.: Programmable stream processors. IEEE Computer 36, 282–288 (2003)
15. Mark, W.R., Steven, R., Kurt, G., Mark, A., Kilgard, J.: Cg: A system for programming graphics hardware in a c-like language. ACM Transactions on Graphics 22, 896–907 (2003)
16. Merrill, D., Grimshaw, A.: Revisiting sorting for GPGPU stream architectures. In: Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques, pp. 545–546. ACM (2010)
17. Satish, N., Harris, M., Garland, M.: Designing efficient sorting algorithms for manycore GPUs. In: IEEE International Symposium on Parallel & Distributed Processing, IPDPS 2009, pp. 1–10. IEEE (2009)
18. Satish, N., Kim, C., Chhugani, J., Nguyen, A., Lee, V., Kim, D., Dubey, P.: Fast sort on CPUs and GPUs: a case for bandwidth oblivious SIMD sort. In: Proceedings of the 2010 International Conference on Management of Data, pp. 351–362. ACM (2010)
19. Thies, W., Karczmarek, M., Amarasinghe, S.: StreamIt: A language for streaming applications. In: Horspool, R.N. (ed.) CC 2002. LNCS, vol. 2304, pp. 179–196. Springer, Heidelberg (2002)

Mining Association Rules from Database Tables with the Instances of Simpson's Paradox

Wojciech Froelich

Abstract. This paper investigates a problem of mining association rules (ARs) from database tables in the case of the occurrence of Simpson's paradox. Firstly, the paper reports that it is impossible to mine reliable association rules using solely objective, data-based evaluation measures. The importance of the problem comes from the fact that in non-experimental environments, e.g. in medicine or economy, the Simpson's paradox is likely to occur and difficult to overcome by the controlled acquisition of data. This paper proposes a new approach that exploits the supplementary knowledge during the selection of ARs, and thus overcomes the presence of Simpson's paradox. In the experimental part, the paper identifies the problem in exemplary real-world data and shows how the proposed approach can be used in practice.

1 Introduction

The main objective of data mining is the development of generalizing models based on limited amounts of raw data. These models can be used for the classification of new observations, prediction or explanatory reasoning. One of the best-known approaches for developing models from data is mining of association rules [1]. Despite of many existing algorithms for mining ARs, one of the main issues that remains unsolved is the applicability and reliability of ARs while using for real-world tasks.

The Simpson's paradox is a phenomenon that can be found in many data sets, and is therefore an important factor that may significantly impair the applicability of ARs. There are many general methods for dealing with Simpson's paradox, and a review of them can be found in [10]. However, there are only few proposals to overcome the paradox in the data mining domain. The method proposed in [4] aims

Wojciech Froelich
Institute of Computer Science, University of Silesia, ul.Bedzinska 39, Sosnowiec, Poland
e-mail: Wojciech.Froelich@us.edu.pl

at the integration of Bayesian networks with the list of occurrences of the paradox. The approach proposed in [3] focuses on detecting the paradox to discover surprising patterns in data. The impact of Simpson's paradox on market basket analysis has been analyzed in [9].

There are only two approaches to deal with Simpson's paradox while mining ARs. The first trivial intuitive solution is to correct (by deleting or supplementing) the set of source data to get rid of the paradox. Intuitively, it is possible to omit the data that have the detected paradox, however, some valuable information can be lost. Consequently, the loss of information can lead to spurious results in the final set of mined ARs. The controlled supplementary acquisition of data, that could lead to the disappearance of the paradox, cannot be performed in non-experimental environments such as medicine or economy. The second known approach to overcome the paradox proposes a new data-driven measure that evaluates whether the ARs are applicable for the prediction task [9].

The first contribution of this paper is the analysis of the applicability of association rules, assuming they are mined from a database table with the occurrence of Simpson's paradox. It is shown explicitly how the issue of Simpson's paradox propagates on the process of forward and background reasoning using a set of the mined ARs. The second contribution is a new context-aware method for the selection of ARs. The approach presented in this paper exploits the general idea proposed by Pearl [10] of using background causal knowledge for overcoming Simpson's paradox in data. However, the method and algorithm proposed in this paper was developed specifically for the selection of ARs. Moreover, this paper also proposes a simple method to exploit the information on temporal sequence of events to construct the required background knowledge base. The proposed method is intended to be used as supplementary (does not exclude or recommend the application of any other method) to other methods for the evaluation and selection of ARs.

The remainder of this paper is organized as follows. Firstly, due to the dependency of two addressed issues, the section 2 and the section 3 present theoretical background of association rules and Simpson's paradox, respectively. After problem formulation in section 4, section 5 illustrates the impact of Simpson's paradox on the applicability of ARs. In section 6, a new method is proposed for the evaluation and selection of ARs. In section 7, the proposed method is illustrated using selected case studies. Section 8 concludes the paper.

2 Association Rules

The advantage of association rules in comparison to classification methods is the possibility to discover long patterns existing within data. The ARs are able to model the chains of probabilistic dependencies between facts instead of single-step functionality represented by, e.g. classification rules. Due to the space limitation, the benefits of using ARs cannot be presented here. The following notation is usually used when analyzing ARs. Let $I = \{i_1, i_2, \ldots, i_n\}$ be the set of so-called items. Every subset of I is called an itemset. Source data for ARs are represented as the set $D = \{T_1, T_2, ..., T_m\}$ of so-called transactions, in which each transaction $T_i \in D$ is an itemset, i.e., $T_i \subseteq I$. The association rule is denoted in the form: $X \Longrightarrow Y$, where $X \subset I, Y \subset I$ are itemsets. Given a set of transactions D, the goal of AR mining is to find and select rules with satisfactory quality. The predominantly used evaluation method [1] of ARs is based on quality (interestingness) measures of support (sup) and confidence ($conf$). The support measure can be applied solely to the itemsets, i.e., $\mathrm{sup}(\mathrm{X}) = \mathrm{card}(\mathrm{X})/\mathrm{card}(\mathrm{D})$. If the $sup(X) \geq sup_{min}$, the itemset X is referred to as frequent. Regarding the association rules the support, $sup(X \to Y) = \frac{card(\{T \in D : X \subset T \wedge Y \subset T\})}{card(D)}$ and confidence, $conf(X \to Y) = \frac{card(\{T \in D : X \subset T \wedge Y \subset T\})}{card(\{T \in D : X \subset T\})}$ measures are considered. Among all possible rules, only those are considered for the intended inference process, for that $sup(X \to Y) \geq sup_{min}$ and $conf(X \to Y) \geq conf_{min}$, where sup_{min} and $conf_{min}$ are the threshold values that are given by experts. Intuitively, a rule that has low support may occur simply by chance. The higher the confidence, the more likely it is that the itemset Y is present in transactions containing the itemset X.

After the introduction of ARs, a discussion occurred on the applicability of the initially proposed quality measures of rules. In spite of many existing successful applications, the sole use of support and confidence for the evaluation of association rules can create problems [14], e.g., redundancy (resulting in a large number of mined rules). The mined rules can be obvious and not interesting for users. These problems can lead to spurious conclusions, examples of which were reported in [2]. Detailed analysis has led to the introduction of an enormous number of interestingness measures for evaluating and selecting ARs. In general, the interestingness measures can be placed in two groups, i.e., measures of objective and subjective interest. The objective measures are data-driven i.e. based on calculations on the mined data. A review of diverse objective interestingness measures and related AR selection methods can be found in [6]. The first association rule mining method based on objective measures was the Apriori[1] algorithm.

The applications that use subjective interestingness measures of ARs refer to some background knowledge that cannot be found in originally mined data. The subjective interestingness measures check usually whether the ARs are surprising or actionable (useful). For instance, the method proposed in [8] classify rules according to the background knowledge, represented as a set of fuzzy rules. The other possibility is the filtering of interesting ARs [12] using predicate formulae in first-order logic that play the role of constraints. Another approach, proposed in [15], introduced various constraints to the mining process, e.g., the selection of rules with a particular itemset as a possible consequent. Reviews that include subjective interestingness measures are given in [11][5].

3 Simpson's Paradox

Simpson's paradox was first described by Yule in 1903 [16] and was named after Simpson who published an article on the subject in [13]. The paradox in its numerical form was presented in [7]. In order to illustrate the paradox, the population of patients in a hospital is considered. The observations of patients [7], including the cardinalities of sub-populations, are presented in Table 1. The set of all patients D, where $card(D) = 80$ is divided equally with respect to gender M, into sub-populations of 40 men $(M = m)$ and 40 women $(M = \neg m)$. The effect of treatment $(\mathrm{T} = \mathrm{t})$ (taking a drug) on recovery $(\mathrm{R} = r)$ with respect to gender M is considered. Both sub-populations of women and men were divided into a group of people who were given a treatment $(T = t)$, with the rest $(T = \neg t)$. Again, in both of these sub-groups, some patients recovered $(R = r)$, and some did not $(R = \neg r)$. On the basis

Table 1 Simpson's paradox in data

(a) m					(b) $\neg m$					(c) summary				
	r	$\neg r$		Rate		r	$\neg r$		*Rate*		r	$\neg r$		*Rate*
t	18	12	30	60%	t	2	8	10	20%	t	20	20	40	50%
$\neg t$	7	3	10	70%	$\neg t$	9	21	30	30%	$\neg t$	16	24	40	40%
	25	15	40			11	29	40			36	44	80	

of Table 1 it is decided whether the treatment should be applied with the intention to achieve the patient recovery (R=r). As can be seen in Table 1a, the recovery rate of males is higher among those men who have not taken a drug. Table 1b shows that the same situation is true for women. In both cases, the drug seems to be harmful. The data are summarized by gender in Table 1c, which shows that drug usage was beneficial and that the recovery rates among all patients who took the drug are higher. The corresponding decision problem can be stated by the question, "Which table should be applied when recommending the drug to a new patient?" Assuming the obvious fact that the gender of a patient is known, the right answer [10] is such that the drug should not be taken by patients.

Another interpretation of the data from Table 1 was given by Lindley and Novick [7]. The assumption was made that the numbers in Table 1 remain unchanged, but, rather than viewing them as related to people being treated with a drug, a different interpretation was used, they were examined from an agricultural context. In this case, T denotes varieties of plants that can be white $(T = t)$ or black, $(T = \neg t)$, the variable M informs whether the plant grew tall $(M = m)$ or short $(M = \neg m)$. The attribute R denotes high $(R = r)$ or low $(R = \neg)$ yields. This time it should be decided which variety of plant should be chosen to achieve better yields $(R = r)$. Obviously, observing the height of a fully grown plant cannot influence the decision concerning which variety to plant. Therefore, the combined Table 1c is consulted. In this case, the white variety is the better variety to plant.

A comparison of the two above-described cases (medical and agricultural) leads to a surprising conclusion. Based on the same raw data, two different decisions were made. In medical case, the data for subpopulations were examined; in the agricultural case, the combined table for the entire population were used. The above situation leads [10] to suspect that some information is missing that is needed to reach the right decision. According to Pearl [10], the knowledge that was missed within the Table 1 and that could help to reach the right decision is related to the cause-and-effect relationship among events. In the medical case, when M denotes gender, it becomes a causal confounder, i.e., it is a common cause of T and R. The gender M can determine whether the treatment T is applied or not. It can also influence the recovery R of a patient, i.e., people of one gender may recover faster than people of the other gender. Therefore, the gender M should be taken into account, and the correct choice is to consult the gender-specific tables and not use the drug. In the agricultural case, when M denotes information on the height of growing plants, the causal influence of M on the variety of plants T cannot be assumed, and the right decision is to plant the white variety (according to the combined Table 1c).

For the purpose of this paper, the Simpson's paradox was detected using the algorithm given by Fabris and Freitas [3]. This algorithm is based on the calculation of the so called probability reversal that was presented by Pearl [10]. Basically, the relationship between conditional probabilities in subpopulations determines the relationships between probabilities within the entire population. The only possibility to obtain the reversal of the probabilities within the entire population with respect to subpopulations is to violate the assumption (1) that gender is independent of taking a drug.

$$\begin{aligned} P(m|t) = P(m|\neg t) = P(m) = \alpha, \\ P(\neg m|t) = P(\neg m|\neg t) = P(\neg m) = 1 - \alpha, \end{aligned} \tag{1}$$

where $\alpha \in [0,1]$ is a constant. Estimating the conditional probabilities from Table 1, it can be noticed that (1) does not hold, i.e., $P(m) = 40/80 = 0.5, P(m|t) = 10/40 = 0.25$ and $P(\neg m|t) = 30/40 = 0.75$, and the reversal of probabilities occur in the given dataset.

Let $\{\mathrm{A_i}\}$ is the set of attributes of a given database table. For a given value of goal attribute $(\mathrm{A_d} = g)$, the algorithm detects two partitioning attributes: A_p and $\mathrm{A_c} \neq A_p$, for which the paradox occurs. It has been assumed [3] that the attribute A_p should be binary and partitions the data into two disjoint subsets (Pop_1, Pop_2). Thus, in every subset, it is possible to compute $Pr(\mathrm{A_d} = g|A_p = true)$ and $Pr(G_2) = Pr(\mathrm{A_d} = g|A_p = false)$. The second attribute A_c (that should be categorical) partitions the subsets Pop_1, Pop_2 into the populations Pop_{1j}, Pop_{2j} for every $A_c = j$. This enables to calculate $\forall j.Pr(G_{1j}) = Pr(\mathrm{A_d} = g|A_p = true, A_c = j)$ and $\forall j.Pr(G_{2j}) = Pr(\mathrm{A_d} = g|A_p = false,\ A_c = j)$. The algorithm detects the paradox if $\forall j.Pr(G_1) > Pr(G_2) \wedge Pr(G_{1j}) \leq \Pr(G_{2j})$ or $\forall j.\Pr(G_1) < \Pr(G_2) \wedge Pr(G_{1j}) \geq \Pr(G_{2j})$. The detection procedure can be called iteratively, assuming all attributes of the table to be A_d. The outcome of the algorithm is a list of attribute-triples $< A_d, A_p, A_c >$. The detection algorithm calculates also the specific measure M to evaluate the degree of the probability reversal detected within the data: $M = (M1 +$

$M2)/2$, where: $M1 = |Pr(G_1) - \Pr(G_2)|, M2 = (\sum_{j=1}^{m} |Pr(G_{1j}) - \Pr(G_{2j})|)/m$ and $m = card(domain(A_c))$.

4 Problem Statement

Let $\{A_i\}$ is the set of attributes of a given database table. Every assignment of the attribute to a certain value that occurs in a table row leads to the construction of the utterance (A_i= value), which is then interpreted as an item in the transaction corresponding to a given row of the table. The antecedent X and consequent Y parts of the rules are itemsets, i.e., $X \subset I$, $Y \subset I$. The rules constitute the rule knowledge base (RKB) that can be used e.g., for the two basic reasoning tasks.

Task 1: The first considered task is one-step forward reasoning. Let $O \subset I$ denotes the itemset that is a known observation and will be the initial set of facts used for the reasoning. The availability of O and the RKB is assumed. The unknown set of items Y that is associated with a given O should be discovered. To perform the reasoning step, the set O is matched $(O \subseteq X)$ to antecedent parts of the ARs. To achieve the reliable prediction of Y (as the consequent of the rule), it is necessary to select the appropriate rule from RKB.

Task 2: Let $G \subset I$ denotes the goal state from which the reasoning process starts. For the purpose of one-step backward reasoning the goal G is matched to the consequent parts Y of the rules. The itemset X (antecedent of the AR) should be discovered that leads to the goal $G \subseteq Y$.

5 Influence of Simpson's Paradox on the Applicability of Association Rules

The set of ARs mined from Table 1 is presented in Table 2. Since the considered goal observation is the patient's recovery (or yield of a plant), the consequent itemsets of ARs are limited to $Y \subseteq \{r, \neg r\}$. The obtained rule knowledge base (RKB) is to be used for the forward or backward reasoning tasks described in section 4. At first, it can be noticed that the values of confidence of rules $\{1, 2, \ldots, 8\}$ and $\{9, 10, \ldots, 16\}$ are complementary, i.e., $conf(rule1) + conf(rule9) = 1$, and that the decisions of the corresponding rules are opposite.

Suppose that the male patient has taken a drug, i.e. the itemset $O = \{t, m\}$ is observed and its consequences should be deduced using RKB. Using forward reasoning, the observation O is matched to the conditional part of the rules. The exact matching $(O = X)$ occurs for two rules: 5 and 13. After comparing the values of the support and confidence of these two rules, rule 5 is selected. The preference of rule 5 over rule 13 can be interpreted such that, after taking a drug, the male patient is more likely to recover than not. This fact spuriously suggests that taking a drug is

Table 2 Rule knowledge base (RKB)

N^o	Rule	Support	Confidence	N^o	Rule	Support	Confidence
1	$t \Rightarrow r$	0.25	0.5	9	$t \Rightarrow \neg r$	0.25	0.5
2	$\neg t \Rightarrow r$	0.2	0.4	10	$\neg t \Rightarrow \neg r$	0.3	0.6
3	$m \Rightarrow r$	0.3125	0.625	11	$m \Rightarrow \neg r$	0.1875	0.375
4	$\neg m \Rightarrow r$	0.1375	0.275	12	$\neg m \Rightarrow \neg r$	0.3625	0.725
5	$t \wedge m \Rightarrow r$	0.225	0.6	13	$t \wedge m \Rightarrow \neg r$	0.15	0.4
6	$\neg t \wedge m \Rightarrow r$	0.15	0.7	14	$\neg t \wedge m \Rightarrow \neg r$	0.0375	0.3
7	$t \wedge \neg m \Rightarrow r$	0.025	0.2	15	$t \wedge \neg m \Rightarrow \neg r$	0.1	0.8
8	$\neg t \wedge \neg m \Rightarrow r$	0.1	0.3	16	$\neg t \wedge \neg m \Rightarrow \neg r$	0.2625	0.7

beneficial to the male patient. In the agricultural case, rules 5 and 13 are interpreted differently. If the plants have grown high and the variety of the plants is white, the yields seem more likely to be high than low. Although, in this particular case, the decision is right, it does not take into account the yields of the plants that grow small $(\neg m)$. Independent of the interpretation of ARs from RKB, the forward-reasoning scheme (Task 1) is not suitable for solving the decision problem described in the example.

Suppose, the goal state consists of only one item, i.e., $G = r$ (the recovery of a patient or high plants). The objective is to find the most suitable set of items that will lead to the achievement of the goal G. After matching G to the consequent parts of the rules, the set of eight rules $\{1,2,\ldots,8\}$ is obtained. It should be decided whether the drug should be taken, therefore rules 3 and 4 (that do not involve items t or $\neg t$ within the premise) are not considered. Applying the confidence measure, rule 6 is selected. In the medical case, if the patient is female, rule 6 cannot be used, as it would contradict real gender. It is not clear whether rule 1 (with greater confidence) or rule 7 (that better reflects our knowledge) should be used. This is exactly the point at which Simpson's paradox occurs. In the agricultural case, rule 6 spuriously suggests the black variety of plant. Even under the assumption that the height of the plant is not known, there is no reason to prefer rule 1 (that would be the right choice in this case) instead of rule 6.

As can be noticed, in both the medical and agricultural cases, the support and confidence measures cannot indicate which ARs should be applied in which of the two cases.

6 Context-Aware Selection of Association Rules

In this section a new method for the selection of association rules is proposed. The method consists of three main processing steps:

1. identification of Simpson's paradox within the mined data,
2. construction of the causal knowledge base (CKB) that constitutes the domain context for the selection of ARs from RKB,
3. integration of the CKB with the set of association rules stored in RKB.

In the first step, the Simpson's paradox is detected using the algorithm of Fabris and Freitas [3]. Due to the limited space of the paper, the algorithm cannot be recalled here. If the paradox occurs, the second step of the proposed method is performed, in other case the algorithm is stopped.

Let $\kappa : A \times A$ be a binary relationship 'is-a-cause' within the set of attributes A. The utterance $A_1 \kappa A_2$ denotes that the setting of the value of attribute A_1 is the cause of observing a particular value of the attribute A_2. In the second step of the proposed method, the CKB is constructed by the domain experts. The CKB contains any syntactically correct, logical sentences built on the basis of the set of attributes A and relationship κ.

Note that the κ relates the pairs of attributes and not the events. In fact, during construction of the *CKB* it is usually not known what particular events are associated. In case of the described medical example, it is prior assumed that the treatment T can influence the state of patient R, i.e. $T \kappa R$. However, it is not known whether the influence is positive, i.e. the event $(T = t)$ leads to the state $(R = r)$ or negative and leads to $(R = \neg r)$. The causal knowledge obtained from experts may be questionable or limited. It is proposed to exploit information on temporal sequences of events. If the elementary events for two exemplary attributes A_1, A_2 occur always in the same temporal order for all their possible values, this fact can be denoted as $A_1 <_t A_2$, where $<_t$ denotes the relationship of temporal precedence. If $A_1 <_t A_2$ holds, it may be assumed that $A_1 <_t A_2 \Longrightarrow A_1 \kappa A_2$ also holds. On the other hand, if for two attributes it is known that $A_1 <_t A_2$, then it can be inferred that the reversed causality does not occur, i.e.: $A_1 <_t A_2 \Longrightarrow \neg(A_2 \kappa A_1)$.

The third step of the proposed method is the integration of CKB with the ARs stored in RKB. For any itemset X, that is the antecedent part of the association rule, a mapping $attr : X -> 2^A$ is defined, i.e. $attr(X)$ returns the subset of attributes of a given itemset $X \subseteq I$. For example, for $X = \{t, m\}$, the $\mathrm{attr(X)} = \{\mathrm{T, M}\}$. Suppose that the reverse mapping $itm : B -> 2^I$, returns the set of all possible items that can be created on the basis of the given subset $B \subseteq A$ of attributes. For example, for $\mathrm{B} = \{\mathrm{T, M}\}$, the $\mathrm{itm(B)} = \{\mathrm{r, m}, \neg\mathrm{r}, \neg\mathrm{m}\}$. The L denotes a set of triples of attributes: $< A_d, A_p, A_c >$, such that for every of them the Simpson's paradox has been detected. The two partitioning attributes are $\mathrm{A_p}$, $\mathrm{A_c} \in \mathrm{A}$ and the goal attribute is $\mathrm{A_d} \in \mathrm{A}$. The input of the integration procedure is a triple $L =$<CKB, RKB, L>. The output is the modified content of the RKB, this set of ARs will be used for reasoning. The following procedure performs the integration of CKB with RKB.

For every triple in L, perform two computational steps.

- Step 3A. Select from RKB the subset of rules (candidates) $\mathrm{RC} \subseteq \mathrm{RKB}$, for which the (2) holds:

$$\forall r \in RC.(A_p \in attr(X) \wedge A_d \in attr(Y)) \tag{2}$$

- Step 3B. Check within the CKB whether the condition (3) holds:

$$(A_c \kappa A_p) \wedge (A_c \kappa A_d) \tag{3}$$

If the condition (3) holds, delete from RKB these rule that belong to the set of candidates ($r \in RC$) for which $\mathrm{A_c} \notin \mathrm{attr(X)}$. Otherwise, if condition (3) fails, delete from RKB the rules $r \in RC$ for which $\mathrm{A_c} \in \mathrm{attr(X)}$.

7 Case Studies

The first case study refers to the data from Table 1 that played a role of the reference example. In the first step of the proposed method, the Simpson's paradox was successfully detected, the names of the partitioning attributes were $(A_p = T)$, $(A_c = M)$ and the goal attribute was $(A_d = R)$. For the construction of CKB the following assumptions were made. In case of medical interpretation of the data, it can be assumed that $(M <_t T)$ and $(M <_t R)$, i.e. the patient's gender is known before a drug is administered and before the possible recovery $(M <_t R)$ so, it was inferred that $(M\kappa T)$ and $(M\kappa R)$ hold. In the agricultural case, it is obvious that the height of plants M is known after they are grown $(T <_t M)$, i.e. after the selection of variety T. As a consequence, the height of plants M cannot be a cause of the color of the variety, i.e., $\neg(M\kappa T)$. Thus the two contradictory conclusions $(M\kappa T)$ and $\neg(M\kappa T)$ in medical and agricultural cases respectively was inferred. In the second step of the proposed method, assuming $(M <_t T)$ and $(M <_t R)$ the content of the first (in medical case) causal knowledge base was assumed as: $CKB1 = \{\mathrm{M}\ \kappa\ \mathrm{T},\ \mathrm{M}\ \kappa\ \mathrm{R}\}$. The content of CKB1 reflected the fact that the gender is a common cause of taking a drug T and recovery R. For both of the previously described cases (medical and agricultural), the value of T had to be discovered, such that could lead to the goal state $(G = r)$. For this purpose, the backward-reasoning was used. The conclusions of rules $9, 10, \ldots, 16$ did not match to the goal G and therefore they were deleted from the RKB. Rules 3 and 4 did not include in antecedent parts any of the items from $itm(T)$, therefore they did not provide information on the value of T. Finally, the following subset of rules were considered RKB=$\{$1,2,5,6,7,8$\}$. In the step 3A of the proposed method, the set RC had to be constructed. Every rule from RKB contained T in the corresponding set $attr(X)$. The goal attribute $R \in attr(Y)$. Therefore, the set of rule candidates was $RC = RKB$. Performing the step 3B of the proposed method, all rules for which $\mathrm{M} \notin \mathrm{attr(X)}$ were deleted from RKB. By this way rules 1 and 2 were filtered out. Rules 5, 6, 7 and 8 leaved in the RKB. As the gender of every patient is obviously known, so, separately for men and women, rules 6 and 8 with the higher confidence were finally selected for the following reasoning process. Both rules 6 and 8 come to the fact $\neg t$ that suggests not to take the drug, irrespective of gender. The similar procedure was performed in the agricultural case. It was assumed that $(T <_t M)$. The $CKB2 = \{\neg(M\kappa T)\}$. The $RC = RKB$ was the same as in medical case. In step 3B, the rules 5, 6, 7, and 8 were filtered out from RKB. Applying the confidence measure to the rest of the rules from RKB=$\{$1,2$\}$, the fact t was inferred that suggests that the white variety of plant should be chosen. In case of the reference example, the results obtained by the method proposed in this paper, delivered the correct results that corresponded to the analysis presented in section 3.

As mentioned in abstract the economy is the other non - experimental domain, where it could be difficult to overcome the occurrence of Simpson's paradox by simple omitting or supplementing the data. On the other hand, the so called basket analysis is the best known application of association rules. Due to these reasons it was decided to test the proposed method using a real-world data set consisted of 10,000 records of shop transactions. The set A was restricted to eight attributes described by the following labels: '1' - cigarettes; '2' - handkerchiefs; '3' - chocolate bars; '4' - lighters; '5' - tickets; '6'- sunflower seeds; '7' - newspapers; and '8' - ball-point pens. The selection of the interesting attributes were made by the management of the shop. The attributes' values were binary (true or false) and indicated whether a corresponding thing was bought by a customer. By this way, the type of data was in fact similar to this that is usually used for basket analysis. The goal of the simulation was to maximize the sales of cigarettes (the most profitable article) and to find what other articles entice smokers to buy cigarettes. Due to this objective, the goal attribute was assumed as (A_d = '1'), its desired value was true.

Firstly, the set of ARs was produced using the Apriori algorithm. The experiment was repeated many times for diverse values of parameters $sup_{min}, conf_{min}$. However, let us remind that the goal of this case study was not to optimize the value of $sup_{min}, conf_{min}$ but rather to test the possibility of overcoming the occurrence of Simpson's paradox by the method proposed in this paper. The finally assumed values $sup_{min} = 0.3$, $conf_{min} = 0.4$ enabled to produce 54 ARs. According to the assumed objective, only the ARs that contained the attribute '5' in their consequent part were considered, the other were deleted fom RKB.

In the first step of the proposed method, 76 instances of Simpson's paradox were detected in data. The instances were sorted with respect to the degree of surprise M. In Table 3 only the 10 most surprising instances of the paradox are shown. In the fourth column, the label of the goal attribute A_d was supplemented by its value (in brackets), for which the paradox occurred. As the goal attribute was A_d='1', among 10 most surprising instances of the paradox, there was only one instance 9 that affected the application of ARs from the RKB (as all rules from RKB contained '1' in its consequent part).

Table 3 The exemplary instances of Simpson's paradox in shop transactions

N^o	A_p	A_c	A_d	M
1	'6'	'4'	'2' [false]	0.3548611
2	'2'	'4'	'6' [false]	0.3548611
3	'4'	'6'	'5' [true]	0.3333333
4	'7'	'2'	'5' [true]	0.3333333
5	'1'	'6'	'5' [true]	0.3333333
6	'6'	'4'	'2' [true]	0.3305195
7	'2'	'4'	'6' [true]	0.3305195
8	'5'	'6'	'4' [true]	0.2916667
9	'5'	'6'	'1' [true]	0.2916667
10	'5'	'2'	'7' [false]	0.2916667

In the second step of the proposed method, the CKB was constructed. It was possible to assume that eating sunflower seeds '6' cannot cause using tickets '5' and smoking cigarettes '1', i.e. $CKB = \{\neg(6\kappa 5)\}$. Therefore, in the step 3A of the proposed method, the condition $(6\kappa 5) \wedge (6\kappa 1)$ did not hold and the ARs that contained the item '6'=true in their antecedent parts were deleted from RKB. By this way the spurious rules were deleted and the Simpson's paradox was overcome. Based on the final RKB, lighters '4', and newspapers '7' were recommended to be sold as the articles that increase the sale of cigarettes.

The above presented analysis illustrate how the method proposed in this paper works in practice. Without doubt, further experiments with real-world datasets are required. However, let us notice two factors that can influence the application of the proposed method and extend substantially the undertaken investigations. The first problem is the possible dependency of Simpson's paradox on the method of dealing with missing data. A trivial solution of deleting incomplete records would modify the statistical properties of dataset and therefore cannot be used in this case. It can be also noticed that in many real-world datasets, the domains of the considered attributes are continuous. The influence of the discretization methods (and their parameters) on the occurrence of Simpson's paradox is the other issue that should be investigated.

8 Conclusions

This paper illustrated the impact of Simpson's paradox on the applicability of association rules. The issue with the forward and background reasoning using ARs was explicitly demonstrated. It was justified that the detection of the paradox is necessary for the reliable reasoning with the use of ARs. A new method was also proposed that overcomes the problem with the paradox. The proposed method requires the availability of background knowledge acquired from experts. It was shown how this background knowledge can be integrated with the mined set of ARs and how it overcomes the detected paradox. The method is independent of the algorithm applied for mining ARs, it is intended to be applied as a post processing procedure after the generation of ARs. The presented case study illustrated and justified the application of the proposed method in practice. Further practical verification of the proposed method is a challenge for future research.

References

1. Agrawal, R., Imielinski, T.: Mining association rules between sets of items in large databases. In: ACM-SIGMOD, pp. 207–216 (1993)
2. Brijs, T., Vanhoof, K., Wets, G.: Defining interestingness for association rules. International Journal: Information Theories and Applications 10, 370–376 (2003)

3. Fabris, C., Freitas, A.: Discovering surprising patterns by detecting instances of simpson's paradox. In: Research and Development Intelligent Systems XVI, pp. 148–160. Springer (1999)
4. Freitas, A., McGarry, K., Correa, E.: Integrating bayesian networks and simpson's paradox in data mining. In: Russo, F., Williamson, J. (eds.) Causality and Probability in the Sciences. Texts in Philosophy, vol. 5, pp. 43–62. University of Kent, College Publications, United Kingdom (2007)
5. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38(3), 9 (2006), doi: `http://doi.acm.org/10.1145/1132960.1132963`
6. Guillet, F., Hamilton, H.J. (eds.): Quality Measures in Data Mining. SCI, vol. 43. Springer (2007)
7. Lindley, D., Novick, M.: The role of exchangeability in inference. Annals of Statistics 9, 45–58 (1981)
8. Liu, B., Hsu, W., Mun, L.F., Yan Lee, H.: Finding interesting patterns using user expectations. IEEE Transactions on Knowledge and Data Engineering 11, 817–832 (1996)
9. Ma, H., Dennis, K.: Effects of simpsons paradox in market basket analysis. Journal of Chinese Statistical Association 42(2), 209–221 (2004)
10. Pearl, J.: Causality, Models Reasoning and Inference. Cambridge University Press (2000)
11. Silberschatz, A., Tuzhilin, A.: On subjective measures of interestingness in knowledge discovery. In: KDD, pp. 275–281 (1995)
12. Silberschatz, A., Tuzhilin, A.: What makes patterns interesting in knowledge discovery systems. IEEE Transactions on Knowledge and Data Engineering 8, 970–974 (1996)
13. Simpson, E.: The interpretation of interaction in contingency tables. Journal of the Royal Statistical Society 13, 238–241 (1951)
14. Srikant, R., Agrawal, R.: Mining generalized association rules. Future Generation Computer Systems 13(2-3), 161–180 (1997)
15. Srikant, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Heckerman, D., Mannila, H., Pregibon, D., Uthurusamy, R. (eds.) Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD, pp. 67–73. AAAI Press (1997)
16. Yule, G.: Notes on the theory of association of attributes of statistics. Biometrika 2, 121–134 (1903)

Synchronization Modeling in Stream Processing*

Marcin Gorawski and Aleksander Chrószcz

Abstract. Currently used latency models in stream databases are based on the average values analysis that results from Little's law. The other models apply theory of M/G/1 queuing system. Theses solutions are fast and easy to implement but they omit the impact of streams synchronization. In this paper, we introduce a heuristic method which measures the synchronization impact. Then we have used this solution to extend the popular model based on average values analysis. This modification allows us to achieve better accuracy of latency estimation. Because schedulers and stream operator optimization require a fast and accurate model, we find our model a good starting point to create better optimizers.

1 Introduction

A well-written and scheduled Data Stream Management System(DSMS) should meet deadlines and provide predictable performance. Real stream databases are highly complex systems which implement different optimization algorithms.

The latency and memory optimization has been widely analyzed in scheduler algorithms [8, 14, 13, 10, 7, 5, 2]. Also, big latency of tuples can result from temporary overload. At such moments, calculations cannot be processed fluently and stream queues rapidly become longer. Eventually, this situation may lead to a

Marcin Gorawski · Aleksander Chrószcz
Silesian University of Technology, Institute of Computer Science,
Akademicka 16, 44-100 Gliwice Poland
e-mail: {Marcin.Gorawski,Aleksander.Chroszcz}@polsl.pl

Marcin Gorawski
Wroclaw University of Technology, Institute of Computer Science,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: Marcin.Gorawski@pwr.wroc.pl

* Supported by EFS Grant POKL.08.02.01-24-019/08.

breakdown because of the shortage of storage resources. If a system specification allows a generation of incomplete results, we can randomly delete some tuples from data streams. Thanks to this, the amount of data to be processed is reduced and latency requirements can be satisfied for part of the data. The above solution is named load shedding and its usability was presented in [15, 3, 16].

The estimation of the average latency time is useful to tune schedulers which switch between time and memory optimization. This value is also helpful to adjust starting time when partial results should be generated when some data streams are temporarily not available. In the paper, we show that the synchronization of inputs in the presence of binary operators such as joins can cause a skew in the way tuples are introduced into downstream operators. This causes a substantial error of latency estimation which we want to reduce. We have run extensive experiments to find a model which can be easily applied in DSMS. Finally, we arrived at some heuristic model which describes the relations between utilization level, query definition and latency. Our model does not take into account moments when a node is temporarily overloaded beyond its CPU capacity. Our solution also does not focus on the impact of network links either. Despite those assumptions, its accuracy is bigger in comparison with popular average values analysis in DSMS.

The rest of this paper is organized as follows: Section 2 introduces the background of modeling stream queries; Section 3 describes the basic approach to modeling; Section 4 defines our theory of modeling stream queries; Section 5 explains how latency is calculated; next in section 6 our model is tested against competitive solutions; and finally Section 7 concludes the paper.

2 Basic Terms

Stream query Q is defined as a directed acyclic graph (DAG), whose nodes and arcs represent stream operators and streams respectively. Query Q is divided into sub queries $Q = \{u_1, u_2, ..., u_n\}$ and each of them is deployed on a calculation node in a distributed system. The calculation node is a separate processor or computer in a stream database. Besides, each stream operator has the following parameters defined:

1. Selectivity s_x is an average number of output tuples resulting from processing one input tuple by operator x.
2. Productivity u_x is the probability of generating at least one tuple by operator x when one input tuple is processed. Productivity and selectivity are equal for *selection*, *projection* and *map* operators. In contrast to them, a *join* operator's productivity and selectivity are different.
3. Processing cost c_x is the time required for an input tuple to be processed by operator x.

A single portion of data transmitted by a stream is a tuple. It has a timestamp which defines the time of its entry into a system. Each stream contains tuples ordered chronologically. Besides, each stream is described by the average tuple latency l. The time when a tuple enters a stream database and the time when this tuple causes the generation of a new tuple at the output of a given operator A constitute the beginning and the end of measured latency for operator A. In the paper, the latency is decomposed on three elements: operator processing time, synchronization time and queuing time.

We can classify stream operators according to the number of their input streams. Operators with one input stream are called unary operators, while operators with two input streams are labeled as binary operators. Stream operators process tuples in chronological order. Consequently, at each point of time instance an operator processes an input tuple with the smallest timestamp. When an operator is of binary type then the tuple with the smallest timestamp can be identified only when neither of the input streams is empty [11, 4].

In a stream database the time associated with an operator corresponds to a tuple timestamp which has been recently popped from its input stream. This time is named operator local time. From this viewpoint, the time flow is frozen between successive tuples. The shorter the interval between consecutive timestamps, the better the freshness of the local time associated with an operator and a stream. Let us define interval τ which is the time between those local time updates. There frequently exist a few tuples with the same timestamps in a stream. This is caused by operators which generate a few result tuples after processing one input tuple. As a consequence, if we want to measure interval τ between local time updates, we cannot divide the time of a stream observation by the number of popped tuples during this time. In the analysis of binary operators more complicated situation can be encountered. They process input tuples as long as both input streams are not empty. When one of them becomes empty, the processing is suspended. As a result, the operator time updates are divided into slots when stream processing is available. As a consequence, the distribution of output tuples can be described as groups of tuples separated by intervals. The bigger the interval between those groups is, the greater the number of tuples appearing per group. The illustration of such an effect is shown in fig. 1. We define the time of tuple group as a timestamp of the last tuple in the group. This metric informs us how often the time associated with a given operator and stream is updated. This is another way of representing data stream freshness [12].

Let us define global selectivity s_{path}. A sequence of operators which connects source a with operator b is defined as $path = \{a, ..., b\}$. The path which connects operators O_0 with O_3 in fig. 2 is defined as $\{O_0, O_2, O_3\}$. Then the global selectivity defined on $path$ equals $s_{path} = \prod_{i \in path} s_i$. Summing up, the value s_{path} represents the average number of tuples generated at the output of operator b when one tuple is processed through this $path$.

Besides, we also define set Src, which consists of source operators $1, ..., K$. The average output rate of those operators is defined by vector $X = (X^{(1)}, X^{(2)}, ..., X^{(K)})$.

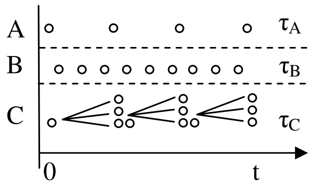


Fig. 1 The formation of groups of output tuples

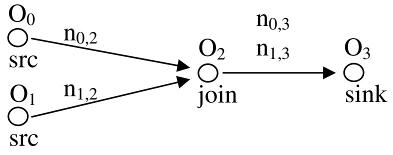


Fig. 2 Sample query graph

3 Basic Approach

The average values analysis that results from Little's law is the basic way of modeling queuing systems. This approach is popular in stream databases [9] because it is created only upon average metrics of query parameters. Our model assumes that a stream database consists of multiple processing nodes which serve multiple classes of clients; each processing node i evaluates sub query u_i.

Figure 2 illustrates a sample query which will be used to explain an operational analysis. Let us assume that $n_{a,b}$ is an average number of tuples flowing into operator b as a result of processing one tuple from source a. Besides, we define average tuple latency $l_{a,b}$ at the output of operator b which is a consequence of processing tuples from source a. Now, we want to estimate cumulative latency l_b, which represents the latency of all output tuples of operator b. Let us notice that the tuple latency of operator O_2 depends on which operator path a tuple has been processed on. For example in fig. 2, there exist two paths connected with operator O_2:$\{O_0, O_2\}$ and $\{O_1, O_2\}$. In order to find cumulative latency l_{O_2}, we have to estimate latencies for both paths and combine them. The formula below defines cumulative latency for operator b:

$$l_b = \frac{\sum_{a \in Src} X^{(a)} n_{a,b} l_{a,b}}{\sum_{a \in Src} X^{(a)} n_{a,b}} \tag{1}$$

Now we will estimate $l_{a,b}$. Each operator is directly or indirectly supplied by sources. The frequencies at which tuples from sources are transfered to operator o are defined by vector $X_o = (X_o^{(1)}, X_o^{(2)}, \ldots, X_o^{(K)})$. The utilization of calculation node i caused by operator o which processes tuples from source k equals:

$$U_i^{(o,k)} = X_o^{(k)} \bar{B}_i^{(o)} = X^{(k)} \bar{D}_i^{(o,k)} \tag{2}$$

where: $\bar{D}_i^{(o,k)} = B_i^{(o)} n_{k,o}$; $\bar{B}_i^{(o)}$ - an average time of processing a tuple by operator o deployed on processing node i.

The utilization of processing node i equals:

$$U_i = \sum_{o \in u_i} \sum_{k=1}^{K} U_i^{(o,k)} \tag{3}$$

This formula shows how long each operator located on processing node i calculates tuples which result from inserting tuples at source streams 1..K. The stream database is in a steady condition when all $U_i < 1$.

The average visit time of tuples queued to operator o at processing node i which results from processing one tuple from source k is:

$$\bar{R}_i^{(o,k)} = \frac{\bar{D}_i^{(o,k)}}{1 - U_i} \tag{4}$$

Finally, latency l_{path} is the sum of values by which latency increases when a tuple passes successive operators on the $path$. After processing one tuple from source k, a group of tuples can appear at operator o. We knew the average visit time at o which defines the time of processing this whole group of tuples. Now we want to find the relation between the average visit time and the value by which the latency is increased after passing operator o. In order to solve this we have assumed that tuples arrive evenly in time according to average throughput. Due to this arithmetic progression is applied to approximate the value by which the latency is increased at operator o: $\frac{\bar{R}_i^{(o,k)}}{2}$. Summing up, latency l_{path} is:

$$l_{path} = \sum_{o \in path} \frac{\bar{R}_{Node(o)}^{(o,k)}}{2} \tag{5}$$

where: $path$ - is the sequence of operators; $Node(o)$ - is the function which returns the processing node for stream operator o.

4 Estimation of Synchronization Impact

The analysis of synchronization impact is divided into two phases. At the beginning, we explain when stream operators require additional time to synchronize streams. The estimation of this time is based on the stream freshness property. In the next subsection, we introduce the algorithm used to calculate this value for each operator input stream.

4.1 Latency Caused by Synchronization

Figure 3 shows the notation of a binary operator and fig. 4 explains its parameters. The vertical markers in fig. 4 represent the moments when consecutive tuple groups arrive at inputs A and B. Variables τ_A and τ_B represent the intervals between those tuple groups for stream A and B respectively. The average latencies of tuples at the input of the analyzed operator are l_A and l_B. In other words, l_a is the amount of time before the tuple in stream A synchronizes with a tuple from stream B and l_b

is the time before the next tuple in stream B synchronizes with the corresponding tuple from stream A. It is worth noticing that we analyze only the sequence in which tuples are processed by a binary operator. We are not interested in the semantics of this binary operator. In order to make the description of the synchronization process easier to follow, τ_A and τ_B have similar values in fig. 4. As a result one tuple appears after another one arrives at the other input. Nevertheless, our model is not limited to this scenario.

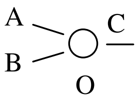


Fig. 3 The binary operator notation

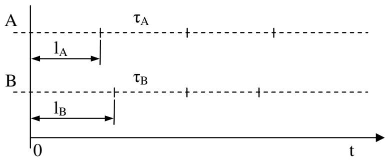


Fig. 4 Graphical representation of latency and τ

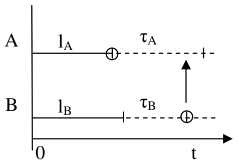


Fig. 5 Pessimistic scenario for input A

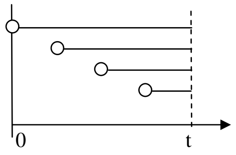


Fig. 6 The explanation of the latency calculation

If tuples arrive at input A with a latency greater than $l_B + \tau_B$ then stream B is not empty. Summing up, when $l_B + \tau_B < l_A$ then tuples at input A are processed directly. The other case is described by $l_B + \tau_B \geq l_A$ and illustrated in fig. 5. Tuples appear at input A with average latency l_A. Because the average interval between successive tuple groups equals τ_A, we assumed that one tuple appeared in the stream within τ_A with 100% probability. An analogous assumption is made for stream B.

The pessimistic scenario occurs when tuples arrive at input A at time l_A while tuples at input B arrive at time $l_B + \tau_B$. In this case tuple buffering lasts longest. Now we calculate the average latency for this pessimistic scenario. Let us notice that the sum of the latencies of all the tuples queued in stream A is the sum of the arithmetic sequence illustrated in fig. 6. Below we repeated the formula for the sum of elements in an arithmetic sequence: $S_n = a_1 + a_2 + ... + a_n = \frac{a_1+a_2}{2}n$. The average number of tuples which appear at input A between consecutive tuples at input B equals: $n_A = \frac{l_B+\tau_B-l_A}{\tau_A}$ In consequence, we arrive at the following formula: $l_{AC} = \frac{l_B+\tau_B-l_A+l_B+\tau_B-l_A-n_A\tau_A}{2}(n_A - 1)$ When we combine the above formulae, we achieve the following average latency for tuples from input A.

$$l_{AC} = \frac{l_B - l_A + \tau_B}{2} \tag{6}$$

Summing up, the stream synchronization of binary operators introduces additional latency, which can be estimated by the following rule.

$$l_{AC} = \begin{cases} 0 & when \quad l_B + \tau_B < l_A \\ \frac{l_B - l_A + \tau_B}{2} & otherwise \end{cases} \tag{7}$$

4.2 Stream Freshness

The average interval τ informs us how frequently the time of a stream is updated. Knowing this, we can calculate the output latency of the operators attached to those streams.

Figure 1 explains why tuple groups are inserted into output streams. Let us assume the average interval between tuple groups in stream A is three times longer than the corresponding interval in stream B. On average, three tuples from input B are consumed at the time of the tuple arrival at input A.

Let us assume that input A started producing tuples and input B started generating tuples after x time units. When $\tau_A > \tau_B$ then we should consider each $x \in (0, \tau_B]$ so as to cover all possibilities. Because tuples are generated evenly, we have limited our observation to the time which passes from the appearance of one tuple from the slower stream to appearance of another one. During this time n tuples in the faster stream appear waiting $\tau_A - x$; and one tuple appears in the slower stream and it waits x time units. Now we can estimate the average interval between tuple groups for a given x and it equals the weighted average value of x and $\tau_A - x$.

$$\tau_C(x) = \frac{x + n(\tau_A - x)}{1 + n} \tag{8}$$

When we assume that x appears with equal probability in the range from 0 to τ_B, we arrive at the formula.

$$\tau_C = \frac{1}{\tau_0} \int_{x=0}^{\tau_0} \frac{x + n(\tau_1 - x)}{1 + n} dx = \frac{-2.5\tau_0^2 + \tau_0\tau_1 - (2\tau_0^2 + \tau_0\tau_1)\ln\frac{\tau_1}{\tau_0 + \tau_1}}{\tau_0} \tag{9}$$

where: $\tau_0 = \min(\tau_A, \tau_B); \tau_1 = \max(\tau_A, \tau_B)$.

This formula allows us to simulate the average interval between groups of tuples depending on the metrics of input streams. Then this interval is important in calculating a latency increase for consecutive operators.

The graph in fig. 7 shows the value τ_C for different ratios τ_A/τ_B. It is worth noticing that the average interval between consecutive groups of tuples is halved when the ratio value ranges from 0.6 to 1. When the ratio value ranges from 0 to 0.6, then the resulting average interval is closer to τ_B. Moreover, the operator is prone to generate peaks of overload.

The broken plot in fig. 7 illustrates a hypothetical case when the binary operator is free of the synchronization burden. A situation like this occurs when the timestamps of all tuples are known a priori. In such a case we divide the period of observation

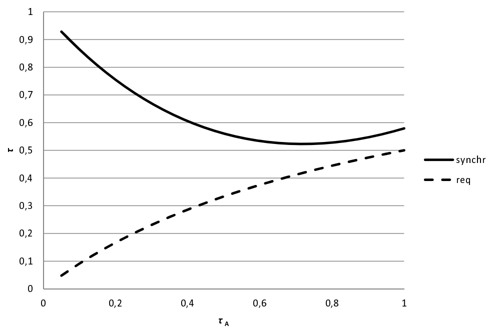


Fig. 7 The average interval between groups of tuples for changing proportion τ_A/τ_B

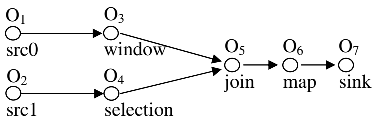


Fig. 8 Query DAG1

τ_B by the number of tuples which were processed by an operator. There is one tuple in stream B and $\frac{\tau_B}{\tau_A}$ tuples in stream A. Summing up, the average interval between groups of tuples is approximated as follows:

$$\tau_C = \frac{\tau_B}{1+\frac{\tau_B}{\tau_A}} \tag{10}$$

The comparison of both plots in fig. 7 shows that the synchronization of binary operators substantially affects the result latency. In order to process streams fluently it is important to have short intervals between groups of tuples. As it is shown in fig. 7, this cannot be achieved when the τ_A/τ_B ratio is under 0.6.

5 Average Latency

The mixed approach combines the impact of synchronization with the operational approach. The query DAG1 shown in fig. 8 will be used to explain the algorithm of latency estimation. Let us assume that operator o has attribute $visited$ which equals $false$ at the beginning of the calculation. There also exists function $next(o)$ which returns the set of operators connected to the output of operator o. Analogously we define function $prev(o)$, which returns the set of operators supplying tuples to operator o.

The latency estimation is a simple bottom-up calculation of $\bar{R}_i^{(o,k)}$ for each operator o and each source k as it is described in section 3. Having used this algorithm upon DAG1, we can achieve the following sequence of calculation: $O_1, O_2, O_3, O_4, O_5, O_6, O_7$. Next, we evaluate latencies on the paths connecting sources with each operator of the query according to alg. 1. Each operator o is described by the following properties: τ - it is an interval between consecutive groups of tuples; L - it is a map of latencies indexed by the source of stimulation.

At the beginning of alg. 1 method $initializeValues(o)$ setups values of source operators. This method gets the value of throughput $X^{(o.id)}$ for operator o, then assigns properties:

1. $o.\tau = \frac{1}{X^{(o.id)}}$
2. $o.L[X^{(o.id)}] = 0$

The consecutive steps of alg. 1 evaluate properties for unary and binary operators. Method $updateOp1(o)$ retrieves operator o_s which supplies operator o with tuples. Next the following properties are calculated:

1. For each source k: $o.L[k] = o_s.L[k] + \frac{\bar{R}_i^{(o.id,k)}}{2}$
2. $o.\tau = \frac{o_s.\tau}{o.u}$

The step 2 of the above method is necessary to model the extension of τ when an operator has low productivity. Method $updateOp2(o)$ retrieves operators: op_{s1} and op_{s2} which supply operator o with tuples. Next, the remaining steps of the method are processed:

1. Latency $l_{s1}(l_{s2})$ is calculated. The output rate for operator o is vector $Y = (Y_o^{(1)}, Y_o^{(2)}, \ldots, Y_o^{(K)})$. Those rates are divided according to data source k. Finally, l_{s1} is defined as:
$$l_{s1} = \frac{\sum_{k\in K} Y_{s1}^{(k)} o_{s1}.L[k]}{\sum_{k\in K} Y_{s1}^{(k)}} \tag{11}$$
   Analogically l_{s2} is calculated.
2. Formula (7) is used to calculate values $l_{s1,o}$ and $l_{s2,o}$
3. The latency is calculated as follows: $l_o = \frac{(l_{s1}+l_{s1,o})a_{s1}+(l_{s2}+l_{s2,o})a_{s2}}{a_{s1}+a_{s2}}$ where $a_{s1}(a_{s2})$ represents the average number of tuples generated by operator $o_{s1}(o_{s2})$ during a time unit.
4. Value $o.\tau$ is updated according to formula (9).
5. For each source k: $o.L[k] = l_o + \frac{\bar{R}_i^{(o.id,k)}}{2}$

When the evaluation of alg. 1 is completed, the latency of a given operator can be calculated by means of formula (11).

Algorithm 1 (Latency evaluation)
```
foreach(op:A) {
  if(op is Source) {
    initializeValues(op);
  } else if(op is unary operator) {
    updateOp1(op);
  } else if(op is binary operator) {
    updateOp2(op);
  }
}
```

6 Tests

The experiments described in this section were conducted on our stream data base simulator which is the result of our previous experience with stream database StreamAPAS. Thanks to this, we were able to isolate the impact of dataset rates or other processes which share the same node. We prepared two types of datasets. One consists of tuples whose timestamps are distributed according to the Poisson process. The other set is based on the time distribution measured in real system [1].The sizes of those datasets were between 100'000 and 800'000 tuples. We calculated average metrics for each dataset and use it during empiric and analytic calculations. Each query has been run multiple times with changing level of utilization, as a result we verified the accuracy of our model for a range of configurations.

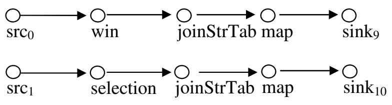


Fig. 9 Query DAG2

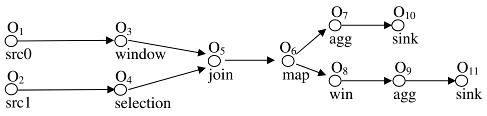


Fig. 10 Query DAG3

Let us now define the notation used in the following figures. A graph name with the suffix 'basic' labels graph which depicts estimation based on the operational approach. A graph name with the suffix 'mixed' shows estimation according to our mixed approach. The remaining graphs with the suffix 'simulation' indicate empirical results. Our aim was to create queries with different topologies. Because of the page limits we confront the empirical results with the analytic estimations for queries DAG1-DAG3, which are shown in figures: 8, 9 and 10. In fig.12 we can see that our mixed model substantially outperforms the popular basic approach that is used by current schedulers and optimizers. In order to measure the impact of binary operator synchronization. We have defined query DAG2 which replaces binary operator O_2 with unary operators with the same selectivity. Figure 13 shows that

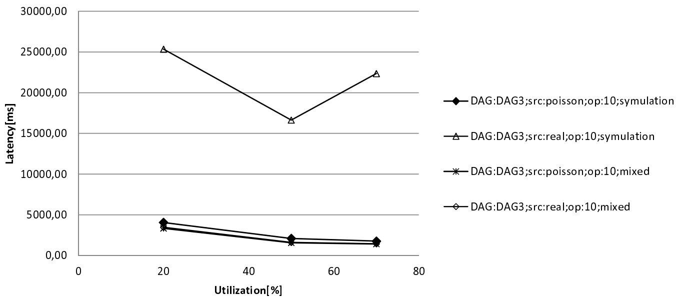


Fig. 11 Comparison of DAG3 results for different datasets

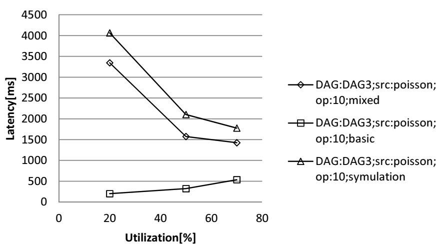


Fig. 12 Comparison of DAG3 results for the operational approach and the mixed approach

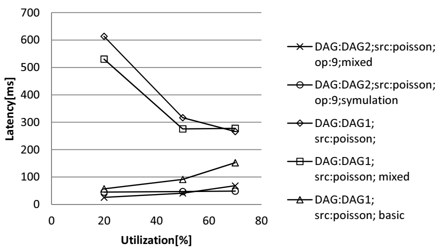


Fig. 13 Comparison of the synchronization impact between DAG2 and DAG1

the latency will be reduced from 300ms to values below 80ms if only synchronization could be avoided. Finally, we have tested how latency changes when real time distribution of data stream is applied. This effect is depicted in fig. 11.

The main conclusion drawn from our analysis of collected results is that the accuracy of our model is substantially higher than that of the basic approach. Despite the fact that our model is based on the analysis of average values, it offers nearly ideal precision for datasets generated by the Poisson process. The estimation accuracy is lower for datasets based on the real distribution of timestamps. This is caused by the fact that we measured average values for wide time windows. If we shorten those time windows then the assumption that data streams are generated by Poisson process will generate smaller error and we can achieve better estimation accuracy.

7 Conclusions

Multi-criteria optimization is a challenge in stream databases. When we consider a distributed system, then we have to monitor the utilization of the processing nodes. We can compose stream operators so as to optimize memory consumption but this optimization strategy affects the model of synchronization. As a consequence the memory optimization changes the latency of result tuples.

These above circumstances make the development of the analytical model important for future stream database systems. Currently the analysis based on average values is the most popular in such systems. Unfortunately, our tests show that this approach is weak when it comes to calculating the latency of result tuples. In the paper we have introduced an additional component, which reflects the impact of synchronization. The new metric in this component is the productivity property defined for each stream operator. By means of this we are able to more efficiently estimate the impact of synchronization. Thanks to this the exactness of latency estimation is substantially improved in comparison with the operational approach. What seems important, is that our component is based on average statistics, which makes our model easy to calculate in real-time systems. Our mixed model joins the analysis of utilization and latency in distributed stream databases which is a good foundation for the future schedulers. Besides, this model can be used to predict places where processing of a stream query can be easily destabilized.

During our further research we plan to create an optimizer which controls the frequency of boundary tuple [6] injection. In contrast to current algorithms which try to achieve the shortest latency, we want to achieve the predefined value. The next area of our interest is the creation of a scheduler which uses our model to strike a balance between memory and time optimization.

References

1. http://ita.ee.lbl.gov/html/contrib/lbl-pkt.html
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Thomas, D.: Operator scheduling in data stream systems. The VLDB Journal 13(4), 333–353 (2004)
3. Babcock, B., Datar, M., Motwani, R.: Load shedding for aggregation queries over data streams. In: ICDE 2004: Proceedings of the 20th International Conference on Data Engineering, p. 350. IEEE Computer Society, Washington, DC (2004)
4. Bai, Y., Thakkar, H., Wang, H., Zaniolo, C.: Optimizing timestamp management in data stream management systems. IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 1334–1338 (2007)
5. Bai, Y., Zaniolo, C.: Minimizing latency and memory in dsms: a unified approach to quasi-optimal scheduling. In: SSPS 2008: Proceedings of the 2nd International Workshop on Scalable Stream Processing System, pp. 58–67. ACM Press, New York (2008)
6. Balazinska, M.: Fault-tolerance and load management in a distributed stream processing system. Ph.D. thesis, Cambridge, MA, USA (2006); Adviser-Hari Balakrishnan
7. Carney, D., Çetintemel, U., Rasin, A., Zdonik, S., Cherniack, M., Stonebraker, M.: Operator scheduling in a data stream manager. In: VLDB 2003: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 838–849. VLDB Endowment (2003)
8. Hammad, M.A., Franklin, M.J., Aref, W.G., Elmagarmid, A.K.: Scheduling for shared window joins over data streams. In: VLDB, pp. 297–308 (2003)
9. Hwang, J.H., Xing, Y., Çetintemel, U., Zdonik, S.B.: A cooperative, self-configuring high-availability solution for stream processing. In: ICDE, pp. 176–185 (2007)
10. Jiang, Q., Chakravarthy, S.: Scheduling Strategies for Processing Continuous Queries over Streams. In: Williams, H., MacKinnon, L.M. (eds.) BNCOD 2004. LNCS, vol. 3112, pp. 16–30. Springer, Heidelberg (2004)
11. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21(7), 558–565 (1978)
12. Sharaf, M.A.: Metrics and algorithms for processing multiple continuous queries. Ph.D. thesis, Pittsburgh, PA, USA (2007)
13. Sharaf, M.A., Chrysanthis, P.K., Labrinidis, A.: Preemptive rate-based operator scheduling in a data stream management system. In: AICCSA 2005: Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications, pp. 46–I. IEEE Computer Society, Washington, DC (2005)
14. Sharaf, M.A., Chrysanthis, P.K., Labrinidis, A., Pruhs, K.: Efficient scheduling of heterogeneous continuous queries. In: VLDB 2006: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 511–522. VLDB Endowment (2006)
15. Tatbul, E.N.: Load shedding techniques for data stream management systems. Brown University, Providence (2007); Adviser-Zdonik, Stan
16. Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Load shedding in a data stream manager. In: VLDB 2003: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 309–320. VLDB Endowment (2003)

Towards Automated Enterprise Architecture Documentation: Data Quality Aspects of SAP PI

Sebastian Grunow, Florian Matthes, and Sascha Roth

Abstract. Well executed, Enterprise Architecture (EA) management is commonly perceived as a strategic advantage. EA management sermonizes IT savvy firms to take profound decisions based on mature EA information. As of today, gathering required information, i.e. documenting the EA, is regarded both, time consuming and error-prone. As a reaction, recent approaches seek to automate EA documentation by extracting information out of productive system environments. In our recent work we showed that a particular Enterprise Service Bus (ESB) namely SAP Process Integration can be used to extract EA relevant information. As a next step towards automated EA documentation, this paper analyzes the quality of the data stored in SAP Process Integration systems in practice. Survey results of 19 industry partners on 4 continents are presented.

Keywords: Enterprise Service Bus (ESB), SAP PI, data quality, automated Enterprise Architecture documentation.

1 Introduction and Motivation

Enterprise Architecture (EA) management is commonly perceived as strategic advantage [16]. Approaches from academia and practitioners, e.g. [23, 3], teach to take profound EA related decisions based on mature EA information. These approaches commonly start an EA endeavor by capturing the current state (as-is) of the EA and create stakeholder-specific visualizations for analyses [17, 10]. As of today documenting the EA requires manual collection of data and thus is regarded as an error prone, expensive, and time consuming task. As a reaction, researchers and

Sebastian Grunow · Florian Matthes · Sascha Roth
Software Engineering of Business Information Systems (sebis),
Technische Universität München, Garching b. München 85748, Germany
e-mail: {grunow,matthes,sascha.roth}@in.tum.de

practitioners [5, 7] seek to automate EA documentation. These approaches focus on extracting information out of productive system environments, but their data quality aspects are not addressed by the authors.

In our recent work [4, 9] we showed that a particular Enterprise Service Bus (ESB) implementation can be used to extract EA relevant information. We investigated an ESB since it can be "considered as the nervous system of an enterprise interconnecting business applications and processes as an information source" [4]. In our analysis we compared concepts contained in the ESB (e.g., interface descriptions, participatinng applications and systems) and the focus was put on the evaluation of the coverage degree to which data of a productive system can be used for EA documentation, i.e. we focused on a model mapping rather than data quality aspects. Results published assume the best data quality (complete, correct, up-to-date) within the productive systems, i.e. data quality aspects are neglected entirely. When applying the idea of an automated EA documentation (cf. [9]) to productive system environments, the actual data quality has a high influence on the outcome of such an endeavor.

In this paper, we analyze data quality aspects of a particular ESB system, namely SAP PI. Analyzed data quality aspects indicate whether or not those systems can be used for an automated EA documentation in practice. To support our recent research question, i.e. 'to which extent can an SAP PI system be used for an automated EA documentation?' (cf. [4]), we conducted a survey among 19 industry partners distributed on 4 continents. These results are a next step towards the practical application of an automated EA documentation.

The remainder of the article is structured as follows: Section 2 presents related work followed by Section 3 that reports results of an EA data quality assessment of ESB systems in productive environments. An interpretation of these results with respect to our research endeavor 'automated EA documentation' is given in Section 4. The paper concludes with an outlook in Section 5 and outlines further research directions.

2 Related Work

Existing EA frameworks covering inter alia *The Open Group Architecture Framework* (TOGAF) [20], *The Integrated Architecture Framework* (IAF) [22], and *Enterprise Architecture Planning* (EAP) [18] commonly do not detail how to acquire and incorporate EA knowledge. Only few approaches considering the documentation of the status quo could be identified. However, the descriptions usually take place on a high abstraction level without consideration of concrete process tasks. For instance, TOGAF suggests the usage of existing architecture definitions as a starting point, which if necessary, can be updated and verified. In case such information is unavailable the collection of data *"whatever format comes to hand"* [20] is advised by TOGAF.

Moser et al. [13] give a first idea for an automated tool-aided collection process by introducing a set of EA process patterns, one of which is called *Automatic Data Acquisition / Maintenance*. The authors propose a process aimed at automatically collecting data from various sources converted into an EA information model instance. However, the considerations do not detail possible information sources including data quality thereof.

Based on identified requirements on an automated documentation process, Farwick et al. [6] develop a basic structure of an automated maintenance process comprising the collection of data as well as the propagation of changes. However, when it comes to implementation details the authors refer to future work.

A more detailed look on an implementation is taken by Buschle et al. in [5], whereby NeXpose, a vulnerability scanner aimed at determining weaknesses within the system landscapes is used. Apart from weaknesses the scanner also collects information about the underlying systems landscape which subsequently is mapped to an EA information model. Hence, information on existing services, installed software, and used operating systems could be gathered. While the information coverage, i.e. the extent to which the demanded EA information can be determined, is considered within the publication the usefulness of the collected data is not discussed leaving open questions – as to the correctness and completeness of the data, for instance. Instead of using a vulnerability scanner, [4] employ an ESB. While the degree of coverage to which data of a productive system can can be used for EA documentation is thoroughly analyzed data quality thereof is not evaluated.

To the authors best knowledge there is no existing research analyzing data quality aspects of ESB systems and in particular SAP PI systems in productive IT environments.

3 SAP PI Data Quality Assessment: Evaluation of the Survey Results

In line with Steen et al. [19] we observed the EAs to be an important starting point for analysis, design, and decision processes. For this to work, EA information must provide an *accurate*, *correct*, and *up-to-date* model of the real world [6]. Accordingly, aiming at developing an automated EA documentation process the suitability of an ESB system as an information source is not only determined by information content but also by the quality of the data saved in the system, e.g., Are the data up-to-date? Are the attribute values correct? and Are the data consistent?.

Subsection 3.1 gives an overview of SAP PI. In order to gain a deeper insight into the data quality of a SAP PI system in practice we conducted interviews with SAP PI experts and an online survey with EA practitioners. Overall, 19 EA practitioners participated in the subsequent online survey and could be identified as responsible for an SAP PI system. Thereby, questions about the data content of SAP PI were assessed in terms of selected quality dimensions (see Subsection 3.2). In Subsection 3.3 we present the survey results in greater detail.

3.1 Overview of the SAP PI data

SAP PI is not a single module but rather a conglomerate of various components, which are not independent but stand in relationship to each other (see Fig. 1). The *System Landscape Directory* represents a central provider of landscape information comprising information about installed and installable software as well as technical details about the underlying infrastructure. Designing, creating and maintaining the interactions between the applications takes place in the *Enterprise Service Builder* which includes amongst others interface descriptions, messages, and exchanged data types. In the *Integration Builder* configurations of communication relationships at run time map *Enterprise Service Builder* elements to the actual execution environment. In order to test and monitor an SAP PI installation the *Runtime Workbench* offers a central entry point putting in place various tools. Finally, the *Integration Server* is responsible for processing incoming messages from sending applications, applying routing and mapping rules, and finally forwarding them to target systems.

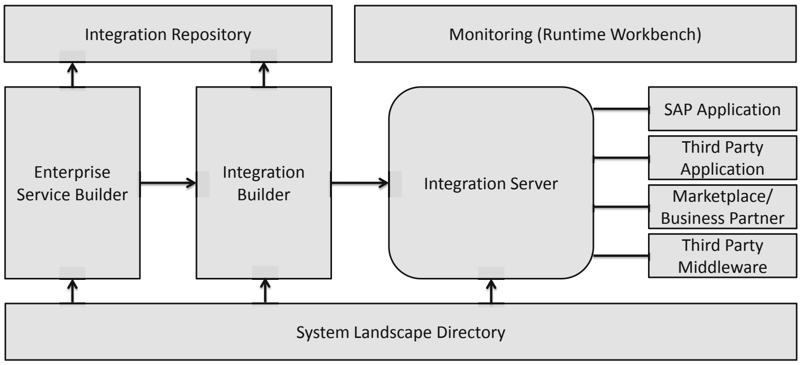


Fig. 1 Architecture of SAP NetWeaver Process Integration [14]

3.2 Data Quality Dimensions

An examination of literature reveals a high variety of quality definitions [11, 1, 8, 12]. For instance, Bednar et al. [15] make a distinction between four views on quality: *quality as excellence*, *quality as value*, *quality as conformance to guidelines* and *quality as meeting or exceeding customer expectations*. The first two views may turn out to be problematic as the assessment of excellence involves a high degree of subjectivity and the determination of a value is highly influenced by a monetary perspective while neglecting further criteria [11]. In contrast, ISO 9001:2008 [2] specifies quality as the “totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs” [1]. Quality respectively means the fulfillment of required characteristics, also referred to as quality attributes.

To establish a link to the first definition both views *quality as conformance to guidelines* and *quality as meeting or exceeding expectations* can be dissembled into a set of quality attributes to be fulfilled, e.g., completeness, accuracy and correctness.

Depending on the research area different frameworks proposing various quality attributes have been developed in an attempt to assess quality, such as software quality ([8, 12]), data quality ([21]), information quality ([11]) and even Enterprise Architecture quality ([6]). Consequently, the first challenge to overcome is to reduce the broad range of existing quality attributes to the essential amount to gain reasonable coverage of the term quality. As the focus lies on the inserted data in SAP PI rather than on the way they are saved, quality aspects regarding the underlying data model as well as its usability, e.g., simplicity, relevance of the data, perception from the user's perspective, semantics, etc. are considered as given and neglected.

A comparison of the different taxonomies reveals many terms including completeness, correctness, and actuality are common to most of them indicating a consensus in research and industry. In addition, attempting among others to determine Enterprise Architecture quality dimensions by conducting a survey Farwick et al. [6] identifies completeness, correctness, and actuality to be ranked highest. Accordingly, to make qualitative statements about the generated EA information the analysis of the SAP PI data quality in terms of these quality dimensions provides a good starting point. A list of the quality dimensions chosen can be found in Table 1.

Table 1 Considered data quality dimensions [11, 1, 8, 12]

Quality Dimension	**Description**
Completeness	The extent to which the expected data are provided according to the SAP PI specification
Correctness	The degree to which the SAP PI data reflect the real world and fulfill the SAP PI guidelines
Actuality	The degree to which the data are up to date

3.3 Assessment of SAP PI

As no research concerned with the quality of ESB systems in general and SAP PI systems in particular could be identified, we conducted an online survey aimed at evaluating the quality of data contained in SAP PI systems in practice. The survey was opened within 45 days. 45 SAP PI experts started the survey, 19 fully completed, 24 partly completed it whereas 2 only answered the first two questions. The last two respondents are neglected in the subsequent analysis.

General information about the respondents: 50 percent of the respondents reside in Asia, 30 percent in Europe and the remainders are distributed equally over North and South America. Out of all respondents 67 percent claimed to work for a consulting company. This circumstance favours the results of the survey as the answers include the experience about the situation in more than one organization.

Perception of the overall quality: All respondents rate their data quality as nearly perfect (80 percent) or even perfect (20 percent) which is partly attributable to high correlation between quality and functioning of the system. Incorrect or out-of-date data in most cases would lead to an unwanted behavior of the system, e.g. malfunctions.

System Landscape Directory quality: According to the official documentation the System Landscape Directory is the primary source for system landscape information. In contrast, most respondents (over 81 percent) stated that all data types except the SAP technical systems as well as the SAP products are only considered in the SLD system insofar as they are of importance for the collaborative processes with some exceptions (see Fig. 2). The SAP products and technical systems are an exception particularly due to the automatic insertion and update of the components in an available SLD system. Apart from completeness in terms of elements stored within SAP PI another important aspect is the completeness of a specific element, i.e., to which extent corresponding attributes are preset with values in SAP PI. In all cases more than 74 percent of the respondents agreed to ‘elements are complete’ or ‘elements are complete with some exceptions’ (see Fig. 3). In the case of SAP products this value even achieves 100 percent. Nevertheless, third party systems, databases as well as third party software products are partly stated as commonly incomplete.

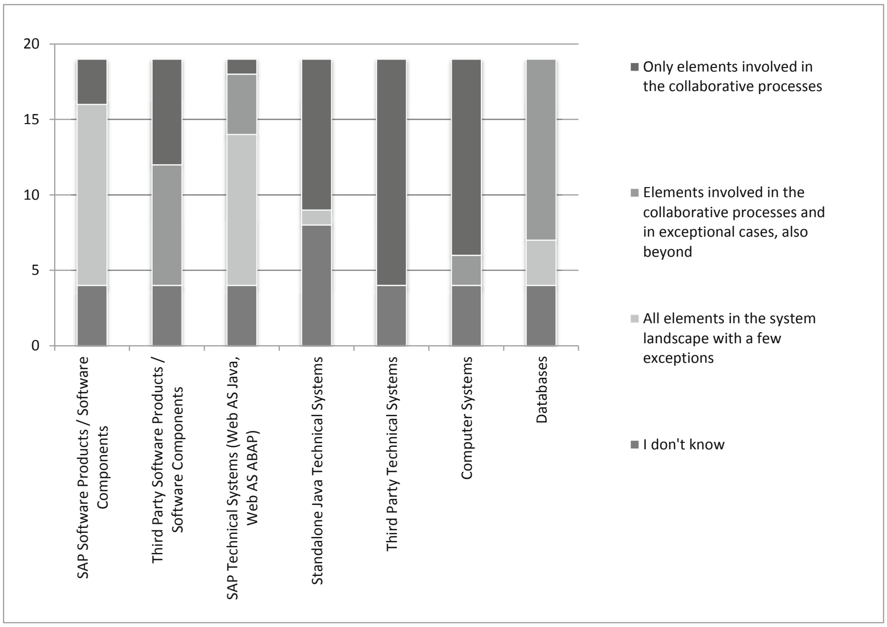


Fig. 2 Elements stored within SLD (n=19)

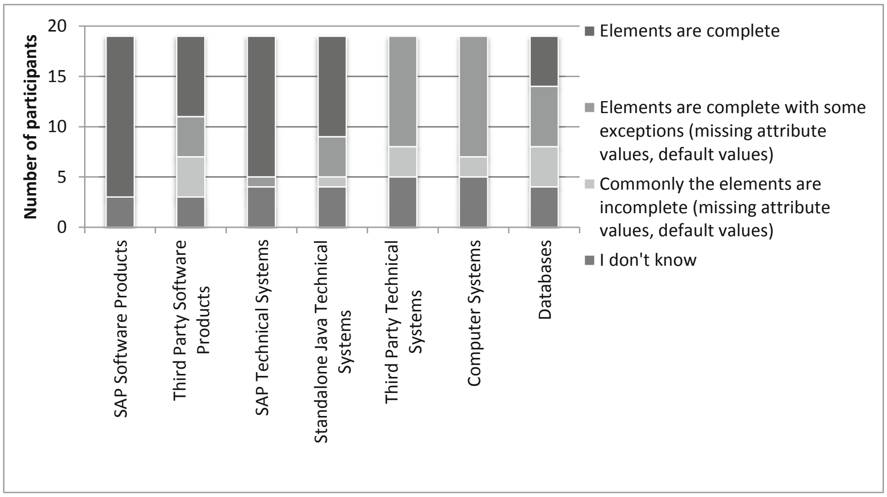


Fig. 3 Completeness of SLD data (n=19)

The assessment of the correctness (cf. Fig. 4) reveals that concerning all data types more than 78 percent agreed data to be either accurate or accurate with some exceptions. In the case of SAP products the proportion is 100 percent. This effect may be attributed to the automated insertion and update of data within the SAP product family.

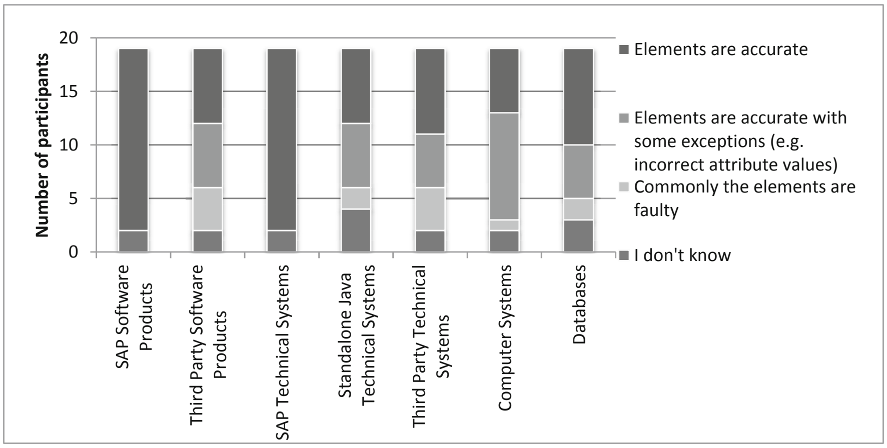


Fig. 4 Correctness of SLD data (n=19)

Focus during the analysis of the dimension *actuality* lies on two questions:

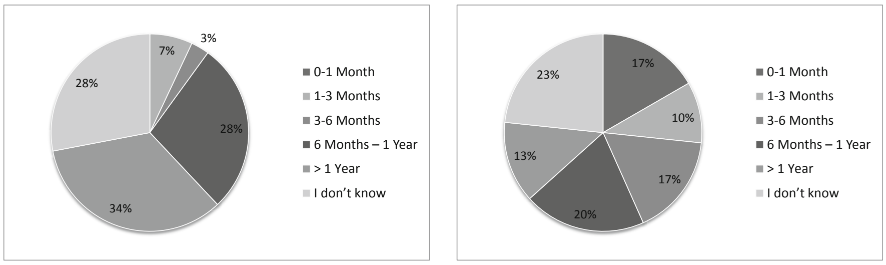
- When are changes to the system as well as application landscape taken into account?
- When are decaying data deleted?

Even though the SLD system is considered as the central information source of the system landscape, most respondents of the survey stated that all data types, except SAP technical systems and SAP products are only considered in the SLD system, with some exceptions, when they become important for the collaborative process.

Out-of-date data pose another problem. In particular, this includes elements which are still saved in the SLD system but not used in practice anymore yielding to a faulty reflection of the world. Out of the respondents 76 percent claimed that old data are deleted within one year and shorter and only 24 percent stated a time interval greater than one year. Within a survey conducted by Farwick et al. [6] the respondents reported that an actuality of EA information within weeks (48 percent) or up to six months (31 percent) is appropriate. While 76 percent claimed to delete out-of-data data within six months or even earlier, no participant stated an interval shorter than one month.

Enterprise Service Builder and Integration Builder quality: Interviews conducted in advance revealed the data of the Enterprise Service Builder as well as the Integration Builder to be nearly error-free and complete due to the mandatory characteristic and consequences to productive system environments in case of errors. Yet, it has to be highlighted that both components only comprise data necessary for the communication processes over the SAP PI system [4]. For instance, unused (legacy) interfaces are not taken into account. This holds true for other data types, e.g. software components, databases and computer systems. As a result, with respect to the quality dimensions (correctness, completeness, actuality), *actuality* was further investigated in our online questionnaire.

Analogous to the SLD system the time to delete is of relevance in order to assess the problem of out-of-date data and the resulting errors (see Fig. 5(a) and Fig. 5(b)). Unfortunately, with respect to the *Enterprise Service Repository* only 15 percent of the respondents reported an interval shorter than six months. In contrast to the



(a) Enterprise Service Builder (b) Integration Builder

Fig. 5 Survey results: Deletion interval (n=19)

average deletion time within the Enterprise Service Builder 56 percent reported a time interval shorter than 6 months for the Integration Builder. 21 percent even agreed to 0-1 month.

4 Effects on the EA Model Quality

Putting our findings in context towards developing and evaluating an automated EA documentation process previous data quality consideration gives a first impression to which extent the generated EA models are affected by the underlying data quality.

Completeness of EA Models: Previous investigations show the information content of EA models is limited to elements used within communication processes. Any information beyond is commonly abstracted even though the SLD officially is meant to be the central information source about the system landscape.

Correctness of EA Models: The previous analysis reveals data of SAP PI systems seem to be correct in most cases especially concerning data stored in the *Enterprise Service Repository* and *Integration Builder*. Accordingly, this also applies to the EA information derived from the data.

Actuality of EA Models: Orphaned data in the SAP PI system pose a problem. In particular, this includes elements which are still saved in the SAP PI system but not used in practice anymore. Consequently, extracted EA information paints a misleading picture not allowing to make appropriate decisions. In contrast to a manual collection in which orphanded data are probably filtered out by the individuals responsible such human filters are dropped within an automation. Losses in quality have to be offset elsewhere.

5 Conclusion and Outlook

This paper gives a first impression which impact data quality aspects of productive IT systems have on automated EA documentation endeavors. Survey results presented show the majority of data in productive SAP PI systems seems to be accurate (complete and correct). Combined with results of our recent study [4], an SAP PI system seems to be 1) suitable and 2) a reasonable starting point for an automated EA documentation endeavor. However, this statement cannot be generalized and further research of productive IT environments is necessary to show the real potential of an automated EA documentation.

Our current efforts are centered around a particular ESB system, namely SAP PI. Further research could integrate Configuration Management Databases (CMDBs) and infrastructure monitoring tools. Both could be very beneficial for impact analyses and strategic planning. Our long term hypothesis is 'when information is gathered from upper layers, e.g. business processes and capabilities, more unstructured information is to expect which would have an impact on 1) the data quality and 2) on the automation potential for EA documentation. With this in mind, a closer look at the field of process mining could be worthwhile.

References

1. A8402 ANSI/ASQC: Quality Management and Quality Assurance - Vocabulary. American Society for Quality Control (1994)
2. A8402 ANSI/ASQC: Quality Management and Quality Assurance - Vocabulary. American Society for Quality Control (1994)
3. Buckl, S., Matthes, F., Roth, S., Schulz, C., Schweda, C.M.: A conceptual framework for enterprise architecture design. In: Aalst, W., Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Proper, E., Lankhorst, M.M., Schnherr, M., Barjis, J., Overbeek, S. (eds.) Trends in Enterprise Architecture Research. LNBIP, vol. 70, pp. 44–56. Springer, Heidelberg (2010)
4. Buschle, M., Ekstedt, M., Grunow, S., Hauder, M., Roth, S., Matthes, F.: Automating enterprise architecture documentation using models of an enterprise service bus. In: Americas Conference on Information Systems, AMCIS (to appear, 2012)
5. Buschle, M., Holm, H., Sommestad, T., Ekstedt, M., Shahzad, K.: A Tool for Automatic Enterprise Architecture Modeling. In: Nurcan, S. (ed.) CAiSE Forum 2011. LNBIP, vol. 107, pp. 1–15. Springer, Heidelberg (2012)
6. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Requirements for automated enterprise architecture model maintenance - a requirements analysis based on a literature review and an exploratory survey. In: ICEIS, vol. (4), pp. 325–337 (2011)
7. Farwick, M., Agreiter, B., Ryll, S., Voges, K., Hanschke, I., Breu, R.: Automation Processes for Enterprise Architecture Management. In: Trends in Enterprise Architecture Research, TEAR, Helsinki (2011)
8. Glass, R.: Building Quality Software, 1st edn. Prentice Hall (1991)
9. Hauder, M., Roth, S., Matthes, F.: Challanges for automated enterprise architecture documentation. In: Trends in Enterprise Architecture Research, TEAR (2012) (in submission)
10. Hauder, M., Roth, S., Schulz, C.: Generating dynamic cross-organizational process visualizations through abstract view model pattern matching. In: Architecture Modeling for the Future Internet Enabled Enterprise, AMFinE (2012)
11. Kahn, B., Strong, D., Wang, R.: Information quality benchmarks: Product and service performance. Communications of the ACM, 184–192 (2002)
12. Kan, S.: Metrics and Models in Software Quality Engineering. Addison-Wesley Longman (2002)
13. Moser, C., Junginger, S., Brückmann, M., Schöne, K.M.: Some Process Patterns for Enterprise Architecture Management. In: Strategy, pp. 19–30 (2009), http://subs.emis.de/LNI/Proceedings/Proceedings150/8.pdf
14. Nicolescu, V., Funk, B., Niemeyer, P., Heiler, M., Wittges, H., Morandell, T., Visintin, F., Stegemann, B.K., Kienegger, H.: Praxishandbuch SAP NetWeaver PI - Entwicklung, 2nd edn. SAP Press, Bonn (2009)
15. Reeves, C.A., Bednar, D.A.: Defining quality: Alternatives and implications. Academy of Management Review 19, 419–445 (1994)
16. Ross, J.W., Weill, P., Robertson, D.C.: Enterprise Architecture as Strategy. Harvard Business School Press, Boston (2006)
17. Schaub, M., Matthes, F., Roth, S.: Towards a Conceptual Framework for Interactive Enterprise Architecture Management Visualizations. In: Modellierung (2012)

18. Spewak, S., Hill, S.C.: Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology, 2nd edn. John Wiley & Sons, London (1993)
19. Steen, M., Strating, P., Lankhorst, M., Doest, H., Eugenia, I.M.E.: Service-Oriented Enterprise Architecture. Idea Group Publishing (2005)
20. The Open Group: TOGAF Version 9 – A Manual, 9th edn. Van Haren Publishing (2009)
21. Wang, R.W., Strong, D.M.: Beyond accuracy: What data quality means to data consumers. Journal of Management Information Systems 12, 5–30 (1996)
22. van't Wout, J., Waage, M., Hartman, H., Stahlecker, M., Hofman, A.: The Integrated Architecture Framework Explained: Why, What, How, 1st edn. Springer, Berlin (2010)
23. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. 26(3), 276–292 (1987), http://portal.acm.org/citation.cfm?id=33596

Two-Stage Stochastic View Selection for Data-Analysis Queries

Rong Huang, Rada Chirkova, and Yahya Fathi

Abstract. We consider the problem of selecting an optimal set of views to answer a given collection of queries at the present time (stage 1) as well as several collections of queries in the future (stage 2), with a given probability of occurrence associated with each collection, so as to minimize the expected value of the corresponding query response time, while keeping the total size of the views within a given limit. We formulate this problem as a two-stage stochastic programming problem. We show that this model is equivalent to an integer programming (IP) model that can be solved via various commercial IP solvers. We also study the relationship between the queries and the views in this context and use this relationship to reduce the size of the corresponding IP model, hence increase the scalability of our proposed approach.

1 Introduction

Many data-intensive systems, such as commercial or scientific data warehouses, store vast collections of data, whose scale tends to grow massively over time. Answering typical data-analysis queries in such systems may involve heavy use of summarization of large volumes of stored data [9]. As a result, brute-force evaluation of such queries tends to be complex and time consuming. One way to reduce the evaluation costs of data-analysis queries is to precompute and store extra relations, called *materialized views*. Intuitively, a materialized view would improve the efficiency of evaluating a query when the view relation represents the result of

Rong Huang · Yahya Fathi
Operations Research Program, NC State University, Raleigh, NC 27695
e-mail: {rhuang,fathi}@ncsu.edu

Rada Chirkova
Computer Science Department, NC State University, Raleigh, NC 27695
e-mail: chirkova@csc.ncsu.edu

precomputation of some "subexpression" of the query of interest, see [18] and references therein. As such, materialized views with grouping and aggregation may be especially attractive for evaluating complex data-analysis queries, because the relations for such views store in compact form the results of (typically expensive) preprocessing of large amounts of data. Hence, answering a query by using an appropriate stored view can be much more effective (i.e., less time consuming) than answering the same query via the original data set.

In theory, in order to maximize the query-processing efficiency, every view that is potentially "beneficial" in the above sense could be materialized and used in this context. But in practice the amount of available storage space and other computational resources typically limit the number of beneficial views that we can actually materialize. The problem of choosing a set of beneficial views under such constraints and for a particular collection of high-priority queries is known as the "view-selection problem."

In recent years a number of research articles have addressed the view selection problem in various deterministic and probabilistic environments. Many articles, e.g., [13,2,10,11,21,20,4,3] address the problem in the context of one query workload and propose both exact and inexact methods for solving the problem in this context. We refer to this problem as the *One-Stage View-Selection Problem*. In a more recent article [12] it is assumed that in addition to the given (current) collection of queries, we also have prior knowledge of future query workloads. This, in turn, allows us to select and materialize an appropriate set of views that are not only beneficial for the current query workload but they are also potentially beneficial in the context of answering the future queries. Yet another line of research has studied the *dynamic view selection problem*, where the views are selected continuously in response to the changes in the query workload over time [7,8,17]. Significant work has also been done on *index* selection, e.g., [1,2,11,7,8,5].

In this paper we introduce and define this problem in a *two-stage probabilistic environment,* with *stage 1* representing the current time, and *stage 2* representing a point of time in the future. We refer to the problem in this environment as the *Two-Stage Stochastic View-Selection Problem*. It is easy to show that the *Two-Stage Stochastic View-Selection Problem* is NP-hard. We formulate this problem as a two-stage stochastic programming model and show that it is equivalent to an integer programming (IP) model, hence it can be solved via commercial IP solvers. We then propose several strategies to reduce the size of this model which, in turn, enhance the scalability of this approach.

2 Preliminaries

We consider a star-schema data warehouse [9] with a single fact table and several dimension tables. We assume that all views are defined, with grouping and aggregation but without selection, on the relation that is the result of the "star-schema join" [9] of all relations in the schema. We refer to this relation as the *raw-data view* throughout this article. We can show formally that for each query posed on such a database, the query can be rewritten equivalently into a query posed on the raw-data view. Using this formal result, in the remainder of the paper we assume that all the queries in the workloads that we consider are posed on the relevant raw-data view. In this context we consider the evaluation costs of answering unnested select-project-join queries with grouping and aggregation using unindexed materialized views, such that each query can be evaluated using just one view (including the raw-data view) and no other data. (This setting is the same as in [4, 13, 16, 19, 24].) A query q can be answered using a view v only if the set of grouping attributes of v is a superset of the set of attributes in the `GROUP BY` clause of q and of those attributes in the `WHERE` clause of q that are compared with constants.

We use v to represent both a view and the collection of grouping attributes for that view, and we use q to represent both a query and the collection of attributes in the `GROUP BY` clause of that query, plus those attributes in the `WHERE` clause of the query that are compared with constants. It follows that query q can be answered by view v if and only if $q \subseteq v$. To evaluate a query using a given view (if this view can indeed be used to answer the query) we have to scan all rows of the view. Hence the corresponding evaluation cost is equal to the size of the view itself; similar cost calculation is used in [4, 13, 16, 19, 24]. One way to estimate the view sizes is to use a "what-if optimizer" [2]. Another way, as suggested in the literature, is by getting a relatively small-size sample of the raw-data view and by then evaluating the view definitions on that table, with a subsequent scaleup of the sizes of the resulting relations. We use a_i to denote the size of each view v_i in the problem input. We also use the parameter d_{ij} to denote the evaluation cost of answering query q_j using view v_i. It follows that for each query q_j we have $d_{ij} = a_i$ if $q_j \subseteq v_i$, and we set $d_{ij} = +\infty$ otherwise, implying that q_j cannot be answered by view v_i in that case.

3 The Two-Stage Stochastic View-Selection Problem

3.1 Statement of the Problem

For the view-selection problem in a two-stage probabilistic environment, we have a query workload Q_1 that we must answer at the present time (stage 1), and a query workload $\mathbf{Q}_2$ that occurs at a future point in time (stage 2). We assume that the workload Q_1 is known and given, but that $\mathbf{Q}_2$ is a random set with a given probability distribution function. (We use boldface to emphasize that $\mathbf{Q}_2$ is a random set, and to differentiate it from a deterministic set such as Q_1.)

At stage 1, we materialize a collection of views S_1 and use these views to answer the queries in Q_1. We assume that we have a storage limit b_1 for these views: that is, the total size of the materialized views must not exceed b_1. At stage 2, once the actual queries Q_2 for this stage are known, we allow for a partial replacement of some of the view relations that we constructed at stage 1, in order to obtain the collection of views S_2 for answering the query set Q_2. (Note that Q_2 represents a realization of the random set $\mathbf{Q}_2$.) In other words, at the end of stage 1 we keep some of the view relations that we materialized at stage 1 to use again at stage 2, while we discard other view relations from stage 1 and replace them with new view relations. We assume that the total size of the new views that we materialize at stage 2 must not exceed a given storage limit b_2. Naturally $b_2 \leq b_1$.

In this context, the technical problem that we need to address prior to stage 1 is to select all views in the set S_1 and a replacement plan associated with each possible realization Q_2 of $\mathbf{Q}_2$. The objective is to minimize the total evaluation cost for the queries at stage 1 plus the expected total evaluation cost for the queries at stage 2. Note that in order to obtain a globally optimal solution for this problem, which we call the ***S***tochastic ***V***iew ***S***election *(SVS)* problem, all decisions regarding both stages (that is, the view set S_1 and the replacement plan for every possible realization Q_2 of $\mathbf{Q}_2$) must be made prior to stage 1.

To illustrate, we present the following numeric example for a data cube [13]. We will use this example later in the paper, to illustrate some of the technical results that we obtain in this context.

Example 1. Given a database with four attributes a, b, c and d, the corresponding *view lattice,* as defined in [13], is shown in Figure 1. In this lattice, each node represents a view, and a directed edge from node v_1 to node v_2 implies that v_1 is a *parent* of v_2, that is, v_2 can be obtained from v_1 by aggregating over one attribute of v_1. At stage 1, we are given queries $q_1 = \{a,b\}$ and $q_2 = \{b,c\}$. At stage 2, queries $q_3 = \{b\}$ and $q_4 = \{a,c\}$ would occur with probability 0.5, and queries $q_5 = \{c\}$ and $q_6 = \{b,d\}$ would also occur with probability 0.5. Equivalently, $Q_1 = \big\{\{a,b\},\{b,c\}\big\}$, $Q_2^1 = \big\{\{b\},\{a,c\}\big\}$, and $Q_2^2 = \big\{\{c\},\{b,d\}\big\}$. We assume the total space limit $b_1 = 30$, and the space limit $b_2 = 15$. Our objective is to minimize the cost of answering the first-stage queries Q_1 plus the expected cost of answering the second-stage queries. Thus, we are to determine a set of views S_1 to materialize at stage 1, with the total size less than or equal to 30, and for each scenario at stage 2, we need to determine another set of views, with the total size not to exceed 15, to materialize as replacement for a subset of S_1.

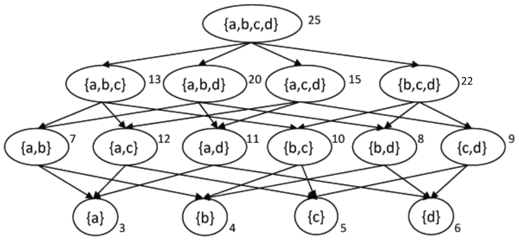


Fig. 1 View lattice for Example 1, with view sizes shown as number of bytes

3.2 A Mathematical Programming Model

We now propose a stochastic programming model for the problem SVS. The general form of our model is valid for any probability distribution function for $\mathbf{Q}_2$. However, for ease of exposition throughout the rest of this paper, we assume that the random query set $\mathbf{Q}_2$ has a discrete distribution with L possible values. More specifically, we assume that $\mathbf{Q}_2$ equals to query set Q_2^ℓ with probability p_ℓ, for $\ell = 1$ to L, where $\sum_{\ell=1}^{L} p_\ell = 1$. We refer to each set of queries Q_2^ℓ as a *scenario*. For the first stage we define the following decision variables for all views $v_i \in V$ (where V is the set of all views) and for all queries $q_j \in Q_1$.

$$x_i = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 1} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \text{ at stage 1} \\ 0 & \text{otherwise} \end{cases}$$

For the second stage we define the following decision variables for all $v_i \in V$, and for all $q_j \in Q_2^\ell$, for $\ell = 1$ to L.

$$u_i^\ell = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 1 and used at stage 2 for query set } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

$$y_i^\ell = \begin{cases} 1 & \text{if view } v_i \text{ is materialized at stage 2 for } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

$$t_{ij}^\ell = \begin{cases} 1 & \text{if we use view } v_i \text{ to answer query } q_j \text{ at stage 2 for } Q_2^\ell \\ 0 & \text{otherwise} \end{cases}$$

The cardinality of the view set V is 2^K, where K is the number of dimension attributes in the database. Let $I = \{1, 2, \ldots, 2^K\}$ be the set of subscripts for all $v_i \in V$; J_1 be the set of subscripts for all $q_j \in Q_1$; and J_2^ℓ be the set of subscripts for all $q_j \in Q_2^\ell$, for $\ell = 1, 2, \ldots, L$. The problem can now be written as the following stochastic programming model [6] that we denote by SP.

$$\begin{aligned} (SP) \quad \text{minimize} \quad & \sum_{j \in J_1} \sum_{i \in I} d_{ij} z_{ij} + \mathbf{E}_{\mathbf{Q_2}} \Psi(\mathbf{x}, \mathbf{Q}_2) & (1) \\ \text{subject to} \quad & \sum_{i \in I} z_{ij} = 1 \quad \forall j \in J_1 & (2) \\ & z_{ij} \leq x_i \quad \forall i \in I \; \forall j \in J_1 & (3) \\ & \sum_{i \in I} a_i x_i \leq b_1 & (4) \\ & \textit{All variables are binary} & \end{aligned}$$

where $\mathbf{E}_{\mathbf{Q_2}}$ denotes the mathematical expectation of response time at stage 2 with respect to $\mathbf{Q}_2$. If the probability distribution of $\mathbf{Q}_2$ is as above, we have $\mathbf{E}_{\mathbf{Q}_2} \Psi(\mathbf{x}, \mathbf{Q_2}) = \sum_{\ell=1}^{L} p_\ell \Psi(\mathbf{x}, Q_2^\ell)$ where $\Psi(\mathbf{x}, Q_2)$ is the minimum response time for a given set of values of the first-stage variables $\mathbf{x} = (x_1, \cdots, x_{|V|})$ and for a realization of the second-stage queries Q_2. For each value of ℓ, the corresponding value of $\Psi(\mathbf{x}, Q_2^\ell)$ is obtained by solving the following view-selection problem.

$$\Psi(\mathbf{x}, Q_2^\ell) = \min \sum_{j \in J_2^\ell} \sum_{i \in I} d_{ij} t_{ij}^\ell \tag{5}$$

$$\text{subject to} \quad \sum_{i \in I} t_{ij}^\ell = 1 \qquad \forall j \in J_2^\ell \tag{6}$$

$$t_{ij}^\ell \le u_i^\ell + y_i^\ell \quad \forall i \in I \;\; \forall j \in J_2^\ell \tag{7}$$

$$u_i^\ell \le x_i \qquad \forall i \in I \tag{8}$$

$$\sum_{i \in I} a_i y_i^\ell \le b_2 \tag{9}$$

$$\sum_{i \in I} a_i (u_i^\ell + y_i^\ell) \le b_1 \tag{10}$$

$$\textit{All variables are binary} \tag{11}$$

In this model constraints (2) and (6) state that each query is answered by exactly one view in the set of materialized views (including the raw-data view that we denote by v_1). Constraints (3) and (7) guarantee that a query can be answered by a view only if the view is already materialized. Constraints (4), (9), and (10) pertain to the given storage limits. Constraint (8) guarantees that the view kept from stage 1 to stage 2 is already materialized at stage 1.

In practice, for each decision variable with a subscript i (associated with view v_i), we only need to consider those views v_i that are relevant in the context of that decision variable. For instance, at stage 1 we do not need to define the decision variable x_i if the corresponding view v_i is not a superset of at least one query in our entire query set. This results in a smaller stochastic programming model that we refer to as SP'.

4 Solving the Problem

In order to solve the problem SVS, we write the stochastic programming model SP' in its *extensive form* [6]. We do so by explicitly substituting Equations (5) through (11) for each value of ℓ in the expression for $\mathbf{E}_{\mathbf{Q}_2}\Psi(\mathbf{x}, \mathbf{Q_2})$ and ultimately in Equation (1) of model SP'. The resulting model is an integer programming (IP) model [23]. Thus, at least in theory, it can be solved using existing IP algorithms such as the branch and bound method. The only difficulty in practice, however, is the relatively large number of variables and constraints in this model, which makes this approach computationally cumbersome. Fortunately, the structural properties of this IP model allow us to reduce its size (i.e., the number of variables and constraints) significantly, hence allowing us to solve relatively large instances of the problem SVS within reasonable execution times using commercial IP solvers such as CPLEX [15]. We summarize these properties in the following observations, which we present here without proof. All the proofs are presented in [14]. In these observations and in the remainder of the paper we use the notation $Q(v)$ to represent a subset of a given set of queries Q consisting of every query in Q that the view v can answer, that is, $Q(v) = \{q \in Q : q \subseteq v\}$. We also use the notation $\widehat{Q}$ to represent the collection of all queries (either in stage 1 or in stage 2) in a given instance of problem SVS, that is, $\widehat{Q} = Q_1 \cup \left\{\cup_{\ell=1}^{L} Q_2^{\ell}\right\}$.

Observation 1. *In an instance of problem SVS, given a view v in the view set* V_1*, if the number of attributes of v is strictly greater than the number of attributes in the union set of the queries in* $\widehat{Q}$ *that v could answer, that is, if* $|v| > |\cup_{q\in\widehat{Q}(v)} q|$*, then there exists an optimal solution such that v is not materialized at stage one.*

For each view v and each query set Q we define the corresponding *benefit* as $d(v,Q) = \sum_{q\in Q(v)} S(q) - S(v)$ where $S(\cdot)$ is the estimated size of the query or view.

Observation 2. *In an instance of problem SVS, given a view v in the view set* V_1*, if v is not equal to a query in* $\widehat{Q}$ *and v satisfies the condition that the size of v is greater than or equal to the total size of the queries in* $\widehat{Q}$ *that v can answer, that is, if* $d(v,\widehat{Q}) < 0$*, then there exists an optimal solution in which v is not materialized at stage one.*

Observation 3. *In an instance of problem SVS, given a view v in* V_1*, if there exists a view* v' *in* V_1 *such that* $v' \subset v$ *and* $d(v',\widehat{Q}) \geq d(v,\widehat{Q})$*, then there exists an optimal solution such that v is not materialized at stage one.*

We make similar observations with respect to each view v in V_2^{ℓ}. These observations allow us to reduce the search spaces of views for V_1 and V_2^{ℓ}, for $\ell = 1$ to L. We illustrate these observations by applying them to the numeric example that we stated earlier.

Example 1 (Continued). Consider view $v_1 = \{c,d\}$ in the set V_1: The view answers only query $q_5 = \{c\}$ in $\widehat{Q}$, yet v_1 has more attributes than q_5. Hence, by Observation 1, there exists an optimal solution that does not contain the view v_1. Consider now view $v_2 = \{a,b,d\}$ in the set V_1; the queries that v_2 can answer are $q_1 = \{a,b\}$, $q_3 = \{b\}$, and $q_6 = \{b,d\}$. Since the total size of these three queries $(4+7+8 = 19)$ is smaller than the size (20) of v_2, Observation 2 implies that there exists an optimal solution that does not contain v_2. Finally, consider view $v_3 = \{b,c,d\}$ in the set V_1; view v_3 is a superset of view $v_4 = \{b,d\}$. We have that $d(v_4,\widehat{Q})$ $(4+5+10-10=9)$ is greater than $d(v_3,\widehat{Q})$ $(4+5+8+10-22 = 5)$. Thus, by Observation 3, there exists an optimal solution that does not contain the view v_3.

We now define the model $IP2$ as an integer programming model that is the same as the model $IP1$, except that we use the reduced search spaces of views in place of the corresponding original search spaces of views in $IP1$. Obviously, the size of model $IP2$ is potentially smaller than the size of the corresponding model $IP1$.

In order to compare the sizes of the models $IP2$ and $IP1$ on an empirical basis, and to evaluate the scalability of our proposed approach, we constructed a collection of randomly generated instances of the problem with varying sizes, using a number of datasets generated via the TPC-H benchmark [22]. We then solved each instance using the models $IP1$ and $IP2$. All of our algorithms are implemented in C++; all the experiments were carried out on a 2.66GHz Intel 2 Quad processor with 3.25 GB RAM running Windows XP Professional. We used CPLEX 11 [15] to solve the integer programming models $IP1$ and $IP2$. We observed that the search spaces of views are significantly reduced in $IP2$ compared with $IP1$, and that we were able to use model $IP2$ to solve relatively large (realistic-size) instances of the problem SVS. For example, for an instance of the problem on the 13-attribute TPC-H dataset, with $L = 2$, $(|Q_1|, |Q_2^1|, |Q_2^2| = (50, 52, 56))$, the corresponding execution time for constructing and solving model $IP2$ was 190 seconds. When we increased the number of queries to $(140, 140, 140)$, the corresponding execution time increased to 451 seconds. For larger instances of the problem, however, the execution time exceeded the 20-minute time limit that we imposed on each instance in this experiment. Details of the numeric results are presented in [14].

5 Concluding Remarks

In this paper we introduced a novel two-stage stochastic view-selection problem and undertook a systematic study of the problem. We introduced a stochastic programming model for the problem, and showed that it is equivalent to an integer programming (IP) model. We studied the structure of this model, and proposed a procedure to efficiently prune the search space of views, hence reducing the size of the IP model. Our computational experiments show that in many instances the reduction in the size of the IP model is significant, and that the resulting IP model can be solved by commercial IP solvers for small to medium realistic-size instances of the problem, within reasonable execution time.

In order to evaluate the effectiveness of our proposed stochastic programming model we compared this model with a simpler deterministic model that we refer to as the *Expected Value Model*, as well as with a model that assumes we have advanced knowledge of the specific query workload that will occur in the second stage. The former model allows us to calculate the *Value of Stochastic Solution*, and the latter model allows us to calculate the *Value of Perfect Information* in the context of the corresponding two-stage problem. Our preliminary computational results show that the proposed stochastic programming model offers distinct advantages over the simpler models in term of the quality of solutions obtained, although its computational requirements are significantly higher. Presently, in order to improve the efficiency and scalability of our approach, we are further investigating the structure of the IP model so as to design more effective exact and inexact methods for solving larger instances of the problem. This, in turn, would allow for a wider potential applicability of our proposed approach.

References

1. Agrawal, S., Bruno, N., Chaudhuri, S., Narasayya, V.R.: AutoAdmin: Self-tuning database systems technology. IEEE Data Eng. Bull. 29(3), 7–15 (2006)
2. Agrawal, S., Chaudhuri, S., Narasayya, V.R.: Automated selection of materialized views and indexes in SQL databases. In: VLDB, pp. 496–505 (2000)
3. Asgharzadeh, Z.T.: Exact and inexact methods for solving the view and index selection problem for OLAP performance improvement. Phd dissertation, North Carolina State University (2010)
4. Asgharzadeh, Z.T., Chirkova, R., Fathi, Y.: Exact and inexact methods for solving the problem of view selection for aggregate queries. International Journal of Business Intelligence and Data Mining 4(3/4), 391–415 (2009)
5. Asgharzadeh, Z.T., Chirkova, R., Fathi, Y., Stallmann, M.: Exact and inexact methods for selecting views and indexes for OLAP performance improvement. In: EDBT, pp. 311–322 (2008)
6. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer (1997)
7. Bruno, N., Chaudhuri, S.: Interactive physical design tuning. In: ICDE, pp. 1161–1164 (2010)
8. Bruno, N., Chaudhuri, S., Weikum, G.: Database tuning using online algorithms. In: Encyclopedia of Database Systems, pp. 741–744. Springer US (2009)
9. Chaudhuri, S., Dayal, U., Narasayya, V.R.: An overview of business intelligence technology. Communications of the ACM 54(8), 88–98 (2011)
10. Chaudhuri, S., Narasayya, V.R., Weikum, G.: Database tuning using combinatorial search. In: Encyclopedia of Database Systems, pp. 738–741. Springer US (2009)
11. Chaudhuri, S., Weikum, G.: Self-management technology in databases. In: Encyclopedia of Database Systems, pp. 2550–2555. Springer US (2009)
12. Duan, S., Franklin, P., Thummala, V., Zhao, D., Babu, S.: Shaman: A self-healing database system. In: ICDE, pp. 1539–1542 (2009)
13. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: SIGMOD, pp. 205–216 (1996)
14. Huang, R., Chirkova, R., Fathi, Y.: Two-stage stochastic view selection for data analysis. Tech. Rep. TR-2011-22, NC State University (2011), ftp://ftp.ncsu.edu/pub/unity/lockers/ftp/csc_anon/tech/2011/TR-2011-22.pdf
15. ILOG: CPLEX 11.0 software package (2007), http://www.ilog.com/products/cplex/
16. Kalnis, P., Mamoulis, N., Papadias, D.: View selection using randomized search. DKE 42, 89–111 (2002)
17. Kotidis, Y., Roussopoulos, N.: A case for dynamic view management. ACM TODS 26(4), 388–423 (2001)
18. Lightstone, S.: Physical database design for relational databases. In: Encyclopedia of Database Systems, pp. 2108–2114. Springer US (2009)
19. Shukla, A., Deshpande, P., Naughton, J.F.: Materialized view selection for multidimensional datasets. In: VLDB, pp. 488–499 (1998)
20. Theodoratos, D., Ligoudistianos, S., Sellis, T.K.: View selection for designing the global data warehouse. Data Knowledge and Engineering 39(3), 219–240 (2001)
21. Theodoratos, D., Sellis, T.K.: Incremental design of a data warehouse. Journal of Intelligent Information Systems 15(1), 7–27 (2000)
22. TPC-H Revision 2.1.0: TPC Benchmark H (Decision Support), http://www.tpc.org/tpch/spec/tpch2.1.0.pdf
23. Wolsey, L.A.: Integer Programming. Wiley (1998)
24. Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In: VLDB, pp. 136–145 (1997)

Choosing Values for Text Fields in Web Forms

Gustavo Zanini Kantorski, Tiago Guimaraes Moraes,
Viviane Pereira Moreira, and Carlos Alberto Heuser

Abstract. Since the only way to gain access to Hidden Web data is through form submission, one of the challenges is how to fill Web forms automatically. In this paper, we propose algorithms which address this challenge. We describe an efficient method to select good values for text fields and a technique which minimizes the number of form submissions and simultaneously maximizes the number of rows retrieved from the underlying database. Experiments using real Web forms show the advantages of our proposed approaches.

Keywords: Crawling, Deep Web, Filling Forms, Hidden Web.

1 Introduction

The surface Web is the portion of the World Wide Web that can be reached by direct link navigation. However, a vast portion of the information on the Web is available in online databases and can only be reached through Web form filling and submission. This portion of the Web is known as *Hidden Web* [16] or *Deep Web* [4] and it is not indexed by conventional search engines. The Hidden Web is more diversely distributed, has a large number of structured databases, and suffers an inherent limitation of crawling [5].

In this context, one of the challenges is how to automatically fill forms in order to gain access to the data. This task is not trivial, since forms were designed to be used by human beings. Most architectures aim at retrieving as much information as possible from the online database behind the form. In order to do that, the simplest

Gustavo Zanini Kantorski · Tiago Guimaraes Moraes · Viviane Pereira Moreira ·
Carlos Alberto Heuser
UFRGS, Porto Alegre, Brazil
e-mail: {gzkantorski,tgmoraes,viviane,heuser}@inf.ufrgs.br

solution would be submitting the combination of all possible field values in a cartesian product. However, this solution is not feasible when the number of fields and possible values are large. For example, a Web form composed of five fields with thirty possible values each, has over 24 million filling combinations. Some combinations are wasteful, *i.e.,* they either return identical results (not adding new data) or fail to retrieve any data.

In this paper, we propose an automatic method for filling forms. Our method explores two strategies. The first is how to select good values, or *queries*, to submit to a particular form in order to retrieve more data with fewer submissions. The second strategy is how to fill the fields efficiently, specially the text fields, which do not have a set of pre-determined values. The strategy to minimize the set of queries, *i.e.,* the number of form submissions, involves pruning the set of all possible queries. As each query is submitted, data extracted from the resulting page is used to identify wasteful queries and prune them. For filling form fields, this paper presents a solution called FTF (*Filling Text Fields*) which focuses on fields which do not have a set of pre-determined values, such as text boxes. Most of the existing work on form filling [1, 6, 8, 10, 11, 13] overlooks the problem of finding good values for text fields. Existing solutions for dealing with text fields usually rely on a list of values previously built by a specialist [10], on a sample of known values [3], or they entail the extraction and understanding of the fields [1, 7, 11] (which depends on the language and on the domain of the forms). Our proposed solution is automatic, requires no training, and relies on feedback from previous submissions. Domain often influences value selection for the fields. Although our method does not use domain knowledge explicitly, our experiments show that the values generated for the fields are domain-related. The results show that in most cases, our approach achieves superior coverage compared to a baseline method.

2 Definitions

Data available on the Hidden Web are accessible through Web forms (usually HTML forms). A form contains fields to be filled so that a form submission may retrieve data, accessing the online database hidden behind the form. Fields can be text boxes, selection lists, check boxes, radio buttons, and submit buttons. They can be classified in two groups: (i) fields with finite domains (such as selection lists) and fields with infinite domains (such as text boxes, in which a user can type any value). Once the values have been filled, the form can be submitted. A form can be submitted by two methods, *get* or *post*. Our strategies work with Web forms of both methods. In the context of this paper, it is important to distinguish between form domain and attribute domain. *Form domain* is the subject or topic to which the form belongs (*e.g.,* books, airfares, hotel, jobs, etc.). For a study on the domains found on the Hidden Web, please refer to Chang *et al.* [15]. The *attribute domain* is the set of values for a field. For example, the domain of a selection list field is presented inside the *option* HTML tag.

Madhavan *et al.* [3] and Cafarella *et al.* [14] divide text boxes into two groups: *(i) generic text boxes* which represent text boxes in which the words are used to retrieve all documents in the backend database containing those words; and *(ii) typed text boxes* which serve as a selection predicate on a specific attribute in the where clause of a SQL query over the backend database. For a set of submissions, we define the concepts of *template* (similar to Madhavan *et al.*'s [3] notion of query template) and *instance template*. Templates are represented by form field combinations. For example, for a form with four fields *A, B, C*, and *D*, we have the following templates: *A, B, C, D, A&B, A&C, A&D, B&C, B&D, C&D, A&B&C, A&B&D, A&C&D, B&C&D and A&B&C&D*. If there are *n* fields in a form, there are $2^n - 1$ templates. The number of fields that make up a template will be referred to as the *dimension of the template*. For example, the dimension of the template *A&B&C&D* is four and the dimension of the template *A&B* is two. An instance template associates a value to each field considered in the form submission. For example, an instance template for the template *A&B&C* could be *A=a1&B=b1&C=c1* and it could be represented by the pair (A&B&C,a1&b1&c1). In practice, instance templates are represented by URLs. More formally we have:

Definition 1. (Template) Let $F=\{f_1, f_2, ..., f_n\}$ be a set of fields located in a Web form and let $T=\{t_1, t_2, ..., t_m\}$ be a set of all combinations of the elements in F. Each element in T is defined as a template.

Definition 2. (Instance Template) Let *t* be a template in T and let $V=\{v_1, v_2, ..., v_m\}$ be the domain of *t*. An instance template is an attribute-value pair (*t*,*v*) where *v* is an element from V.

3 Related Work

The literature has several solutions for automatic form filling. Raghavan *et al.* [10] manage Web form filling through tables called *Label Value Set* (LVS). LVS tables are associated to form fields. The main issue in this method is filling the LVS table with the desired values for queries and the association between values and fields.

In Liddle *et al.* [8], automatic form filling is carried out by assigning default values to form fields. Text fields are ignored and, if they are mandatory, user intervention is needed.

Barbosa *et al.* [6] present an approach for filling forms based on keywords. The discovery of words is based on the data coming from the database itself, instead of random word generation. On the other hand, they do not handle Web forms that do not contain keyword fields.

Wu *et al* [12] present a form filling technique based on a feedback process of the values filled in the form. A limitation is that, in the query, just one form field can be used at a time–combining form fields for several queries is not possible.

Jian *et al.* [1] present a method in which each keyword obtained from the result pages is encoded as a tuple representing its linguistic, statistic, and HTML features.

These tuples (issued keywords) are used to train a model using machine learning algorithms which will be used to evaluate the harvest rates for un-issued keywords.

Toda *et al* [11] describe a form filling solution based on value extraction from a text document. This solution exploits features related to the content and to the style of values, which are combined through a Bayesian framework. The approach relies on the knowledge obtained from the values of previous submissions for each field.

The work by Madhavan *et al.* [3] describes a system for surfacing the content of the Hidden Web. They improved the keyword selection algorithm by ranking keywords with respect to their TF-IDF.

Existing methods for filling form fields [1, 3, 6, 8, 10, 11, 13] focus on selecting values for fields with finite domains and for fields with infinite domains classified as *keyword text boxes*. A gap is still open in the selection of good values for fields with infinite domains classified as *typed text boxes* (See Section 2). Proposed solutions usually employ a set of pre-defined values defined by a specialist [10], require the extraction and understanding of field labels [1, 7, 11], use the generic frequency distribution of each keyword [3, 6, 8, 13] or use a sampling of the known values [3]. Our main contribution is a totally automatic approach for finding values for typed text boxes. To the best of our knowledge there are no automatic solutions for filling this type of field.

4 Automatic Filling of Web Forms

Figure 1 shows the proposed architecture. Our main contributions are in the highlighted modules *Value Selection* and *Instance Template Generation and Submission*. The *Candidate template generation* module represents the HTML form processing to generate all possible templates. The *Value Selection* module finds the values to fill each field in the form. Here the main problem is how to choose values for fields with infinite domains. Details are discussed in Section 4.2.

The *instance template generation and submission* module attaches input values to each field of each template, creating its set of instance templates. More details are

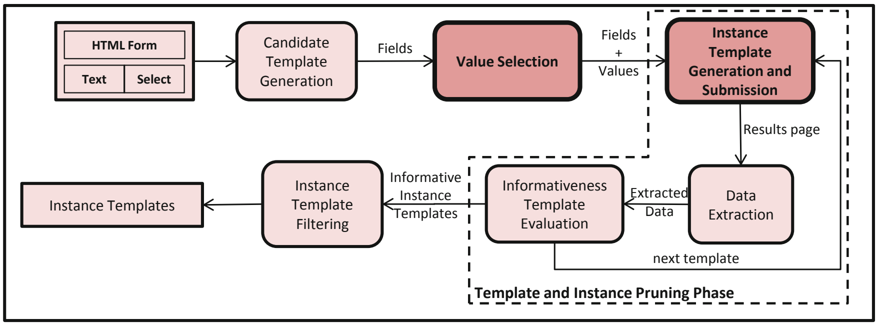

HTML Form
Text
Select
Candidate Template Generation
Fields
Value Selection
Fields + Values
Instance Template Generation and Submission
Results page
Data Extraction
Extracted Data
Informativeness Template Evaluation
next template
Informative Instance Templates
Instance Template Filtering
Instance Templates
Template and Instance Pruning Phase

Fig. 1 Proposed Architecture

given in Section 4.1. The *Data extraction* module extracts the information from the pages resulting from form submissions. This extraction is needed to find where the data region is located in the result page. Information about ads and presentation from the result page are discarded. The extracted data is used to evaluate each template. This data is also used to populate a database, which is serves as basis for the generation of values for fields with infinite domains (*Value Selection* module). The intuition is that higher coverage may be achieved if instead of using randomly selected data, values resulting from previous submissions are used. The *informativeness evaluation* module checks if the template is informative after the submission of the selected instance templates. A template is *informative* if its instance templates retrieve sufficiently distinct data. If a template t is considered non-informative, higher order templates including t will be discarded. At the end of the process, after processing all templates, the *Filtering Instance Templates* module determines the minimum number of instance templates needed to retrieve all distinct rows extracted from the pages resulting from form submissions.

4.1 Instance Template Generation and Submission

The main idea here is that, for each new submission, information from previous submissions is considered in order to avoid wasteful submissions which will incur in blank pages, error pages, or pages with duplicate rows which do not add new information to the existing set. The Instance Template Pruning (ITP) method is used to prune the wasteful instances of a template. In order to identify such wasteful instances, we employ the concept of informativeness of an instance template. Each instance template submitted is evaluated as informative or non-informative. All non-informative instance templates are added to a *pruning list* and are discarded in higher-order templates. This process is shown in Figure 2. The pruning list is initially empty and starts being filled as non-informative instances are discovered.

For each template, a set of possible instance templates are generated, taking into account the pruning list. Instances are submitted one by one and, for each of them,

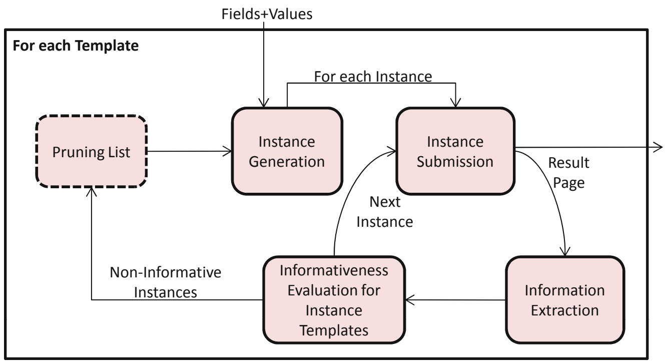


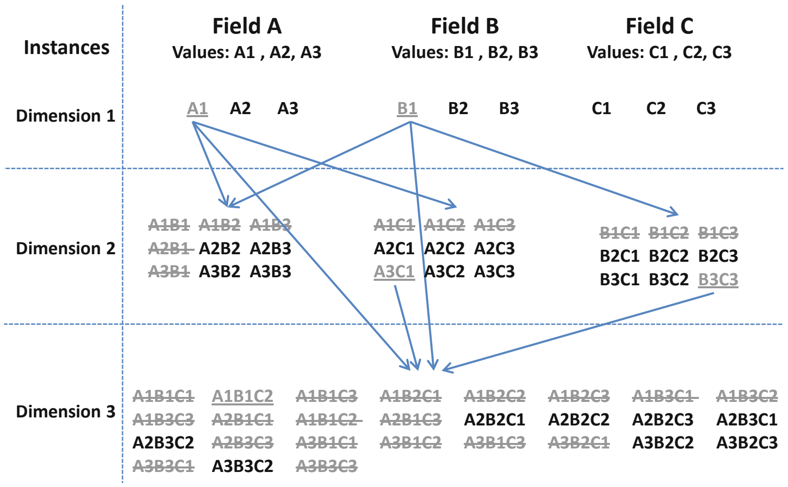
Fig. 2 Instance Template Generation and Submission module

the information from the response page is extracted. The extracted data feeds the informativeness evaluation module. This process is repeated for all generated templates and avoids unnecessary submissions in templates of higher order. Note that an instance template considered non-informative implies the pruning of other instance templates in templates of higher order. The method usually generates templates with order less than three because most templates of order greater than three do not return distinct data and thus are discarded [3].

Figure 3 shows an example of instance template pruning. Three fields (A, B, C) are considered, with three possible filling values each. The underlined values were submitted and considered non-informative. The strikethrough values were pruned and not even submitted. The other values were submitted and considered informative. The arrows show which values were considered in the pruning list for the creation of the instances for each template. For example, the instance of the template A, A = A1 was considered non-informative. Thus, other templates that take the filling of field A (templates AB, AC and ABC) into account had instances with value A1 removed from the group of possible instances for submission. This reduces the number of instances to be tested. In a naive method that considers the cartesian product of possibilities, 63 instance templates would be submitted. In the ITP method, that number is reduced to 33 since with the information that there are four non-informative instances, the system has learned that there are another 30 instances which would not return new data and thus do not need to be submitted.

4.2 *Value Selection*

The goal of selecting good values for text boxes is to discover the values to be used in queries in order to retrieve as many rows hidden behind the form, at an acceptable



Instances
Field A
Values: A1 , A2, A3
Field B
Values: B1 , B2, B3
Field C
Values: C1 , C2, C3
Dimension 1
A1 A2 A3
B1 B2 B3
C1 C2 C3
Dimension 2
~~A1B1~~ ~~A1B2~~ ~~A1B3~~
~~A2B1~~ A2B2 A2B3
~~A3B1~~ A3B2 A3B3
~~A1C1~~ ~~A1C2~~ ~~A1C3~~
A2C1 A2C2 A2C3
A3C1 A3C2 A3C3
~~B1C1~~ ~~B1C2~~ ~~B1C3~~
B2C1 B2C2 B2C3
B3C1 B3C2 B3C3
Dimension 3
~~A1B1C1~~ A1B1C2 ~~A1B1C3~~ ~~A1B2C1~~ ~~A1B2C2~~ ~~A1B2C3~~ ~~A1B3C1~~ ~~A1B3C2~~
~~A1B3C3~~ ~~A2B1C1~~ ~~A1B1C2~~ ~~A2B1C3~~ A2B2C1 A2B2C2 A2B2C3 A2B3C1
A2B3C2 ~~A2B3C3~~ ~~A3B1C1~~ ~~A3B1C2~~ ~~A3B1C3~~ ~~A3B2C1~~ A3B2C2 A2B3C3
~~A3B3C1~~ A3B3C2 ~~A3B3C3~~

Fig. 3 An example for ITP

cost. We present the *Filling Text Fields* (FTF) method which generates values for text boxes. FTF is based on a feedback loop, in which each element has an effect on the next one, until the last element produces feedback on the first element. The idea is to use information from the form itself plus the data retrieved from previous submissions as input to future submissions.

The generation of values for fields with finite domains is fairly easy as the values which may be selected are found inside the Web form code. However, the same does not happen for fields with infinite domains. It is not possible to know beforehand the quantity and the quality of the desired values. By quantity we mean the number of selected values which will lead to the desired coverage and by quality we mean the choice of values that retrieve more distinct data. These characteristics turn the task of finding good values for text boxes into a challenging problem. One way to find initial values could be to design a list of words associated with the text box. However, this is not feasible since there are Web forms in multiple domains that may have similar fields whose values present a wide variation.

In the FTF method, the selection of values for fields with infinite domains is divided into two steps. In the first step, a score r_1 is calculates with the aim of selecting meaningful tokens from previous submissions. The second step catches tokens from the first step to compose new queries and a new score r_2 is generated. Pages resulting from each submission are stored in a database. We calculate the score r_{1t} (Equations 1 and 2) for each token, and take the k-highest scoring tokens to use in future submissions. The intuition here is that, by associating cf, more records are retrieved. The *idf* component is used to remove tokens present in all results, for instance, header and title table. For each new template containing fields with infinite domains, the database is analyzed again and new k tokens are extracted. Thus, for distinct templates that have fields with infinite domains, distinct values are generated. Finally, we calculate the score r_{2t} (Equation 3) for each token. The ranking is based on number of rows retrieved by tokens (r_2). The difference between r_1 and r_2 is that, r_1 uses the values for discovering new values for text fields while r_2 uses the values for filling text fields. The FTF method works well for both keyword fields and for typed fields.

$$r_{1t} = cf_t \times idf_t \tag{1}$$

and

$$idf_t = log\frac{N}{df_t} \tag{2}$$

and

$$r_{2t} = nr \times idf_t \tag{3}$$

where cf_t is the number of times a token t appears in the database containing the rows obtained so far; N is the number of submissions; df_t is the number of result pages containing t; and nr is the number of records retrieved from the submission.

For our purposes, we divide the Web forms in two types: those that contain only text boxes and those that contain, besides text boxes, finite domain fields, such as selection lists. The division is necessary, because depending on the type of form, the generation of initial values for the fields is different.

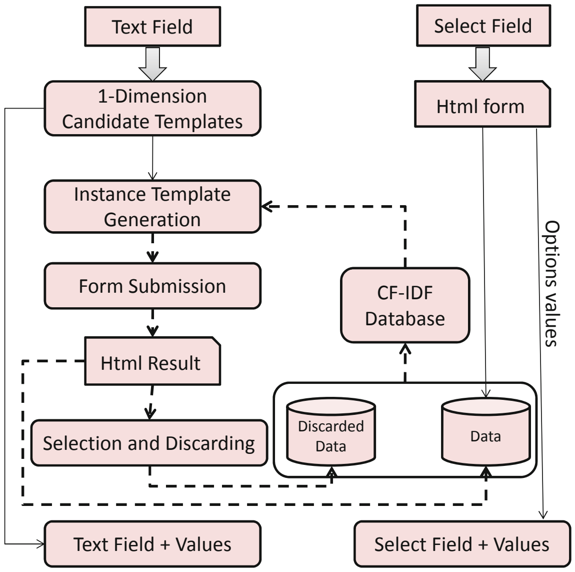


Fig. 4 Value Selection Module

Figure 4 shows the proposed process for discovering good values for filling text boxes. The *Select Field* component represents the fields with finite domains. These fields are filled by the values extracted from the code of the form, in the *option* tag from HTML form. Queries are generated by values extracted from option tag and then submitted. Query results are stored in a database.

The order of submission of queries always starts by finite domain fields followed by infinite domain fields. The database containing query results will be used later to generate values that are used to fill infinite domain fields. The information inside the HTML page which contains the form is extracted only when the form has just infinite domain fields. The information is tokenized and ranked according to Equations 1 and 2. The n most frequent terms are selected to compose new queries.

The *Text Field* component represents the text box fields. Our proposal selects values only for dimension-1 templates. These values will be combined for higher dimension templates. For each template, n instances are generated. The submission of all instances is called *iteration*

The results for each instance are evaluated by checking the number of records retrieved (r_2). Queries that do not return records are discarded. Queries that return just a single row are also discarded because its likely that the single row contains just the heading and not data. In addition, the terms that return the fewest rows are also discarded. The intuition is that, by discarding values that return few rows, we are making room for the selection of new terms that will return more rows in the next iteration.

The FTF method works well for both keyword fields and for typed fields. The main difference between the FTF method and other techniques [1, 6, 8, 10, 11, 13] is that it generates good values for typed fields in a fully automatic manner.

5 Experiments

This section presents experiments performed in order to evaluate the proposed strategies for input value selection and instance template pruning. All experiments were carried out using real Web forms. The proposed strategies and the architecture are domain independent. Forms from several domains (such as, Jobs, Books, Movies, and Food), and sizes were used in experiments. Table 1 shows details about the forms used in experiments. The number of web forms used in our experiments is similar to what is used in related work [6, 8, 10, 11].

Three metrics were used in the evaluation. The coverage [6] C_f, of a form f is the number of distinct records extracted during the whole process. The execution efficiency [13], EE_f, for a form f is the ratio between coverage and the number of URLs submitted ($Total_{submitted}$) in the process. The indexing efficiency, IE_f, for a form f is the average number of records obtained for each distinct URL indexed ($Total_{indexed}$). For 1-dimension templates and finite domain fields, URL submissions are generated according to the options that exist for each field. For n-dimension templates, where $n > 1$, there are several strategies for selection. Kantorski *et al* [2] discuss four strategies to generate URLs for templates. In this paper, the *k-allValues* strategy was used. This strategy uses all values of fields at least once. The intuition behind this method is that the selection of all values can reach higher data coverage. All instance templates (URLs) are generated and only a subset is submitted according to the selected values by *k-allValues* strategy. For the data extraction process, the MDR method [9] is used.

Our evaluation is divided in two parts. In the first, we test the ITP method, and, in the second, we test the FTF method. The Web forms used in each part are different because the FTF method needs forms with text and select fields and forms with only

Table 1 Form properties

ID	Web Form	#Fields		Id	Web Form	#Fields	
		text	select			text	select
Forms for FTF method				Forms for ITP/ITTP method			
1	http://www.beerintheevening.com/pubs/	3	1	1	http://www.foodandwine.com/search/	3	1
2	http://www.foodandwine.com/search/	3	1	2	http://www.global-standard.org/	1	3
3	http://www.mymusic.com/advancedsearch.asp	6	2	3	http://onlineraceresults.com/search/index.php	2	0
4	http://www.posteritati.com/advanced_search.php	1	1	4	http://www.phillyfunguide.com/	1	2
5	http://www.e4s.co.uk/	1	2	5	http://www.whoprofits.org/	2	1
6	http://www.whoprofits.org/	2	1	6	http://www.hcareers.com/seeker/search/	1	5
7	http://www.mldb.org/search-bf	4	0	7	http://www.rtbookreviews.com/rt-search/books	1	2
8	http://usajobs.opm.gov/	2	0	8	http://formovies.com/search/combined.html	3	0
9	http://www.movlic.com/k12/search.asp	2	0	9	http://www.careerbuilder.com/	2	1
10	http://jobs.careerbuilder.com/	2	0	10	http://www.policechiefmagazine.org/magazine/	1	2
11	http://onlineraceresults.com/search/my_results.php	2	0				

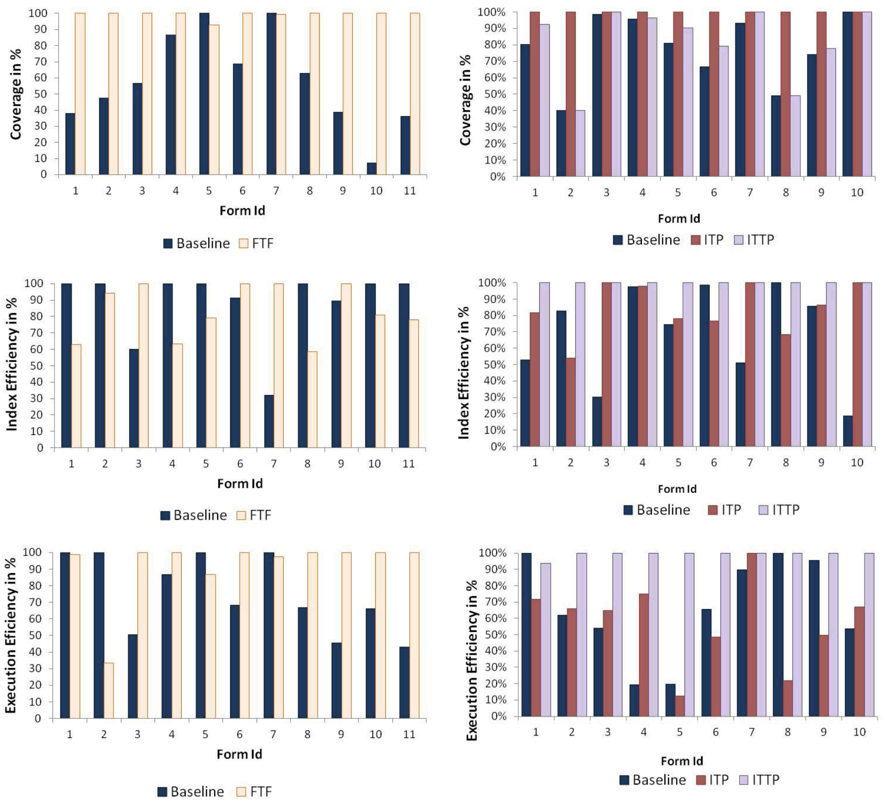


Fig. 5 Evaluation results for ITP, ITTP and FTF methods

text fields. In both cases, our baseline is an implementation based on the method proposed by Madhavan *et al.* [3] (see Section 2). Two versions of our pruning method were tested: ITP which selects informative instance templates (as described in Section 4) and ITTP which, as proposed by Madhavan et al. [3], employs an additional test on the informativeness of each template.

Figure 5 shows the coverage, and the execution efficiency normalized by the best method for each metric and form. In all forms tested, the coverage of the ITP method had the best performance because it uses all templates which means it makes further exploration of the submission possibilities. In some cases the ITTP method obtained the same results of ITP, and both were always better than the baseline. This occurs because ITP does not prune the templates (just the instances), submitting more instances compared to ITTP. On the execution efficiency aspect, ITTP was better than ITP in all forms. This happens because the ITTP method does not submit non-informative templates, reducing the number of non-informative instance templates submitted. On the other hand, compared to the baseline, which also considers only informative templates, the ITTP method submits fewer instance templates with a greater probability of having better quality, since the baseline considers the all the instances of an informative template as informative.

For the FTF method, the number of URLs indexed was higher in all forms. This happens because the generated values are better, i.e., they retrieve more rows. Thus, the likelihood of a template being informative is higher for our method than for the baseline. Another finding is that, for forms that contain only text fields, the templates with dimension greater than one have a small probability of being informative. Our experiments showed that all 1-dimension templates are informative, increasing the chance of templates with higher dimension also being informative. The indexing efficiency presented by FTF was lower than the baseline. This happened because the baseline generated fewer URLs. For instance, form id 2 has 852 URLs for FTF and 135 URLs for the baseline. The templates from the baseline method, which contain the text fields, were considered non-informative. The FTF method reached a higher coverage compared to the baseline. The average coverage of the baseline was 47,8%, while the average coverage by FTF was 81,8%. This shows an advantage of our method. The *e4s* dataset (form id 5) showed a distinct behavior between baseline and FTF. This happened because the text field from this form is of keyword type and the remaining fields are selection lists. FTF was always better for forms that contain only text box fields.

6 Conclusion and Future Work

This paper described an approach for filling Web forms automatically. It includes two main methods for improving the search on the Hidden Web: *i.* ITP, a method to minimize the number of queries submitted to the form; and *ii.* FTF, an automatic method for selecting values for text fields. The ITP strategy evaluates each submission of an instance template. These instances are classified as informative (which retrieve new data) or non-informative (which do not return new distinct data). We can reduce the number of submissions by pruning the non-informative instances. The FTF method extracts values for text fields based on a feedback loop. The advantage is that FTF is totally automatic and can be used in any Web form that has text fields. Although the FTF method is domain-independent, the values selected for text box fields adapt to the domain. Our experiments demonstrate that our approaches are able to properly deal with text fields and query selection and have shown to be useful as a solution for filling Web forms automatically.

In future work, we will carry on with the research into two phases. In the first phase, we will use machine learning techniques to discover field values in templates with order higher than one. The second phase will entail the definition of a new method to determine values for text fields considering the number of distinct rows and the order of instance template submissions to assess the quality of the selected values. Further investigations will also include handling fields generated dynamically and considering dependencies between values in distinct fields of a form.

Acknowledgements. This research was partially supported by the National Counsel of Technological and Scientific Development, CNPq/Brazil, project number 480283/2010-9.

References

1. Jiang, L., Wu, Z., Zheng, Q., Liu, J.: Learning Deep Web Crawling with Diverse Features. In: WI/IAT, pp. 572–575 (2009)
2. Kantorski, G., Moraes, T., Heuser, C.: Strategies for Automatically Filling Web Forms. Technical Report RP367-PPGC, Instituto de Informatica, UFRGS, Brazil (2012)
3. Madhavan, J., Ko, D., Kot, Ł., Ganapathy, V., Rasmussen, A., Halevy, A.: Google's Deep Web Crawl. Proc. of the VLDB Endowment 1(2), 1241–1252 (2008)
4. Bergman, M.K.: The Deep Web: Surfacing hidden value. Journal of Electronic Publishing 7(1), 07–01 (2001)
5. He, B., Patel, M., Zhang, Z., Chang, K.C.C.: Accessing the Deep Web. Communications of the ACM 50(5), 94–101 (2007)
6. Barbosa, L., Freire, J.: Siphoning Hidden-Web Data through keyword-based interfaces. In: SBBD, pp. 309–321 (2004)
7. Khare, R., An, Y., Song, I.Y.: Understanding deep web search interfaces: A survey. SIGMOD Rec. 39(1), 33–40 (2010)
8. Liddle, S.W., Embley, D.W., Scott, D.T., Yau, S.H.: Extracting Data behind Web Forms. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 402–413. Springer, Heidelberg (2003)
9. Liu, B., Grossman, R., Zhai, Y.: Mining data records in Web pages. In: SIGKDD, pp. 601–606. ACM (2003)
10. Raghavan, S., Garcia-Molina, H.: Crawling the Hidden Web. In: VLDB, pp. 129–138 (2001)
11. Toda, G.A., Cortez, E., da Silva, A.S., de Moura, E.: A probabilistic approach for automatically filling form-based web interfaces. Proc. of the VLDB Endowment 4(3), 151–160 (2010)
12. Wu, P., Wen, J.R., Liu, H., Ma, W.Y.: Query selection techniques for efficient crawling of structured web sources. In: ICDE, p. 47. IEEE (2006)
13. Ntoulas, A., Zerfos, P., Cho, J.: Downloading textual hidden web content through keyword queries. In: JCDL, pp. 100–109 (2005)
14. Cafarella, M.J., Madhavan, J., Halevy, A.: Web-scale extraction of structured data. SIGMOD Rec. 37(4), 55–61 (2009)
15. Chang, K.C.C., He, B., Li, C., Patel, M., Zhang, Z.: Structured databases on the web: Observations and implications. SIGMOD Rec. 33(3), 61–70 (2004)
16. Florescu, D., Levy, A., Mendelzon, A.: Database techniques for the World-Wide Web: a survey. SIGMOD Rec. 27(3), 59–74 (1998)

A Relative Feature Selection Algorithm for Graph Classification

Yaser Keneshloo and Sasan Yazdani

Abstract. Graph classification is one of the most important research fields in data mining nowadays and many algorithms have been proposed to address this issue. In practice, labeling large or even medium-size graphs is a hard task and we need experts to do so. The biggest challenge in graph classification is extracting a set of proper features from graphs. Since graphs are represented by a complex data structure, this issue has been dealt with for a long time. Previous methods focused on extracting features from a certain class in a dataset. In this paper we propose a new feature selection method that extracts features from each graph rather than extracting them from a certain class in the dataset. We extract only frequent subgraphs as features. These subgraphs are chosen according to their number of occurrences in a graph. Moreover, we proposed a new formula which calculates the minimum number of occurrences required for a subgraph to be considered as frequent. We experimented on five real datasets and reached 7-17% higher accuracy than previously proposed methods.

1 Introduction

In recent years, there has been a great emphasis on graph mining due to its vast application in web, bio-informatics, social sciences and networks. Great number of algorithms has been proposed on graph clustering, graph classification, graph querying, motif finding and finding dense patterns in graph [3, 1, 17, 2]. Among these fields, graph classification recently has been focused on greatly. Labeling graphs is usually time consuming, expensive and requires experts. For example in order to label a chemical compound of an AIDS sample, we need to perform lots of

Yaser Keneshloo · Sasan Yazdani
Iran University of Science and Technology
e-mail: {yaser_keneshloo,sasan_yazdani}@comp.iust.ac.ir

experiments to extract important information required to determine whether the virus is active or not [10]. In intrusion detection systems (IDS) a network analyzer is needed to detect an intrusion based on network logs accumulated over months and years [16]. Therefore if we can build a model to detect these structures we can cut these complexities both in cost and time consumption. The biggest challenge in graph classification is feature extraction, thus the better the features extracted from the graph, the higher the chance of correct classification. For example, in Fig. 1, we presented the output of a Decision Tree classifier used in our experiments on AIDS dataset. As we can see in this figure, at level two of this decision tree we have the chemical compound, $C-1-C-1-C-1-N-1-C$[1]. With a simple threshold value we can easily separate two classes, CA and CI from each other. In common approaches, a graph dataset is postulated and for each class in the dataset, a set of features known as frequent subgraphs is extracted. For labeling an unlabeled graph, classification is continued by checking whether each class's features are occurred in the unlabeled graph or not. Afterwards, classifier measures the structure similarity between the unlabeled graph and every class in the dataset. Finally, graph's label will be whichever class its structure similarity is closest to. Instead of extracting a feature set of frequent subgraphs for each class in dataset, we extract a set of features for each graph in the dataset and then unlabeled graphs are classified based on their distinct frequent subgraphs. After feature extraction, graphs are represented by a real valued vector which can then be used by any known classifier for classification.

Current approaches in graph classification suffer from some inefficiencies as described below:

1. Initial Parameter: All recent graph classification approaches use an initial parameter known as *Min_Sup*, which is set by user. If this value is not set appropriately, algorithm's performance are affected extremely [21, 9, 13].
2. Dataset size: If a dataset is small or has small number of samples in a specific class, usually previous algorithms are unable to find a proper feature set because they are concentrated to extract features from a class as a whole rather than extracting features from each sample.

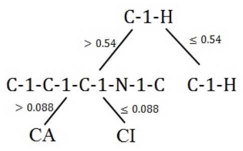


Fig. 1 Separation of two classes using a single rule at level 2

[1] When we show a chemical compound as $C-1-C-1-C-1-N-1-C$ it means the first Carbon joins second with a one bond join and the second is jointed with the third with another one bond join and so on.

In this paper we propose a novel model, *RelativeGraphMiner* (RgMiner), for graph classification which can classify graphs without any starting parameters. To the best of our knowledge, this model is the first model to classify graphs without starting parameters. Moreover, It is the first model to use frequent subgraphs extracted from a single graph as the kernel for classification task. In this approach, by adding a new graph to a dataset, we only need to find frequent subgraphs for this new graph while in previous approaches features must be re-extracted from scratch.

In the following sections, we first point some preliminaries in section 2. Section 3 will focus on related work in graph classification proposed in recent years. Section 4 presents our proposed method. In section 5, we'll present our experimentations on real datasets. Finally section 6, concludes materials proposed in this paper.

2 Preliminary

This section provides basic definitions needed for the rest of this paper. Weighted graph is a common representation in graph classification problems. A weighted graph G can be represented as a quaternary $G = (V, E, l(V), l(E))$ where V is the set of nodes, E is the set of edges and $L(v)$ and $L(E)$ are two functions that label nodes and edges, respectively in G. A subgraph G_s of G, denoted as $G_s \subseteq G$ and is represented as follows:

$$G_s = (V_s, E_s, l_s(V_s), l_s(E_s))$$

Where $V_s \subseteq V, E_s \subseteq E, L_s(V_s) = L(V)$ and $L_s(E_s) = L(E)$. Support of a graph G_s in dataset D is calculated by the number of graphs, $G \in D$ in dataset where $G_s \subseteq G$. We define a new concept of support and show the support of G_s as S_s and define it as number of occurrences of G_s in G. In both definitions G_s is called frequent if number of its occurrences in G is greater or equal to a threshold parameter called *Min_Sup*. G_1 is isomorph with G_2, if there exists a bijection function $f : V(G_1) \to V(G_2)$ such that for any two adjacent vertex u and v in G_1, $f(u)$ and $f(v)$ are also adjacent in G_2.

3 Related Works

As mentioned before the most important part of classification is extraction of a good feature set. Frequent subgraphs are one of the most common features in graph classification problems. Various methods have been proposed for extracting frequent subgraphs from a graph dataset [12, 13, 9, 15, 21]. All previous methods extract frequent subgraphs from a graph dataset hence they cannot be used for extracting frequent subgraphs, existing in a single graph.

gActive[10] and gSSC[18] used gSpan as their core feature extracting method. In these approaches, for a dataset, all frequent subgraphs are extracted and if a subgraph occurs in a graph G, its corresponding entry in G's feature vector will set to 1, otherwise it will be set to zero. There are other approaches like fragment based approaches [6, 20], kernel based method [19] and topological methods [14]. In fragment based approaches each graph is considered as structures like frequent subgraphs. Kernel based approaches check two graphs similarity using different kernels like random walks, shortest path, cyclic pattern, subtree and graphlet. For detailed discussion refer to [19]. Topological approaches select different characteristics in a single graph like average degree, average clustering coefficient, effective radius and diameter and etc and use them for classification [14].

Our proposed approach doesn't fall in neither of these categories. It takes some of its concept from algorithms like [10] and some from fragment based approaches. Our algorithm is based on extracting frequent subgraphs in a single graph, not a graph dataset. We could see the discriminative power of frequent subgraphs in different categories of application. In [22] the authors used frequent subgraphs as a tool for filtering irrelevant graphs when it is searching for a query graph in a dataset. Grafil attempts to filter out as many irrelevant graphs as possible to apply the graph isomorphism operation, which is proved to be an NP-Complete problem, on small number of graphs. Although frequent subgraphs are powerful feature, to discriminate graphs for all algorithms that used this frequent subgraphs as features, there is another important parameter namely, *Min_Sup*, that should be selected carefully. *Min_Sup* determines the minimum number of occurrences of each subgraph in order to be considered as frequent. So, an important question in these kinds of algorithms is to figure how we can determine a proper value for this parameter. All feature extraction algorithms that use frequent subgraphs as features have an important parameter namely, *Min_Sup*, that should be selected carefully. *Min_Sup* determines minimum number of occurrences of each subgraph in order to be considered as frequent. Therefore, an important issue in these kinds of algorithms is to figure how we can determine the proper value for this parameter. All previous algorithms considered this parameter as user-defined, so if this parameter is set inappropriately, the final result is affected extremely [9, 21]. In order to find the best value for *Min_Sup*, one can tune this parameter by running algorithm several times with different *Min_Sup* values and then pick the best result, which of course is time-consuming.

4 RgMiner Algorithm

In this section first we describe how features are extracted from graphs in a dataset. Then we propose a formula which helps us extract suitable features from graphs with respect to their size and density. Afterwards in section 4.3, we explain how normalized feature vectors can help improving classification results. Finally, we describe how to use extracted features for classification.

4.1 Creating Frequent Subgraphs

In *RgMiner*, each graph is represented by a vector of its frequent subgraphs. This can best be shown by an example. Suppose we want to extract features for the graph in Fig. 2(a). As we can see, Fig. 2(b) and 2(c) are two arbitrary subgraphs extracted from Fig. 2(a), where the subgraph represented in Fig. 2(b) has support value of 1 and the one in Fig. 2(c) has support value of 4, which is the number of its isomorph as shown in Fig. 2(c) through Fig. 2(f). Usually many subgraphs exist in a graph, so finding all of them is time consuming. In order to solve this issue we only find and extend subgraphs which their support are greater than *Min_Sup*. Therefore, subgraphs not satisfying this rule are discarded. Also, in order to decrease *RgMiner*'s complexity even more, we only evaluate subgraphs that are simple walks [5]. A simple walk is a path which nodes can repeat more than once but edges cannot be repetitive. After extending new subgraphs, their support must be calculated in order to find whether they are frequent or not. If a subgraph's support is equal or greater than *Min_Sup* then it is counted as frequent. Also, this subgraph will be considered for next expansions.

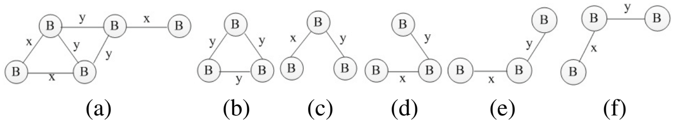


Fig. 2 A simple graph with some of its subgraphs. Graph (b) has support value=1 and graphs (c-f) are isomorph with support value=4.

4.2 *Relative_Min_Sup*

As described in section 3, *Min_Sup* plays an important role in finding frequent subgraphs. If *Min_Sup* is set inappropriately large, algorithms may not be able to find any frequent subgraph at all. On the other hand if *Min_Sup* is set too small, many frequent subgraphs will be found. In previous approaches, this value is always set by user and if it is not set properly or if the user has a little or no knowledge about the dataset to select this value correctly, classification results will be poor. In order to prevent this kind of problems we proposed *Relative_Min_Sup* which finds the value of *Min_Sup* relatively based on graph's size and density. *Relative_Min_Sup* works based on the following rule: The bigger and denser a graph, the higher its *Min_Sup*.

In recent approaches, first a threshold value $\theta \in [0,1]$ is specified by user for a dataset, then *Min_Sup* value for this dataset is calculated using $\lfloor \theta \times |D| \rfloor$, where $|D|$ shows the number of graphs in dataset. We can use this naïve approach for our problem, i.e. setting a threshold for all of our graphs and multiplying it by the graph size ($\lfloor \theta \times |V| \rfloor$). However, this approach has a big disadvantage: through our experiments on chemical datasets, we figured that determining *Min_Sup* value for each graph does not have a linear relation with its size and density. In the naïve approach this relation is calculated with a linear function. We can show the effectiveness of a feature extracting algorithm using *feature variance*. *feature variance* can be used to show how well an algorithm performs in extracting features for a given graph. In order to calculate *feature variance*, first we need to calculate how many features are extracted from a graph with respect to its size, i.e. δ= *(number of frequent subgraphs / number of graph nodes).* Clearly more frequent subgraphs are extracted from bigger graphs. Therefore, to be able to compare two graphs, no matter how big or small, we divided the number of frequent subgraphs by the number of graph's nodes. A dataset $D = \{G_1, \ldots, G_N\}$ can be represented by $\delta_D = \{\delta_1, \ldots, \delta_N\}$, where δ_i is the corresponding feature to size ratio for G_i. We can then define *feature variance* of an algorithm A on dataset D using δ_D as follows:

$$feature\ variance = \gamma_D^A = \max(\delta_D) - \min(\delta_D) \tag{1}$$

This measure can be used to compare how well two feature extraction algorithm are doing on dataset D. A small γ_D^A shows that algorithm A extracts appropriate number of subgraphs proportional to graph's size, while a large γ_D^A means that A is performing poorly on D. We compare *feature variance* of our proposed measure against naïve approach at the end of this section.

To determine a suitable *Relative_Min_Sup* for a graph we follow a simple rule: The sparser a graph, the less the number of frequent subgraphs. With this in mind, a formula should be proposed to determine *Min_Sup* with respect to graph's size and density. In graph theory, density of a graph can be calculated using the following formula:

$$density = \frac{2|E|}{|V|(|V|-1)} \tag{2}$$

For a sparse graph like a chemical compound, density is close to zero while for a dense graph's density is close to one. Because of the denominator, for two large chemical compounds, *density* will be nearly equal and close to zero, which is a big issue. Therefore, density cannot help us in finding a good *Min_Sup* in chemical compounds so we propose a new measure which takes advantage of density's statistics and also resolves the nonlinearity issue:

$$Relative\_Min\_Sup = M_{R_i} = \lfloor \frac{|E|}{\log(|V|(|V|-1)/2)} \rfloor \tag{3}$$

In this equation, we injected graph's statistics like number of edges and number of nodes to find a proper *Min_Sup*. Moreover, *Relative_Min_Sup* behaves like density, but does not have density's denominator drawback, which is because *Relative_Min_Sup* uses logarithm function to reduce the impact of $|V|$ in denominator. As we can see in Eq. 3, by increasing the number of nodes in a graph, the value of its *Min_Sup* will increase inverse-logarithmically, which is of course in a non-linear manner, so *Relative_Min_Sup* satisfies the condition required for selecting a proper value for *Min_Sup*.

We used PDC-FR dataset, which is described in section 5, to compare the *feature variance* of our approach with naïve approach. For naïve approach the value of *feature variance* was $\gamma_{PDC-FR}^{Naive} = 38.6$, while our approach reached the value of $\gamma_{PDC-FR}^{Relative} = 12.85$, which is clearly better.

4.3 Normalized Feature Vector

In our proposed approach instead of using binary features we used integer valued features, since number of occurrences of a frequent subgraph is very important by itself. Each integer represents the corresponding frequent subgraph's support. But support value of a specific subgraph in a large graph is much greater than its support in a small one, while this specific subgraph may have equal impact in these two graphs' structures. So we need a normalization factor to reduce the effect of this issue. Therefore, in order to get better results, we need to normalize them with respect to their corresponding graph's size, i.e .x_i^j=(S_j/n) where S_j is support of frequent subgraph g_j and n represents number of G_i's nodes.

4.4 Feature Aggregation

Usually there are many frequent subgraphs in large graphs but even for two graphs with a little difference in their structure, there are some distinct subgraphs which make them different. In order to consider all of these features, we use an aggregate feature vector (AFV) for classification. Aggregate feature vector contains all distinct frequent subgraphs occurring in a dataset. By using AFV, each graph in our dataset will have n_{AFV} number of features. So, the feature vector of a graph G_i is created as follows:

$$x_i^j = \begin{cases} \frac{S_j}{n} & \text{if } g_j \subset G_i \text{ and } S_j \geq M_{R_i} \\ 0 & \text{if } g_j \not\subset G_i \text{ or } S_j < M_{R_i} \end{cases}$$

Where M_{R_i} is the calculated relative *Min_Sup* for the graph G_i.

5 Experiments

We used five real world datasets to evaluate *RgMiner*. Table 1 presents more information about these five datasets. All datasets are comprised of graphs, each representing a chemical compound. Each graph node represents a chemical element and each edge shows a bond between two chemical elements in that compound.

Table 1 Real world dataset used for *RgMiner*

Dataset	#Positive	#Total
AIDS	518	1237
PTC-FM	78	201
PTC-MM	67	189
PTC-FR	61	201
PTC-MR	69	193

1. AIDS Antiviral Screen Data[2]. This dataset is gathered by examining tens of thousands of chemical compounds to identify whether they are HIV-active or not. Chemical compounds in this dataset are divided in three categories: Active (CA), Inactive (CI) and Moderately active (CM). We considered CA and CM compounds as negative class and CI as positive class, which are the same settings used in [10]. Finally, to have a fair classification we considered all graphs from CM and CI and randomly chose the same amount of graphs from CA.
2. Predictive-Taxonomy Challenge (PTC): Last four datasets are taken from the challenge proposed in Predictive-Taxonomy website[3]. The goal of this classification challenge was to detect chemical compounds with carcinogenicity activity among them for four types of animal: Male Mouse (MM), Female Mouse (FM), Male Rat (MR) and Female Rat (FR). The compound in this dataset divided in three categories: P: Positive, N: Negative and E: Equivocal. We used only positive and negative data for classification and ignored E samples.

5.1 Classifier Production Methods

In order to evaluate our method and compare it with previous approaches, we employed three different classifiers and compared their results with previous works' best accuracies. All classifiers have been created using Weka 3.6 [7] using 10-fold cross-validation. Classifiers used for comparison are Decision Tree (J48), Random Forest, LibSVM. We used these diverse classifiers in order to have different perspectives on discriminant functions in feature space and also to have a comprehensive evaluation. Also, we built a binary version of our approach called *RgMiner_0-1* which instead of using normalized support and real valued feature vectors, uses binary valued feature vectors. This means, if a frequent subgraph occurs in a graph the corresponding value in its feature vector is set to one, otherwise it is set to zero. We created *RgMiner_0-1* in order to be able to fairly compare our approach with gActive and also, to support two main idea of our approach: first, we wanted to show that if we extract relevant information from each graph rather than extracting them from graphs belonging to a certain class in a dataset, we'll get better results. Second, we wanted to show that by using real values rather than binary ones, we can achieve even higher accuracy. Fig. 3(a) through 3(c) shows final results for these three classifiers based on their accuracy, F-Measure and MCC metrics, respectively. We used gActive and gSSC which are both popular in graph classification. gSSC uses semi-supervised feature extraction algorithm and based on experiments in [11] , best accuracy gained by this method for PTC-FM and PTC-MM datasets are 57% and 57.7%, respectively, while our approach classifies these two datasets with 74.12% and 64.67% accuracy in best case scenario which improves gSSC algorithm by about 17% and 7%. On the other hand, gActive results in [10] for AIDS dataset

[2] Available at: `http://dtp.nci.nih.gov/docs/aids/aids_data.html`

[3] Available at: `http://www.predictive-toxicology.org/ptc/`

shows 67% classification accuracy, while our method reaches 75.9% in accuracy which is an improvement about 9% for this dataset.

Table 2, represents these three classifier's detailed results on each dataset based on different performance metrics we used for evaluating *RgMiner*. Classifiers which work best on each dataset based on different metrics are represented in bold.

Table 2 Result from *RgMiner* using different classifier and metrics

Classifier/Metric		AIDS	PTC-FM	PTC-MM	PTC-FR	PTC-MR
LibSVM	Accuracy	70	66.67	64.55	**69.65**	66.32
	F-Measure	0.68	0.62	0.59	0.63	0.62
	MCC	0.34	0.19	0.09	0.10	0.16
Random Forest	Accuracy	**75.9**	**74.12**	**64.67**	64.65	**67.16**
	F-Measure	**0.759**	0.62	0.61	**0.63**	**0.64**
	MCC	**0.50**	0.20	0.14	**0.11**	**0.22**
J48	Accuracy	69.84	63.68	62.96	60.69	59.06
	F-Measure	0.70	**0.63**	**0.62**	0.61	0.57
	MCC	0.38	**0.21**	**0.15**	0.09	0.05
RgMiner_0-1	Accuracy	75.66	63.68	64.55	71.14	68.39
	F-Measure	0.757	0.60	0.61	0.67	0.65
	MCC	0.50	0.16	0.13	0.20	0.24

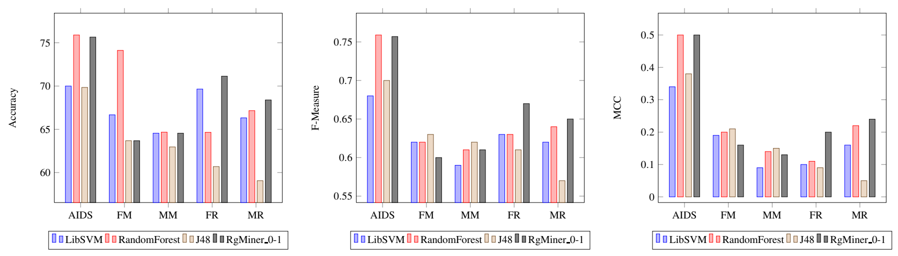
5.2 Discussion

In this section, we describe the result achieved by *RgMiner*. We used different metrics for our evaluation since Accuracy is not a good indicator of behavior of a classifier by itself, especially when datasets are unbalanced. Therefore, we need to use some complementary measures to evaluate our proposed method thoroughly. In addition to accuracy, we used the *F-Measure* and *Matthews Correlation Coefficient (MCC)* to validate our result. MCC is a performance quality measure used in two-class classification problems and is an interesting performance metric in bioinformatics. MCC measure has -1 as its lower bound and +1 as its upper bound, where +1 indicates perfect classification, -1 shows inverse classification, and 0 represents a random classifier. MCC can be measured using Eq. 4.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4}$$

In Eq. 4, parameters TP, TN, FP, FN stands for true positive, true negative, false positive and false negative extracted from classifier's confusion matrix.

We employed LibSVM which uses C-SVM to build the model and with RBF as its kernel, which is also the default setting in Weka. RBF kernel function maps the original feature space to a higher-dimension feature space and then, the classifier separates two classes in the dataset linearly with a spherical plane. However, RBF kernel function results in poor classification performance. This happens because in our created dataset, number of features dominates the number of samples by about a



(a) Accuracy Measure (b) F-Measure (c) MCC Measure

Fig. 3 Result of classification using *RgMiner* on different classifiers

factor of 14. As stated in [4], in these situations, this method is likely to give us poor results. In these cases since we have an enormous difference between number of samples and features, we do not need yet another kernel function to map our feature space into a higher dimension one. So, it is better to use a linear kernel to do the classification.

Second classifier used for our evaluation was decision tree (J48 in Table 2), which is a supervised rule-based classifier and use simple if-then rules to determine each sample's class. The prominent feature of decision tree among other classifiers is the high interpretability of this classifier. This classifier shows us a good representation to describe our algorithm better. Fig. 1 shows a part of our final decision tree classifier created for AIDS dataset. As we can see in second level of this partial tree, a chemical compound can be classified by a simple rule of checking whether a $C-1-C-1-C-1-N-1-C$ subgraph occurs in its structure. If it does and has a normalized support of equal or greater than 0.088 then the graph is classified as CI, Otherwise graph's label is considered CA. One important characteristic of a decision tree classifier is that no matter how many features used to describe an instance, it only chooses a subset of features that is enough to discriminate existing classes in a dataset against one another. For instance only 11 features were used to build a decision tree classifier for PTC-MR dataset, which compared to 2829, total number of features in AFV for this dataset, is much smaller, which shows our feature extracting approach is performing so well that only 0.4% of features extracted from the dataset is enough for its classification.

Also, we used Random Forest (RF) which technically is an ensemble of classifiers to show that we could reach even better results by incorporating ensemble of classifiers that take advantage of different set of features and samples in building classifiers. Finally, in Table 2 we can see the performance of *RgMiner* using binary values for each feature vector. As we can see, performance of this classifier is near to *RgMiner* but in some datasets it is less preferable. This supports the idea that higher performance can be achieved by simply extracting frequent subgraphs from each graph in a dataset rather than extracting them for each class separately. Using real valued feature vectors represents that relative support does even better than binary values.

Comparing previously proposed approaches with our approach, we can see ours has four great advantages: First of all we use very simple subgraphs instead of using complex ones. This reduces graph isomorphism's cost. Second, we consider all subgraphs no matter how big or dense they are. This fact can be seen in level one and two of the tree shown in Fig. 1. Third, we employ *Relative_Min_Sup* to find a proper minimum number of occurrences of a subgraph, in order to be considered frequent. Finally, we use real valued feature vectors by including the number of occurrences of each subgraph, as an important factor for classification. This way each frequent subgraph has its own impact on classification and is not considered equal to others.

6 Conclusions

In this paper we addressed the problem of graph classification and proposed a new algorithm for this issue. In our proposed algorithm we used the concept of frequent subgraphs in a single graph to create feature sets. Also we proposed an innovative formula to determine the value of *Min_Sup* dynamically based on graph's size and density, finally we normalized the value of supports calculated for each subgraph with respect to graph's size to emphasize on the relative importance of each subgraph. In spite of prior approaches that extract feature sets from and for a certain class in a dataset, we create this feature set from each graph and use it for classification. Through experiments on five real world datasets, we showed that our approach reached up to 17% higher accuracy than prior approaches.

References

1. Aggarwal, C.C.: On classification of graph streams. In: Proceedings of Eleventh SIAM International Conference on Data Mining, SDM 2011, Arizona, Mesa (2011)
2. Aggarwal, C.C., Li, Y., Yu, P.S., Jin, R.: On dense pattern mining in graph streams. PVDLB 3(1), 975–984 (2010)
3. Aggarwal, C.C., Zhao, Y., Yu, P.S.: On clustering graph streams. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, Columbus, Ohio, USA (2010)
4. Auria, L., Moro, R.A.: Support Vector Machines (SVM) as a technique for solvency analysis. Discussion Papers of DIW Berlin (2008)
5. Bondy, J.A.: Graph Theory with Applications, pp. 12–21. Elsevier Science Ltd. (1976)
6. Cheng, H., Yan, X., Han, J., Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: Proc. of ICDE, Istanbul, Turkey (2007)
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, L.H.: The Weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)
8. He, H., Singh, A.K.: Closure-Tree: An index structure for graph queries. In: ICDE 2006, Atlanta, Georgia (2006)
9. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraph in the presence of isomorphism. In: Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM 2003, Melbourne (2003)

10. Kong, X., Fan, W., Yu, P.S.: Dual active feature and sample selection for graph classification. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California (2011)
11. Kong, X., Yu, P.S.: Semi-supervised feature selection for graph classification. In: Proceeding of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, Washington, DC (2010)
12. Krishna, V., Suri, N.N.R.R., Athithan, G.: A comparative survey of algorithms for frequent subgraph discovery. Current Science 100(2) (2011)
13. Kuramochi, M., Karypis, G.: An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1038–1051 (2004)
14. Li, G., Semerci, M., Yener, B., Zaki, M.J.: Graph Classification via Topological and Label Attributes. In: 9th Workshop on Mining and Learning with Graphs (with SIGKDD) (August 2011)
15. Nijssen, S., Kok, J.: A quick start in frequent structure mining can make a difference. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 647–652. ACM (2004)
16. Parveen, P., Weger, Z.R., Thuraisingham, B.M., Hamlen, K.W., Khan, L.: Insider Threat Detection using Stream Mining and Graph Mining. In: IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI 2011, Boca Raton, FL (2011)
17. Singh, A.K.: Graph querying, graph motif mining and the discovery of clusters, United State Patent, Santa Barbara, CA (2011)
18. Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H., Smola, A., Song, L., Yu, P.S., Yan, X., Borgwardt, K.M.: Near-optimal supervised feature selection among frequent subgraphs. In: SIAM International Conference on Data Mining (2009)
19. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, I.R., Borgwardt, K.M.: Graph Kernels. Journal of Machine Learning Research 11, 1201–1242 (2010)
20. Wale, N., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. In: Proc. of ICDM, Hong Kong, pp. 678–689 (2006)
21. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM 2002 (2002)
22. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: SIGMOD Conference (2005)

Generic Conceptual Framework for Handling Schema Diversity during Source Integration

Semantic Database Case Study

Selma Khouri, Ladjel Bellatreche, and Nabila Berkani

Abstract. DataBase Integration Systems (*DIS*) aim at providing a unified view of data stored in different *heterogeneous* local sources through a global schema. The schemas of global and local sources are represented by *a-priori known* logical representations. This makes the potential deployment model of the DIS, like for Data Warehouse (*DW*) systems, *rigid* and *inflexible*. To overcome these limitations, we claim that the integration process should be completely performed at the *conceptual level* independently of any implementation constraint. After studying existing database (*DB*) integration systems, we propose through this paper a generic conceptual framework for *DB* schema integration. This framework is generic as it *subsumes* most important *DB* integration systems studied. It is defined based on the description logic formalism. We then instantiate it through a case study considering a set of Oracle semantic *DB*, that are *DB* storing their own conceptual model, generated using Lehigh University BenchMark (LUBM). These *DB* participate in the construction of a semantic *DW*.

Keywords: DataBase Integration, Description Logic, Semantic Databases.

1 Introduction

From the last decade, integrating heterogeneous and autonomous data sources generate a real industry related to *integration* [13]. High demand on integration solutions is identified for various IT applications: data warehouse, peer to peer data,

Selma Khouri · Ladjel Bellatreche
LIAS/ISAE-ENSMA
Poitiers University, France
e-mail: (selma.khouri,bellatreche)@ensma.fr

Selma Khouri · Nabila Berkani
National High School for Computer Science (ESI), Algiers, Algeria
e-mail: (s_khouri,n_berkani)@esi.dz

e-commerce and Web services, etc. The goal of a *DIS* is to provide a global schema representing a unified view of data stored in different heterogeneous sources. Each source is associated with a local schema. Formally, a *DIS* may be represented by the triple $< G, S, M >$, where G, S and M describe respectively the global schema of the *DIS*, a set of local schemas that describe the structure of each data source, and the mapping between G and S [15]. The global and local schemas were represented in first *DIS* by their logical models, i.e. data models of current DBMS. Usually, the relational logical model was used due to its popularity and its performances. Mappings between these schemas are defined using SQL queries (views) based on the traditional operators of the relational algebra. An important disadvantage of such systems is that the potential deployment model of the final *DIS* is already known and frozen from the beginning of the integration process. In a real scenario, especially for *DW* systems that can be seen as a special case of a *DIS* where data sources are duplicated in the same repository, the deployment model can follow different representations according to specific requirements. To overcome this problem and to give schema mappings a clear semantic, we claim that the integration process must be achieved fully at the conceptual level. This assumption is actually shared by some recent research studies that will be discussed in the related work section.

In these studies, we noticed that many *DIS* using conceptual models are *ontology-based* integration systems. Domain ontologies have been introduced in the data integration field in order to manage syntactic and semantic conflicts between data [1]. They are actually seen as conceptual models, providing a unified and consensual view of a given domain. Note that ontologies exist in various domains: engineering, medicine, environment, etc. Different structures of ontology-based *DIS* have been proposed where a domain ontology plays the role of the global schema G in the framework $< G, S, M >$. Many of these studies make the hypothesis that each data sources has its own local ontology [1]. With the spectacular development of ontologies and the explosion of ontological instances [10], academicians and industrials identify the need to propose a new type of *DB* that allows local ontology to be stored in the same repository of data instances. This type of *DB* is called $\mathscr{S}$emantic *DB* (*SDB*). A large panoply of *SDB* exist [9, 16, 18]. They differ according to their *architectures* and the target *storage models*. The presence of these *SDB* puts them as serious candidates for a *fully conceptual integration* and they should *co-habit with the traditional sources*.

The objective of this paper is twofold. Firstly to provide a generic conceptual integration framework. To manage this diversity of the nature of sources participating in the integration process, such a framework represents a crucial issue that should be addressed. This diversity calls into question the traditional SQL view mapping. The *description logics* (DL) formalism may be a solution for upgrading these mapping. The framework proposed uses description logics for expressing mapping. (2) Secondly to instantiate our framework for integrating *SDB* in a DW.

This paper is structured as follows: section 2 presents the related work, where different integration systems are studied and classified using four orthogonal criteria. Section 3 presents the background where we define the notions of ontologies, *SDB* and the DL formalism. Section 4 proposes the generic integration framework.

Section 5 presents the case study where the proposed framework is instantiated on *SDB* used to construct *DW* system. Section 6 concludes the paper.

2 Related Work

We present in this section a classification of integration systems enriching the classification we proposed in [1] by the fourth criterion: abstraction level of the *DIS*. Initial classifications of integration systems used one criterion: *the sense of mapping* between the global and local schemas distinguishing Global As View (*GaV*) and Local As View (*LaV*) approaches . The second criterion is *data representation*. In a *materialized* architecture, data of local sources are duplicated and stored in a single database called the data warehouse. In the *virtual* architecture, data are remained in local sources and accessed through a mediator. Other contributions consider the number of used *ontologies for automating the integration process*. We find systems using *manual* mappings, *semi-automatic* approaches based on vocabularies or linguistic ontologies, and *automatic* approaches by using conceptual domain ontologies as a reference model. The last criterion considers the *abstraction level* of the global and local schemas (logical or conceptual levels) where we distinguish two main architectures of *DIS*: *partial* conceptual *DIS* and *fully* conceptual *DIS*.

Partial Conceptual *DIS*: In the first architecture, only the global schema is presented by its conceptual model. Sources schemas are expressed in their logical model. [6], [3] and [4] proposed DIS defining mappings between relational sources and conceptual global schemas respectively formalized in : an enriched *Entity-Relationship* (ER) model described in *DLR* formalism, *DL-Lite*$_A$ model, and *OWL2DL* formalism.

Fully Conceptual *DIS*: In this second architecture, both the global and local schemas are described using a conceptual representation. [5] proposed a *DIS* where the global schema is defined by a *DLR* model, and sources are local ontologies seen as a set of extensions (instances) of an ontology-concept. This system defines mappings at the *extensional* level (source instances), which can be time consuming. We thus claim that mappings should be abstracted to the *intensional* level. [12] describe a *DIS*, where all schemas are described in *OWL-DL* language. [1] proposed an *DIS* where sources are *SDB*. Mappings used in this study are simple and relate each concept of the global schema to a local concept of a source. Note that all formalisms used (DLR, DL-Lite$_A$, OWL2DL, OWL-DL) are subsets of Description Logics (DL) formalism, that define logics specifically designed to represent structured knowledge and to reason upon.

We notice however that these studies usually concentrate on one specific conceptual model, or do not provide a full methodology in order to integrate database schemas. Most of them do not deal with *SDB*. As most of these proposals use fragments of DL formalism to represent conceptual models, we use this formalism as the basis of our generic framework. We instantiate it then by using *SDB*. We describe in the following section the notion of semantic databases, and present the main semantics of DL formalism.

3 Background

3.1 Semantic Databases

Ontologies are currently extensively used in various domains. As a consequence, a big mass of data referencing these ontologies were produced. These data are called *ontology-based data*. First solutions proposed to manage ontology-based data in *main memory*. These important amounts of data generated needed more efficient management systems. *SDB* have been proposed, presenting a new structure of *DB* storing data and ontologies [9].

Both industrial and academic communities proposed *SDB* solutions like: Ontodb [9], IBM SOR [16] and Oracle [18]. These *SDB* differ according to: (i) the *ontological formalisms* used to define the ontology like RDF, RDFS, OWL, PLIB, FLIGHT, etc. (ii) The *storage schema* of the ontology and of the data model. We distinguish three main *relational* representations: *vertical*, *binary* and *horizontal*. Vertical representation stores data in a unique table of three columns (subject, predicate, object). In a binary representation, classes and properties are stored in different tables. Horizontal representation translates each class as a table having a column for each property of the class. (iii) The *architecture* of the *SDB*: *SDB* can use the same or different schemas for storing the ontology model and the data model, giving rise to different architectures (figure 1).

The first architecture (*type I*) is similar to traditional *DB* using two parts: *data schema part* and the *meta schema part*. In the data schema part, ontology instances (data) and also the ontology model (concepts and properties) are stored, like in Oracle [18]. A second architecture (*type II*) separates the ontology model from its data and is thus composed of three parts: the ontology model, the data schema and the meta-schema (eg. IBM SOR [16]). A third architecture (*type III*) extends the second one by adding a new part which is *the meta-schema* of the ontology like in OntoDB [9].

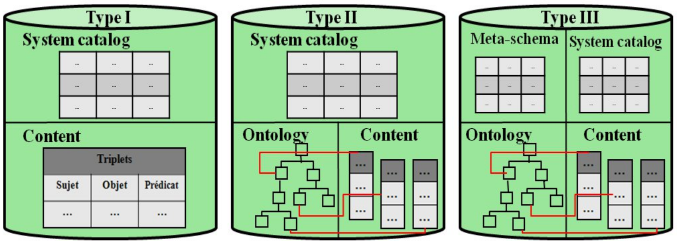


Fig. 1 *SDB* architecture (Type I, II and III)

3.2 Description Logics

Description logics (DLs) are a family of knowledge representation languages [7]. DL are recognized as the formalism able to capture the most popular data *class-based modeling* formalisms presently used in *databases* and *Information system* analysis [7]. It also captures ontological languages like the standard OWL. In DL, structured knowledge is described using *concepts* denoting unary predicates and *roles* denoting binary predicates. Concepts denote sets of individuals, and roles denote binary relationships between individuals. Two types of concepts and roles are used: *atomic* and *complex* concepts (and roles). Complex concepts are defined based on other concepts (atomic or complex) by applying suitable DL *constructors*, equipped with a precise set-theoretic semantics (Exp: negation $\neg A|$, intersection $C \sqcap D|$, value restriction $\forall R.C|$, etc).

A knowledge base in DL is composed of two components: the TBOX (Terminological Box), and the ABOX (Assertion Box). The TBOX states the *intensional* knowledge of the modeled domain. In the most general case, terminological axioms have the form of inclusions: C $\sqsubseteq$ D (R $\sqsubseteq$ S) or equalities: C $\equiv$ D (R $\equiv$ S). The ABOX states the *extensional* knowledge of the domain and defines assertions about individuals, called membership assertions. For example, if we take the small fragment of university model in Figure 3 composed of the two entities *Person* and *Publication* and the relationship *author* between them, it can be formalized using the following axioms (See paper [7]):

$\exists$ author $\sqsubseteq$ Publication (relationship *author* has the entity *Publication* as domain)
$\exists author^{-} \sqsubseteq$ Person (relationship *author* has the entity *Person* as range)
Person $\sqsubseteq \exists$ author (cardinality minimum equal to 1).

4 Generic Integration Framework

The generic framework we propose *DIS*: <Global schema G, local source S, Mapping M>, takes into account new parameters that were not considered before, like the architecture of the database and the storage model of its conceptual model (if stored). We choose description logics formalism as a generic conceptual model. We point out that, as for most of database schema integration, *G* only represents intentional knowledge (the TBOX). The extensional knowledge or instances are stored in local sources *S*.

4.1 The Global Schema $< \underline{G}, S, M >$

The global schema is defined by its conceptual structure that we call *Information Model* (IM). IM is defined as follows IM: <C, R, Ref (C), formalism> : **(1)** *C*: denotes *Concepts* of the model (atomic concepts and concept descriptions), **(2)** *R***:** denotes *Roles* (relationships) of the model. Roles can be relationships relating concepts to other concepts, or relationships relating concepts to data-values (like Integers, Floats, etc), **(3) Ref**: C $\rightarrow$ (Operator, Exp(C,R)). Ref is a *function* defining terminological axioms of a DL TBOX. Operators can be inclusion ($\sqsubseteq$) or equality ($\equiv$). Exp(C,R) is an expression over concepts and roles of IM using constructors of description logics such as union, intersection, restriction, etc (as presented above), **(4) Formalism** is the *formalism* followed by the global model like ER model, UML class diagram, PLIB, OWL, DL-Lite, etc. For example, an OWL-DL ontology can be represented as follows: <Classes, Properties, Ref:<Operator: Inclusion or Equality, *Exp*: expression using SHOIN(D) operators>, OWL-DL>

4.2 The Local Source <G,S,M>

S presents each local and is defined as follows: S: <IM, I, Pop, SMIM, SMI, Ar>. Each local source is defined by its information model (as the global model) if available, and its data part (the instance I) which is mandatory as follows: **(1)** *IM***:** <C, R, Ref, formalism> is the *information model* of the source, **(2)** *I***:** presents the *instances* or data of the source. It would individuals if the source is an ontology, tuples if the source is a relational database, **(3)** *Pop*: C $\rightarrow 2^I$ is a *function* that relates each concept to its instances, **(4)** $\mathbf{SM}_{IM}$: is the *Storage Model* (eg. Relational, Triples, etc) of the information model, **(5)** $\mathbf{SM}_I$: is the *Storage Model* of the instances part I, **(6)** *Ar*: is the *architecture* model of the source, where three architectures type I, II and III are distinguished. Traditional sources follow architecture I. Ontological sources can be constructed using the three architectures (as explained in section 3.1). For example, *relational sources* can be represented by instantiating S as follows: $< \phi$, Tuples, Pop (Query selecting tuples), ϕ, relational, Type I>. *Sor SDB* is represented as follows: <IM(OWL-DL), I, Pop (obtained by a Sparql query), binary relational, horizontal relational, type II >

4.3 The Mappings DIS: <G, S, M>

The mappings are defined between global and local schemas as follows M:< MapSchemaG, MapSchemaS, MapElmG, MapElmS, Interpretation, SemanticRelation, Strength, Type>. This formalization is based on [8] formalization and [2] meta-model defined for conceptual mappings (Note that discovering such mapping is related to the domain of schema and ontology matching/alignment, which is out of the scope of this paper): **(1) MapSchemaG** and **MapSchemaS**: present respectively the *mappable schema* of the global schema and of the local schema. The MapSchemaG is its information model (IM), and MapSchemaS can be either its IM (if available) or its instance part (I), **(2) MapElmG** and **MapElmS**: present respectively the *mappable element* of the global schema and of the local source. This element can be a simple concept or an expression over the schema defined at the intentional or extensional level, **(3) Interpretation**: presents the *Intentional* interpretation or *Extensional* interpretation of the mapping, **(4) SemanticRelation**: present the type of semantic relationship between MapElmG and MapElmS. Three relationships are possible: *Equivalence*, *Containment* (Sound, Complete) or *Overlap*. Equivalence states that the connected elements represent the same aspect of the real world. Containment states that the element in one schema represents a more specific aspect of the world than the element in the other schema. Overlap states that some objects described by the element in the one schema may also be described by the connected element in the other schema [2], **(5) Type**: present the *sense of the mapping*: GaV, LaV or GLaV (can be deduced from the MappableElement).

Mapping between global and local schemas will be defined by a set of mapping assertions following this formalization proposed. For example, [4] *DIS* use the mapping signature M:<IM (DL-lite R), Relational, Conjunctive query expression, SQL query, Sound containment, GLaV>. [12] integration system use the mapping signature M:<IM (OWL-DL), IM (OWL-DL), Conjunctive query expression, Conjunctive query expression, Equivalence or containment or Overlap, GLaV>

In the following case study, we will see the impact of the new parameters considered in our framework, especially the storage model and the architecture of the source.

5 Case Study: Integrating *SDB*

In this section, a case study using *Lehigh University BenchMark* (LUBM) [11] is conducted, based on an industrial *SDB*: Oracle. It adopts a *vertical storage* system and defined two subsets of OWL-DL: OWLSIF and a richer fragment OWLPrime. We use OWLPrime fragment which offer more constructors[1]. The scenario adopted consists on the creation of 3 Oracle *SDB* considered as sources (S1, S2 and S3) and populated locally using Lehigh University BenchMark (LUBM) as illustrated in figure 2. We consider a scenario where a director organism (for example the ministry) imposes the same vocabulary for all universities. Each university refers to the same global schema (LUBM schema), extracts its local schema from this global schema using mappings, and populates this schema locally (using LUBM instances). Each source stores its schema and instances in an Oracle *SDB*. In order to perform some analysis studies, these *SDB* need to be integrated in a *DW* system using *ETL* (Extract-Transform-Load) process.

5.1 Instantiation of the Generic Integration Framework

The generic integration framework <G, S, M> can be instantiated for oracle *SDB* as follows:

- The global schema G: The global schema is formalized by its information model : IM_{Oracle}: <Classes C, Properties P (Datatype Property and Object Property), Ref:(Operator, Expressions), OWLPrime>. Ref is a function over classes

[1] `http://www.w3.org/2007/OWL/wiki/OracleOwlPrime`

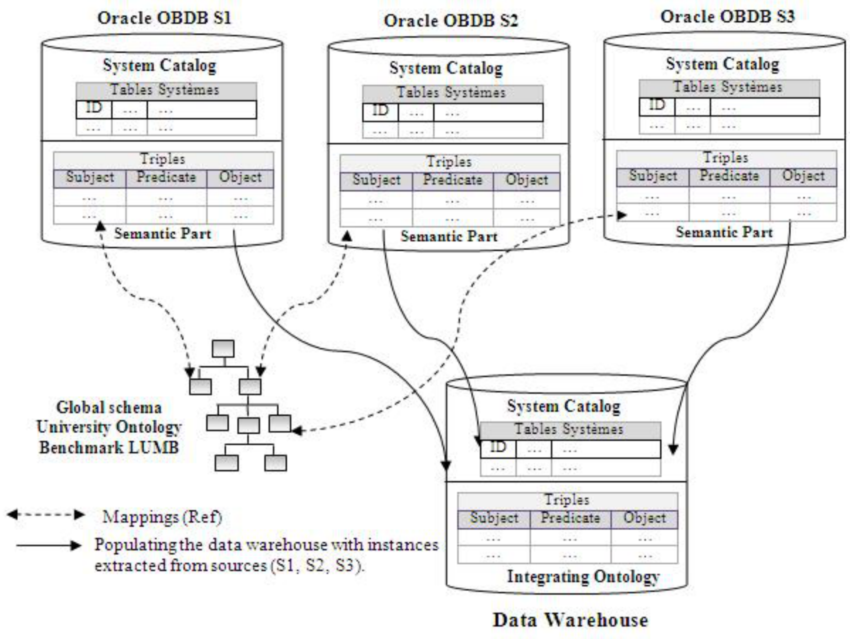


Fig. 2 Integrating Oracle *SDB* using Benchmark LUBM

and properties using OWLPrime constructors. For example: *Ref (University)*= (rdfs:subClassOf, Organization).

- The local source S: Each local source Si is instantiated as follow: $< IM_{Oracle}$, Individuals (instance RDF), Pop is given in tables RDF_link$ and RDF_values$, Vertical, Vertical, type I$>$. Vertical storage is a relational schema composed of one table of triples. For example: (Student, type, Class) and (student#1, type, Student).
- The mappings: The mapping assertions between the global and local schema are instantiated as follows: Mapping M:$< IM_{Oracle}$ of each source, IM_{Oracle} of the global schema, Expression over G, Class from MapSchemaS, Intentional interpretation, Containment (owl:SubClassOf in OWLPrime), GaV$>$. The mappings between each source and the global schema are defined using GaV approach. Each class of a local source is defined as an expression over classes and roles of LUBM global schema.

5.2 Creation of Oracle SDB Sources

We first describe the meaning of the different types of fragments. We tried to draw a parallel between fragments defined in distributed databases to define fragments over ontologies. A ***vertical*** fragment of ontology corresponds to a projection over a set of classes of the ontology. Each class inherits all her properties. A ***horizontal*** fragment

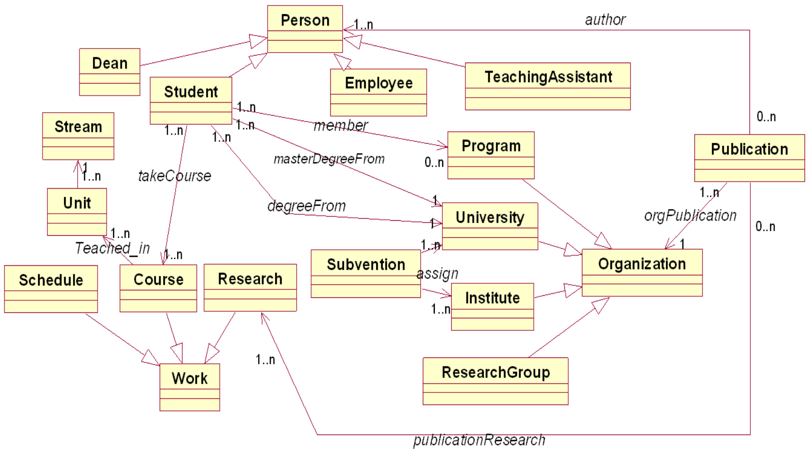


Fig. 3 LUBM global schema

of ontology is a selection of a set of properties for each class of the ontology. The other classes are identical to the classes of the ontology. A ***mixed*** fragment is defined as the process of applying simultaneously horizontal and vertical fragmentation on the ontology.

We have created three Oracle *SDB*, respectively as vertical, horizontal and mixed fragments over LUBM ontology. The sources are populated using instances available in the benchmark LUBM: **(1) Source S1:** is a vertical fragment over LUBM ontology containing three classes: Person, Student and GraduateStudent. **(2) Source S2:** is a horizontal fragment over LUBM ontology containing all its classes.A projection on the class person is done on the properties : Age, EmailAdress. **(3) Source S3:** is a mixed fragment over LUBM ontology containing three classes : Person, Student, GraduateStudent. A projection is done on two properties of the class Person: Age, EmailAdress.

The goal of the integration process is to define an *integrating ontology* (Schema + Instances) that covers all the *SDB*. According to user's requirements, one can define the schema of the integrating ontology (IO) from the schema of the global ontology (G). Three scenarios can be defined: (1) $IO = G$: G corresponds exactly to user's requirements, (2) $IO \subseteq G$: the IO is extracted from G using modularity methods, (3) $IO \supseteq G$: G does not fulfill the whole users' requirements. The designer extracts the fragment of the G corresponding to requirements and can locally enrich it with new concepts and properties. Once the schema of IO defined, we need to populate it with available instances extracted from sources using ETL process.

5.3 ETL Process

For transforming record-sets from sources to target datastore (the *DW*), we need to define an ETL process. [17] defined six generic types of *conceptual opera-*

tors typically encountered in an ETL scenario, which are: (1) **EXTRACT(S,E)**: used to identify elements E (Class, Property) of the source schema S from which data should be extracted, (2) **RETRIEVE(S,C)**: retrieve instances of the class C from the source S; (3) **MERGE(S,I)**: used to merge instances belonging to the same source, (4) **UNION (C,C')**: used to merge instances whose corresponding classes C and C' belong to different sources S and S' respectively; (5) **JOIN (C, C')**: used to combine instances whose corresponding classes C and C' are related by a property, (6)**STORE(S,C, I)**: represent the loading of the instances I corresponding to the class C in the target data store S. We can use the operators described above to populate our target datastore, following Algorithm 1.

```
begin
    Input:
    1. IO: The Integrating Ontology
    2. Si: Local source i (SDB)
    Output: The integrating ontology IO (schema + instances)
    for Each C : Class of ontology IO do
        $I_i = \phi$
        for Each source Si do
            if C exists in Si then
                C'= EXTRACT(Si,C) /*identify the element (class, property)
                from sources*/ if classes have the same super class then
                    Ii= Union (Ii, MERGE (Si,RETRIEVE(Si,C'))) /*retrieve
                    instances associated to C and merge them. Then unites
                    these instances with other instances identified from other
                    sources*/
                end
                if classes are related by same property then
                    Ii= JOIN (Ii, MERGE (RETRIEVE(Si,C'))) /*retrieve
                    instances associated to C and merge them. Then join these
                    instances with other instances identified from other
                    sources*/
                end
            end
        end
        STORE(IO, C, Ii) // store instances related to the class C on the
        ontology IO
    end
end
```

Algorithm 1. Algorithm for populating target datastore using conceptual ETL operators

The result of this algorithm is an integrated ontology populated with data selected from the three *SDB* sources S1, S2 and S3 participating in the integration process. The ETL operators used are generic and need to be instantiated according to our case study. Each operator will correspond to some Sparql queries used by oracle. For example, the conceptual operator UNION is translated to the following Sparql query:

```
PREFIX univ-bench-O1:<http://www.lehigh.edu/~zhp2/2004/0401/univ-bench1.owl#>
PREFIX univ-bench-O2:<http://www.lehigh.edu/~zhp2/2004/0401/univ-bench2.owl#>
SELECT ?x  ?y  WHERE
{{?x univ-bench-O1:takesCourse ?y}UNION {?x univ-bench-O2:takesCourse ?y}}
```

5.4 Results of Data Sources Integration

The result of data sources integration is a *DW* with a conceptual model corresponds to the integrating ontology *IO*. In our scenario, we considered the case where the global schema fits exactly to the users' requirements ($IO = G$). The conceptual model of the resulting *DW* corresponds to the *ontology university* of the benchmark LUBM (Fig. 3). We can already store this IO in an Oracle *SDB*. The final *DW* must obviously organize its data following the multidimensional paradigm. We gave here only the initial *DW* schema. The other design steps have been studied in previous papers [14], but are out of the scope of this paper. We can notice from this experimentation that a *conceptual integration process* frees the designer from all implementation issues. The only effort that must be done is to translate the conceptual ETL operators according to storage schema of the ontology, the rest of the integration is automatic, and the final *DW* model can be deployed using different logical platforms (multidimensional, relational, etc).

6 Conclusion

The relational logical model *reigns* for four decades in the *DB* source storage. As consequence, the proposed data integration solutions were proposed according to this model. Recently, *DB* undergone considerable evolutions, by handling new types of *DB* such as *SDB*, *Cloud Column-Oriented DB*, where each type may support various *architectures* and *storage models*. Note that a *SDB* stores their own ontology (the conceptual model) in the same repository corresponding to the data instances. This conceptual model is then translated to the logical model supported by the target *SDB*. After studying literature related to *DIS*, we noticed that *conceptual schema integration* is the best way to ensure the construction of an integration system independent of all implementation decisions. Ontologies have been extensively in this way, where sources participating to the integration process are related to one or more ontologies in order to solve syntactic and semantic conflicts, and to ensure automatic integration. We first proposed a *generic conceptual framework* for *DIS* that we instantiated using Oracle *SDB*, as they present perfect candidate for conceptual integration. The case study treated uses LUBM benchmark schema and instances, and is conducted following an iterative process starting with the creation of data sources, the mapping with the shared ontology, the generation of the integrating ontology by selecting a fragment from the shared ontology corresponding to the users' requirement and then populating the ontology with the corresponding instances extracted from Oracle sources using ETL operators.

As perspectives, we first project to study other architectures of *SDB* (type II and type III) and their behaviour in an integration process, and the impact of this integration in the whole *DW* design.

References

1. Bellatreche, L., Nguyen Xuan, D., Pierra, G., Dehainsala, H.: Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. Computers in Industry Journal Elsevier 57(8-9), 711–724 (2006)
2. Brockmans, S., Haase, P., Serafini, L., Stuckenschmidt, H.: Formal and Conceptual Comparison of Ontology Mapping Languages. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 267–291. Springer, Heidelberg (2009)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M.: Data Integration through $DL\text{-}Lite_{\mathscr{A}}$ Ontologies. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 26–47. Springer, Heidelberg (2008)
4. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini Rosati, R., Ruzzi, M.: Using owl in data integration. In: Semantic Web Information Management, pp. 397–424 (2009)
5. Calvanese, D., Giacomo, G., Lenzerini, M.: A framework for ontology integration. In: SWWS, pp. 303–316 (2001)
6. Calvanese, D., Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. Int. J. Cooperative Inf. Syst. 10(3), 237–271 (2001)
7. Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In: Logics for Databases and Information Systems, pp. 229–263 (1998)
8. Catarci, T., Lenzerini, M.: Representing and using interschema knowledge in cooperative information systems. Int. J. Cooperative Inf. Syst. 2(4), 375–398 (1993)
9. Dehainsala, H., Pierra, G., Bellatreche, L.: OntoDB: An Ontology-Based Database for Data Intensive Applications. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 497–508. Springer, Heidelberg (2007)
10. Goasdoué, F., Karanasos, K., Leblay, J., Manolescu, I.: View selection in semantic web databases. PVLDB 5(2), 97–108 (2011)
11. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. Journal of Web Semantics, 158–182 (2005)
12. Haase, P., Motik, B.: A mapping system for the integration of owl-dl ontologies. In: IHIS, pp. 9–16 (2005)
13. Halevy, A.Y., Ashish, N., Bitton, D., Carey, M.J., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise information integration: successes, challenges and controversies. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 778–787 (2005)
14. Khouri, S., Bellatreche, L.: A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In: DOLAP 2010, pp. 19–24 (2010)
15. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS, pp. 233–246 (2002)
16. Lu, J., Ma, L., Zhang, L., Brunner, J.S., Wang, C., Pan, Y., Yu, Y.: Sor: A practical system for ontology storage, reasoning and search. In: VLDB, pp. 1402–1405 (2007)
17. Skoutas, D., Simitsis, A.: Ontology-based conceptual design of etl processes for both structured and semi-structured data. Int. J. Semantic Web Inf. Syst. 3(4), 1–24 (2007)
18. Wu, Z., Eadon, G., Das, S., Chong, E., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an inference engine for rdfs/owl constructs and user-defined rules in oracle. In: ICDE, pp. 1239–1248 (2008)

Adaptation of Asymptotic Trust Algorithm

Reputation on Internet Forums

Konrad Leszczyński and Maciej Zakrzewicz

Abstract. Internet is a great medium that allows us to share knowledge with people from all over the world. On the other hand, information from the internet is unreliable because one cannot tell whether it was provided by an expert or a novice. In many cases it does not matter because everybody has the right to his/her opinion about for example a video clip. However there are situations when the credibility of the information is vitally important. Often one wants to find a solution to an important problem, therefore she/he needs some method to ignore jokes, spam and wrong answers by unexperienced users and consider only credible answers. We have designed Asymptotic Trust Algorithm (ATA) to manage users' reputation in customer-to-customer environment like auction sites. In this paper we present an adaptation of ATA which allows to managed reputation of users of internet forums. We belive that our method will be useful for every internet forum which cares about reliability.

Keywords: reputation system, trust management method, internet forums.

1 Introduction

There are many internet forums where reliability of users is vital. In this paper we will refer to forums for software developers as an example but our algorithm may be applied as well to any kind of forums where people ask and answer questions (for example `answers.yahoo.com`[18], `wiki.answers.com`[17] or `allexperts.com`[14]). In case of forums for software developers like

Konrad Leszczyński · Maciej Zakrzewicz
Poznań University of Technology,
Piotrowo 2, 60-965 Poznań, Poland
e-mail: `{konrad.leszczynski,maciej.zakrzewicz}@cs.put.poznan.pl`

stackoverflow.com[16] it is very important to know if proposed solution will solve one's problem or make her/him waste time.

Especially when the subject of the forum is important, users need a way to ignore information written by spammers or so called "trolls" (users who post silly information to make a joke or to annoy others). Most of the internet forums do not use any reputation mechanisms or use very simple mechanisms based on a vote count. On the one hand internet users are used to simple clicking "like" or "dislike" (vote up or vote down) to assess forum posts. This way of giving feedbacks is probably most common in the internet. On the other hand simple sum or average of feedbacks is unsufficient in many ways. In this paper we present *Asymptotic Trust Algorithm (ATA)*, which is a method that exploits this modest feedbacks but still provides readers with a lot of information about writers and their posts. Information that may be useful to quickly and accurately assess users' reputation and usefulness of posts. What is more, ATA allows to show the whole history of a user's reputation on a single chart. One picture speaks a thousand words, so a chart with reputation history of a writer allows readers to see if the writers posts are getting better (she/he is getting more experienced and get more positive feedbacks) or reputation is falling (because same users may stop warring about feedbacks once they gained reputation). We also pay some consideration to different methods of drawing reputation history charts. We realised that showing reputation on a time scale allows readers to see how much time a writer needed to get her/his reputation and whether she/he was active recently or stops visiting the forum.

2 Related Work

Research shows that reputation systems are very important in most Web 2.0 multi-agent applications (to name just a few: [6, 2, 3, 10]). Many trust models and reputation systems have been proposed. HISTOS[12], SPORAS[12] and REGRET[8] are probably most cited. Another interesting trust models, that also give good results in predicting future users' behaviour are FIRE[4], AFRAS[7] and RRM[13]. For more information about many types od reputation systems see [9].

These trust models provide good estimation of users' reputation, unfortunately they are often difficult to introduce to existing portals. Moreover, users like the simplicity of voting up and down. Methods with a lot of parameters (like for example [11]) or sophisticated probabilistic methods (like The Beta Reputation System [1]) may confuse user, therefore internet sites' owners are hesitant of introducing more sophisticated reputation management methods.

Our intention is to build a trust model which is easy to comprehend for users and keeps the simple interaction method (voting for post up or down) just like we build a trust model for online auction sites which extend but not replace eBay's reputation system [5].

3 Algorithm to Manage Reputation of Internet Users

Our method initially designed for auction portals can easily by applied to any Web 2.0 portal. In this paper we present an adaptation of ATA to work with forums where users ask and answer questions. In this section we describe the way it works and compare it briefly to the auction portal's version.

3.1 Desirable Properties of the Reputation System

We would like to create a trust mechanism inspired by people's behaviour in the real live. ATA is based on the following principles:

- Easy to use and with simple interaction mechanism. Users may simply vote up or down marking each post as good (i.e. interesting, useful, helpful) or bad.
- The reputation of a user who gets only positive feedbacks asymptotically approaches the value of 1 (maximum 100% reputation value). Also the reputation of a user who constantly gets negative marks asymptotically approaches the value of 0.
- Old ratings are less important. Users are gaining experience and we do not want to punish them for mistakes they have made being beginners. Also we cannot allaw users to rest on their laurels and stop writing well-thought-out answers.
- The value of reputation varies from 0 to 1. Values are easy to compare and the reputation can be represented as a percentage value (understandable for users), or even acts as a rough estimator of probability that the user will answer correctly next time.
- Answers to difficult questions should change reputation more.
- The mechanism should be reasonably simple, we would like to avoid complex graph algorithms and intricate probabilistic methods to make our algorithm easy to understand by users.

3.2 Reputation in Online Auction Portals

In eBay-like auction portals users give feedbacks (positive, neutral or negative) after each finalised transaction. The reputation of an eBay's user is the number of positive feedbacks he/she gets minus the number of negative ones. Also internet forums sometimes provide some way to rate posts by voting them up or down. And also reputation of a user is some kind of sum or average of positive and negative votes. This feedback count system is unsufficient in many ways but users like the simplicity of use and straightforward interaction method. In [5] we proposed a trust algorithm based on desirable properties analogous to properties we want to achieve now (see above 3.1).

In a nutshell ATA for auction portals calculates reputation R_i after i-th interaction (i-th transaction of the user) using the following formula:

$$R_i = \begin{cases} R_{i-1} + ((1 - R_{i-1}) * F(p_i) * \alpha) & \text{for positive feedback} \\ R_{i-1} & \text{for neutral feedback} \\ R_{i-1} - (R_{i-1} * F(p_i) * \alpha) & \text{for negative feedback} \end{cases} \quad (1)$$

Where $F(p)$ is a function of price and α is a scaling factor.

3.3 Reputation in Internet Forums

To manage reputations of users of internet forums we have needed to make certain adaptations to ATA. Of course in forums there is nothing like transaction between two users. Users only ask and answer questions, but they also can rate answers. Many forums already allow users to click "like" or "vote up" and "dislike" or "vote down" to express their opinion about each posted answer. For example in case of stackoverflow.com[16] the user who ask question may accept/approve certain answers. Usually other people may have similar problems (or want to join the discussion to help or to practice their own skills) and they also may vote for good and bad answers. This is all the interaction we need, every vote for a user's answer is a single interaction. ATA calculates user's reputation after i-th interaction (every time someone finds the user's posted answer useful and votes up or finds it misleading and votes down).

The second difference between auctions and forums is price, answers do not have prices. However, answers are related to questions which may be difficult or simple. In this version of algorithm we have replaced the function of price $F(p)$ with *The Function of Question Difficulty* $F(q)$. It is of course impossible to decide objectively which questions are more difficult, therefore we propose an estimation. We based the estimation on the assumption that unexperienced users often ask simple question whereas when a professional needs to ask a question on the forum it is probably a difficult one. The $F(q)$ is therefore equal to the value of reputation of the user who had asked the question. The scaling factor α which is responsible for suddenness of reputation changes (the higher α the faster the reputation value will change) is chosen arbitrarily from the range $(0, 1]$ and is constant for the reputation system.

To meet our requirements (see 3.1) we have designed the following algorithm. To calculate the reputation value after i-th interaction we use the equations below.

$$R_i = \begin{cases} R_{i-1} + ((1 - R_{i-1}) * F(q) * \alpha) & \text{for vote up} \\ R_{i-1} - (R_{i-1} * F(q) * \alpha) & \text{for vote down} \end{cases} \quad (2)$$

The previous reputation value R_{i-1} is increased by a fraction $(F(q) * \alpha)$ of the complementary reputation $(1 - R_{i-1})$ in case of positive feedback but in case of negative feedback the reputation value is decreased by $(F(q) * \alpha)$.

In our implementation the reputation value of a newcomer $R_0 = 0.1$. R_0 can be different but should be reasonably low because a high value of initial reputation may encourage dishonest people to create a new account after committing losing reputation.

3.4 Reputation of Posted Answers

The posted answers also need to have some kind of reputation value, a value that represents its potential *usefulness*. After all, the whole point is to allow users to quickly decide which answer may present the best solution and is potentially most useful and helpful. Of course next to each answer we can present the value of reputation of the user who posted the answer. Most internet forums show brief information about users next to their posts. This is often unsufficient and we want to allow even the very beginners to stand out and present their ideas. We introduce *The Value of Usefulness U* and we use the same ATA method to calculating it as in case of users' reputation. Every vote up or down related to the posted answer is an interaction that changes *The Value of Usefulness* and we use the following formula to calculate it:

$$U_j = \begin{cases} U_{j-1} + ((1 - U_{j-1}) * F(q) * \alpha_a) & \text{for vote up} \\ U_{j-1} - (U_{j-1} * F(q) * \alpha_a) & \text{for vote down} \end{cases} \qquad (3)$$

Where j is the number of votes related to the answer. Note that the number of interactions related to a user (i in the Formula 2) is the sum of interactions related to all his/her answers (j in the Formula 3) and that every vote invokes both calculations for R_i and U_j independently. In the formula for U_j (Formula 3) we use the same *Function of Question Difficulty* $F(q)$ ($F(q)$ is equal to the value of reputation of the user who had asked the question) as in the formula for R_i (Formula 2) but the scaling factor α_a is different than α. Single answers get few or several votes therefor their usefulness must change faster and the scaling factor for answers should be higher $\alpha_a > \alpha$. Newly posted answers need to have some *Value of Usefulness* before anyone vote it up or down. Since it may be posted by an expert or a beginner, the initial value should depend on reputation of the user who posted the answer, therefore initially *Value of Usefulness* equals the reputation of a user who posted the answer $U_0 = R$.

On purpose in calculations we do not use the value of reputation of the user who votes for the answer. Users who vote for answers are supposed to use the solution and validate it. In case of forums for software developers the unexperienced users usually seek for solution and mostly they will be happy finding a particular answer for a problem. We do not want to obligate the most experienced users to become judges and moderators of the forum. We also do not want to allow dishonest users to use their heigh reputation to raise reputation of their friends to quickly.

4 Experimental Evaluation

This section provides some examples to illustrate how the algorithm works. To verify if ATA meets our expectations we performed a series of experiments. For the purpose of these experiments we have developed a piece of software in C# which simulates users' behaviour using random number generator. We have tried different ways of presenting reputation history because we belive that a small chart shown next to a posted answer may provide additional information. In this section we also show a screenshot of a forum that implements ATA.

4.1 Users' Reputation Growth

Let us consider a system with the following parameters: $\alpha = 0.1$, $R_0 = 0.1$. That low value of the scaling factor α ensures smooth growths and degradations of reputations. In other words, one mistake will not ruin reputation and a single good answer will not make anyone an expert. Reputation for a newcomer R_0 must be very low (to discourage users from creating new accounts after losing some reputation). Also R_0 should not be equal to 0 since it is used to calculate reputation growths for other users, because reputaion of a user who is asking the question is used as $F(q)$ in Formulas 2 and 3.

Figure 1 presents reputation growths of three users. The interactions (on the X-axis) are votes for users' answers, every vote is given to an answer to a random question (random $F(q)$). In this experiment we used random number generator to decided which votes/feedbacks are positive. $R1$ represents reputation changes of the least experienced user - probability that this user's answer gets positive feedback is $p = 0.33$. $R2$ is the reputation of a user who gets more votes up than votes down (probability that this user's answer gets positive feedback is $p = 0.66$). $R3$ is a reputation of an expert whose answers are always useful and correct (probability that this user's answer gets positive feedback is $p = 1$).

As we can see, the reputation of the user who gets only positive feedbacks asymptotically approaches the maximum value of 1. As intended reputations of users who get both votes up and votes down "oscillate" around values respective to positive/negative votes ratio.

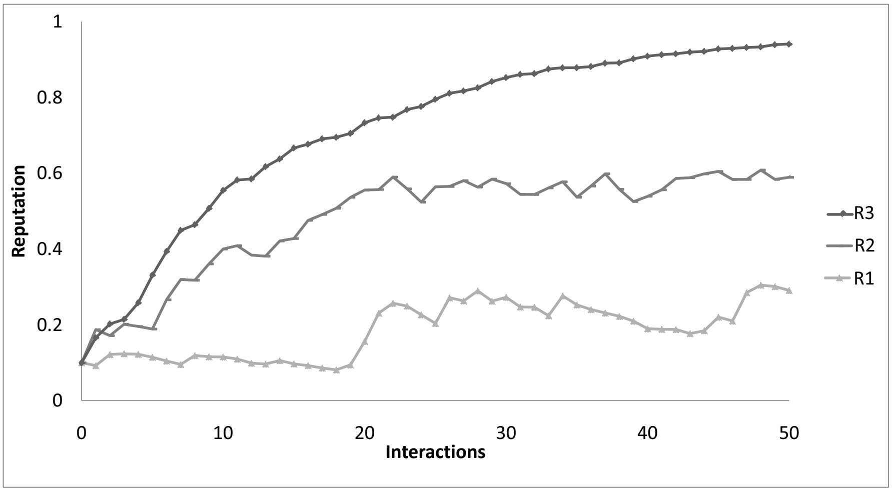


Fig. 1 History of reputation of 3 different users

4.2 Usefulness of Answers

Figure 2 shows changes of usefulness of three answerers. In this experiment we set the scaling factor $\alpha_a = 0.3$. Let us consider a rather experienced user who already has the value of reputation $R = 0.7$. This user posts three answers (*Function of Question Difficulty* of each questions is $F(q) = 0.5$).

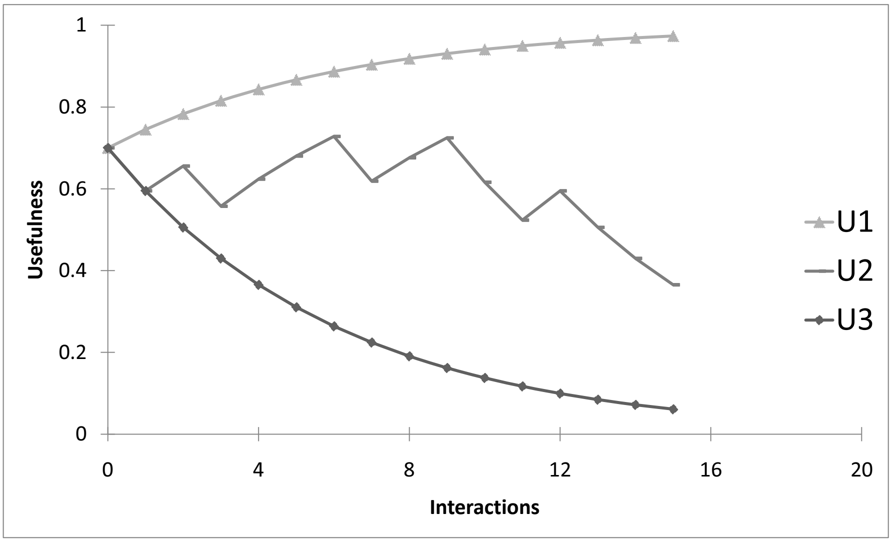


Fig. 2 Answers' usefulness changes

The first answer was very helpful so it gets only positive feedbacks (all votes for this answer are up), therefore its *Value of Usefulness* $U1$ grows quickly. Second answer gets both votes up and down, so $U2$ is changing both in plus and in minus. Third answer turns out to be misleading, so it get only votes down. As intended, wrong answer quickly lose its *Value of Usefulness.* $U3$ starts from rather heigh value (when it was posted reputation of the user was $R = 0.7$) but drops considerably just after a few votes. This is a clear sign to the user who posted this answer to review her/his ideas and correct or remove the answer.

4.3 Presentation in the Time Scale

We can present the whole history of user's reputation on a single chart. Interesting idea is to use time scale instead of interaction number on the X-axis.

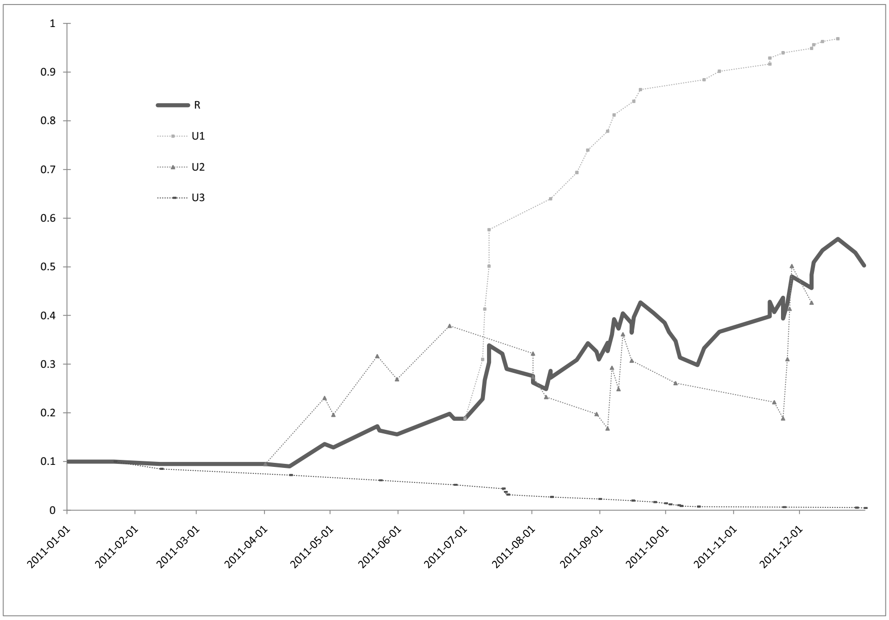


Fig. 3 Reputation of a user and usefulness of her/his answer in time scale

Figure 3 shows a complex history of a user's feedbacks. On a single chart we were able to show both history of her/his reputation and history of others opinions about each of the user's answers. On this chart we can see every event related to this user.

In this experiment we simulate a user who posted three answers in random moments of time. Ones posted, answers get feedbacks in random moments in time.

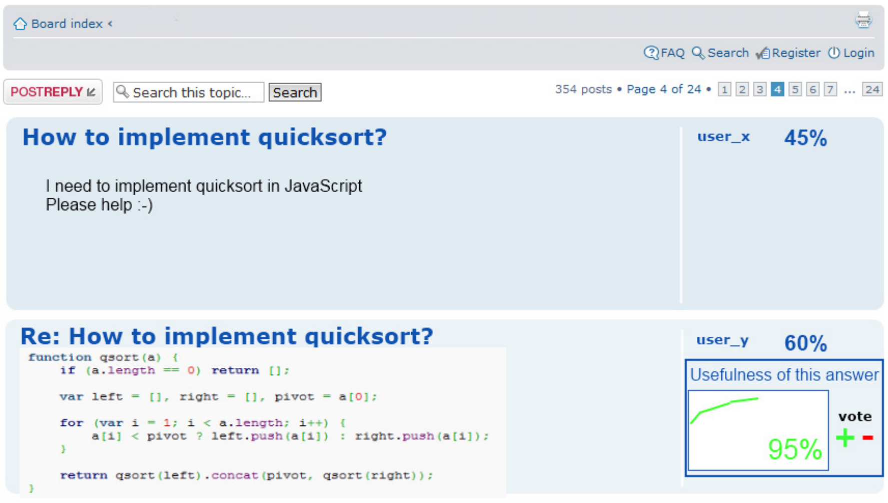
- Answer 1 is represented by the chart $U1$ and gets only positive feedbacks. Answer 1 was posted on the first of July.
- Answer 2 – chart $U2$, gets both positive and negative feedbacks (randomly). Answer 2 was posted on the first of April.
- Answer 3 – chart $U3$, gets only negative feedbacks. Answer 3 was posted on 22th of January.

In each case $F(q) = 0.5$. Every interaction (i.e. every vote) is represented as a marker on the chart.

We can see that every vote changes the Value of Usefulness of related answer and also changes (but to a lesser extend) overall reputation value – as intended. Charts similar to this may be shown on users' details pages of forums. This method of data presentation shows not only the reputation values changes but also shows when a user has posted an answer and when the answer was assessed. At a glance we can say when the user was most active and whether her/his answers are getting better or worse.

4.4 Real Dataset Experiments

Currently we want to gather data by using phpBB framework [15] to create a forum that implements ATA to manage reputation. Then we will be able



How to implement quicksort?

user_x 45%

I need to implement quicksort in JavaScript
Please help :-)

Re: How to implement quicksort?

user_y 60%

```
function qsort(a) {
    if (a.length == 0) return [];

    var left = [], right = [], pivot = a[0];

    for (var i = 1; i < a.length; i++) {
        a[i] < pivot ? left.push(a[i]) : right.push(a[i]);
    }

    return qsort(left).concat(pivot, qsort(right));
}
```

Fig. 4 Screenshot of the forum that uses ATA to manage users' reputation and answers' usefulness

to conduct experiments on real data. Figure 4 shows the prototype of how ATA fits in the standard phpBB layout. Next to users' names there are values of their reputations in percentage terms. To see the chart with detailed history of reputation one must visit the user's info page. The answer's usefulness is presented in percentage terms and in addition a small graph with usefulness history is shown. Buttons "+" and "-" allow users to vote the answer up or down. Currently we are gathering data from this long-term experiment.

5 Conclusion

In this article we have shown that Asymptotic Trust Algorithm (ATA) can be easily use in internet forums. ATA is a trust management system that gives good estimation of reputation of users and their posts. At the same time ATA does not need to change the way users interact by clicking "likes" and "dislikes". Users are used to this simple way of interaction and it is much easier to extend existing reputation system instead of replacing it completely. ATA is designed in such a way that it can coexist with the commonly used reputation systems simultaneously. It provides users with more detailed information about each other which allows better judgement.

Reputation is not earned forever but one has to constantly look after it like in "real life" human relationships. In our system it is easy to spot when a user starts posting unproven solutions. Negative feedbacks should encourage users to review their posted solutions (of course a forum must allow users to delete or change her/his wrong post).

References

1. Commerce, B.E., Josang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Electronic Commerce Conference (2002)
2. DeFigueiredo, D., Barr, E.T., Wu, S.F.: Trust Is in the Eye of the Beholder. In: CSE, vol. (3), pp. 100–108 (2009)
3. Gómez, M., Carbó, J., Earle, C.: Honesty and trust revisited: the advantages of being neutral about others cognitive models. Autonomous Agents and Multi-Agent Systems 15, 313–335 (2007), http://dx.doi.org/10.1007/s10458-007-9015-8, doi:10.1007/s10458-007-9015-8
4. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: Fire: An integrated trust and reputation model for open multi-agent systems. In: Proceedings of the 16th European Conference on Artificial Intelligence, ECAI, pp. 18–22 (2004)
5. Leszczyński, K.: Asymptotic Trust Algorithm: Extension for reputation systems in online auctions. In: KKNTPD 2010 - III Krajowa Konferencja Naukowa Technologie Przetwarzania Danych (2010)
6. Malaga, R.A.: Web-Based Reputation Management Systems: Problems and Suggested Solutions. Electronic Commerce Research 1, 403–417 (2001)

7. Rubiera, J.I.C., Molina, J.M., Muro, J.D.: Trust management through fuzzy reputation. Int. J. Cooperative Inf. Syst. 12(1), 135–155 (2003), doi: http://dx.doi.org/10.1142/S0218843003000681
8. Sabater, J., Sierra, C.: Regret: reputation in gregarious societies. In: Proceedings of the Fifth International Conference on Autonomous Agents, AGENTS 2001, pp. 194–195. ACM, New York (2001), doi: http://doi.acm.org/10.1145/375735.376110
9. Schlosser, A., Voss, M., Brückner, L.: On the simulation of global reputation systems. Journal of Artificial Societies and Social Simulation 9 (2005), http://ideas.repec.org/a/jas/jasssj/2005-13-2.html
10. Turilli, M., Vaccaro, A., Taddeo, M.: The case of online trust. Knowledge, Technology & Policy 23, 333–345 (2010), http://dx.doi.org/10.1007/s12130-010-9117-5, doi:10.1007/s12130-010-9117-5
11. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Transactions on Knowledge and Data Engineering 16, 843–857 (2004)
12. Zacharia, G., Maes, P.: Trust management through reputation mechanisms. Applied Artificial Intelligence 14(7), 881–907 (2000)
13. Zhang, H., Duan, H.X., Liu, W.: RRM: An incentive reputation model for promoting good behaviors in distributed systems. Science in China Series F: Information Sciences 51(11), 1871–1882 (2008)
14. All Experts Questions and Answers. A directory of subjects with volunteer experts to answer posted questions, http://www.allexperts.com/
15. phpbb. PhpBB is a free flat-forum bulletin board software solution, http://www.phpbb.com/
16. Stack Overflow. A language-independent collaboratively edited question and answer site for programmers, http://stackoverflow.com/
17. WikiAnswers. Questions and answers in a shared, wiki format, http://answers.com/
18. Yahoo! Answers. Yahoo! Answers is a place where people ask and answer questions on any topic, http://answers.yahoo.com/

PropScale: An Update Propagator for Joint Scalable Storage

Paweł Leszczyński and Krzysztof Stencel

Abstract. In the era of Web 2.0 and the apparent dawn of Web 3.0 web pages are dynamic and personalized. As the result, the load of web servers rapidly increases. Moreover, the upcoming load boost is impossible to predict. Although deceptively funny, the term of *success-tolerant* architectures has been coined. A number of web services actually failed *because of* their initial success. In order to achieve success-tolerance the server architectures must be scalable. Nowadays almost all components of systems can certainly be multiplied. The only exception is the storage constituent. The usual solution with one strong relational database is unsatisfactory. Thus, designers introduce additional (NO)SQL storage facilities. From this point one has a number of separate data sources that can apparently get inconsistent with each other. Special software must be developed to synchronize them. This means more bugs to fix, more code to maintain and more money to spend. In this paper we present a new technique to introduce a number of non-homogenous storage units into a system. The solution consists of an algorithm that propagates updates among disparate (NO)SQL storages built into a system.

Keywords: multi storage, key-value storage, scalability, data consistency, web applications.

1 Introduction

Traditional databases do not scale well. This problem is well-recognized by the industry. When a database workload increases, it is a common practice to split the

Paweł Leszczyński
Allegro Group, ul. Marcelińska 90, 60-324 Poznań
e-mail: pawel.leszczynski@allegro.pl

Krzysztof Stencel
Faculty of Mathematics and Computer Science Nicolaus Copernicus University,
Chopina 12/18, 87-100 Toruń
e-mail: stencel@mat.umk.pl

database into smaller parts, and distribute it onto several servers. However, this is rather an ad-hoc fix and not a scalable architecture. The other option is to migrate some data into scalable storages. For this purpose one can apply a local NoSQL storage, or use external ones like Amazon S3, Amazon SimpleDB, or others. Whatever solution has been chosen, the database is eventually split into several smaller instances running on different storage engines and servers. This however, makes the overall system architecture more complicated. Thus the application gets harder to maintain and the whole development process is more expensive. Sometimes the same data are stored in several locations, and the application's logic needs to keep the replicated data in the consistent state. This requires developers to take care of all data writes and apply them on several storages. When dealing with big applications, this can lead to errors that are hard to detect and repair.

In this paper, we describe *PropScale*, a novel data propagation system for joint storages that maintains replicated data in multiple sources in the consistent state. Figure 1 shows its architecture. It also allows storing redundant and pre-computed statistical data for fast access. Imagine a product page in an e-commerce application that displays the number of customers who bought a certain product. The read access to this counter must be fast. Thus the storage serving should scale well. On the other hand it is a usual business requirement to store financial data in a relational database. This count is a simple stastistic based on a *one-to-many* association between two relations (a product and its sold items). *PropScale* allows statistics like *count*, *sum*, *avg*, *min* and *max*. The last two may be useful when storing the last customer who bought a certain product. All such statistics can be outsourced to several storages. When an update on a data source occurs, our system modifies data in other storages, if it is needed. The proper update propagation on underlying storages allows constructing a scalable joint storage with all advantages of underlying storages. The paper makes the following contributions. (1) We introduce a novel architecture to build several storages into a system. (2) We present the experimental evaluation of applying its proof-of-concept implementation *PropScale* in real world scenarios.

The paper is organized as follows. In Section 2 we address the related work. Section 3 presents the update propagation algorithm, while in Section 4 we present experimental evaluation of performance of *PropScale*. Section 5 concludes.

2 Related Work

Several publications address scalability and consistency. The paper [1] describes design choices and principles for a scalable storage. The authors address the gap between key-value and relational storage. This gap is investigated in our research.

An interesting, ongoing research is a modular cloud storage system called Cloudy [9]. It is built on the top of different storage engines similarly to our system. Cloudy provides interfaces for read and write operations. This makes underlying storages invisible for an application server and is a clear design pattern. In our system we only care for updates and assume that retrievals are done by individual component

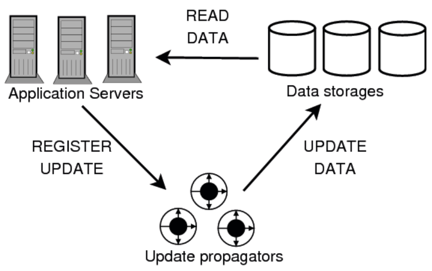


Fig. 1 *PropScale* architecture

data sources (cf. Fig. 1). Cloudy also integrates retrievals. This means a significant increase in its complexity and cost of maintenance when one takes into account that numerous NoSQL storages are being rapidly developed. These NoSQL storages provide plenty of API clients like JSON, XML, THRIFT [11] etc. and rewriting all of them is almost impossible.

The general problem we consider is keeping data consistent in different storages. The authors of [12] claim that managing a heterogeneity of data sources in a multidatabase systems is a hard problem, and as one of solutions, restricting database functionality is given. This also happens in *PropScale*, however we managed to restrict only the functionality of data writes and not retrievals. This is a significant improvement. We have also examined possible solutions of similar problems, like consistent caching. Authors of [7, 6, 10] present a model that detects potential inconsistency based on statements' templates. However, their approach cannot handle joins of attribute families or aggregations that are common in web applications. Our approach is based on a graph with edges that determine the impact of update operations on the cached data. The idea of the graph representation has been presented in [2, 4, 5]. The vertices of the graph are instances of update statements and cached data objects. In our approach the dependency graph also has vertices that represent updates, retrievals and data items. However, they are classes and not individual instances. This means that the complexity of *PropScale* depends only on the count of design items (statements, relations, columns).

3 Propagation Algorithm

3.1 Data Model

Suppose our data consists of relations: $R_1, R_2, \dots, R_k$. Each relation has exactly one primary key element *id*. *One-to-many* associations between relations R and S are denoted by $R \prec_{r_i} S$. Additionally for any two relations R and S, we say that they are associated, $R \lhd S$, if there exist relations $S_1, S_2, \dots, S_i$ and attributes $r_1, r_2, \dots, r_{i+1}$ such that:

$$R \prec_{r_1} S_1 \prec_{r_2} S_2 \dots S_{i-1} \prec_{r_i} S_i \prec_{r_{i+1}} S. \tag{1}$$

One-to-many associations between attributes of the same relation, $R \prec_r R$, are useful to represent hierarchical data.

These rules allow building rich web applications and are frequently obeyed in the industry. Allthough the data architecture of most e-commerce systems obeys these rules, it is stored across different storages and some storages are not relational. We believe that each underlying storage contains a projection of the schema. For each attribute of a schema relation, there exists at least one storage that projects tuples containing this attribute. Moreover some attributes may be stored redundantly in more than one storage. A data item may be stored in a distributed database for fast read accesses when generating item pages. However, its financial data must be stored in a relational ACID database for safe transactional access.

3.1.1 Write Operations

We put some restrictions on updates and retrievals. We assume that each update modifies a single tuple specified by *id* parameter. We distinguish three types of write operations: adding a new tuple, editing a tuple attributes' except for *id* and deleting it. In general case, an update can be represented as

$$(R_U, type, value_{id}, \{(r_i, value_{r_i}), \dots, (r_j, value_{r_j})\}). \tag{2}$$

3.1.2 Underlying Storages

Suppose R is a relation and $S = (S_1, S_2, \dots, S_i)$ is a sequence of relations associated with R, i.e.

$$R \lhd S_1 \wedge R \lhd S_2 \wedge \dots \wedge R \lhd S_i. \tag{3}$$

For each $S_j \in S$ we take relations $S_{j,1}$, $S_{j,2}$, $\dots S_{j,k}$ and attributes r_{j_1}, r_{j_2}, $\dots$, r_{j_k} such that

$$R \prec_{r_{j_1}} S_{j,1} \prec_{r_{j_2}} S_{j,2} \prec_{r_{j_3}} \dots \prec_{r_{j_k}} S_j. \tag{4}$$

Then we define

$$R_{S_j} = S_{j,1} \bowtie_{S_{j,1}.id=r_{j_2}} S_{j,2} \bowtie \dots S_{j_k} \bowtie_{S_{j,k}.id=r_k} S_j. \tag{5}$$

Suppose $r_1, r_2, \dots, r_i$ are attributes of $R, R_{S_1}, R_{S_2}, \dots, R_{S_i}$. We allow projections of the form

$$\pi_{R.id, r_1, r_2, \dots}(R \bowtie_{R.id=r_{1_1}} R_{S_1} \bowtie_{R.id=r_{2_1}} R_{S_2} \dots \bowtie_{R.id=r_{i_1}} R_{S_i}). \tag{6}$$

In other words we allow joins between R and arbitrary number of relations associated with R. We require that the primary key of R is projected and allow arbitrary attributes from $R, R_{S_1}, \dots, R_{S_i}$ to be projected. We call such projection *a safe projection* and R is denoted *the primary relation* of the projection.

Given a *safe projection* Π and a projected attribute r_p, we introduce the *trace* of r_p in Π which determines how r_p is projected. Suppose a projection attribute r_p is projected from:

$$R_{t_1} \bowtie_{R_{t_1}.id=t_2} R_{t_2} \dots \bowtie_{R_{t_{m-1}}.id=t_m} R_{t_m} \tag{7}$$

where $t_1, t_2, \dots, t_m$ are attributes of relations $R_{t_1}, R_{t_2}, \dots, R_{t_m}$ respectively. Additionally $R = R_{t_1}$ is the primary relation of Π, and let r_p be an attribute of a projection Π that projection an attribute t_m of a relation R_{t_m}. Then $tr(\Pi, r_p)$ is defined as:

$$tr(\Pi, r_p) = (R_{t_1}.id, R_{t_1}.t_1, R_{t_1}.id, R_{t_1}.t_2, \dots, R_{t_m}.t_m) \tag{8}$$

Since we allow a single relation attribute to be projected several times, we need to take care of how it is projected. The function tr allows recovering the chain of joins that has been applied to project an attribute. For each projection, we introduce the set of attribute traces that contains all pairs of relation attribute and their *traces*:

$$Trace(\Pi) = \{(r_p, s) : tr(\Pi, r_p) = s\} \tag{9}$$

The underlying storages can also contain processed results of *safe projections*. We allow selection operations on projections' data with some limitations. We require that a tuple stored in a storage can be properly updated based on its current value and an update, According to this, we allow a *count* selection. However, a *sum* selection can be used only in case of projections where no tuples are modified nor removed.

3.2 Dependency Graph

A dependency graph G is a triple $(V, E_{strong}, E_{weak})$ where V is the set of vertices, and E_{strong}, E_{weak} are sets of directed edges. Let

$$A = Attr(R_1) \cup Attr(R_i) \cup \dots \cup Attr(R_k) \tag{10}$$

be the set of all attributes of all schema relations. $Attr(R)$ is the set of attributes in relation R. Let:

$$P = \{P_1, P_2, P_3, \dots\} \tag{11}$$

be the set of all safe projections stored in the underlying storages. We introduce the function: $Map(U) = (R_U, type, \{r_i, \dots, r_j\})$. It maps a write operation so that two updates that perform the same operation on the same attributes are treated as the same entity. Next we define

$$M = \{Map(U_1), Map(U_2), Map(U_3), \dots\} \tag{12}$$

as the set of values of Map for all data modifications. Then the set of vertices V of the dependency graph is the union $A \cup P \cup M$.

Next we define the edges of G. For each $R \prec_r S$, the foreign key $S.r$ is connected by a strong edge with the primary key of its relation $S.id$ and with the primary key of the foreign relation $R.id$. Thus,

$$\{(S.id, S.r), (S.r, R.id)\} \in E_{strong}. \tag{13}$$

Each projection vertex Π is connected by a strong edge with the primary key of its primary relation. The edge goes from the primary key to the projection vertex, i.e. $(R.id, \Pi) \in E_{strong}$. Each projected attribute r is connected by weak edges with Π: $(r, \Pi) \in E_{weak}$. Given a vertex $Map(U)$, it is connected by a strong edge with $R.id$ and by weak edges with all modified attributes, i.e.

$$(Map(U), R_U.id) \in E_{strong} \wedge \forall_{i=1,\dots,j} (Map(U), R_U.r_i) \in E_{weak}. \tag{14}$$

This ends the definition of the dependency graphs. An example G is shown on Figure 2.

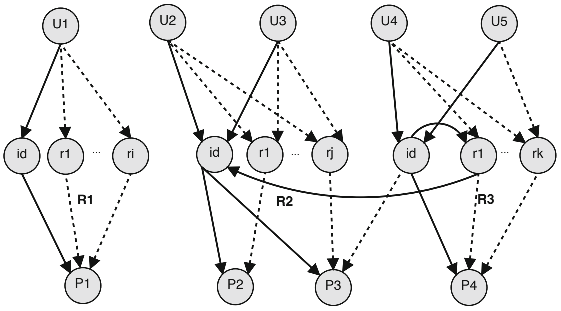


Fig. 2 The figure shows an example of the dependency graph. $U_1, \dots, U_5$ denote update vertices, $R_1, \dots, R_3$ denote schema relations and $P_1, \dots, P_4$ projections stored in underlying systems. We assume $R_2 \prec_{R_3.r_1} R_3$.

3.3 Underlying Storages' Drivers

Now we describe functions implemented in underlying storages' drivers that are used by the algorithm. Given an update U and a *safe projection* Π, we distinguish two functions that add a new tuple, namely *add* and *addPrimary*. The *addPrimary* returns the primary key of the new tuple, while *add* inserts a tuple with a specified primary key. The *modify* function updates a single tuple in the projection Π with the given primary key. Our algorithm also uses the *retrieve* function that returns the value of a projection attribute r_p from the tuple in the projection Π with the primary key equal $value_{id}$. The last function needed is *delete*, which removes the specified tuple from a projection. We assume those functions to be implemented in drivers of underlying storages'. We do not restrict to any database types nor vendors, and only assume a few basic functions that need to be implemented by a storage.

3.4 The Algorithm

When an update U is submitted to the system, it first checks which projections are affected. We denote such a set $Proj(U)$ which is:

$$Proj(U) = \{\Pi \in P : \exists_{r \in A}(Map(U), r) \in E \wedge (r, \Pi) \in E\} \tag{15}$$

Then we investigate elements of $Trace(\Pi)$ that has been defined in (9). Given r_p, we investigate $t = tr(\Pi, r_p)$ such that $(r_p, t) \in Trace(\Pi)$. According to the definition (8), $tr(\Pi, r_p)$ contains a sequence of primary and foreign key attributes, from the primary key of the projection to the attribute r. Additionally we know that each update vertex is connected by a strong edge with the primary key of the modified relation. Moreover, each projection is accessed by a strong edge from its primary relation. As a result, for each t we can construct a strong path $SP(t)$ from the update to the projection vertex. *A strong path* is a path composed solely of strong edges. Then given an update U, we define the $Path(U, \Pi)$ function as a set of all strong paths $SP(t)$ from $Map(U)$ to Π.

Strong paths are used for evaluating the primary keys of modified tuples in the projections. Given values of U, we define $Find(U, \Pi)$ that returns primary keys of modified tuples in Π. Let AV_{t_i} denote a value of an attribute t_i in the modified tuple and suppose $AV_{t_0} = AV_{R_U.id} = value_{id}$. Suppose t_i and t_{i+1} are attributes of relations R_{t_i} and $R_{t_{i+1}}$ respectively. Then $AV_{t_{i+1}}$ is determined as follows:

$$AV_{t_{i+1}} = \begin{cases} retrieve(t_{i+1}, AV_{t_i}), & R_{t_i} = R_{t_{i+1}} \\ AV_{t_i}, & R_{t_i} \neq R_{t_{i+1}} \end{cases} \tag{16}$$

where $retrieve(t_{i+1}, AV_{t_i})$ retrieves a value of an attribute t_{i+1} from a row with the primary key equal AV_{t_i} from a projection that stores a relation attribute t_{i+1}. The algorithm iterates through the path's attributes from $Path(U, \Pi)$, and computes values of the joined tuple attributes until it fetches the attribute $R_\Pi.id$. This is possible since we iterate through strong edges, and in case of connecting attributes, strong edge connects either an attribute with the primary key attribute of the same relation or foreign key attribute with the primary key of the associated relation. As a result $AV_{R_\Pi.id} = AV_{t_m}$.

We have constructed a function, that given a path from U to Π and the primary key of the updated tuple, returns the primary key of the modified tuple corresponding to a strong path between $Map(U)$ and Π. We denote that function $g(U, p)$. When finding all modified tuples, the algorithm examines all paths from $Path(U, \Pi, r)$. Note that from the dependency graph construction, it follows that U modifies an attribute r when $(Map(U), r) \in E$. In general $Find(U, \Pi)$ returns the maximal set that contains pairs of strong paths between update and projection vertices associated with the primary key of the column that has to be modified.

This leads us to a main algorithm section:

1. If U is *add*, let Π be the primary projection of a modified relation and let p be the strong path p containing $(Map(U), R_U.id, \Pi)$. Apply *addPrimary* and append its result to values of U as the primary key of the new tuple.
2. Let us define a set of projections T that are going to be updated. If type of U is *add*, then $T = Proj(U) \setminus R_U$, in other case let $T = Proj(U)$.
3. For each $\Pi \in T$:

a. For each $(value_{\Pi.id}, p) \in Find(U, \Pi)$:
   i. If a strong path p contains 3 vertexes, then simply apply the requested add, $delete$ or $modify$ on a driver according to the update value.
   ii. Otherwise update the existing value according to the selection type.

4 Benefits of Applying PropScale

In this section we describe interesting scenarios of applying *PropScale*. Most of them is provided with experimental results that evaluate the improvement when compared to traditional methods.

4.1 Introduced Overhead

As the first test environment, we have chosen a bookstore with relations: *book*, *user*, *book_sold* and *book_comment*. The database stores books' data, data on customers, information on sold items and book comments put by users. In our benchmarks we have assumed 1 million of books and users, 5 millions of sold items and 10 millions of comments. Each tested storage has been run on a single Intel i5-2400 machine with 3.10GHz CPU's and 4GB RAM. Separate machines of the same type have been used to generate the workload and for *PropScale* web service.

In the first test we assume that the whole relation *book* is stored only in PostgreSQL. We add new tuples to the relation and test the overhead introduced by the propagator web service layer.

Figure 3 presents the results. It shows the total time of a client request to the propagator compared to the time consumed by PostgreSQL. The result is satisfactory. The overhead of the propagator seems independent of the workload and remains at the acceptable level.

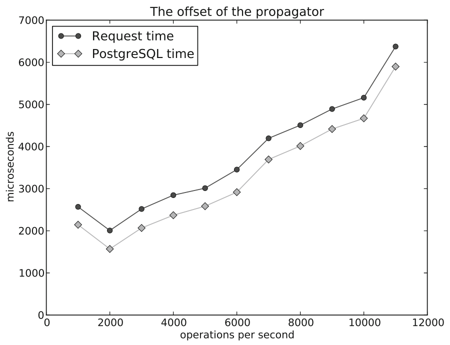


Fig. 3 The results of the experiment on the overhead of the propagator. The gap between the curves is the additional time above the PostgreSQL operation needed when using the propagator.

Next we evaluate the offset between actual update operations performed in the storages. When applying changes on multiple storages *PropScale* updates at first the specified projection. It is called the primary projection of a relation and is predefined in the schema. As the storage with the primary projection is updated, the response is returned to the client, while other storages are updated asynchronously. In this test we examine a scenario where book data are distributed among different databases: PostgreSQL for storing financial data, Mongo as a scalable storage with book information, and Redis as fast storage for simple book statistics. The projection stored in PostgreSQL has been defined as the primary projection of the relation *book*, thus changes are first applied to PostgreSQL. Then we measure the offset before they are applied in Mongo and Redis. Figure 4 shows the results. Again we can observe that the tested offset did not grow when the workload increased. The relational schema in this test is the same as in the previous one, where we have inserted all book data into RDMBS. Note that on the functional level this is the same. Although storing the same data in different storages introduced extra workload, the system runs more than 12000 operations per second, i.e. more than in the first test where only PostgreSQL database was used.

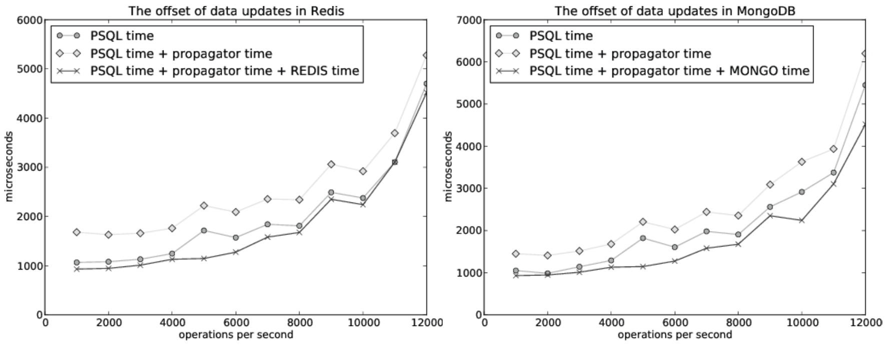


Fig. 4 The offset between updating data in the primary storage and in secondary storages. The bottom curve represents time spent in the primary storage PostgreSQL. The middle curve includes the offset introduced by the propagator. The upper curve presents the total time till all changes appeared in secondary storage.

4.2 *PropScale for Cloud Integration*

Cloud databases became an interesting issue in recent years and several companies outsource their databases to external services like Amazon SimpleDB or others. In the bookstore example it is worth storing book comments in a cloud. In that case, one does not have to take much care of scalability issues and service layer agreement is clearly defined. However, companies may be reluctant to store their financial or customer data outside of their company's system. Thus, the business critical data are kept in a local storage, while outsourcing less-critical data to cloud database suppliers. The research described in [8] compares the cost of a single request from different cloud database providers measured under a different workload. As the result external cloud databases are cheaper (per request cost), when a number of requests increases. According to this, it may be a business decision to move data, that is frequently read, to cloud data stores while leaving rest in local database. In that case a problem of integrating storages into a single system arises and *Propscale* solves it. It tracks data stored in different locations and makes sure that storages constitute a consistent storage.

4.3 Custom Statistics

The real advantage of the *PropScale* is the ability to define frequently accessed statistics and keep them in storages with quick data accesses like *key-value* ones. *PropScale* allows directly defining which statistics have to be stored. To evaluate this feature we introduce a benchmark built on a simple community forum. Suppose the system contains three relations: *forum*, *thread* and *post*. In our benchmark we assume 100 forums, 10K threads with 100 posts each, which results in 10 million posts.

We examine the following data access patterns which we believe are the most frequent: adding a single post, retrieving a list of forums, a list of threads within a forum and a list of posts within a thread. When retrieving a list of forums and threads the system needs to read the information on the last post in a thread/forum, its author and the date when an item has been added. Moreover when listing forums/threads we need to retrieve the number of contained threads/posts. In our benchmark we examine the queries that add a single post and retrieve the first 20 posts of a thread. These three queries are run randomly by a benchmark with the probability of: 10% for adding posts and 45% for retrieving posts/threads.

In the naive architectural choice, we store data in MySQL in the third normal form without any redundant columns. In that case queries that add or retrieve posts perform well, however retrieving threads with all needed data is singificantly slower that the acceptable level. In that case the achieved throughput is 5 operations per second when the workload is generated by a single thread. At this workload the average time for retrieving thread is 376 milliseconds, while adding and retrieving posts requires 4 and 49 milliseconds respectively.

The obvious countermeasure is to add redundant data to the system. This can be done by adding extra columns to relational database or by storing the redundant data in other storage. We compare these two options while the second is implemented by the *PropScale* and the redundant data is stored in *Redis*. The clear advantage of the second choice is that it scales well. The other disadvantage of the first choice is that when a new post is added, the database needs to update the corresponding thread and the forum tuple. We know that query caches implemented in relational database perform well when data are not modified. However the frequent read and write accesses to the same tuples significantly reduce the performance of most RDBMS. This has been confirmed in our tests and the results are presented in Figure 5. It is

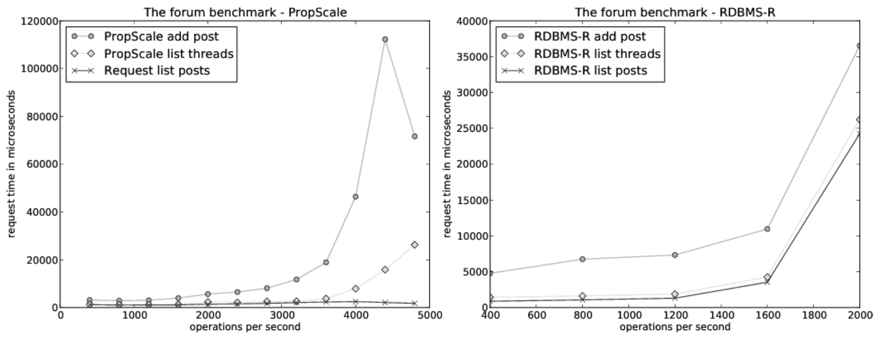


Fig. 5 RDBMS-R and PropScale correspond to a forum application built on: MySQL with redundant columns and MySQL/Redis integrated with PropScale respectively. The end of graph determines the maximal number of ops per second such that, the benchmark did not fail, i.e. an expected number of operations has finished.

even hard to present them on a single graph since when storing data only in MySQL the request time started to increase at the rate of 2000 ops per second and the next test with 2400 ops per second failed. The similar test with *PropScale* performed well until more than 3000 ops per second and failed on 5200.

5 Conclusion

According to the CAP theorem [3], there exists a trade-off between consistency and availability. Some storages like relational databases provide ACID but do not scale well. Possibly inconsistent NoSQL storages offer high availability. We believe that our system allows tuning this trade-off in a better way. Within our model, data can be cheaply split into smaller chunks put into custom storages for better performance.

Creating a scalable database storage is a valid research problem. We have focused on web applications with additional assumptions: (1) retrievals are dominant and (2) several consistency levels are needed in different contexts.

We have presented a scalable joint storage system based on several underlying storages that propagates updates to keep all data copies consistent with each other. We have shown the architecture. We also described basic implementation assumptions and the update propagator algorithm. The idea of the joint storage allows taking advantage of different architectures that suit best specific data.

We believe that it allows building scalable web applications at the lower cost, because it eliminates the risk of programming faults affecting the data consistency that are difficult to fix and detect.

References

1. Agrawal, D., El Abbadi, A., Antony, S., Das, S.: Data Management Challenges in Cloud Computing Infrastructures. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) DNIS 2010. LNCS, vol. 5999, pp. 1–10. Springer, Heidelberg (2010)
2. Iyengar, A., Challenger, J.R., Dias, D., Dantzig, P.: High-performance web site design techniques. IEEE Internet Computing 4, 17–26 (2000)
3. Brewer, E.A.: Towards robust distributed systems (abstract). In: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, PODC 2000, p. 7. ACM, New York (2000)
4. Challenger, J.R., Dantzig, P., Iyengar, A., Squillante, M.S., Zhang, L.: Efficiently serving dynamic data at highly accessed web sites. IEEE/ACM Trans. Netw. 12, 233–246 (2004)
5. Challenger, J.R., Iyengar, A., Dantzig, P.: A scalable system for consistently caching dynamic web data (1999)
6. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable consistency management for web database caches. computer science. Tech. rep. (2006)
7. Garrod, C., Manjhi, A., Ailamaki, A., Maggs, B., Mowry, T., Olston, C., Tomasic, A.: Scalable query result caching for web applications. Proc. VLDB Endow. 1, 550–561 (2008)
8. Kossmann, D., Kraska, T., Loesing, S.: An evaluation of alternative architectures for transaction processing in the cloud. In: Elmagarmid, A.K., Agrawal, D. (eds.) SIGMOD Conference, pp. 579–590. ACM (2010)
9. Kossmann, D., Kraska, T., Loesing, S., Merkli, S., Mittal, R., Pfaffhauser, F.: Cloudy: A modular cloud storage system. PVLDB 3(2), 1533–1536 (2010)
10. Manjhi, A., Gibbons, P.B., Ailamaki, A., Garrod, C., Maggs, B.M., Mowry, T., Olston, C., Tomasic, A., Yu, H.: Invalidation clues for database scalability services. Tech. rep. In: Proceedings of the 23rd International Conference on Data Engineering (2006)
11. Thrift (2012), http://thrift.apache.org/
12. Valduriez, P.: Principles of Distributed Data Management in 2020? In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011, Part I. LNCS, vol. 6860, pp. 1–11. Springer, Heidelberg (2011)

An Indexing Structure for Dynamic Multidimensional Data in Vector Space*

Elena Mikhaylova, Boris Novikov, and Anton Volokhov

Abstract. The multidimensional $k-NN$ (k nearest neighbors) query problem is relevant to a large variety of database applications, including information retrieval, natural language processing, and data mining. To solve it efficiently, the database needs an indexing structure that provides this kind of search. However, attempts to find an exact solution are hardly feasible in multidimensional space. In this paper, a novel indexing technique for the approximate solution of $k-NN$ problem is described and analyzed. The construction of the indexing tree is based on clustering. Indexing structure is implemented on top of high-performance industrial DBMS.

1 Introduction

Search efficiency is crucial for modern information retrieval. Commonly, search queries retrieve a relatively small portion of information. In this case, indexing search is much more efficient than a full scan. Any given process or stored object can be characterized by a set of features that are usually called attributes. The purpose of any index is quick access to the object based on the values of some of its attributes. In other words, the indices provide effective implementation of associative search.

Multidimensional searching is primarily associated with processing spatial and spatio-temporal data. Another class of applications processing multidimensional data includes systems based on various flavors of a text vector model for methods of data mining, pattern recognition, data compression etc. Data obtained from the application of this class are typically characterized by high dimensionality.

Elena Mikhaylova · Boris Novikov · Anton Volokhov
Saint Petersburg University
e-mail: egmichailova@mail.ru, b.novikov@spbu.ru,
a.v.volokhov@gmail.com

* This research is supported by Russian Foundation for Basic Research, grant 10-07-00156.

In this work we present an approach to approximate $k-NN$ multidimensional searching. Its idea is based on tree-clustering structure. Due to the curse of dimensionality we refused to provide exact $k-NN$ queries and have concentrated on approximate ones.

The remaining part of the paper is organized as follows. Section 2 contains the overview of related work. Section 3 informally outlines the techniques used in our approach. Our algorithms and data structures are presented in section 4, followed by analysis of experiments and results in section 5.

2 Related Work

Years of multidimensional searching evolution led to development of various indexing algorithms. The most respected tree structure to provide multidimensional indexing in generic metric space is M-Tree [5]. To improve the above-mentioned algorithm, a static tree reclustering technique is proposed in bulk loading algorithm [4]. Slim tree [20, 18] is developed to minimize overlap of nodes at the same level. Attempts to partition the initial set of points to disjoint subsets tuned to provide k-NN queries are made in the family of algorithms based on Voronoi diagram: D-Tree [22] treats Voronoi diagram buckets as nodes, and VoR-Tree [15] starts query processing with estimating a near point using R-Tree [10, 11] and then treat this point as a start point for querying Voronoi diagram built on leaf points.

In last decade multidimensional hashing has become a serious competitor to traditional tree approach. Locality-sensitive hashing [7] uses a priori information about a dataset to build a fast and efficient structure for approximate querying. This approach was introduced [12] and well developed [6, 1]. Yet, when dataset is changing rapidly, this idea is not applicable.

In many cases, it is computationally complex not only to find and to prepare vectors (eg. feature extraction of images), but also to calculate the function of similarity (eg. the Levinstein distance, quadric form distance or tree edit distance). Search could be greatly accelerated if it is possible to store a matrix of pairwise distances between objects. This idea was first presented in AESA [21]. To efficiently answer k-NN queries, Ak-LAESA [13] has been suggested. Unfortunately, this solution is not scalable because of the quadric behavior of index size.

In [19] k-NN search problem deals with the double filtering effect of clustering and indexing. The clustering algorithm ensures that the largest cluster fits into main memory and that only clusters closest to a query point need to be searched and hence loaded into main memory. In each cluster data is distributed with ordered-partition tree (OP-tree) main memory resident index, which is efficient for processing k-NN queries.

In [3] divide and conquer methods for computing an approximate k-NN graph are proposed. A hash table is used to avoid repeating distance calculations during the divide and conquer process.

Different aspects of the curse of dimensionality are known to present serious challenges to various machine-learning methods and tasks. In [14] a new aspect of the curse of dimensionality, referred to as hubness, is explored. Origins of this phenomenon, and its connection to k-NN query evaluation are discussed.

High-dimensional clustering is used by some content-based image retrieval systems to partition the data into groups (clusters), which are then indexed to accelerate processing of queries. Recently, the Cluster Pruning approach was proposed in [8] as a simple way to produce such clusters. The evaluation of the algorithm was performed within an image indexing context. The paper [8] discusses the parameters that affect the efficiency of the algorithm, and proposes changes to the basic algorithm to improve performance.

[16] presents an adaptive Multi-level Mahalanobis-based Dimensionality Reduction (MMDR) technique for high-dimensional indexing. The MMDR technique discovers elliptical clusters for more effective dimensionality reduction by using only the low-dimensional subspaces, data points in the different axis systems are indexed using a single B+-tree. The technique is highly scalable in terms of data size and dimensionality. It is also dynamic and adaptive to insertions. An extensive performance study was conducted using both real and synthetic datasets, and the results show that the proposed technique not only achieves higher precision, but also enables queries to be processed efficiently.

In paper [9] a new clustering method is proposed: to group together only points that are close to each other. The remaining points are stored separately, not clustered. Such a structure is obtained unbalanced. In order to create the indexing structure, in [2], the distances between the selected set of pivots and the data objects is computed, sorted and nearest distances are stored in separated tables. For search at first query is compared with pivots.

In resent research [17] general problems of metric access methods are discussed. Relevance of metric generalization in questioned. Idea of using more data-specific approaches to indexing is proposed.

3 The Approach and Rationale

Due to survey [17], most common metric in this case is Lp. The goal of this work is to construct dynamic bottom-up built tree for approximate nearest-neighbor search objects in high-dimensional vector space Lp. To build the indexing structure we use tree based paradigm with data points stored in leafs and routing points in non-leaf nodes. Parent points (centroid) are means of all its children. Tree is balanced by the construction. Volume of node is a parameter of algorithm and depends on operation system page size.

As we refused to use the metric approach, we don't have upper bound for k-approximate nearest-neighbor queries points, retrieved as nearest neighbors, are allowed to be arbitrarily far from the query point. According to this approach we choose precision (number of actual $k-NN$ in retrieved set divided by the value of k) as our main metric.

The idea of the algorithm is not to split overcrowded node, but to recluster the whole level of the tree, distinguishing new node in this level. This technique is believed to catch the inherent structure of initial dataset. As a result we can obtain more compact nodes. So the node size is a parameter of the algorithm. However, inserting is performed in traditional technique $1 - NN$ query is implemented and new point is added to retrieved leaf node.

The algorithm is developed with keeping in mind the idea of dynamic tuning: since in every non-leaf node we store only total number of its leaf children, the default approach is to descend only to the nearest leaf during searching. However, the algorithm implies the possibility of tuning the number of nodes to descend to perform better querying. In this case we choose total number of leaf nodes to be examinated and descend to several nodes, if number of children in nearest node is less than chosen threshold.

In order to evaluate the characteristics and performance of this structure, we have implemented the model over the data provided industrial relational DBMS. Perhaps this decision affects the efficiency but also ensures quick implementation, suitable for use in the prototype, for example, in an experimental system for content based image retrieval. That is, the developed system allows finding images on a number of characteristics similar to the one desired.

4 Algorithm Description

4.1 Data Structure

For indexing structure we used 4 tables. First table is a Table of points. In order the table structure was independent from the space dimension, each vector is represented in the table as a set of rows - one row for each vector attribute. Each table row has the $Point\_id$, $Attribute\_id$ and $Attribute\_value$. The table of centroids stores mass centers for every cluster and has the same structure as Table of points.

The third table is used to store the tree structure. Each row has $Point\_id$ and $Child\_id$.

Also we use the temporary tables for mass centers and tree structure. The rows have $Point\_id$, $Attribute\_id$, $Attribute\_value$ and $Iteration$.

4.2 Index Structure Parameters

The proposed algorithm has parameters. The parameters might affect the efficiency of the algorithm. Parameters are:

- distance measure,
- maximum number of points in one node maxNodeSize,
- balancing factor to prevent frequent re-clustering balancingFactor

In our experiments distance was calculated using the $L2$ metric , but algorithm works in any vector space.

We can choose maximum number of points in one tree node - *maxNodeSize*. In our experiment *maxNodeSize* was changed from 10 to 100. This number depends on the space dimensionality.

In order to avoid frequent reorganization of the tree, we dont fill all clusters to *maxNodeSize* by re-clusterization. Number of points in each node cant exceed two thirds of the *maxNodeSize*.

4.3 Index Structure Construction

Indexing algorithm starts with an empty set of indexed points and one tree node, considered as root node. When new point arrives, we recursively descent to node, that is nearest to arrived point. When the leaf node is reached, new point is added to corresponding node in tree structure. After that, algorithm rises to the root node, recalculating all centroid points to hold keep them being mean of their children.

When node overflows, we get all children its parent's node brothers and perform k-means clustering with following number of centers:

$$k = \max\left(1, \frac{n}{k} \times maxnodesize \times balancingFactor - k\right)$$

where n number of obtained, k number of brothers.

If constructed cluster is overflowed, we recursively recluster upper level of tree structure. When root node is reached, the tree is pulled up.

Initial values for k-means clustering are set to previous node centers. The point, that overflowed cluster become the one of new centers, the others are taken randomly from effected points. K-means algorithm is chosen according to the fact that centroids in tree are already similar to k-means centers every centroid contain points, that more similar to this centroid, than to the others. In this case we can assume, that nodes don't need many k-means iterations to return values of new centroids. On the other hand, k-means doesn't keep in mind inherent database properties and with some assumptions has exactly Voronoi diagram in a result. Algorithmic complexity of constructing the indexing structure equals

$$O(n \times \log^2 n)$$

4.4 Search

The search algorithsm can be described with the following steps:

1. Start from the top level.
2. Compute distances between points of current level (inside current node) and query point, retrieve list of points, ordered by distance to the query point

3. Pick points from the top of the list, until total number of point's leafs is bigger than number of points we want to examine.
4. Descend to chosen nodes and repeat $2-3$ until leaf nodes are reached.
5. Get all points from the leaf nodes.
6. Sort obtained points and retrieve first k points, k is defined by query type.

Algorithmic complexity of searching equals:

$$O(d \times \log n)$$

where d dimensionality of initial dataset, n cardinality of initial dataset.

5 Experiment and Analysis

The algorithm described above, was analyzed on two different datasets.

The first one is 41-dimensional representation of image characteristics with cardinality of 30000. The second one is 100-dimensional synthetic dataset with values from Gaussian distribution. Cardinality of second dataset is also 30000 vectors.

The method was implemented within the DBMS Oracle. Tables were constructed and procedures were written. Maximum node size was set to 10/50/100.

Table 1 Experimental results (1% examinated)

	Image representation dataset			Gaussian dataset	
maxNodeSize	10	50	100	50	100
$1-NN$ query	21%	28%	85%	20%	31%
$10-NN$ query	27%	45%	32%	3%	5%
$100-NN$ query	45%	38%	18%	5%	3%

Table 2 Experimental results (10% examinated)

	Image representation dataset		Gaussian dataset	
maxNodeSize	50	100	50	100
$1-NN$ query	56%	92%	42%	50%
$10-NN$ query	60%	47%	20%	23%
$100-NN$ query	61%	35%	20%	21%

Each experiment was performed twice. First time we set the total number of leaf points to be examined to 1% of initial dataset, and second time to 10%. Results for the fast one-percent scan are shown on the figure 1 and the result for more precise ten-percent scan are on the figure 2.

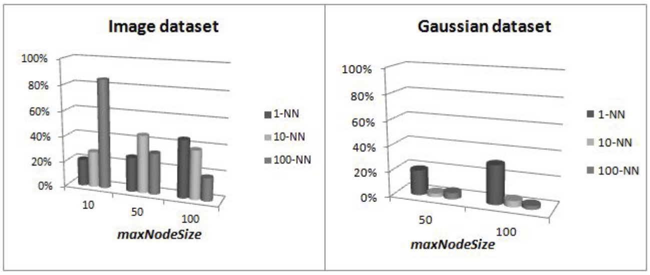


Fig. 1 The results for scanning 1% of the dataset

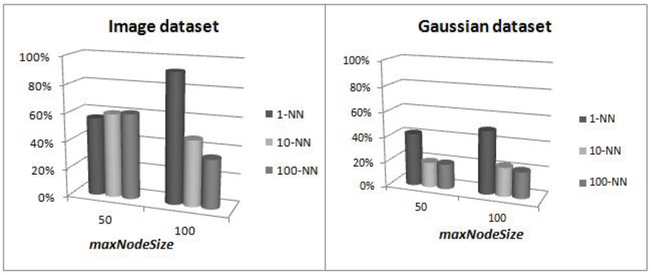


Fig. 2 The results for scanning 10% of the dataset

6 Conclusion

We present the algorithm that can deal with rapidly changing data in high-dimensional vector space. Experiments show high precision on not very high dimensional space. Precision in high-dimensional space strongly depends on size of a tree node: when nodes size is approximately equal to the value of k in $k-NN$ query, search algorithm visits only a few number of nodes and retrieves only a few actual nearest neighbors.

However to resolve this problem we can increase the number of nodes examined on each level: experiments show that scanning only 10% of the database results in significant increase of precision.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51(1), 117–122 (2008), http://doi.acm.org/10.1145/1327452.1327494, doi:10.1145/1327452.1327494

2. Barton, S., Gouet-Brunet, V., Rukoz, M.: Large scale disk-based metric indexing structure for approximate information retrieval by content. In: Proceedings of the 1st Workshop on New Trends in Similarity Search, NTSS 2011, pp. 2–7. ACM, New York (2011), http://doi.acm.org/10.1145/1966865.1966869, doi:10.1145/1966865.1966869
3. Chen, J., Fang, H.R., Saad, Y.: Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection. J. Mach. Learn. Res. 10, 1989–2012 (2009), http://dl.acm.org/citation.cfm?id=1577069.1755852
4. Ciaccia, P., Patella, M.: Bulk loading the m-tree. In: Proceedings of the 9th Australasian Database Conference, ADC 1998, pp. 15–26 (1998)
5. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB 1997, pp. 426–435. Morgan Kaufmann Publishers Inc., San Francisco (1997), http://dl.acm.org/citation.cfm?id=645923.671005
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, SCG 2004, pp. 253–262. ACM, New York (2004), http://doi.acm.org/10.1145/997817.997857, doi:10.1145/997817.997857
7. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco (1999), http://dl.acm.org/citation.cfm?id=645925.671516
8. Gudmundsson, G.T., Jónsson, B.T., Amsaleg, L.: A large-scale performance study of cluster-based high-dimensional indexing. In: Proceedings of the International Workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval, VLS-MCMR 2010, pp. 31–36. ACM, New York (2010), http://doi.acm.org/10.1145/1878137.1878145, doi:10.1145/1878137.1878145
9. Günnemann, S., Kremer, H., Lenhard, D., Seidl, T.: Subspace clustering for indexing high dimensional data: a main memory index based on local reductions and individual multi-representations. In: Proceedings of the 14th International Conference on Extending Database Technology, EDBT/ICDT 2011, pp. 237–248. ACM, New York (2011), http://doi.acm.org/10.1145/1951365.1951395, doi:10.1145/1951365.1951395
10. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, SIGMOD 1984, pp. 47–57. ACM, New York (1984), http://doi.acm.org/10.1145/602259.602266, doi:10.1145/602259.602266
11. Guttman, A.: R-trees: a dynamic index structure for spatial searching. SIGMOD Rec. 14(2), 47–57 (1984), http://doi.acm.org/10.1145/971697.602266, doi:10.1145/971697.602266
12. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC 1998, pp. 604–613. ACM, New York (1998), http://doi.acm.org/10.1145/276698.276876, doi:10.1145/276698.276876
13. Moreno-Seco, F., Milcó, L., Oncina, J.: A modification of the laesa algorithm for approximated k-nn classification. Pattern Recogn. Lett. 24(1-3), 47–53 (2003), http://dx.doi.org/10.1016/S0167-8655(02)00187-3, doi:10.1016/S0167-8655(02)00187-3

14. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. J. Mach. Learn. Res. 11, 2487–2531 (2010), http://dl.acm.org/citation.cfm?id=1756006.1953015
15. Sharifzadeh, M., Shahabi, C.: Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. Proc. VLDB Endow. 3(1-2), 1231–1242 (2010), http://dl.acm.org/citation.cfm?id=1920841.1920994
16. Shen, H.T., Zhou, X., Zhou, A.: An adaptive and dynamic dimensionality reduction method for high-dimensional indexing. The VLDB Journal 16(2), 219–234 (2007), http://dx.doi.org/10.1007/s00778-005-0167-3, doi:10.1007/s00778-005-0167-3
17. Skopal, T.: Where are you heading, metric access methods?: a provocative survey. In: Proceedings of the Third International Conference on SImilarity Search and Applications, SISAP 2010, pp. 13–21. ACM, New York (2010), http://doi.acm.org/10.1145/1862344.1862347, doi:10.1145/1862344.1862347
18. Skopal, T., Pokorný, J., Krátký, M., Snášel, V.: Revisiting M-Tree Building Principles. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 148–162. Springer, Heidelberg (2003)
19. Thomasian, A., Zhang, L.: Persistent clustered main memory index for accelerating k-nn queries on high dimensional datasets. Multimedia Tools Appl. 38(2), 253–270 (2008), http://dx.doi.org/10.1007/s11042-007-0179-7, doi:10.1007/s11042-007-0179-7
20. Traina Jr., C., Traina, A.J.M., Seeger, B., Faloutsos, C.: Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 51–65. Springer, Heidelberg (2000), http://dl.acm.org/citation.cfm?id=645339.650146
21. Vidal, E.: New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). Pattern Recognition Letters 15(1), 1–7 (1994)
22. Xu, J., Zheng, B., Lee, W.C., Lun Lee, D.: The d-tree: An index structure for planar point queries in location-based wireless services. IEEE Trans. on Knowl. and Data Eng. 16(12), 1526–1542 (2004), http://dx.doi.org/10.1109/TKDE.2004.97, doi:10.1109/TKDE.2004.97

Social Network Analysis on Highly Aggregated Data: What Can We Find?

Mikołaj Morzy and Kamil Forenc

Abstract. Social network analysis techniques have been often used to derive useful knowledge from email and communication networks. However, most previous works considered an ideal scenario when full raw data were available for analysis. Unfortunately, such data raise privacy issues, and are often considered too valuable to be disclosed. In this paper we present the results of social network analysis of a very large volume of the telecommunication data acquired from a mobile phone operator. The data are highly aggregated, with only limited amount of information about individual connections between users. We show that even with such limited data, social network analysis methods provide valuable insights into the data and can reveal interesting patterns.

1 Introduction

Social network analysis methods have proven useful in many areas of computer science. In particular, the study of the network structure can provide interesting insights into the properties and behavior of the network. In this paper we tackle the problem of studying the communication network of phone calls and SMS messages. However, contrary to previous works we are faced with an important obstacle. Due to privacy preservation concerns and to safeguard very sensible data, we have obtained the dataset from a mobile phone operator that contains the data already in the aggregated format, without an insight into raw data statistics and distributions, not to mention individual voice calls or SMS messages. Therefore, the main aim of this work was to determine the usefulness of highly aggregated data from a communication network for social network analysis techniques. We have decided to see if

Mikołaj Morzy · Kamil Forenc
Institute of Computing Science, Poznań University of Technology,
Piotrowo 2, 60-965 Poznan, Poland
e-mail: {Mikolaj.Morzy,Kamil.Forenc}@put.poznan.pl

the aggregated data still withheld enough patterns and structural motives to unveil interesting, useful and actionable knowledge.

To achieve this goal we began with the statistical analysis of the dataset. We have identified most useful measures of importance and prestige that could be applied to the nodes in the network. Then, we have asked business questions that correspond to use-case analysis, and we have tried to utilize the data in the aggregated format to provide satisfactory responses to these questions.

The paper is organized as follows. In Section 2 we review the existing literature and discuss previous approaches. Section 3 contains basic definitions used throughout the paper. In Section 4 we provide a detailed description of the dataset and discuss the difficulties in analyzing these data. The results of social network analysis methods applied to the dataset are reported in Section 5. The paper concludes in Section 6 with brief conclusions.

2 Related Work

The idea of analyzing phone call networks on a large scale can be related to the seminal experiment performed by Stanley Milgram, in which he has formulated the theory of small worlds in social networks [16]. The original experiment consisted in sending dozens of parcels using 300 randomly selected people to a distant city, where each participant of the experiment could have sent the parcel only to a person whom she/he had known personally. Later, similar experiments were conducted using the web by Adamic [1], Kleinberg [9] or Dodds *et al.* [6], and the criticism toward the original experimental setup has been raised by Kleinfeld [10]. All these works concentrated on examining the phenomenon of small diameter of real world networks with huge numbers of nodes.

Our current work is strongly influenced by the recent developments in the fields of communication networks analysis. For instance, Onnela *et al.* [15] present an interesting method for measuring structural properties of communication networks by reverse engineering the network, i.e., they measure the consistency of the network under sequential elimination of edges from the network. Inspired by Kovanen *et al.* [11] we scrutinize the reciprocity in the network.

Privacy preservation of sensitive user data has long been the subject of intensive research. Bonneau *et al.* show how easy it is to scrap sensitive data from social networks [5]. Gross and Acquisti show an empirical study on the amount of sensitive personal data shared publicly by Carnegie Mellon University students and provide scenarios of potential attacks on various aspects of data privacy [7]. There have been also studies aiming at the characterization of potential privacy leakages, e.g. [12]. Several solutions have been proposed to remedy this problem. Zheleva and Getoor propose a set of graph anonymization techniques [18] which consist mainly in the removal of sensitive data. A thorough overview of anonymization techniques previously proposed in the field of knowledge discovery can be found in [2].

Finally, the work presented in this paper is strongly influenced by recent developments in the analysis of social network dynamics. Important works in this direction include the discovery of densification laws and shrinking diameters by Leskovec *et al.* [14] and the research into the micro-evolution of social network structures by Kumar, Backstrom *et al.* [13], [3].

3 Basic Definitions

This section contains definitions of basic entities and measures used throughout the paper. We use primarily the notation and nomenclature introduced in [17] and [8], treating the network in the light of the graph theory. The graph $G = (V, E)$ is a pair of sets V, E, where V is a non-empty set of nodes (vertices), and E is a set of pairs of elements from V called *edges*, $(v,w) \in E \Leftrightarrow v \in V \wedge w \in V$. If edges in the graph are ordered pairs of nodes, then the graph is called a *directed graph*, otherwise the graph is *undirected*. The *neighborhood* of the node v is the set of nodes $N_v = \{w : w \in V \wedge ((v,w) \in E \vee (w,v) \in E)\}$. The number of nodes in the neighborhood of the given node v is called the *degree* of the node v and is denoted by $deg(v)$. The set of nodes from the neighborhood of the node v and all edges between these nodes constitute the *ego-network* of the node v.

A *walk* in the graph G is a finite sequence of neighboring nodes. The number of nodes on a walk is called the *length* of the walk. A walk such that every node appears at most once is called a *path*. The shortest path between nodes v and w is denoted as d_{vw}, and its length is denoted as $d_G(v,w)$. The number of shortest paths between nodes v and w is denoted as δ_{vw}. A graph G where for each pair of nodes (u,v) there exists a path between these nodes is called a *connected graph*. A *connected component* is a maximal connected subgraph of G. Each node belongs to exactly one connected component.

For our analysis we have studied several measures of centrality, coherence and efficiency of networks. The *density* of the network is the ratio of the existing edges to the maximal number of edges that might exist in the network. For social networks the density of the network is typically close to 0, because the average node degree is orders of magnitude smaller than the number of available nodes. For a directed graph the density is defined as $\rho_D = |E|/(|V| * (|V|-1))$, for an undirected graph the density is $\rho_G = 2 * \rho_D$.

The *betweenness* of a node v is defined as the ratio of the number of shortest paths containing the node v to the total number of shortest paths existing in the graph. Formally, $C_B(v) = \sum_{u \neq w \in V} \frac{|\rho_{u,w}(v)|}{|\rho_{u,w}|}$. Betweenness of a node is one of the popular *centrality measures* used to assess the importance of nodes in the network. Betweenness informs on the relative influence of a node in transmitting information through the network. In the context of a telecommunication network nodes with high betweenness correspond to users who form bridges between separate clusters (subgroups, communities) of users.

Another centrality measure used in our analysis is the *closeness* of a node v. This measure is defined as the average length between the node v and all remaining nodes in the network, which can be formalized as $C_C(v) = \sum_{u \neq v \in V} d_G(v,u)|V|^{-1}$.

Closeness is a useful measure to assess the importance of a given node in terms of the accessibility of the node, its radius of influence and the number of other nodes to which the given node can transmit information in short time.

The *diameter* $d_{\varnothing}$ of the network is the longest among the shortest paths between any pair of nodes in the network. Sometimes, to offset the influence of outlier nodes, the diameter of the network is computed over the 95th percentile of nodes. When computing the diameter of the network it is often useful to compute the *average distance between nodes* $\hat{d}$, which is the arithmetic average over all shortest paths between pairs of nodes. Formally, $d_{\varnothing} = max_{u,v \in V}(d_G(u,v))$ and $\hat{d} = \sum\limits_{u,v \in V} \frac{2 d_G(u,v)}{|V|(|V|-1)}$.

Another measure useful in our analysis is the *reciprocity*. This measure is defined only for directed graphs and it represents the ratio of bidirectional relationships between pairs of nodes to the total number of relationships. This can be expressed as $R = |\{(u,v) : (u,v) \in E \wedge (v,u) \in E\}| * |E|^{-1}$. In the case of our analysis reciprocity is essential for distinguishing between incidental communications (e.g., a marketing call) from regular relationship.

A fundamental local measure applied to individual nodes is the *local clustering coefficient*. This measure can be interpreted as the degree of acquaintanceship between the neighbors of a given node. The local clustering coefficient is defined as the ratio of the number of existing relationships in the ego-network of a node. Nodes that have higher values of the local clustering coefficient belong to coherent communities of nodes, for which the average density of relationships between the nodes within the community is much higher than the density of relationships with the nodes from outside of the community. Formally, the local clustering coefficient of a node v for a directed graph G (for an undirected graph the numerator is multiplied by 2) is defined as $C(v) = \frac{|\{(u,w) \in E : u \in N_v \wedge w \in N_v\}|}{deg(v)(deg(v)-1)}$. The local clustering coefficient can be easily aggregated to a single value describing the entire network. The *global clustering coefficient* is simply the local clustering coefficient averaged over all nodes: $C = |V|^{-1} \sum_{v \in V} C(v)$ but it should be noted that an alternative formulation of this measure is commonly used in sociology, where the global clustering coefficient is defined as the ratio of the number of triangles in the network (i.e., the number of cliques of the size 3) to the number of triplets (i.e., the number of sets of three nodes connected by paths of the length of at most 2).

One of the most prominent characteristics of social networks is the abundance of power laws governing the growth and evolution of these networks [4]. Whenever a phenomenon can be described using the distribution of the form $P(k) \sim k^{-\gamma}$, with $\gamma \in \langle 1,3 \rangle$, we can assume that the phenomenon can be described using power laws.

4 Dataset Description

The data have been provided by one of the major European mobile phone operators. The dataset describes both the voice communication and SMS communication. For voice communication we can know whether nodes A and B are subscribed to the network, the day of the last connection between A in B within the current month, the number of calls, total duration of calls and numbers of calls under 1 minute, between 1 and 5 minutes, between 5 and 15 minutes, over 15 minutes, respectively. Apparently, we do not have detailed information about individual calls, nor on their timing. The data have been aggregated to monthly statistics, with the total number of calls aggregated to discrete bins. Similarly, we do not have the dates of connections (only the day of the last connection), thus, we approximate the regularity of voice calls by dividing the last date of connection by the number of monthly connections (assuming that the dates of connections between any two nodes can be reasonably described using the uniform distribution). For SMS communication we know the direction of the communication, the day of the last connection between A in B within the current month, and the number of SMS messages exchanged between the nodes. We approximate the regularity of communication using SMS messages similarly to the previous case. Due to privacy preservation concerns we do not have the timestamps of individual messages, so we cannot follow SMS conversations (e.g., bursts of messages between two nodes in a short period of time).

The data can be analyzed both as an undirected graph (where an edge between nodes represents any communication between the nodes) or as a directed graph (where each edge represents communication from one node to another node). Below we present basic characteristics of the datasets, using the following notation: G_v is the undirected graph of voice connections, G_s is the undirected graph of SMS connections, G_t is the undirected joint graph of total connections (both voice and SMS), and D_v, D_s, D_t are the directed versions of these graphs, respectively. Table 1 presents the basic characteristics of these datasets.

Table 1 Characteristics of graphs

size	G_v	G_s	G_t	D_v	D_s	D_t
Nodes	23 188 616	12 512 998	24 897 382	23 188 616	12 512 998	24 897 382
Edges	61 411 576	22 346 286	69 720 794	77 036 240	31 604 367	90 717 966
Density	$2,28 \cdot 10^{-7}$	$2,85 \cdot 10^{-7}$	$2,25 \cdot 10^{-7}$	$1,43 \cdot 10^{-7}$	$2,02 \cdot 10^{-7}$	$1,46 \cdot 10^{-7}$
Avg. degree	5,9	3,57	5,6	—	—	—
Reciprocity	—	—	—	25,4%	41,4%	30,1%

An interesting feature visible in the above characteristics is that the reciprocity of SMS communication is almost two times greater than the reciprocity of voice connections. This suggests that SMS communication is more often utilized between people who are socially close. There is also a great overlap between users of voice and SMS communication, although we observe a significant portion of the population (almost 7%) who use only SMS communication and make no voice calls.

Figure 1 presents the distribution of node degrees, the distribution of number of connections between pairs of nodes, the distribution of summary connection times and the distribution of the text messages sent. All these distributions show characteristics of power laws which often emerge in the context of social network properties.

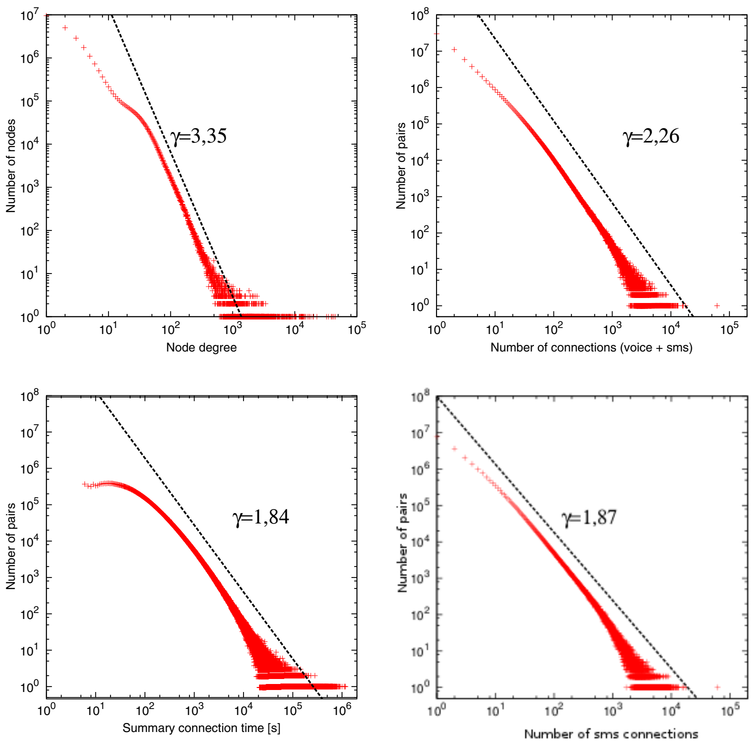


Fig. 1 Basic distributions

Table 2 The size of the largest component

size	G_v	G_s	G_t	dataset
Nodes	$99,64\%$	$97,79\%$	$99,72\%$	full
Edges	$97,04\%$	$96,67\%$	$96,57\%$	full
Nodes	$89,89\%$	$82,18\%$	$83,27\%$	constrained
Edges	$97,25\%$	$88,82\%$	$95,04\%$	constrained

For connection times the modal value of the connection time is 21 seconds, but it suddenly drops to only 9 seconds if we constrain the dataset to contain only the communication between the subscribers of the mobile phone operator who provided the dataset (i.e., when excluding the communication from outside the provider's network).

Another interesting property of the network emerges if we observe the numbers and sizes of components in the network. Table 2 presents the size of largest component in the full dataset and the dataset constrained to the mobile phone operator's subscribers, respectively, where the size is expressed either as the percentage of nodes or edges contained in the largest component. Again, as expected, the vast majority of nodes and edges are within a single giant connected component. Interestingly, we observe a smaller percentage of nodes in the largest connected component in the constrained dataset. This clearly suggests, that some external nodes (i.e., users who are not subscribed to mobile phone operator's service) serve as bridges that provide the connectivity for subscribers. Also, this suggests that isolated components are comprised mostly of operator's subscribers. We can see that G_t has a lower percentage of nodes than G_v (and only slightly higher than G_s), which means that isolated components of G_v and G_s do not share any nodes. Figure 2 shows the distribution of isolated components sizes. These distributions are govern by power laws as well, and the size of the second largest component never exceeds 50 nodes, independently of the size of the largest component.

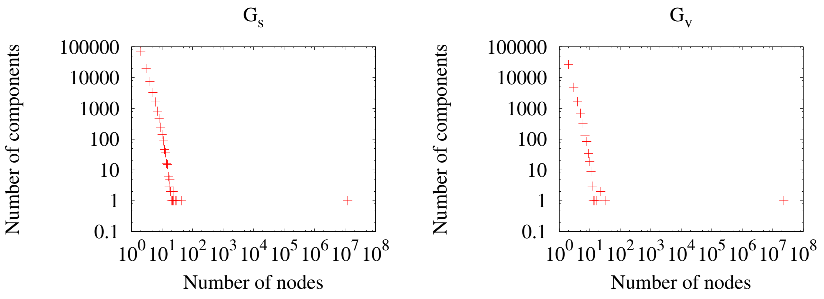


Fig. 2 Distribution of sizes of isolated components

5 Experiments

Before starting the experiment, we have removed the outliers from the dataset. We have decided to use the degree of a node as an indicator of outlier nodes, setting the filtering thresholds at $deg(v) \leq 200$ and $deg(v) \leq 1000$. This filtering had no significant impact on the results of the analysis because, depending on the type of the graph, between 99.938% and 99.9998% of original nodes and edges were preserved after outlier elimination. Below we present our experiments aiming at providing answers to the following questions.

5.1 Are There Any Areas with a Higher Density of Relationships?

To answer this question we have analyzed the distribution of the number of relationships between nodes of particular degrees. We were trying to identify subsets of users who engage in communication (either voice or SMS) more frequently than others. The results are summarized in Figure 3.

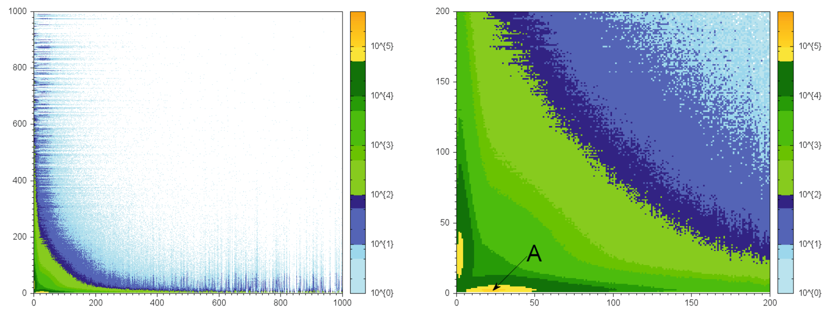


Fig. 3 Number of relations in voice calls. The left figure presents nodes with $deg(v) < 1000$, the right figure presents nodes with $deg(v) < 200$. The abscissae represent the degree of the caller, the ordinates represent the degree of the person being called.

We observe that the densest region of relationships spans from nodes of degree $10-40$ to nodes of degree less than 5. This indicates that users who do not have many friends are mostly passive receivers of communication (e.g., these could be parents of adult children). The distribution (excluding the dense area marked with the letter A) can be approximated by the formula $f(x,y) = cx^{\gamma}y^{\delta}$, where $\gamma = -1.2$ and $\delta = -1.3$. More surprisingly, if we constrain the dataset to contain only the subscribers of the mobile phone operator, the same analysis yields different results, depicted in Figure 4. In this case, the dense relationship area is constrained by a circle with the center in the point $(30, 30)$ and the radius of 30. Again, users with low degrees make outbound calls less frequently and more often than not are on the receiving end of the call.

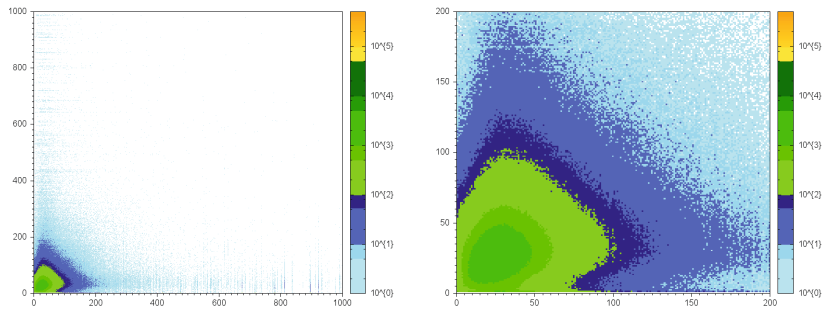


Fig. 4 Number of relations in voice calls constrained to only the subscribers of the mobile phone operator. The left figure presents nodes with $deg(v) < 1000$, the right figure presents nodes with $deg(v) < 200$. The abscissae represent the degree of the caller, the ordinates represent the degree of the person being called.

5.2 Is There a Point That Makes Reciprocal Relations More Probable?

Our next question concerns the existence of a user profile that guarantees, with a high probability, that any new relationship will be returned, making the relationship reciprocal. First, we check the reciprocity w.r.t. different means of communication.

Table 3 Reciprocity of various means of communication

parameter	one-directional	reciprocal	% of reciprocal
number of SMS	47 742 626	250 330 869	84%
number of seconds	11 604 404 901	21 223 013 808	64,7%
number of voice calls	99 820 901	201 380 401	66,9%
number of connections	109 407 327	475 147 144	81%

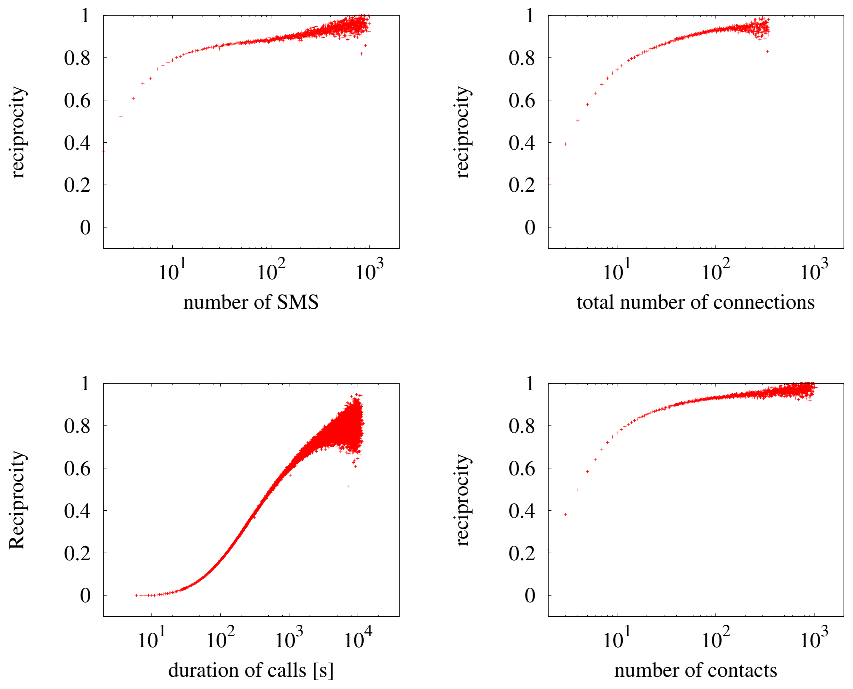


Fig. 5 Reciprocity of communication. Number of SMS messages (upper left), number of voice calls (upper right), summary duration of voice calls (lower left), number of contacts (lower right)

Surprisingly, we have found that there are no strong correlations between the duration of the call and the reciprocity of communication. Figure 5 presents the detailed examination. All connections with the summary duration of calls lower than 12 seconds are uni-directional. It is a strong indicator that such connections are either due to a mistake, or serve as notifications only. An important result of this analysis is that the reciprocity grows the slowest w.r.t. the total duration of calls.

5.3 *Is There a Limit on the Number of Contacts, above Which SMS Communication Is Not Feasible Anymore?*

We theorized that above a certain threshold of the node's degree the communication using texting becomes too cumbersome and users tend to switch to voice communication. Figure 6 presents the results. On the left we compare the number of relations where SMS communication dominates over voice calls (i.e., we count the number of pairs of users who text more often than make voice calls). On the right we show the ratio of the SMS communication to all forms of communication. Although there are a few areas with slightly increased domination of SMS communication (as visible on both figures), one may formulate a rule of thumb that above the degree of 40 SMS communication is clearly replaced by voice calls. This can be explained by the burden of texting when communicating with a large set of friends.

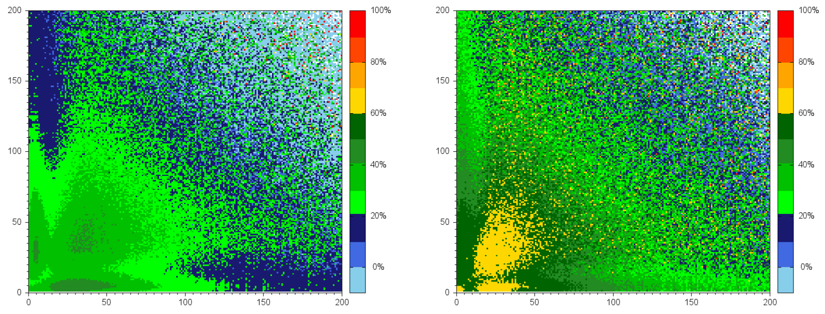


Fig. 6 The percentage of communication with the majority of SMS messages (left), and the ratio of SMS messages in the overall communication. The abscissae represent the degree of the caller, the ordinates represent the degree of the person being called.

6 Conclusions

In this paper we have presented an analysis of a large social network created from a telecommunication company dataset. The data have been highly aggregated, thus precluding many core analysis advocated by the literature. However, even in the presence of aggregations, social network analysis methods proved useful in unearthing interesting patterns. Among non-trivial results of the analysis we include the following findings:

- the reciprocity of communication does not depend on the duration of calls,
- the threshold of SMS communication profitability is around 40 friends,
- the most talkative and active users have between 20 and 50 friends,

References

1. Adamic, L.A.: The Small World Web. In: Abiteboul, S., Vercoustre, A.-M. (eds.) ECDL 1999. LNCS, vol. 1696, pp. 443–452. Springer, Heidelberg (1999)
2. Aggarwal, C.C., Yu, P.S.: A general survey of privacy-preserving data mining models and algorithms. In: Privacy-Preserving Data Mining. The Kluwer International Series on Advances in Database Systems, vol. 34, ch. 2, pp. 11–52. Springer US, Boston (2008)
3. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 44–54. ACM, New York (2006)
4. Barabási, A.-L., Albert, R.: Emergence of Scaling in Random Networks. Science 286(5439), 509–512 (1999)
5. Bonneau, J., Anderson, J., Danezis, G.: Prying Data out of a Social Network. In: Social Network Analysis and Mining, International Conference on Advances in, pp. 249–254. IEEE, Los Alamitos (2009)
6. Dodds, P.S., Muhamad, R., Watts, D.J.: An Experimental Study of Search in Global Social Networks. Science 301(5634), 827–829 (2003)
7. Gross, R., Acquisti, A.: Information revelation and privacy in online social networks. In: Proc. of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, pp. 71–80. ACM, New York (2005)
8. Hanneman, R.A., Riddle, M.: Introduction to Social Network Methods. University of California (2005)
9. Kleinberg, J.: The Small-World Phenomenon: An Algorithmic Perspective. In: Proc. of the 32nd ACM Symposium on Theory of Computing, pp. 163–170 (2000)
10. Kleinfeld, J.: Could It Be A Big World After All? The ”Six Degrees of Separation” Myth. Society (2002)
11. Kovanen, L., Saramaki, J., Kaski, K.: Reciprocity of mobile phone calls (2010)
12. Krishnamurthy, B., Wills, C.E.: Characterizing privacy in online social networks. In: Proc. of the First Workshop on Online Social Networks, WOSN 2008, pp. 37–42. ACM, New York (2008)
13. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 611–617. ACM, New York (2006)
14. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proc. of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, pp. 177–187. ACM, New York (2005)
15. Onnela, J.P., Saramäki, J., Hyvönen, J., Szabó, G., Lazer, D., Kaski, K., Kertész, J., Barabási, A.L.: Structure and tie strengths in mobile communication networks. Proc. of the National Academy of Sciences 104(18), 7332–7336 (2007)

16. Travers, J., Milgram, S.: An Experimental Study of the Small World Problem. Sociometry 32(4), 425–443 (1969)
17. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences). Cambridge University Press (1995)
18. Zheleva, E., Getoor, L.: Preserving the Privacy of Sensitive Relationships in Graph Data. In: Bonchi, F., Malin, B., Saygın, Y. (eds.) PInKDD 2007. LNCS, vol. 4890, pp. 153–171. Springer, Heidelberg (2008)

An Adaptive Navigation Method for Semi-structured Data

Rim Zghal Rebai, Corinne Amel Zayani, and Ikram Amous

Abstract. The navigation adaptation is the solution that supports the user during his interaction with the system. In the literature, several works that deal with the navigation adaptation are proposed. They guide the user from a document to another, provide the user with a set of links leading to the pertinent documents, or apply on simple links the suitable adaptive navigation technologies. In this paper, we contribute to propose a method that identifies the best navigation path between semi-structured result documents according to the user's needs, history and device.

1 Introduction

As the information systems evolve, they become more and more difficult to navigate. Therefore, users can lose their time in order to find the pertinent documents or can be lost in the huge number of links. Zhu [15] says ”moving back and forth between links and the main nodes creates disruption and discontinuity” causing ”disorientation and cognitive overload”. That is why navi-gation adaptation becomes a necessity because it helps the user to easily find the required information and reduces the disorientation problem.

Several adaptive navigation methods [12][3] and technologies [2] have been proposed. They adapt the navigation by (i) guiding the user from a document to another, (ii) providing the user with a set of links leading to the pertinent documents and (iii) applying on simple links the suitable adaptive navigation technologies. Generally, navigation adaptation support is strongly based on user modeling, his goals, necessities and interests. These information are stored in a user profile. According to this latter, adaptive systems provide the user with help to easily navigate.

Rim Zghal Rebai · Corinne Amel Zayani · Ikram Amous
MIRACL, ISIMS, Cité El Ons, Route de Tunis Km 10, Sakiet Ezzit 3021, Sfax, Tunisia
e-mail: rim_zghal@yahoo.fr, zayani@irit.fr, ikram.amous@isecs.rnu.tn

In order to adapt the navigation and specifically to reduce the required number of steps to locate the pertinent information, we propose a method that provides the best navigation path between semi-structured result documents. To restrict the navigation space, we propose as well the idea of a new adaptive navigation technology that allows to reduce the number of pertinent simple links by using the XLINK extended links (W3C[1]). So, we can apply our method on both simple and extended links which is not provided by the existing works.

This paper is organized as follows.Section 2 presents a state of the art of some works dealing with the navigation adaptation. In section 3, we expose the proposed parameters for the navigation adaptation. In section 4, we detail our proposed navigation adaptation method. In section 5, we evaluate our proposed method. Finally, we present the conclusion and our perspectives.

2 State of the Art

All materials on several works studying navigation problems have been proposed to provide the user with an adaptive navigation support. We propose to classify the works that we will study in this state of the art into three categories.

The first category mainly adapts the navigation by only adapting the presentation of links according to the user's preferences, knowledge, history, etc., by means of the adaptive navigation support technologies. The AHA! [4], for example, applies the link hiding technology [2][7] to irrelevant links and the link annotation technology [2][7] to the remaining links by using different colors depending on the user's model. Web Watcher [1] and ELM-ART [11] are the most popular adaptive hypermedia systems that use the direct guidance technology [2] to suggest a link to the "next best" page or document for the user to visit. Hypadapter [8] is the first system that introduced the link ordering technology. The idea is to put the links in order of relevance according to the user's model. WebIC [15] and ELM-ART [11] are among the systems that use the link generation technology [2]. This technology provides new links to documents deemed relevant to the user's profile.

The second category aims to provide one or more links to the best nodes (document, page). These latter are identified by means of different methods that vary from one system to another depending on the objectives and the application areas. The adaptive system proposed by Verma et al. [10] calculates and ranks the weight of each web page in the priority of descending order according to click-count, hyperlink weight and most frequent visits to the webpage. Then, it proposes a direct link to the first page. Wanga et al. [12] analyze navigation paths of websi-te visitors to identify the frequent surfing paths and provides the user with a set of links that leads to the next visited web pages. Seo et al. [9] propose two methods to navigation adaptation. The first one suggests the next link to be followed by the user and the second one generates quick links as additional entry points into Websites.

[1] http://www.w3.org/TR/xlink11/

The third category suggests the best navigation path allowing the user to reach relevant information with fewer clicks. The identification of this path varies from one system to another. The system proposed by Chiou et al.[3] provides the best navigation path between u-learning objects according to the student's personal and environmental situation. This path is determined by using a meta-heuristic or a heuristic based algorithm. These algorithms take as input a set of ubiquitous environment specific parameters that are related to the learner, the environment and the learning objects. In [5] the authors propose a system that analyses Web logs to identify the best navigation path by skipping irrelevant nodes and providing shortcuts to popular nodes. The system proposed in [15] extracts the links and the key words from the already visited pages in order to propose a set of links that lead to the relevant pages.

We notice that all the already mentioned works adapt the navigation without offering the user the opportunity to choose the next visited document. Moreover, the used navigation adaptation technologies and the proposed methods are applied only on simple links lead to only one document or page, we propose an adaptation navigation method that identifies the best navigation path. This method allows a better use of the adaptive navigation technologies in order to allow the user to freely choose the next visited document. Also, we propose the idea of a new adaptive navigation technology called ”Extended Link Technology” based on the XLINK extended links. So, our method can be applied to simple and extended links.

3 Proposed Parameters for Adaptation

The main objective of our work is to provide the user with the best navigation path between the semi-structured result documents which depends on his current situation. We propose to describe this situation by using three parameters:

- The related parameters of user's history: these parameters are a part of the user profile. They allow to describe the history of user. We distinguish two parameters: one for the session and another for the user. The first one is consisted of the number of sessions and their duration. As for the second one, it is made up by the visited documents which are identified by, the number of access to each document, the spent time on each document, the visited links and the number of clicks on each link. In the end of each session, these parameters are updated or stored in the user's profile in order to be taken into account in the next sessions.
- The related parameters of the result documents: these parameters allow to describe a document by three elements: (i) the Unified Resource Locator, (ii) the required device configuration which is limited to the Operating System and the Random Access Memory and (iii) the specific domains, knowing that a document can belong to one or several domains/sub-domains. For each domain, we specify the benefit of the document which depends on the domain. It is similar to the profit of learning object proposed in [3]. It is a value ranged from 0 to 1. It can be identified by the document's author based on the rele-vance of the document's content to each domain/sub-domain.

- The Related Parameters to user's device: the user can access the system via different devices with various configurations. In our work, we are interested in using only the operating system and the size of the RAM to avoid the blocking state of the device while displaying a document. These parameters are detected and used only in the current session.

4 Proposed Navigation Adaptation Method

To solve the disorientation problem, we propose a method for adapting the navigation that allows to identify the best navigation path between the returned documents. This method is based on a new adaptive navigation technology called ”Extended Link Technology” and based on the XLINK extended links. Our method is the core of the Navigation Adaptation Engine (NAE) which is the main component responsible for the navigation adaptation in our architecture ”MEDI-ADAPT” [14]. It is mainly based on a heuristic algorithm using the already presented parameters (See section 3) and the calculation equations related to documents (See section 4.2).

4.1 Proposed Heuristic Algorithm

The proposed heuristic algorithm takes the previously cited parameters and the result documents as input in order to provide the best navigation path between these documents as output. It is illustrated by the flowchart in figure 1.

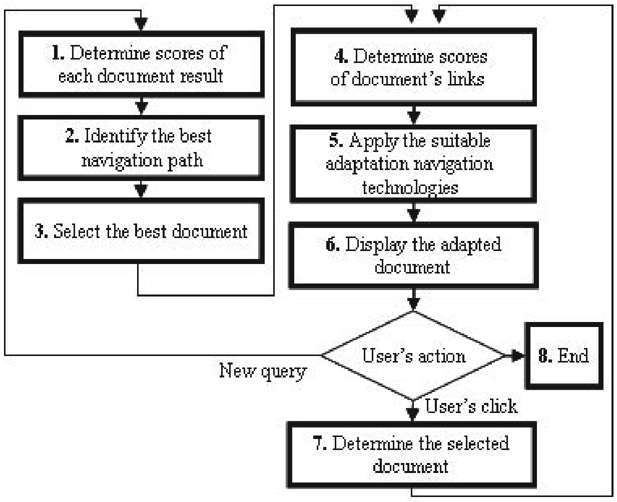

1. Determine scores of each document result
2. Identify the best navigation path
3. Select the best document
4. Determine scores of document's links
5. Apply the suitable adaptation navigation technologies
6. Display the adapted document
User's action
New query
8. End
User's click
7. Determine the selected document

Fig. 1 Flowchart of the proposed heuristic algorithm

When the user launches a query, the system executes this query and receives result documents from sources. The Navigation Adaptation Engine receives these documents and all the required parameters in order to start the navigation adaptation process. First, the algorithm calculates each document's score by using the equation 1 (See subsection 4.2). Then it identifies the best navigation path between these documents. Next, it selects the first document of the path and determines the scores of each link. Finally, it adapts the document by means of the suitable navigation adaptation technologies and display it to the user.

Whenever the user clicks on a link, the target document is selected, and the algorithm reiterates from the determination of the links' scores (Step 4) until the end of the session. In case of a new query, the algorithm restarts from step 1.

4.2 Identification of the Navigation Path

In order to identify the best navigation path between the result documents (Step 4 in the flowchart of figure 1), we propose an algorithm called "Identify the navigation path" (See Algorithm 2). To distinguish between result documents, in terms of relevance, this algorithm must firstly assigns a score to each document (Step 3 in the flowchart of figure 1). This score is calculated by a function called "Calculate_ Document_ Score" (See Algorithm 1) using the equation (1), (2) and (3).

$$Doc\_Score(di) = \frac{benefit(di) + Doc\_Freq(di) + tm(di)}{Cp} \tag{1}$$

benefit(di), Doc_Freq(di) and tm(di) denote respectively the access benefit of the document di, the frequency of the access to the document di (equation (2)) and the spent average time on the visit of the document di (equation (3)). Cp is a constant that assumes 1 as a value when the user's device manages to view the document, otherwise, it assumes 10. This value is resulted from a comparison between the configuration of the used device and the required device configuration to display the document.

$$Doc\_Freq(di) = \frac{nb\_acces(di)}{nb\_total\_doc} \tag{2}$$

nb_acces(di) represents the access number to the document di and nb_total_doc represents the total number of the accessed documents during all sessions.

$$tm(di) = \frac{\sum t(di)}{\sum_{k=1}^{nb_session} tk} \tag{3}$$

tm(di) corresponds to the ratio of the total time of visiting the document di and the total time of all sessions .

The algorithm 1 describes the function " Calculate_Doc_Score" that calculates the score of document. This function takes as parameters the history of user (user_history), the selected document (doc_reslt) and its benefit (benefit(doc_reslt)).

It returns the score according to the equation (1) which uses the frequency of document (doc_freq(doc_reslt)), the average time spent (tm(doc_reslt)) and the constant (Cp). These latter, are respectively presented in lines (3), (4) ,(5) of the algorithm 1.

```
1.Algorithm 1: Calculate_Doc_Score
2.Input:  doc_reslt,user_history,benefit(doc_reslt)
3.Output: Doc_Score(doc_reslt)
3.begin
4.    Doc_Freq(doc_reslt)=Calculate_freq(doc_reslt,
          user_history);
5.    tm(doc_reslt)=Calculate_tm(doc_reslt, user_history);
6.    Cp=Calculate_Cp(doc_reslt);
7.    Return  Doc_Score(doc_reslt);       //(cf. Equation1)
8.end.
```

The algorithm 2 "Identify the navigation path", allows to identify the best navigation path between the result documents. For each document in result documents (list_doc_reslt), it extracts the benefit (line (7)) from meta-document and calculates its score. This score assumes the benefit's value when the document has never been visited (line (11)). Otherwise, it is calculated by means of the equation (1) using the user history parameters (line (9)).

```
1.Algorithm 2: Identify the navigation path
2.Input:list_doc_reslt,user_history,best_navigaion_path
3.Output: best_navigaion_path
4.begin
5.    for each(doc_reslt in lis_doc_reslt)
6.    begin
7.        benefit[doc_reslt]= Extract_benefit(doc_reslt);
8.                if(doc_reslt in user_history)then
9.                    Doc_Score[doc_reslt]=Calculate_Doc_Score
                          (doc_reslt,benefit,user_history);
10.               else
11.                   Doc_Score[doc_reslt]= benefit[doc_reslt];
12.       end
13.       best_navigaion_path=Determine_path(Doc_Score, lis_doc_reslt);
14.   Return  best_navigaion_path;
15.end.
```

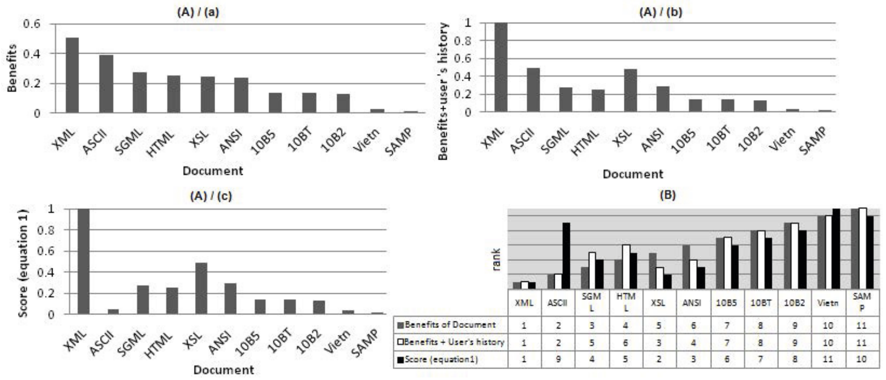
4.3 The Used Navigation Adaptation Technologie

Brusilovsky [2] "Adaptive navigation support is a specific group of technologies that support user navigation in hyperspace, by adapting to the goals, preferences and knowledge of the individual user". These technologies help the user to easily differentiate between links and find his needs. Many links (contextual, non-contextual, index, etc.) can mislead the user to docu-ments related to a domain different to the current one. So, we propose to apply on these links the "disabling link technology". To guide the user to the next best unvisited document in the defined path, we propose to use the "direct guidance technology". When the link does not exist, we generate a direct link and apply the "annotation link technology". Otherwise, we only apply the "annotation technology". In order to provide the user with the opportunity to choose the next document to visit among the returned documents, we propose to generate in the current docu-ment an index table that leads to the remaining path documents. This table is generated by using the ”generation technology” and ordered by the ”ordering technology”. To reduce the number of contextual links in the document and enable the user to have an idea about the related domain of each link, we suggest to generate, at run time, a set of XLINK[2] extended links from simple links. An XLINK extended link ”is a link that associates an arbitrary number of resources. The participating resources may be any combination of remote and local”. So, we propose to regroup several simple links that belong to the same sub-domain into a single extended link. The founded extended links will be generated and take the sub-domain's na-me as a title. Then, the resources of each extended link are subsequently reordered by using the ”ordering technology” and annotated by the ”annotation link technology”.

5 Evaluation

In order to evaluate the proposed method, we use a corpus of 110000 documents of INEX 2007 French version, which is a part of the collection WIKIPEDIA XML. Documents in this corpus are related to one or more domains/sub-domains and containing XLINK simple links. To apply our method, an additional work in our laboratory was performed in order to add to these documents a correspondent domain/sub-domain benefits. Given one user launches this query: ”Documents related to the norms and computer standards”. As a result to this query, the system provides 11 documents (links) in random order. We propose to evaluate the first four steps of our proposed heuristic algorithm (the documents' score and rank). Figure 2 shows (i) the variation of the returned documents scores based on (A) the benefits, (B) these latter and the user's history related parameters , (C) the equation 1, and (ii) the variation of the returned documents rank based on the different scores (D). We notice that the scores of documents change according to the parameters taken into account (See figure 2/A). According to these scores, the system affects the document ranks in the navigation path (See figure 2/B). The obtained rank based on the benefit considers only the content of documents. Thus, the result navigation path is mostly not adapted to the user's current situation. So, the best navigation path is the one that takes into account at the same time the benefit, the user's navigation history and the used device. In fact, the content of the ASCII document appropriately responds to the user's query (Figure 2/A/a) and it is frequently visited (Figure 2/A/b). But, by taking into account the device, this document takes the position 9 (from 2 to 9; see figure 2/B). This is due to the incompatibility of configurations of device and document. Furthermore, XML document is frequently visited and the configuration of the user's device is compatible with the required one, thus it has the same rank in the three cases (See figure 2/B). Moreover, the XSL document is frequently visited and does not cause a compatibility problem (Figure 2/A/b and 2/A/c). So, its rank is well improved by the user's history (from 5 to 3) and by equation 1 (from 5

[2] http://www.w3.org/TR/xlink11/



	XML	ASCII	SGML	HTML	XSL	ANSI	10B5	10BT	10B2	Vietn	SAMP
■ Benefits of Document	1	2	3	4	5	6	7	8	9	10	11
□ Benefits + User's history	1	2	5	6	3	4	7	8	9	10	11
■ Score (equation1)	1	9	4	5	2	3	6	7	8	11	10

Fig. 2 Evaluation of the first four steps of the proposed algorithm. (A) Variation of the documents'scores given by (a) document benefits, (b) this latter and the user's history and, (c) the equation 1. (B) Variation of the documents'rank.

to 2). This means that the use of the equation 1, and more explicitly the combination of the user's history with the device's configuration, can improve the navigation adaptation.

6 Conclusion

In this article, we have proposed a navigation adaptation method that provides the user with the best navigation path between the returned semi-structured documents meeting his request and taking into account his navigation history and device's configuration. In this method, we also propose the idea of a new adaptation navigation technology which is mainly based on the XLINK extended links. As it is shown in the evaluation, our proposed method reorders documents result according to the current situation of each user in order to minimize wasted time and reduce the disorientation problem. In the continuation of our work, we aim firstly to apply the XLINK extended links as a new adaptive navigation technology and evaluate its use. Secondly, we intend to evaluate our proposed method with more than one user and evaluate their satisfaction. Then, we plan to propose a generic model of the user profile independent of the application area. Finally, we are going to propose and implement a learning method that reduces the profile to the most relevant content.

References

1. Armstrong, R., Freitag, D., Joachims, T., Mitchell, T.: WebWatcher: A learning apprentice for the World Wide Web. In: Proceeding Of AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments, pp. 6–12. AAAI Press (1995)

2. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. In: User Modeling and User-Adapted Interaction, pp. 87–129 (1996)
3. Chiou, C.-K., Tseng, J.C.R., Hwang, G.-J., Heller, S.: An adaptive navigation support system for conducting context-aware ubiquitous learning in museums. Journal: Computers and Education (2010)
4. De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N.: AHA! The Adaptive Hypermedia Architecture. In: Proceedings of the ACM Hypertext Conference (2003)
5. Doerr, C., Dincklage, D.V., Diwan, A.: Simplifying Web Traversals By Recognizing Behavior Patterns. In: HT 2007: Proceedings of the 18th Conference on Hypertext and Hypermedia, pp. 105–114 (2007)
6. Hohl, H., Böcker, H.-D., Gunzenhäuser, R.: Hypadapter: An adaptive hypertext system for exploratory learning and programming. In: User Modeling and User-Adapted Interaction, vol. 6, pp. 131–156 (1996)
7. Knutov, E.: GAF: Generic Adaptation Framework. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 400–404. Springer, Heidelberg (2008)
8. Lassila, O.: Web Metadata: A matter of Semantics. IEEE on Internet Computing 2(4), 30–37 (1998)
9. Seo, J., Diaz, F., Gabrilovich, E., Josifovski, V., Pang, B.: Generalized Link Suggestions via Web Site Clustering. In: WWW 2011: Proceedings of the 20th International Conference on World Wide Web (2011)
10. Verma, S., Patel, S., Abhari, A.: Adaptive web navigation. In: SpringSim 2009 Proceedings of the 2009 Spring Simulation Multiconference (2009)
11. Weber, G., Brusilovsky, P.: ELM-ART: An adaptive versatile system for Web based instruction. International Journal of Artificial Intelligence in Education 12(4), 351–384 (2001)
12. Wanga, Y.-T., Lee, A.J.T.: Mining Web navigation patterns with a path traversal graph. Expert Systems with Applications 38, 7112–7122 (2011)
13. Zayani, C.: Towards an Adaptation of Semi-structured Document Querying. In: CIR 2007 (2007)
14. Zghal Rebai, R., Zayani, C., Amous, I.: MEDI-ADAPT: A distributed architecture for personalized access to heteroneneous semi-structured data. In: WEBIST, pp. 259–263 (2012)
15. Zhu, T., Greiner, R., Haeubl, G.: Learning a model of a web user's interests. In: 9th International Conference on User Modeling (2003)

Relational Schema Summarization: A Context-Oriented Approach

Marcus Sampaio, Jefferson Quesado, and Samarony Barros*

1 Introduction

Query a database by users unfamiliar with its schema can be a challenging test due mainly to the difficulty of understanding dozens or more of possibly poorly designed inter-linked tables, beyond outdated or missing documentation (usability problem). Such users include database developers and sophisticated users: they may eventually need to acquire detailed knowledge of the schema, and then their ability to do so would be greatly improved if they could start with a simplified, easy-to-read schema. Simplified and easy-to-read schemas have been studied within a research direction called database schema summarization [8, 10, 11, 12, 13].

Existing summarization algorithms [10, 11] generate general purpose summaries, ie query-independent. The drawbacks of these solutions seem evident: it is quite possible that a generic summary is not exactly suitable for many types of query, although such a supposition should be confirmed experimentally.

In this paper, we propose a new model of relational schema summarization, based on the notion of context. A context is a subschema of a database schema that is complete for a set of related queries. Our summarization model aims to summarize contexts, so that the summaries are well designed and documented, structurally simple and still complete.

We developed a programming language to generate context summaries as they are formally specified by domain experts. We conducted an experimental

Marcus Sampaio · Jefferson Quesado · Samarony Barros
State University of Ceará, Brazil
e-mail: {sampaio.marcus,jeff.quesado,samarony.barros}@gmail.com

* The authors would like to thank the Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) and Conselho Nacional de Pesquisa (CNPq) for financial support.

study on the usability of a public database with the help of its context summaries, comparing our model with a previous seminal generic method: the gains in usability with our context summaries far surpassed those with generic summaries.

Section 2 is an extensive example of use of our context-oriented summaries. Related work is the subject of Sect. 3. Our summarization model is detailed in Sect. 4. A model solution is presented in Sect. 5. A model evaluation is discussed in Sect. 6. Section 7, Conclusion, closes the paper.

2 Context-Oriented Summaries: An Extensive Example

Consider the well-known database *The Internet Movie Database* (IMDb)[1], having movie as its main entity. Its relational schema is very complex, about 60 tables, and has evolved in an ad hoc way, with inevitable design problems.

In Fig. 1, we see a part of the IMDb schema: it is linked to the rest of the schema from the tables Movies and Actor, as follows: table Movies is referenced by other 22 tables not appearing in Fig. 1 and Actor by 1 table.

It is natural that for a user group with similar information needs about movies, only a small subset of the IMDb schema – a context – is likely to be of interest.

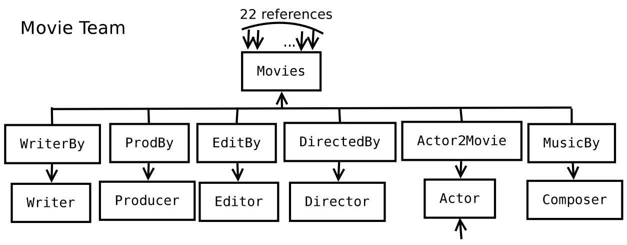


Fig. 1 *Movie Team* IMDb Context

Figure 1 also shows one context: *Movie Team*. Regarding this context, users are interested in movie teams (tables Actor, Composer, Director, Editor, Producer and Writer) and their related entities movies (table Movies).

Basically, the first idea of context is to circumvent the complexity of the underlying database: we can say, for example, that the context *Movie Team* is structurally very simple, compared to the whole IMDb schema. Unfortunately, it still suffers of the idiosyncrasies of IMDb: we limit ourselves to quote one design problem, that is the lack of standardization of names of

[1] Script for creating and loading the database downloaded from: `http://www.dis.uniroma1.it/~degiacom/didattica/semingsoft/seminari-studenti/08-09-09%20-%20SIS%20-%20Valerio%20Del%20Grande%20-%20Junio%20Valerio%20Franchi/DB/`

tables and columns. What was the reason for Movies in plural in the context *Movie Team*, while the names of all other tables are in singular? Why not Movie StarredBy Actor instead of Movies Actor2Movie Actor? Indeed, several other kinds of problems exist, which makes it difficult to use the IMDb.

Other issues: (1) users might be interested only in table projections; (2) as actors, composers, directors, editors, producers and writers are persons, a single table Person could represent them in order to simplify the schema; and (3) lack of proper documentation.

IMDb usability must improve with the summarization of this context. For instance, Fig. 2 is a *Movie Team* summary for the *Movie Team* context in Fig. 1. Note how it is well designed: the entities Movie and Person, represented by their homonym tables, are related by the N:M HasTeam relationship, represented by the homonym table and whose primary key is the composition of the foreign keys codMovie and codPerson. All tables have their names in the singular and their first letters are capitalized. The first letter of the columns is lowercase. A primary key is always a surrogate, and its name is standardized as cod<table>. The summary is much simpler than its context: the table Person is a generalization of 6 tables of the context Actor, Composer, Director, Editor, Writer and Producer; in turn, the table HasTeam is a generalization of 6 context tables: WriterBy, ProdBy, EditBy, DirectedBy, Actor2Movie and MusicBy. The columns of the tables Movie and Person are only those of the *Movie Team* context that were identified as relevant for a certain user group. Very interestingly, the summary tables are abstract, ultimate reason for their simplicity.

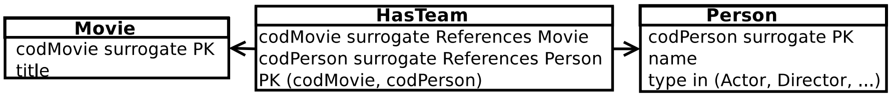

Movie
codMovie surrogate PK
title
HasTeam
codMovie surrogate References Movie
codPerson surrogate References Person
PK (codMovie, codPerson)
Person
codPerson surrogate PK
name
type in (Actor, Director, ...)

Fig. 2 *Movie Team* Summary

The summary in Fig. 2 must be programmed by an expert, using a summary definition language. Basically, the program must contain the specification of the summary tables and the summary-context mapping rules as partially explained in the previous paragraph.

Assuming that the user wants to know the director and actors who worked in the film "The Godfather", he easily formulates his SQL query directly over the *Movie Team* summary, as in Fig. 3 **a**. But this query is not executable. However, with the help of the summary-context mapping rules the user can be guided to the equivalent complex executable query in Fig. 3 **b**. This process – the summary at the first place, then its context – should be more productive than querying directly over the context.

In Sect. 4 we detail our summarization model, while in Sect. 5 we discuss a model solution that assures consistent queries over context summaries.

```
select m.title, p.name, p.ptype
from Movie m, Person p, HasTeam ht
where m.title ="Godfather, The"
and m.codmovie = ht.codmovie
and ht.codperson= p.codperson
and p.ptype in ("Actor", "Director");
```

(a)

```
select m.title, d.name, d.surname,
        a.name, a.surname,
from Movies m, Actor a, Actor2Movie am,
DirectedBy db, Director d
where m.title= "Godfather, The"
and m.id=db.movie
and m.id=am.movie
and am.actor=a.id
and db.directorN=d.name
and db.directorS=d.surname;
```

(b)

Fig. 3 Query over: **a** *Movie Team* Summary **b** *Movie Team* Context

3 Related Work

First of all, let us examine how view and summary have been addressed in the literature. The result of our investigation is synthesized in Table 1.

Table 1 View and Summarization in the Literature

Name	What is	Generation mode	Application
Materialized View[5]	Table schema and its extension	SQL conjunctive query	Query optimization
Materialized View[1]	Table schema and its extension	SQL conjunctive/ aggregate query	Query optimization
Materialized View[2]	Table schema and its extension	Unrestricted SQL query	Query optimization
Restructured View[7]	Table schema and its extension	Extended relational algebra	Query optimization
Restructured View[6]	Table schema and its extension	Conjunctive/ aggregate S-LOG rule	Query optimization
Schema Summary[11]	Importance / coverage oriented schema	Algorithm BalanceSummary	Database usability
Schema Summary[10]	Importance / similarity oriented schema	Algorithm GreedyClus	Database usability

Summaries are much more flexible and powerful than view: they are database schemas *à part entière*, and are generated by specialized algorithms for database usability purposes (see the last two rows of Table 1). The most obvious requirement of a summary is that it 'summarizes', ie the size of a summary[2] should be smaller than the size of the schema that is the object of summarization. But each approach has its specific requirements.

The summaries in [11] and [10] differ in their properties: importance and coverage in the first model, and importance and similarity in the second. Importance of a table in [11] is measured by the number of links to the table and by its cardinality; importance in [10] extends the concept in [11] weighing the contents of the table. Thus, both BalanceSummary algorithm [11] and GreedClus algorithm [10] recognize that the Movies table (see Fig. 1)

[2] The size of a schema is measured by the number of its tables and of its table columns.

is a right candidate to appear on the summarization of IMDb. While BalanceSumary also explores the coverage (via referential links) of an important table, GreedClus searches for other tables that are similar to an important table, according to a distance criterion. The two summarization algorithms may generate summaries with abstract elements (for example, a virtual table that is a cluster of real tables). In common, the two algorithms totally ignore user queries: they implement generic summarization models. It is therefore unclear that they can be useful for many usage scenarios.

The proposals in [8] and [12] deepen the discussion on database usability, but in both the basic summarization algorithm is still BalanceSummary. The work in [13] is not exactly on summarization; however, it discusses important schema summarization principles.

Context-aware database is also a topic related to our work. A context in [9] associates differente context-relevant entities under certain critera. Unfortunately, the model does not support the fundamental requirement of abstract entity.

Unlike the generic summarization models in [11] and [10], our summarization model is targeted for specific summaries, based on the hypothesis that the gains in usability with specific summaries can far surpass those with generic summaries.

4 Summarization Model

In this section, we give details of the two models that build up our summarization framework.

4.1 Context Model

Conceptually, a context is a subset of a conceptual database schema, with these peculiar characteristics: some of the entities are pivotal, while some others are terminal. The remaining entities, if any, are ordinary. Pivotal entities are those of primary interest of the users of the context; a terminal entity is the last entity in a chain of related entities since its pivotal entity, possibly including ordinary entities in the middle. Formally, we have:

Definition 1 (Conceptual Context). A conceptual context CC of a conceptual database CD is a proper subset of the **entities** of CD, ie CC $\subset$ CD. Each entity e $\in$ CC has a type t $\in$ {**pivotal, terminal, ordinary**}.

For each pair (pivotal $\in$ CC, terminal of pivotal $\in$ CC), there exists at least a path of entities and their **relationships** $\rightarrow_{ER}$, in the following sequence: pivotal $\rightarrow_{ER}$ $e_1; \ldots ; e_n$ $\rightarrow_{ER}$ terminal, in which $e_1, \ldots , e_n \in$ CC, $e_i \in \{e_1, \ldots , e_n\}$ is ordinary and $\{e_1, \ldots , e_n\}$ may be $\emptyset$.

Logically, a context is a subset of a relational database schema, in which its *pivotal*, *terminal* and *ordinary* tables respectively represent the pivotal, terminal and ordinary entities of the concerning conceptual context. The remaining tables of the logical context, if any, are referential tables and represent relationships between entities of the conceptual context. Formally, we have:

Definition 2 (Logical Context). A logical context LC of a relational database RD is a proper subset of the **tables** of RD, LC $\subset$ RD, in the following way: (1) each tab $\in$ LC represents either an entity or a relationship between entities of the underlying conceptual context; and (2) each tab $\in$ LC has a type t $\in$ {**pivotal, terminal, ordinary, referential**} depending on the type of the concept that it represents: pivotal entity, terminal entity, ordinary entity or relationship between entities.

For each pair (pivotal $\in$ LC, terminal of pivotal $\in$ LC), there exists at least a path of tables, in the following sequence: pivotal; $tab_1; \ldots; tab_n$; terminal, in which $tab_1, \ldots, tab_n \in$ LC, $tab_i \in \{tab_1, \ldots, tab_n\}$ is ordinary or referential and $\{tab_1, \ldots, tab_n\}$ may be $\emptyset$.

Remember the *Movie Team* context in Sect. 2. Its pivotal entities are Actor, Editor, Director, Writer, Producer and Composer, while Movies is their common terminal entity. The pivotal-terminal paths are: Actor Actor2Movie Movies; Editor EditBy Movies; Director DirectedBy Movies; Writer WriterBy Movies; Producer ProdBy Movies; and Composer MusicBy Movies. There is no ordinary entity.

The corresponding logical context is illustrated in Fig. 1, Sect. 2. Pivotal tables: Actor, Editor, Director, Writer, Producer and Composer. Common terminal table: Movies. Referential tables: Actor2Movie, EditBy, DirectedBy, WriterBy, ProdBy and MusicBy. Pivotal-terminal paths: Actor Actor2Movie Movies; Editor EditBy Movies; Director DirectedBy Movies; Writer WriterBy Movies; Producer ProdBy Movies; and Composer MusicBy Movies.

A logical context must observe this size constraint:

Constraint 1 (Size constraint). *The size of a logical context LC must be smaller than the size of the underlying relational database RD, that is to say,* $|LC|<|RD|$.

Taking into account only the tables, the *Movie Team* context size is 13 ($\ll$ 60).

Syntactically, our notion of context exhibits some similarity with entity template in [4]: however, in a entity template there is only one pivotal table.

4.2 Summary Model

A summary must summarize its context: in this sense, it is context-oriented. The summarization is achieved by means of reductive abstractions of entities in the context. In addition, a summary shall not propagate the possible idiosyncrasies of the design of the context. Otherwise, there is no formal difference between summary and context.

Summary abstractions must be compliant with the mapping constraint and with the structural constraint:

Constraint 2 (Mapping constraint). *Every abstract entity in a summary must be mapped onto concrete entities in its context.*

Constraint 3 (Structural constraint). *Abstract entities must not be structurally more complex than their corresponding entities in contexts.*

Corollary 1. *Given a summary S and its context C, the size of S is less than or equal to the size of C, ie $|S| \leq |C|$.*

Summary design must be compliant with the design constraint as follows:

Constraint 4 (Design constraint). *A summary must comply with best practices of database design* [3].

The constraints 2, 3 and 4 ensure that summaries summarize and are well designed. More precisely:

Well-formed summary. *A summary is well-formed if it is compliant with the mapping, structural and design constraints.*

The reader can easily check that the *Movie Team* summary in Fig. 2 – Sect. 2 – is well-formed.

5 A Model Solution

In this section, we present our algorithm *Summarizer* which generates context-oriented summaries according to the summarization model discussed in Sect. 4.

5.1 System Architeture

The system architecture is shown in Fig. 4.

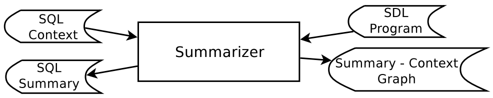


Fig. 4 The *Summarizer* Architecture

Summarizer is an interpreter of our Summary Description Language (SDL). A domain expert formalizes his summary in the form of an SDL program. The inputs to *Summarizer* are: the logical SQL context of the summary and the SDL program. *Summarizer* then interprets the program with the help of the SQL context and produces as outputs: the logical SQL summary and a graph with the mapping rules of the summary onto the context.

5.2 Summary Definition Language

An SDL program basically consists of: (1) logical description of the summary (tables, columns and integrity rules); and (2) as the logical summary must be mapped onto its SQL context.

```
SUMMARY MovieTeam FROM CONTEXT MovieTeam (
  SUMMARY TABLE Person (
    SPECIALIZATION (Actor, Director)
    COLUMNS (
      codPerson IDENTIFIER USE SPECIALIZATION
      name USE SPECIALIZATION.(name || surname)
      ptype ENUM ("actor" MAP Actor, "director" MAP Director)
    )
  )
  SUMMARY TABLE Movie (
    COLUMNS (
      codMovie IDENTIFIER USE Movies
      title MAP Movies.title
    )
  )
  SUMMARY TABLE HasTeam (
    SPECIALIZATION (Actor2Movie, DirectedBy)
      COLUMNS (
        codPerson REFERENCES Person  REF_MAP (
        Actor2Movie.actor REFERENCES Actor.id, DirectedBy.(directorN,
            directorS) REFERENCES Director.(name, surname))
        codMovie REFERENCES Movie REF_MAP ((Actor2Movie,
            DirectedBy).movie REFERENCES Movies.id)
      )
      IDENTIFIER (codMovie, codPerson)
  )
)
```

Fig. 5 An SDL Program (Partial)

Figure 5 is a partial SDL program that specifies the logical *Movie Team* summary together with its mapping rules onto the SQL *Movie Team* context: for brevity and without loss of clarity, we omitted the context tables Composer, Editor, Producer and Writer in the mapping rules. Notice that the context is easily obtained from the SQL schema of the database IMDb (see Definition 2 – Sect. 4.1).

The SDL program starts with the SUMMARY clause, containing the summary and context names. The program continues with three SUMMARY TABLE clauses concerning respectively the tables Person, Movie and HasTeam (see also Fig. 2 – Sect. 2).

In a general way, a summary table specification involves: table name, column names, integrity rules and mapping rules. A mapping rule can be one of these types: SPECIALIZATION, USE, ENUM, MAP and REF_MAP.

The specification of the table Person tells that it is a specialization of the tables Actor and Director in the context (mapping rule SPECIALIZATION). Column codPerson is a surrogate, indicated by IDENTIFIER: the mapping rule USE SPECIALIZATION indicates that it is mapped onto the primary keys of the context tables Actor and Director. The mapping rule for the column name is also USE SPECIALIZATION, indicating now that Person.name is a concatenation of name and surname, in Actor and Director. Finally, for the column ptype the enumeration values actor and director are respectively mapped (MAP) onto the context tables Actor and Director.

The novelty in the specification of the table HasTeam is that each its foreign key (clause REFERENCES) has associated to it a mapping rule REF_MAP, that maps foreign keys in the summary onto foreign keys in the context.

With basis on the mapping rules of a SDL program, Summarizer automatically generates a directed graph Summary-Context as shown partially in Fig. 6 for *Movie Team* summary and context.

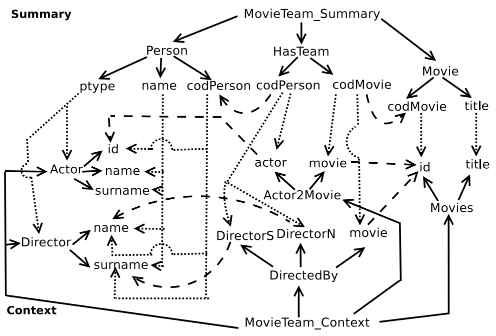


Fig. 6 *Movie Team* Summary-Context Graph (Partial)

In the graph, filled arrows are structural links, dashed arrows are referential links and dotted arrows are mapping links[3].

The upper graph represents the *Movie Team* summary, while the bottom one represents its context. The two parts are connected by mapping links.

All the elements of the graph are found in the SDL program in Fig. 5 and in the SQL context. The need of the SQL context is justified by the fact that identifiers in summaries are always mapped onto primary keys found in contexts (mapping rule IDENTIFIER). Thus, for example, the identifier Person.codPerson is mapped onto Actor.id and Director.(name, surname), where id and (name, surname) are primary keys in the *Movie Team* context.

[3] Our summary-context graph keeps strong similarities with the schema graph in [11]: the differences are mainly syntactical.

5.3 Summarization Methodology

The question that is posed now is: how should a development process be so that a sophisticated user or a user programmer can program their SQL queries over complex databases with quality and low cost, with the help of summaries?

The first thing to remark is that the conceptual and logical designs of summaries are the responsibility of domain experts. The activities of an expert include five steps:

Step 1: Conceptual Entity-Relationship (ER) design of the context;
Step 2: Logical SQL design of the context;
Step 3: Conceptual ER design of the summary;
Step 4: SDL programming: logical summary + summary-context mappings;
Step 5: Run *Summarizer* and save the SQL summary and the summary-context graph;

With regard to the user, the two steps are the following:

Step 6: SQL query over the SQL summary;
Step 7: SQL query over the SQL context with the help of the SQL query over the SQL summary and of the summary-context graph.

Consider that the SQL summary and the summary-context graph generated in the step 5 refer to the *Movie Team* summary and context. Now suppose the user wants to submit to the summary this query *Who are the director and actors in the film* The Godfather? His ultimate goal is to write an executable SQL query over the *Movie Team* context, as in Fig. 3 b. – Sect. 2. First (step 6), he quickly writes a SQL query over the *Movie Team* summary, as in Fig. 3 a. – Sect. 2. Next (step 7) he maps, with the help of the *Movie Team* summary-context graph, the query over the summary onto the executable query over the *Movie Team* context. For this, the user navigates on the graph in Fig. 6 from top (summary) to bottom (context).

6 Experimental Evaluation

The goal of our experiments was to confirm the hypothesis that the gains in database usability with context-oriented summaries can far surpass those with generic summaries (usability hypothesis). For testing, we used the database IMDb, with its 60 tables.

To validate the usability hypothesis, our algorithm *Summarizer* was compared with the algorithm BalanceSummary [11]. The metric chosen was that of [11], called Query Discovery Cost (QDC). QDC is the effort required on the user to locate his query elements in a summary-context graph (Step 7 - Sect. 5.3). We assume that the user 'visits' one element at a time, and charges one unit for each element visited that is not in the query. The QDC for a particular query is the total charge accumulated by the time all elements in the query are visited.

Table 2 synthesizes our comparative QDC study. We generated 5 *Summarizer* summaries (the 5 columns of the middle part on *Summarizer*, Table 2) with their summary-context graphs. For each summary, each one of two lecturers of computer science and three application developers randomly selected 2 exclusive queries in natural language – intention queries – and then calculated the QDC of each intention query (grand total of $5 \times 5 \times 2 = 50$ intention queries). The average QDC values by summary appear in the line "w / summary". The second line, "saving", contains values that indicate how much the QDC with the use of summary-context graphs was lower than without them, ie, using only the IMDb schema for the computation of the QDC.

Table 2 Comparison *Summarizer* versus *BalanceSummary*

Summarizer						
QDC (Avg)	Movie Team	Studio & Distributor	Country	Producer	Actor	All 50 queries
w/ summary	2.4	1.8	1.0	1.1	2.0	1.5
saving	18.1	24.2	43.5	39.6	21.8	25.6
BalanceSummary						
QDC (Avg)	10-size	25-size	40-size	60-size	75-size	
w/ summary	37.6	41.5	28.4	22.5	26.7	
saving	1.2	1.1	1.5	1.9	1.6	

For the tests with BalanceSummary, our version of the algorithm generated 5 generic summaries, with sizes 10, 25, 40, 60 and 75 (the 5 columns of the middle part on BalanceSummary, Table 2): their summary-context graphs were constructed manually, based on the summary-database graph in [11]. The same 50 intention queries were submitted to each one of the BalanceSummary summaries.

To allow a direct comparison of algorithms, the column "All 50 queries" in the part *Summarizer* of Table 2 contains the w / summary and saving averages for all 50 intention queries. This column is directly comparable with the 5 columns in the part BalanceSumary: the advantages of *Summarizer* are always meaningful.

We can conclude that the goal of our experiments has been achieved: the gains in database usability with *Summarizer*, whose model is context-oriented, can far surpass those with BalanceSummary, whose model is generic.

7 Conclusion

Database systems can often have complex and poorly designed schemas. We proposed a relational schema summarization model suitable for database usability, in order to manage these problems of complexity and design. Unlike previous summarization models, our model relies on the idea of context of summarization. A context is a subschema of a database schema that is complete for a set of related queries. Our summaries are then context-oriented. We suggested mapping, structural and design constraints as three relevant constraints by which to judge the quality of a context summary, with regard to database usability. We presented a methodology to develop high quality context summaries, based on our summarization model. The feasibility of our summarization model was evidenced through its empirical evaluation: we are not aware of another context-oriented solution that reached our level of quality of schema summaries.

References

1. Afrati, F., Chirkova, R.: Selecting and Using Views to Compute Aggregate Queries. In: International Conference on Database Theory, pp. 383–397 (2005)
2. Agrawal, S., Chaudhuri, S., Narasayya, V.: Automated Selection of Materialized Views and Indexes for SQL Databases. In: 26th International Conference on Very Large Databases, pp. 496–505 (2000)
3. Blaha, M., Premerlani, W.: Object-Oriented Modeling and Design for Database Applications. Prentice-Hall (2008)
4. Chakaravarthy, V.T., et al.: Efficiently Linking Text Documents with Relevant Structured Information. In: VLDB 2006, pp. 667–678 (2006)
5. Chaudhuri, S., Krishnamurthy, R., Potaminianos, S., Schim, K.: Optimizing Queries with Materialized Views. In: 11th IEEE International Conference on Data Engineering (IEEE ICDE 2011), pp. 190–200 (1995)
6. Chen, D., Chirkova, R., Sadri, F.: Query Optimization Using Restructured Views: Theory and Experiments. Information Systems 34, 353–370 (2009)
7. Cunningham, C., Galindo-Legaria, C.A., Graefe, G.: PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. In: 30th VLDB Conference, pp. 998–1009 (2004)
8. Jagadish, H.V., et al.: Making Database Systems Usable. In: SIGMOD 2007, pp. 13–24 (2007)
9. Roussos, Y., Stavrakas, Y., Pavlaki, V.: Towards a Context-Aware Relational Model. In: Proceeding of the Contextual Representation and Reasoning Workshop of the 5th International and Interdisciplinary Conference on Modeling and Using Context, CONTEXT 2005 (2005)
10. Yang, X., Procopiuc, C.M., Srivastava, D.: Summarizing Relational Databases. In: VLDB 2006 (2009)
11. Yu, C., Jagadish, H.V.: Schema Summarization. In: VLDB 2006 (2006)
12. Yu, C., Jagadish, H.V.: Querying Complex Structured Databases. In: VLDB 2007, pp. 1010–1021 (2007)
13. Wu, W., et al.: Discovering Topical Structures of Databases. In: SIGMOD 2008, pp. 1019–1030 (2008)

Semantic Approach to Cluster Validity Notion

Elena Sivogolovko and Bernhard Thalheim

Abstract. In our research we formulate new concepts of the cluster quality based on semantic point of view. In the presented cluster validity approaches quality of clustering is measured according to correspondence between dataset and cluster structure or some cluster structure properties. Cluster semantic and user interests are not considered. We present a semantic approach to cluster validity and a methodology of its evaluation.

1 Introduction

Clustering techniques are widely used in many different areas, for example in web documents mining, search results representation, ontology matching, personal information mining, decision making and etc. In recent years, a variety of different algorithms - density-based, hierarchical, graph-based and others - were developed in order to perform heterogeneous clustering tasks. With no doubts evaluation of clustering results is an important part of the clustering process. With no quality estimation obtained cluster structure obviously could not be considered as reliable.

Discussions of the clustering quality should begin with a definition of what is it. Obviously, quality itself is very complex notion. In the Data Quality area it is often considered as the measure how well the data fits to their intended use. In Software Quality area the ISO 9000 definition of quality is used:

"The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [2]

Elena Sivogolovko
Saint-Petersburg State University
e-mail: efecca@gmail.com

Bernhard Thalheim
Institut für Informatik der Christian-Albrechts-Universität zu Kiel
e-mail: thalheim@saturn.is.informatik.uni-kiel.de

In comparison with such definitions the existing cluster quality concepts look one-sided. For example:

We define the "optimal" clustering scheme as the outcome of running a clustering algorithm (i.e. partitioning) that best fits the inherent partitions of the data set [7]

and

The adequacy of a clustering structure refers to the sense in which the clustering structure provides true information about the data, or the ability of recovered structure to reflect the intrinsic character of the data [9].

According to these definitions the question of cluster validity can be listed as "How well does the obtained cluster structure fit the inherent partitions of the dataset?" These definitions take into account only the correspondence between data set structure and cluster structure but not the user needs or some semantic reasons. We suppose that cluster validity should be considered from the user point of view and cluster semantic is also meaningful for cluster quality estimation. In general, data mining aims to extract some knowledge from the given dataset. Consequently, the question of cluster quality can be reformulated as "What knowledge was extracted from the dataset by this clustering scheme?" We called this semantic approach to cluster validity. In our work we describe the methodology how semantic cluster validity can be defined and measured.

2 Background

In conceptual modelling area the notion of quality is often divided on three different concepts. Namely: a syntactic quality, a semantic quality and a pragmatic quality. The syntactic quality is how well the model corresponds to the language. The semantic quality is how well the model corresponds to the domain. The pragmatic quality is how well the model corresponds to its audience interpretation [11]. We adapt these notions to the cluster validity. We consider the syntactic cluster quality as quality which can be measured according to some cluster structure features or given dataset features as well. For example, well known criteria of the cluster structure quality, which are often used in the relative approach to the cluster validity, can be listed as following [4].

- **Compactness:** The member of each cluster should be as close to each other as possible.
- **Separation:** The clusters themselves should be widely separated.

These criteria are used if we want to choice one "the most optimal" cluster structures from several ones. They are derived from the clustering task definition: to organize a collection of data items into clusters, such that items within a cluster are more "similar" to each other than they are to items in the other clusters. One can see, that the compactness represents the similarity between elements in the same clusters and the separation represents the similarity between elements in different clusters correspondingly. This cluster quality definition correlates with the quality notion, which was described in the previous section in a following way: a clustering algorithm should construct a cluster structure which should fit a inner structure of a given dataset and its clusters should be as compact and separate as possible. Different cluster validity indexes measured the compactness and the separation in different ways. For example, in the Dunn index [6], which was constructed for crisp and non-hierarchical cluster structures, the compactness is represented as a cluster diameter:

$$diam(c_i) = \max_{x,y \in c_i} \|x - y\|$$

and the separation is considered as the distance between the most closest elements

$$d(c_i, c_j) = min_{x \in c_i, y \in c_j} \|x - y\|$$

$diam(c_i)$. A general index formula can be listed as

$$D = min_{i,j \in \{1...c\}, i \neq j} \left\{ \frac{d(c_i, c_j)}{\max_{k \in \{1 \cdots c\}} diam(c_k)} \right\}$$

For well-separated and compact clusters the distance between clusters is expected to be large and the diameter of a cluster is expected to be small. Thus, according to Dunn index definition, large values of the index indicate a “good” cluster structure.

Another approach to cluster quality, which is usually used in external and internal cases, when user wants to validate an obtained cluster structure by itself, considers the cluster quality from the statistic point of view. A cluster structure is valid if it is ”unusual” in some sense [9]. In general, clustering can be considered as retrieval of some ”unusual” patterns in the data, and described approach to the cluster quality is based on this clustering representation. In this case the question of cluster validity can be formulated as: “Is the obtained cluster structure better than some random cluster structure of the same dataset?” According to this approach, the process of validity evaluation can be described follows:

1. Select some cluster validity statistic, for example, Hubert Γ Statistic [9]:

$$\Gamma = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} X(i,j) Y(i,j)$$

   where $X(i,j)$ is the proximity matrix of a given dataset and $Y(i,j)$ can be considered as [7]

$$Y(i,j) = \begin{cases} 1 \text{ , if i and j belong to the same cluster} \\ 0 \text{ , otherwise} \end{cases}$$

2. Select some random cluster hypothesis H_0
3. Select some significance level α
4. Model validity index distribution on the given dataset by, for example, Monte-Carlo method
5. Check that obtained cluster structure is ”unusual” according to the accepted significance level.

As one can see, user and user needs are not mentioned in these cluster validity concepts. It is implicit that user wants to find right cluster structure in a given dataset. In many cases it is definitely right and syntactic quality approach is quite enough for cluster validity estimation. However sometimes user does not want to find real cluster structure, he wants to construct some cluster structure model, which fits his needs, for example, he may want to construct not all cluster structure, but just a part of it or structure, which should satisfy some special conditions. User may have some concept of different clusters or cluster structure like, for example, in conceptual clustering [13]. In this case syntactic validity methods may evaluate obtained model as “not valid” just because it does not completely fit to a dataset structure or syntactic quality criteria. Therefore we suppose that cluster validity should be considered from the user point of view. In this case evaluation of clustering results can help the analysts to understand is obtained cluster structure really useful for further usage or not.

3 Semantic Quality

As it was mentioned above, in general, data mining aims to extract some knowledge from a given dataset and the question of cluster quality can be reformulated as “What knowledge was extracted from the dataset by this clustering scheme?”. The knowledge is another very complex term. Actually, nobody knows how to define it. The concept of knowledge is composed of multiple components, aspects and conditions. The core component unit of knowledge is the information. The required information can be qualified as knowledge, if the information is true by certain notion of “truth”, it is reusable in a rule system for new information, and it is not equivalent to other information, which can be generated with the aid of facts or preliminary information in the particular inventory of knowledge by a rule system.

Knowledge underlies a learning process, in which the current information is gathered, accepted, validated on the basis of facts and preliminary information by a rule system, compared contingently and stored in the particular inventory of knowledge. We consider the requested knowledge of users as a composition of understanding and information demand, whereby the information demand is an aggregated component of life cases, motivation, intention and quality.

Because we try to represent the cluster quality as a kind of knowledge quality we need to illustrate the knowledge quality notion too. In general, we consider it as shown at Fig. 1 [10]. We adapted some knowledge quality dimensions to the cluster validity.

The semantics has many different definitions in different research areas. According to the Cambridge Encyclopedia of Language, semantics is the study of meaning, i.e. how meaning is constructed, interpreted, clarified, obscured, illustrated, simplified, negotiated, contradicted and paraphrased [16]. We use the term “semantics” as some equivalent of “meaning”. In conceptual model quality, the semantic quality means the degree of correspondence between the model and the domain [11], we

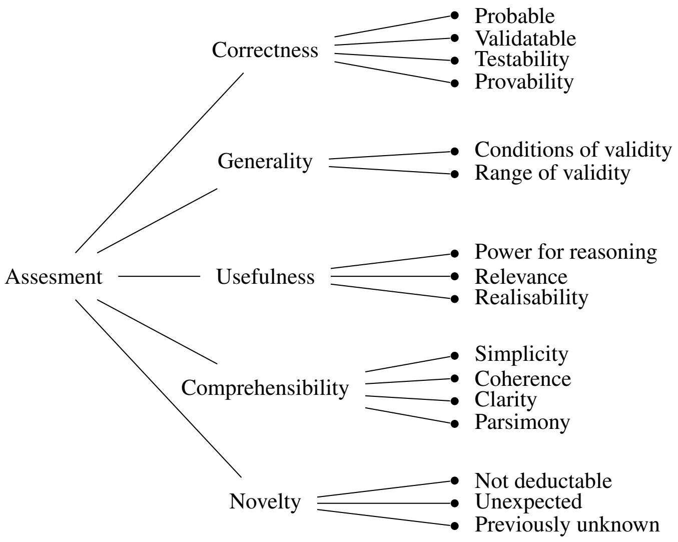


Fig. 1 Knowledge quality in general

consider the semantic quality as the difference between the knowledge which can be extracted from an obtained cluster structure and the all knowledge in the dataset or the user knowledge. Here "user" is considered as a role: as a group of persons or a community with similar knowledge.

3.1 Cluster Model

Before a presentation of the semantic quality model we should say a few words about a cluster model. Sometimes the cluster model notion is used as equivalent of a cluster structure. We suppose that in case of the cluster quality estimation this notion should be wider. We considered the cluster model as the concept which describes all information about clustering process. In our understanding, the cluster model consists of following parts: 1) data representation method (for example, tf-idf vectors for text documents), 2) similarity metric (euclidean metric, cosine metric, etc.), 3) clustering algorithm, 4) clustering algorithm parameters, 5) obtained cluster structure. Note that in some cases (for example for DBScan algorithm) the first four parts uniquely determine the fifth one — cluster structure, but in some other cases (for example for KMeans algorithm) the final cluster structure also depends on a random distribution of cluster centroids on initial stage and can not be determined by the previous parts of the model. Relationships between parts of cluster model are shown on Fig. 2. We suggest that all these parts have significant impact on the final cluster quality in both semantic and syntactical cases. Usually analysts concentrate on an algorithm and algorithm parameters impact only, but a data representation and

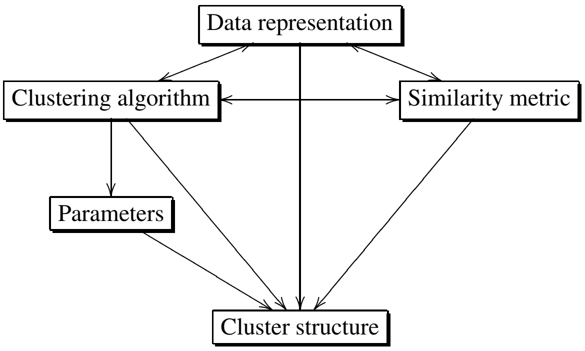


Fig. 2 Cluster model

a similarity metric should also be taken into account, because wrong choice on this stage could "break" all clustering process.

3.2 Semantic Quality Model

Our semantic cluster quality model consists of the following parts: Dataset knowledge, Cluster knowledge, Cluster model (which was defined in the previous section), extraction process (the process of the knowledge extraction from cluster model), User knowledge and validity conditions. Let's describe the following "sets":

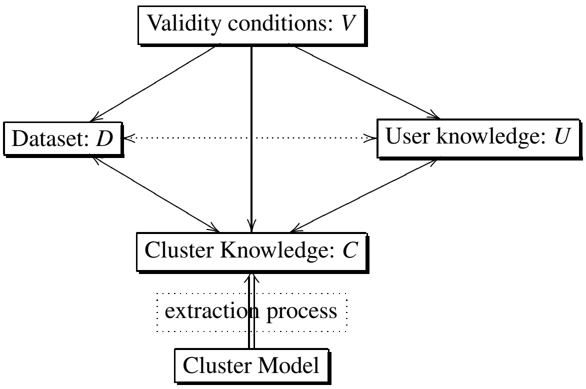


Fig. 3 Semantic quality diagram

1. D — all knowledge which is contained in the dataset and can be extracted by clustering.
2. C — all knowledge which is contained in the constructed cluster model

3. U — user knowledge. It consists of three different parts: $U = U_K \cup U_G \cup U_E$, where U_K is a current user knowledge about the dataset, U_G is the user needs and U_E is the expectations (something that user can expect and explain according to his knowledge).
4. V — validity conditions.

A general scheme of our semantic validity model is presented on Fig. 3. As it was described at the beginning of this section, we consider the semantic quality as the degree of correspondence between C and D or U. According to these notions and general notions of the knowledge quality we conceptualize our quality notion in terms of data quality dimensions, like in the data quality area. Their definitions are listed below and the suggested way of theirs measurement are presented in Sect. 3.3.

Incompleteness: $D \cap C$. In the ideal case, clustering should extract all knowledge, which can be extracted. $D \backslash C \neq \varnothing$ means that some information was not extracted by the current clustering model. Obviously, this notion can be used only if some benchmark dataset is given and all D is known.

Novelty: $C \cap U_K$. Here $C \subseteq U_K$ means that the extracted information is already known. Sometimes it is not an "error". If the user wants to confirm some known information about the dataset and he can do it with the obtained clustering model - it means that the model fits the user needs. Otherwise, if the goal is to obtain some new knowledge, no new information is an "error". $C \backslash U_K$ means how "much" new knowledge was obtained from the cluster model.

Interpretability: $(C \cap U) \subset U_E$ represents the degree of understandability of the clustering model. The worst case is if $U_E \cap C = \varnothing$. This means that user obtained some clustering model, but he can not explain it. The model is completely meaningless for the user.

Relevance: $U_G \cap C$ is the measure of relevance. The worst relevance case is if $U_G \cap C = \varnothing$. The obtained cluster model can not be used for the user needs. Maybe user can explain it but for a current goal he needs some other model. Therefore it is an "error". In some cases, additional restriction can be added in this quality dimension: $(U_G \cap C) \nsubseteq U_K$ — the extracted knowledge is relevant, if user needs it and does not know it yet.

In these quality dimensions we tried to combine the knowledge quality notions with the user needs and the clustering goals.

3.2.1 Discussion

Besides four main semantic quality dimensions we formulate four additional dimensions. Theirs definitions are not really strong and theirs measurement can be a good point for the future work, because at the moment it was not developed.

Validity: consists of two different parts. First of them considers the correspondence between C and V. It is obvious that cluster model should not used if it does not satisfy the general validity conditions. The second part of validity dimension considers the correspondence between C and U_E. If user explains something which contradicts to obtained cluster knowledge, this means that probably our cluster model is not valid.

Coherence: can be considered as the measure of contradiction. The knowledge extracted form different clusters or group of clusters should be coherent and also extracted knowledge C should be coherent with U_K.

Comprehensibility: might be a property of the user. Whenever the user knows something from the clustering or data set then the user can explain that. Also it can be reformulated as "How difficult is to extract and explain the knowledge from the cluster structure?"

Testability: How difficult is to test cluster structure for correctness? Here the correctness is considered in both syntactical and semantic senses.

Also we should note the following: in our model it is difficult to separate the cluster quality from the extraction process quality. We consider not the cluster structure as is, but the extracted knowledge and the extraction process has some impact on this knowledge. Some errors could be caused by the extraction and not by the clustering process.

In this work we suggest that cluster analyst and user have the same understanding, terminology, learning process and etc. In general, analyst and user can easily use, for example, a different terminology and in this case constructed C and U could be hardly compatible. For example, user wants to obtain two clusters with labels "cold water" and "warm water", but analyst obtained two clusters with labels "dirty water" and "clean water". Was this a cluster algorithm error or an extraction process error or just different terminology? Actually, we can not answer this question in our model, we can just signalize, that probably we hae some errors.

It is easy to see, that in our model the user is always right. We can not establish that there are some errors in the user knowledge but not in the clustering one.

3.3 Semantic Quality Measurement

The core point of our semantic quality representation is the knowledge representation and knowledge measurement. We suggest using the Resource Description Framework (RDF) [12] for knowledge representation. The RDF was developed for representing information about resources in the World Wide Web. Nowadays it has been widely accepted as a standard for semantic representation for the next generation of the Web. In general, this framework can be used not only for Internet resources description but also for any knowledge representation too. Therefore we can represent D, C and U sets with help of RDF. The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object (Fig. 4). A set of such triples is called an RDF graph. Any knowledge, which core elements can be presented as RDF triples, can be presented as RDF graph.

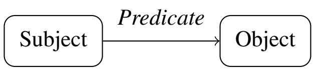


Fig. 4 RDF triple

Several frameworks (for example Jena [1]) are developed for RDF graphs construction. Figures 5 and 6 show the knowledge about the dataset from user point of view. The user knows, that there are two clusters in the dataset (Fig. 5) and the user wants to obtain some representatives for these clusters (Fig 6).

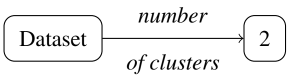


Fig. 5 U_K — user known information

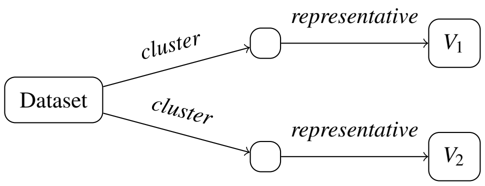


Fig. 6 U_G — user goals

As it listed before in Sect. 3.2 we consider the cluster quality as the degree of correspondence between two knowledges, for example D and C. If both of these knowledges are presented in the RDF graph form, the cluster quality measurement can be considered as graph matching or sub-graph isomorphism or graph similarity measurement tasks.

We suggest two different ways for the semantic quality measurement.

First of them is Ontology Matching. In computer science, Ontologies are considered as formal, explicit specifications of shared conceptualizations of a given domain of discourse. Generally, an ontology for a domain contains a description of important concepts, properties of each concept as well as restrictions and axioms upon properties. Ontologies are often used to provide interpretation to the contents of the Semantic Web data. RDF is also used as ontologies representation language in the Research Community. In recent years, several Ontology Matching methods were presented for RDF graph matching [3], [5], [8], [17]. In this case compared knowledges are considered as two different ontologies, and ontology matching between them is constructed. If graphs can not be matched exactly, then sub-graph isomorphism should be constructed. After that this matching is manually checked by an analyst. Therefore this method of semantic validity measurement could be quite expensive.

The second approach is the semantic similarity metric on RDF graphs. For example, SemMF [14] or VSM Semantic Similarity [15].

3.4 Methodology

In this work we do not consider the problem how to combine semantic and syntactic quality methods in one methodology. We suggest the methodology for semantic cluster validity measurement and it is listed as method 1. For different users or user communities different quality dimensions can be important. For example, for accountants the most important dimension is Relevance, but for scientists it can be Novelty. Therefore we did not put all dimensions in one general formula.

Algorithm 1. Semantic cluster quality measurement

```
Require: $C$ — cluster model
  Model knowledge: extract all knowledge $C$ from the obtained cluster structure and construct RDF tree representation $T_C$ for it.
  if Dataset is a benchmark then
    Dataset knowledge: represent all knowledge about Dataset in RDF: $T_D$
    Calculate $Incompletemess = Sim(T_D, T_C)$
  end if
  User Knowledge: define all user known information about dataset and represent it as $T_{U_K}$
  Calculate $Novelty = Sim(T_{U_K}, T_C)$
  Expectations: define all user theories or expectations about information which can be obtained form data set by clustering and represent them as RDF graph $T_{U_E}$
  Calculate $Incomprehensibility = Sim(T_{U_E}, T_C)$
  Goals: define user goals and interests and represent them as RDF tree $T_{U_G}$
  Calculate $Usefulness = Sim(T_{U_G}, T_C)$
```

4 Conclusion

In our work we described the new approach to the cluster validity. We suggest to differ the syntactic cluster quality, which based on correspondence between the cluster structure and the dataset structure, from the semantic cluster quality which based on difference between extracted knowledge and user needs. We present the concept of semantic cluster validity and introduce different dimensions of the semantic validity. Four of these dimensions – Incompleteness, Novelty, Interpretability and Relevance – were presented as measurable and the method of theirs measurement based on the RDF graphs similarity was suggested.

The implementation of our methodology and its approbation on artificial and real-world datasets can be a good point for future work.

Another part of future work can be the pragmatical quality definition. In this work we considered user as role but not as an actual person, and the quality notions for the actual people should be also studied.

References

1. http://jena.sourceforge.net/
2. Iso standard 9000-2000: Quality management systems: Fundamentals and vocabulary (2000)
3. Albagli, S., Ben-Eliyahu-Zohary, R., Shimony, S.E.: Markov network based ontology matching. In: Proc. The 21st International Joint Conference on Artifical Intelligence IJCAI (2009)
4. Berry, M., Linoff, G.: Data Mining Techniques For Marketing, Sales and Customer Support. John Wiley & Sons, Inc. (1996)
5. Doshi, P., Kolli, R., Thomas, C.: Inexact matching of ontology graphs using expectation-maximization. Journal Web Semantics: Science, Services and Agents on the World Wide Web 7(2) (2009)
6. Dunn, J.: Well separated clusters and optimal fuzzy-partitions. Journal of Cybernetics 4, 95–104 (1974)
7. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Intelligent Information Systems Journal 17, 107–145 (2001)
8. Hu, W., Jian, N., Qu, Y., Wang, Y.: Gmo: A graph matching for ontologies. In: K-Cap 2005 Workshop on Integrating Ontologies (2005)
9. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall (1988)
10. Kidawara, Y., Zettsu, K., Kiyoki, Y., Jannaschk, K., Thalheim, B., Linna, P., Jaakkola, H., Duzí, M.: Knowledge modeling, management and utilization towards next generation web. In: Proc. of the 2010 Conference on Information Modelling and Knowledge Bases XXI (2010)
11. Lindland, O.I., Sindre, G., Solvberg, A.: Understanding quality in conceptual modelling. IEEE Software, 42–49 (1994)
12. Manola, F., Miller, E. (eds.): W3C Recommendation, chap. RDF Primer (2004), http://www.w3.org/TR/rdf-primer/
13. Michalski, R.S., Stepp, R.E.: Learning from observation: Conceptual clustering. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine Learning: An Artificial Intelligence Approach, Tioga, ch.11, pp. 331–364 (1983)
14. Oldakowski, R., Bizer, C.: Semmf: A framework for calculating semantic similarity of objects represented as rdf graphs. In: Poster at the 4th International Semantic Web Conference, ISWC 2005 (2005)
15. Tous, R., Delgado, J.: A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 307–316. Springer, Heidelberg (2006)
16. Wanner, P. (ed.): The Cambridge Encyclopedia of Language. Cambridge University Press (1987)
17. Zhang, R., Wang, Y., Wang, J.: Research on ontology matching approach in semantic web. In: Proc. International Conference on Internet Computing in Science and Engineering ICICSE 2008, pp. 254–257 (2008)

An Automated Approach of Designing Multiplex PCR Primers for the Amplification of Exons

Adam Skowron and Rafal Pokrzywa

Abstract. This paper presents a new program to design specific and reliable primers for amplification of exons in a multiplex PCR. The program is composed of two components: a data acquisition component and a data processing component. The data acquisition component automates time–consuming steps of preparing data for the primers design algorithms. It provides up-to-date information about selected genes including coding sequences and locations of SNPs. The data processing component automates the process of grouping candidate primers into an optimal set for the multiplex PCR by checking their specificity. It is an original and unconventional method that combines the hierarchical clustering and the separate-and-conquer strategy. The results obtained for various genes and presented in this paper prove that the proposed method is an effective way to design an optimal set of primers for the multiplex PCR.

Keywords: multiplex PCR, primer design, primers specificity.

1 Introduction

Databases are extensively used to collect and process data coming from various biological experiments (DDBJ, EMBL, GenBank, Entrez, UniProt, Swiss-Prot etc.). The rapid increase in the number and size of biological databases (Figure 2) motivates the development of new information systems integrating various databases and extending their functionality (e.g. BioMart) [13, 8, 11]. To extract the most relevant information from the huge amount of stored and shared records more and more advanced methods of knowledge discovery are needed. In recent years, the

Adam Skowron · Rafal Pokrzywa
Institute of Informatics, Silesian University of Technology, Akademicka 16, Gliwice, Poland
e-mail: {adam.skowron,rafal.pokrzywa}@polsl.pl

most popular are the methods of data mining commonly applied for example for pattern recognition, prediction, classification and clustering [1].

Primers design for a multiplex polymerase chain reaction (multiplex PCR) is a biological problem that requires a combination of databases, tools and methods of data mining. While in the standard PCR the amplification focuses on one region of double-stranded DNA the aim of the multiplex PCR is to simultaneously amplify multiple regions of DNA in a single tube, which helps to reduce costs and time of an experiment [19, 3] (Figure 1). However, the process of the multiplex PCR is more complex than the singleplex PCR, where the amplification focuses only on one region of DNA at the same time. The key problem in both variants of PCR is to design optimal and specific primers (two short nucleic acid sequences complementary to double-stranded DNA that serve as the starting point of the PCR process) [10, 12, 21, 7]. Although the conditions for designing primers for singleplex PCR are well known they are not sufficient for the multiplex PCR [21, 6]. The process of designing primers for the multiplex PCR is more advanced and time consuming [7, 4]. It requires application of data mining techniques for extensive analysis of large amount of data coming from various biological databases. One should also consider limitations of the complex nature of the multiplex PCR process in which several regions of DNA must be amplified simultaneously and independently of each other [19, 10, 17]. To make this feasible, all possible interactions between the primers need to be checked and all primers that are able to work without any competition or interaction must be combined in sets[19, 20, 23, 27]. In general, the reliable application for designing primers for the multiplex PCR should consider three main aspects: designing primers as good as for the singleplex PCR, excluding incompatible primers (interacting with other primers and giving the non-specific products) and grouping compatible primers into an optimal set.

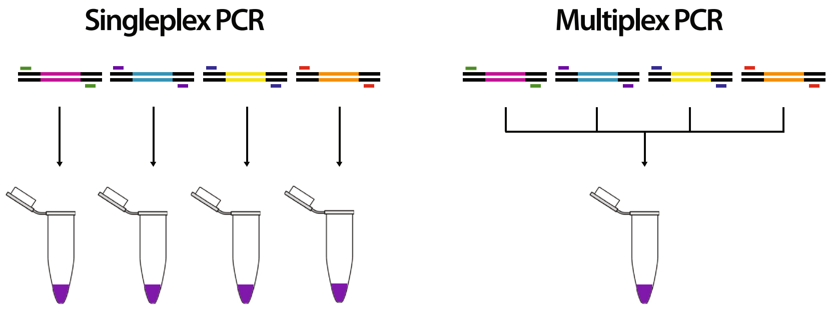


Fig. 1 A general outline of the PCR process and the difference between singleplex and multiplex reactions. The main objective of both PCR reactions is the multiplication of the double-stranded DNA. Crucial to this are short and specific sequences of nucleic acid called primers. In the case of multiplex PCR primers are combined in groups therefore, it is important that the primers in the groups can not interfere with each other.

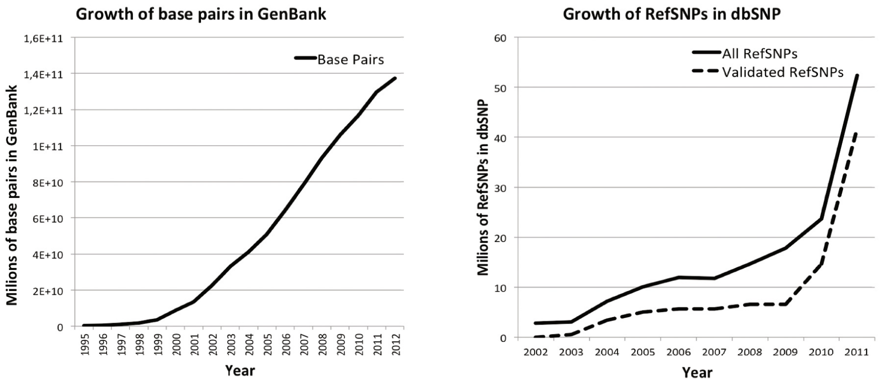


Fig. 2 The example of rapid increase in the number of the data in biological databases. In the case of GenBank database the exponential growth continues, while in the dbSNP database the strong growth has just begun.

To deal with these limitations and conditions we present a new approach to the problem of multiplex PCR primers design that integrates several databases, services and methods. These are: the database of human genes from the NCBI (to obtain data about genes, exons and sequences), the BioMart service (to retrieve information about single nucleotide polymorphisms, SNPs) [11], the Primer3Plus program (to design candidate primers) [22, 25], the Smith-Waterman algorithm (to avoid cross-reactivity between primers) [24] and the BLAST program (to avoid reactions with more than one region of DNA – primers specificity) [2]. The proposed program uses a popular method of data mining called hierarchical clustering combined with the separate-and-conquer strategy [5, 9]. It is an original and unconventional way of computing the optimal set of primers for the multiplex PCR.

The main goal of this work is to design a group of primers for the multiplex PCR that is optimal for two proposed measures: grouping efficiency and reduction of primers.

The grouping efficiency is denoted as a formula:

$$efficiency = (1 - ng/nspp) * 100\% \tag{1}$$

where ng is a number of created groups and nspp is a number of specific parts of the coding sequence. We have used the formulation of the number of parts of the coding sequence, because for some reason long coding sequences were divided into smaller parts that were treated as separate regions.

On the other hand the reduction is denoted as a formula:

$$reduction = (1 - sp/p) * 100\% \tag{2}$$

where sp is a number of specific parts with primers and p is a number of all parts.

The paper starts with a comparison of related works in Section 2. Section 3 describes the process of data acquisition including finding genes and exons, assigning SNPs and designing primers. The automated approach of grouping primers for the multiplex PCR using two methods of data mining (the hierarchical clustering and the separate-and-conquer strategy) is presented in Section 4. Section 5 presents the results of experiments including the grouping efficiency for various genes. The paper ends with a conclusion and a discussion of further work.

2 Related Work

There are a few applications that deal with the problem of designing primers for the multiplex PCR. However, these applications mostly focus on the process of grouping primers into optimal sets and do not provide any support for data acquisition and preprocessing that is equally important. Moreover, according to our knowledge, even if the applications use some databases, they usually base on certain (typically taken during the process of writing of the application) versions of the databases. However, in the case of frequently updated biological data, those applications do not consider the latest scientific discoveries, especially in relation to a growing number of locations of SNPs [11].

To group primers that are able to work together the applications use different approaches. Method based on the graph theory is commonly used by several applications for designing primers [19, 20, 23]. MuPlex uses a graph, which has nodes with multiple states and all nodes that are compatible with each other are marked in the matrix E_{xy} with value 1, otherwise 0. MuPlex searches the state assignment matrix to achieve maximal covering with disjoint cliques [20, 19]. Although the MuPlex verifies the specificity of the primers, it uses BLAST-like alignment tool (BLAT) in place of Basic Local Alignment Search Tool (BLAST) which may produce non-specific primers [16, 2]. Apart from checking all interactions, the state matrix approach in the MuPlex application seems similar to our method, but it is limited only to the discrete state. Our program additionally allows to use the function describing compatibility in a continuous manner. The second application that uses graphs is MPprimer, which utilizes the Primer3 program to design singleplex PCR candidate primers and applies the MPEprimer to verify the specificity of the candidates [23, 18]. To determine the optimal set of primers for multiplex PCR MPprimer uses the graph-expanding approach, which is different to MuPlex where the whole matrix of state is searched [20, 19, 23].

PrimerStation is another program that designs human-specific primers for multiplex PCR. PrimerStation looks for primers that have similar executable temperature [27]. It uses precomputed primers that are in some cases outdated as the size and the contents of related biological databases have changed.

Other applications for multiplex PCR primers design such as G-PRIMER, PDA-MS/UniQ or Greene SCPrimer focus on the problem of a minimum primer set and pay no attention to study the relationship between primers and their specificity [26, 15, 14]. Therefore, they are applicable only for specific cases.

3 Data Acquisition

The rapid increase of the amount of biological data motivates the development of new information systems, which automate the process of data acquisition and integration. Our approach follows this trend.

Data acquisition is the first step in the process of designing primers for the multiplex PCR. The workflow of this step is presented in Figure 3. The process (Step 1) is initiated by the user entering the name of the gene. Then the program calls the Entrez query and database system at the National Center for Biotechnology Information (NCBI). Our program uses the Entrez Programming Utilities (E-utilities) to access the Entrez system. E-utilities is the set of server-side programs accessible over HTTP or SAOP protocol. First, the ESearch utility is called (Step 2), which returns a XML document with the list of best hits (genes IDs) in GenBank database. The list of gene IDs is then sent to the ESummary utility (Step 3) with the request for additional information (e.g. name, description, aliases, position on the chromosome etc.). In Step 4, the information about the selected gene is used to prepare a final query to EFetch utility. It returns the complete entry of the selected gene, including the sequence and the positions of coding regions (exons). These data is then used for PCR primers design. It is worth noting that one of the adopted requirement is to design primers allowing amplification of the entire exon sequence. This restriction forces us to retrieve the sequence of exon with extra flank regions of nucleotides from both sides (additional fragments of DNA). The influence of the size of flank regions on the final results will be discussed later. In Step 5, the user selects exons for the amplification. Steps from 1 to 5 can be repeated several times, until all coding regions (possibly of different genes) for the biological experiment are selected. The next steps of the proposed algorithm are performed automatically without the user interaction.

One of the most important parts of the designed system and probably the first such solution is fully automated method of retrieving information about locations of SNPs. Information about positions of SNPs is used to avoid designing primers in these locations, because these primers could not work with the DNA of the person who has a given mutation. For the purpose of automating, we use the BioMart service, which provides comprehensive access to the dbSNP and allows to obtain a complete list of SNPs locations based on the chromosome number and the selected positions on this chromosome (interval from-to). The query is written in XML language and it is sent to BioMart using RESTful Access [11]. The output is a list of SNP locations. One of the difficulties in merging sequence data from GenBank with locations of SNPs from BioMart is different positioning. The BioMart service requires absolute locations of the selected positions in relation to chromosome while

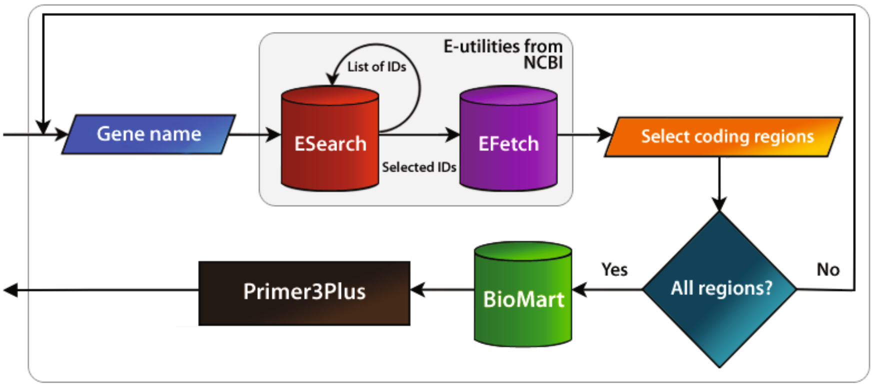


Fig. 3 The workflow od data acquisition

coding regions from GenBank database have the relative positions compared to the same chromosome.

In the final step of data acquisition we use the Primer3Plus program to design candidate primers for PCR. The query is prepared using the data obtained earlier and it is sent over HTTP protocol. The key elements of the query are: the sequence of the exon with extra flank regions, the relative positions of SNPs, the locations of the coding region (to ensure that the entire region will be covered) and additional settings for the program like the number of expected results.

The size of the flank regions of the exon affects the process of designing candidate PCR primers with the Primer3Plus program [25]. This is because the program actually tries to design primers inside the flank regions as the result of the adopted requirements (fragments of the sequence that are complementary to the primers are not included in the product of amplification). The PCR primers usually have a length

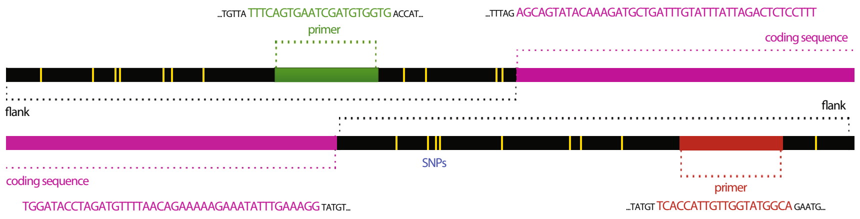

...TGTTA TTTCAGTGAATCGATGTGGTG ACCAT...
primer
...TTTAG AGCAGTATACAAAGATGCTGATTTGTATTTATTAGACTCTCCTTT
coding sequence
flank
flank
SNPs
coding sequence
primer
TGGATACCTAGATGTTTTAACAGAAAAAGAAATATTTGAAAGG TATGT...
...TATGT TCACCATTGTTGGTATGGCA GAATG...

Fig. 4 The graphical interpretation of the problem of designing primers including examples of primers, flanks, SNP locations and the coding region. In the figure the long sequence was divided in half, so the upper part should be understood as the left part and the lower as the right respectively. Additionally, the double-stranded DNA was combined into a one line, but keep in mind that the primers are single-stranded.

of 18 to 27 nucleotides but the flank regions are full of forbidden places (e.g. SNPs locations) that may occur in small intervals from each other. Therefore, to increase the likelihood of designing optimal primers the size of the flank regions should be properly chosen.

The second key element of the query for the program is the value of expected results. This parameter affects the processing step and will be discussed later. It is worth mentioning that the bigger value has a positive effect, because many primers are very similar (the difference is in the 1-2 characters/nucleotides). A greater value of parameter is equivalent to greater diversity of primers, but only by increasing the frequencies.

4 Data Processing

Data processing begins when all the candidate primers are obtained. Then we use local installation of the BLAST program to check the specificity of candidate primers (to check that candidate primers do not cover more than one region of DNA). The Basic Local Alignment Search Tool (BLAST) it the most popular program for searching regions of local similarity between sequences [2]. In the proposed approach the database of human genome is used, but the solution can be easily extended to other organisms. The human genome is downloaded from NCBI servers and is converted from the FASTA format into a BLAST database. The output of BLAST (we are interested in the part containing matches of each primer to the genome sequence called hits) can be in many formats (XML, text) but the best performance is achieved using the text format and the efficient parser. This is because the input of the program has many sequences and the output contains a lot of data, whose size can easily exceeds hundreds of megabytes. It seems that a compromise between the file size and the performance of using the BLAST is to run the program separately for each of the selected coding regions. In other words, in our approach the number of primers corresponds to the value of the parameter of the expected results from the Primer3Plus program.

The goal of checking specificity step is to discard candidate primers that are non-specific, that is, they cover more than one region of DNA. Primers specificity is checked by examining matches to the human genome of each candidate primer. To ensure that the primers are specific the BLAST program is run with the parameter E-value equal to 7 (that returns also less significant hits, that is, hits with more mismatches to the checked sequence). The key assumption in the analysis of tens of thousands of hits is to ensure the absence of occurrences of matches within a certain distance from each other. The assumption is that only these primers can be assumed to be specific, which give only one product with a desired size but the products of much larger size (in proposed approach more than 3000 characters) do not impinge on the specificity.

The next and the most important part of the data processing is to assign a set of candidate primers into groups in the way that all primers within one group can work together. For this purpose, the hierarchical clustering and the separate-and-conquer strategy are used. In the hierarchical clustering in order to decide which primers can be combined the upper or lower triangular matrix of distance is required. The matrix is composed for all obtained and non-rejected primers. The cells of this matrix represent the relationship between corresponding primers. The value of each cell is calculated using the distance function and is equal to 0, if primers are able to work in one group without affecting each other, otherwise the distance is equal to 1. The distance function checks the complementarity of primers and the cross specificity. None of the conditions should be met.

The complementarity is calculated using the Smith-Waterman algorithm, which is a popular algorithm for determining the similarity between two sequences. The rejection threshold is equal to 8 and the parameters are: match 1, mismatch -1, gap open and gap extend -2. The same algorithm is used to calculate the complementarity between endings of each of primers pair. The reason is that the ends can connect with other primers regardless of the full complementarity of primers (in this case the rejection threshold is equal to 4).

The specificity is checked using the same results from BLAST, but the specificity is checked by the impact on cross. In order to illustrate the evaluation let LP1 and PP1 denote the left and right primers for the first coding sequence, and LP2 and PP2 denote the left and right primers for second coding sequence. The checking pairs can be presented as: LP1 – PP2, PP1 – LP2, LP1 – LP2, PP1 – PP2. The primers are specific if none of combinations work (the pairs for the same coding sequence are checked earlier).

To create optimal groups of primers the separate-and-conquer strategy is involved. The main idea of this method is to find the biggest possible group of primers within distance matrix (the conquer step). If two or more groups have the same number of primers then the first one is chosen. After the group is selected, it is added to a final set and the assigned primers are removed from the matrix (the separate step). In addition, the matrix is removed from primers attached to the same coding regions as primers separated. Then, the next group is searched on the smaller distance matrix. The step is repeated until no more groups can be created. The remaining primers are added to the final set as a separate group.

The method of hierarchical clustering is used in the conquer step. However our approach is unconventional, because in the grouping primers case there is no need to cut the tree. None of primers can have relation to others so the interesting group is on the ground level. For the same reason, the complete link cluster method is used. It should be noted that the complexity for complete-linkage clustering is $O(n^2)$, however the number of state matrix rows in real life cases do not exceed a few thousand (the largest tested square matrix had below 2000 rows).

The reason that the separate-and-conquer method was chosen is that for each of coding regions only one pair of primers should be in one of groups in the final set. The problem of grouping primers into a few sets for the multiplex PCR can be also considered as a set of sub-problems with lower complexity (the focus is on creating a single group rather than all at once). The separate step ensures that redundant primers are removed and in the next iteration the next biggest group can be found. On the other hand the conquest step guarantees to find the largest possible group thus allowing for the maximum limiting reagents needed for the laboratory experiments and that in turn reduces costs.

5 Results

The Table 1 shows the results of grouping for various genes. We tried a few genes in order to check a grouping efficiency for different and unrelated primers.

It can be seen that our approach gives promising results. The average value of grouping efficiency is 78%, which means that using our method to design primers and groups for experiment of the multiplex PCR could reduce the number of singleplex experiments by 78%. This can also be understood as cost and time reduction of laboratory experiments.

Table 1 The results of the effectiveness of primers grouping

Gene	Regions (Parts)	Regions and specific primers (Parts / Reduction)	# Groups	Grouping efficiency %
CFTR	27 (32)	25 (29 / 9%)	6	79
NF1	58 (63)	42 (44 / 30%)	9	80
CFTR+NF1	85 (95)	68 (74 / 22%)	13	82
PARP1	22 (23)	19 (20 / 13%)	4	80
BRCA1	22 (34)	15 (27 / 21%)	7	74
BRCA2	26 (51)	21 (35 / 31%)	9	74
BRCA1+BRCA2	48 (85)	34 (60 / 29%)	12	80
ABCC1	31 (32)	23 (24 / 25%)	6	75
		Average reduction: 23%	Average efficiency: 78%	

Interestingly, the grouping efficiency is the best when the problem is the most complex (the largest number of parts of coding regions are considered). Moreover a combination of primers in the groups (CFTR+NF1 and BRCA1+BRCA2) increases the grouping efficiency. It seems that a greater number of primers has a positive effect on grouping performance because there are more primers not affecting each other. However, to confirm this thesis further studies are required.

On the other hand the reduction shows how many parts are rejected because the Primer3 program designs for them non-specific primers. One possibility to decrease the reduction value is to change parameters of BLAST, but that change may lead to worse results of laboratory experiments. Another way to minimize this value is to design more primers in the hope that at least some of them will be specific. It seems that the second possibility is safer but requires the optimal value, because larger number of primers simultaneously increase the number of iterations and thus the time of the algorithm. Those parts of coding regions for which the Primer3Plus program was unable to design primers also increase the value of reduction. The reason for this could be too large a number of SNPs, the low-complexity of sequences of flanks, self complementarity of primers or primers were rejected because of other parameters. To limit the impact of the lack of primers on the reduction value the greatest hopes are in further research on the optimal setting.

It can be noticed that the combination of genes (CFTR + NF1) gives 68 parts with specific primers whereas separately these genes give 67 parts (the same situation is with genes BRCA1 and BRCA2, which give 36 (separately) and 34 (in combination) respectively parts). A possible explanation could be that the Primer3Plus program returns different set of primers and for one region all of these primers are non-specific. It is worth mentioning that the Primer3Plus program often returns several primers that differ by 1-2 characters. Such groups with very similar primers are often non-specific precisely because of the small changes. Only the larger differences in these short sequences (ie more than 5 characters) may make it appear specific primers. Therefore, the larger number of candidates obtained from Primer3Plus could give better results.

Results of the reduction of regions and the grouping efficiency confirm the thesis that both steps (the acquisition stage and the processing step) are equally important in the overall performance of the multiplex PCR. However, due to the overall performance of both PCR processes (singleplex and multiplex), the reduction is more important. The smaller value of reduction may also adversely affect the grouping efficiency, because of a bigger number of primers for clustering (but up to now the larger number of primer has a positive effect). The reduction measure also provides information about the complexity of the problem for various genes. Note, that for the CFTR gene it is rather easy to get primers, but for two genes (BRCA2 and NF1) almost one third of all regions received no primers. To minimize the measure of reduction the further research on the influence of parameters on the results (especially for complex genes) should be carried out.

6 Conclusions

We propose an original and unconventional approach to design reliable and specific primers for the multiplex PCR. The process is divided into two phases: the data acquisition and the data processing. In the data acquisition step we demonstrate an automatic way to obtain primers for coding sequences including the known locations of SNPs. In the step of data processing the program ensures that received primers are specific and uses two well-known algorithms called the hierarchical clustering combined with the separate-and-conquer strategy in order to group primers in the same tubes without any undesirable interactions between the primers.

To evaluate the approach we examine a few genes with many exons and the results tend to be very promising. The grouping efficiency is nearly 80%, which proves that the proposed approach is effective and can be successfully used in the multiplex PCR reactions. To ensure that all primers and groups work the laboratory experiments were conducted with the primers designed by the program. However, we did not perform all the experiments and devise results that could be published. The biological experiments are also outside the scope of this article.

Future efforts will focus on the development of the data processing algorithm. In the hierarchical clustering the matrix of distance is filled with discrete values and is identical to the matrix in the application MuPlex [20, 19]. In the conquer step of separate-and-conquer strategy if two groups have the same number of primers, the first one is chosen. It seems obvious to develop an evaluation function that will point a better set. We also plan to provide an efficient web application and a stand-alone version of our program. Another aim may be to optimize all input parameters of applications: Primer3Plus, BLAST and parameters of our algorithm.

Acknowledgements. This work was supported by the European Union from the European Social Fund (grant agreement number: UDA-POKL.04.01.01-00-106/09) and by the NCN grant UMO-2011/01/B/ST6/06868.

References

1. Aggarwal, C.C., Yu, P.S.: Data Mining Techniques for Associations, Clustering and Classification. In: Zhong, N., Zhou, L. (eds.) PAKDD 1999. LNCS (LNAI), vol. 1574, pp. 13–23. Springer, Heidelberg (1999)
2. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215(3), 403–410 (1990)
3. Chamberlain, J.S., Gibbs, R.A., Ranier, J.E., Nguyen, P.N., Caskey, C.T.: Deletion screening of the duchenne muscular dystrophy locus via multiplex dna amplification. Nucleic Acids Res. 16(23), 11141–11156 (1988)
4. Chen, et al.: Design of multiplex pcr primers using heuristic algorithm for sequential deletion applications. Comput. Biol. Chem. 33(2), 181–188 (2009)
5. Defays, D.: An efficient algorithm for a complete link method. The Computer Journal 20(4), 364–366 (1977)
6. Dieffenbach, C.W., Lowe, T.M., Dveksler, G.S.: General concepts for pcr primer design. PCR Methods Appl. 3(3), S30–S37 (1993)
7. Edwards, M.C., Gibbs, R.A.: Multiplex pcr: advantages, development, and applications. PCR Methods Appl. 3(4), S65–S75 (1994)
8. Ncbi-genbank flat file release, ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt
9. Fürnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review 13, 3–54 (1999)
10. Gervais, A.L., Marques, M., Gaudreau, L.: Pcrtiler: automated design of tiled and specific pcr primer pairs. Nucleic Acids Res. 38(Web Server issue), W308–W312 (2010)
11. Guberman, et al: Biomart central portal: an open database network for the biological community. Database (Oxford) 2011, bar041 (2011)
12. Henegariu, O., Heerema, N.A., Dlouhy, S.R., Vance, G.H., Vogt, P.H.: Multiplex PCR: critical parameters and step-by-step protocol. Biotechniques 23(3), 504–511 (1997)
13. dbsnp summary, http://www.ncbi.nlm.nih.gov/SNP/snp_summary.cgi
14. Huang, Y.C., Chang, C.F., Chan, C.H., Yeh, T.J., Chang, Y.C., Chen, C.C., Kao, C.Y.: Integrated minimum-set primers and unique probe design algorithms for differential detection on symptom-related pathogens. Bioinformatics 21(24), 4330–4337 (2005)

15. Jabado, O.J., Palacios, G., Kapoor, V., Hui, J., Renwick, N., Zhai, J., Briese, T., Lipkin, W.I.: Greene scprimer: a rapid comprehensive tool for designing degenerate primers from multiple sequence alignments. Nucleic Acids Res. 34(22), 6605–6611 (2006)
16. Kent, W.J.: Blat–the blast-like alignment tool. Genome Res. 12(4), 656–664 (2002)
17. Markoulatos, P., Siafakas, N., Moncany, M.: Multiplex polymerase chain reaction: a practical approach. J. Clin. Lab. Anal. 16(1), 47–51 (2002)
18. Qu, W., Shen, Z., Zhao, D., Yang, Y., Zhang, C.: Mfeprimer: multiple factor evaluation of the specificity of pcr primers. Bioinformatics 25(2), 276–278 (2009)
19. Rachlin, J., Ding, C., Cantor, C., Kasif, S.: Computational tradeoffs in multiplex pcr assay design for snp genotyping. BMC Genomics 6, 102 (2005)
20. Rachlin, J., Ding, C., Cantor, C., Kasif, S.: Muplex: multi-objective multiplex pcr assay design. Nucleic Acids Res. 33(Web Server issue), W544–W547 (2005)
21. Robertson, J.M., Walsh-Weller, J.: An introduction to pcr primer design and optimization of amplification reactions. Methods Mol. Biol. 98, 121–154 (1998)
22. Rozen, S., Skaletsky, H.: Primer3 on the www for general users and for biologist programmers. Methods Mol. Biol. 132, 365–386 (2000)
23. Shen, et al.: Mpprimer: a program for reliable multiplex pcr primer design. BMC Bioinformatics 11, 143 (2010)
24. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147(1), 195–197 (1981)
25. Untergasser, A., Nijveen, H., Rao, X., Bisseling, T., Geurts, R., Leunissen, J.A.M.: Primer3plus, an enhanced web interface to primer3. Nucleic Acids Res. 35(Web Server issue), W71–W74 (2007)
26. Wang, J., Li, K.-B., Sung, W.-K.: G-primer: greedy algorithm for selecting minimal primer set. Bioinformatics 20(15), 2473–2475 (2004)
27. Yamada, T., Soma, H., Morishita, S.: Primerstation: a highly specific multiplex genomic pcr primer design server for the human genome. Nucleic Acids Res. 34(Web Server issue), W665–W669 (2006)

Functional Dependencies on Symbol Strings Generated by Extended Context Free Languages[*]

Gyula I. Szabó and András Benczúr

Abstract. In this paper, we first rephrase the notion of regular functional dependency. The original definition based upon the dual language emerging from a regular grammar. We define now regular functional dependencies on finite state automata constructed from regular expressions. We extend the definition of regular functional dependency to extended context free languages. We define the syntactical form of functional dependencies on the graph of the FSA, constructed from the regular expression denoting the right side of the production rules. The semantics of the functional dependency will be given on the generated language. Using this model we can handle extended relations generated by recursive regular expressions too. The implication problem of our class of dependencies is decidable by a version of Chase algorithm specified on the graph of the associated FSA.

1 Introduction

Functional dependency (FD) is perhaps the most important integrity constraint for any data model, but surely, it is the most exhaustively analyzed one of them. In relational data model, FDs are well studied and are widely used in normalization theory([1]). XML has recently been practically the standard format of data exchange over the world wide web. XML was originally defined for describing and presenting individual documents, but it has been used for building databases too. Because of the use of XML as database model, one needs XML integrity constraints, and also XML

Gyula I. Szabó · András Benczúr
Eötvös Loránd University, Faculty of Informatics, Hungary
1118, Budapest, Pázmány Péter sétány, 1/C
e-mail: gyula@szaboo.de, abenczur@inf.elte.hu

[*] The research was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

functional dependency concepts. The main problem with defining functional dependency in the XML context is the lacking ”tuple” concept for XML. An instance of a relational schema is a set of tuples, and one can easily select pairs of tuples from this set for comparing in order to check whether the instance satisfies a given functional dependency defined on the relational schema. In the XML world there is no general accepted definition for the concept of tuple, and even if one chooses a collection of elements and declares them to be a ”tuple”, it is very hard to find a proper matching algorithm for them. Arenas and Libkin defined ”tree tuples” in their seminal work [5], based upon DTD schema. Vincent et al.[8] described some cases, not covered with ”tree tuples”, and invented the notion ”closest node” to deal with them. They defined functional dependency on XML trees without any schema, and used DTDs just to prove that their definition is equivalent with ”tree tuples” for some classes of DTDs.

All XFD concepts are very intricate, compared with the classical functional dependency concept for relational databases. In the case of XML data model they base mostly upon path expressions.

We have proposed recently ([7]) a new functional dependency concept, regular FD, applicable for data models, those extended ”tuples” are sentences from a regular language. Our main motivation was to find a simple, but general definition of functional dependency for a broad family of data models: our only assumption was that the ”tuples” should be sentences of a given regular language (i.e., they should be generated by a regular grammar).

We generalized in [7] the concept of functional dependency defined on attributes of a relational schema in a natural way to the sentences of the dual language associated to the ”tuple”-generator regular language using the finite state automaton associated to the regular grammar.

In this paper we rephrase the concept of regular FD when the concerned regular language is given by a regular expression then extend it to scoped functional dependency defined on extended context free languages.

2 Regular Expressions and Associated FSAs

We want to define regular functional dependencies for a regular language on the graph of the corresponding ((non)deterministic) finite state automaton (FSA). This automaton can be created from the grammar of the language by Alg. 1:

Algorithm 1. *Regular Grammar to FSA.*
Input: G regular grammar, generating the language L(G);
Output: M(G) finite state automaton, accepting L(G);
Let G=G(N,T,S,P),
then the corresponding FSA: $M(G) = (N, T, \delta, S, \mathscr{E}\mathscr{N}\mathscr{D})$, *where*

N is the (finite) set of the internal states of M,
T is the input alphabet,
δ is the transition function: for each $X \in N, y \in T, \delta(X,y) \in N$, iff, when there is a production rule $X \Rightarrow xY \in P$,
S is the initial state,
$\mathscr{END}$ is the (unique) final state.

In the usual case (e.g. XML schema language) the regular language will be given in the form of a regular expression built up from symbols picked from the alphabet of the language: the collection of these symbols is the set of terminal values of the regular grammar generating the language, when this grammar is given.

We need for our definition of regular functional dependency a finite state automaton accepting the regular language. There exists a great number of algorithms for the efficient construction of a finite automaton from a given regular expression. There are two main types of them according to the working-method of the resulting state machines: non-deterministic (NFA, e.g. Glushkov automaton) and deterministic (DFA, e.g. Brzozowski's construction). We borrow here the classical algorithm of Berry and Sethi [4] that constructs efficiently a DFA from a regular expression when all symbols are distinct. The states of this DFA can be associated uniquely to the symbols. The construction works via Alg. 2, for the definition of regular expression derivative, derivative extension, function Δ and continuation we refer to [4].

Algorithm 2. *Construction Berry-Sethi [4].*
Input: distinct regular expression E (built from the alphabet Σ,
Output: deterministic finite state automaton M(E) accepting L(E).

1. *M(E) has a state for the continuation of each (distinct) symbol in E.*
2. *Construct a transition from state p to the state of the continuation for $\alpha \in E$, if and only if, when p is for some continuation C and C can generate a string with a leading α.*
3. *The start state is for the whole expression E. A state is an accepting state iff, when it is a continuation C and $\Delta(C) = 1$.*

Definition 1 (FSA for Regular Language). Let L be a regular language, generated by either a regular expression $E \in RE$, or by a regular grammar G, then the FSA M(L) accepting L is either M(G), constructed by Alg. 1, or M(E), constructed by Alg. 2.

Example 1. Let $G(\{S,A,B\},\{a,b\},S,P)$ be a regular grammar, where
$P = \{S \Rightarrow aS, S \Rightarrow bS, S \Rightarrow aA, A \Rightarrow bB, B \Rightarrow a\}$.

The regular expression $E = (a+b)^* aba$ generates the regular language L(G) too. Figure 1 shows the FSA constructed by Alg. 2, using distinct symbols in E.

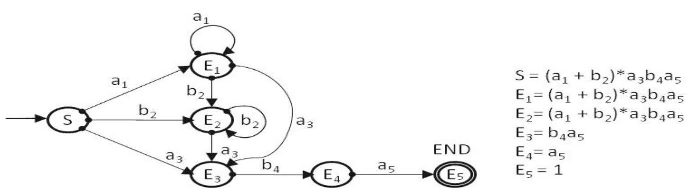


Fig. 1 Example FSA graph for Example 1

3 FD on Regular Languages

We can now generalize the notion of the dual language introduced in [7].

Definition 2 (Dual Language for Regular Language). Let L be a regular language, accepted by the FSA M(L). The alphabet of the dual language for L consists of the states of M(L), the sentences of the dual language are the strings of states visited by the accepting traversings (from START to END) of M(L).

According to the concept of relational functional dependency we can select two subsequences on each dual sentence, as the left and right side of the dependency, let we state this as the syntactic specification for the dependency. The set of sentences R satisfies this dependency, when there exist no two tuples in R so, that they are identical in all the nonterminals fitting to the left side subsequences, but on the subsequences selected for the right side, they differ on at least one position.

Notice, that we specify the syntax of the dependency on the dual language, its satisfaction will be checked on the accepted sentences of the language L(G).

If the definition of the language L is non-recursive, then the associated L' dual language is finite, because there is only a finite number of paths in the graph M(L). We can define functional dependencies (left and right side) on each one of these dual sentences, we can check the logical implications among them.

If the definition of the language L is recursive then the associated L' dual language is infinite. We can use the pumping process to select substrings of dual sentences for defining functional dependencies on them. We can select sub-paths on the non-pumped area as described before,(pumping 0-times the circles), then on the pumped part we can select left and right side of the dependency and pumping them together with the same frequencies. It is important, that we should always consider the visited nodes for a whole path (from $\mathscr{START}$ to $\mathscr{END}$) in the sequence of the processing, and the repeatings (recursions) should also be taken in sequence. For all dual sentences should be the selection unique for the left and right sides of the dependency.

In order to specify the (left and right) sides of a functional dependency we should pick up two sets of nodes from the graph M(L): one set for the left side (denoted by X), another one for the right side (denoted by Y). We can choose nodes visited by a traversing and state that each visiting of these nodes would be selected. We can choose starting and ending points for a path in the traversing, so that this pair of nodes will be selected at each closing of that path.

Definition 3 (Assignment). Let L be a regular language, let M(L)=(V,E) be its FSA as defined in Def. 1. We say that the tuple Y=(Y_1,Y_2), where $Y_1 \subseteq V$ and Y_2 is a subgraph of the transitive closure of M(L) is an assignment on M(L). Y_1 is taken from the non-recurred part of M(L), Y_2 refers to nodes and edges whose are (could be) repeatedly visited during a traversing.

Let Y be an assignment, Y selects a unique subsequence from a given dual sentence as follows:

Definition 4 (Selection). Let $Y = (Y_1, Y_2)$ be an assignment and let w be a dual sentence over M(L). Let $walk(w) = \{v_1, v_2, \ldots, v_n\}$ be a traversing on M(L). The symbols in Y_1 will be selected in order of their exploration (when visited). For each edge $e \in Y_2$ when the edge will be closed on the shortest path between its endpoints during the traversing on $walk(w)$, these two endpoints will be selected in their succession order (when visited at all). By the end of the selection the from w selected symbols build up the (possibly empty) array $w[Y] = \left\{v_{i_1}, \ldots, v_{i_k}\right\}$ $(1 \leq i_1 < i_2 < \ldots < i_k \leq n\,(k \geq 0))$.

Let t∈ L, the corresponding dual sentence w ∈ L'. We can interpret the w[Y] sequence of symbols as set of "attributes", that projects the "tuple" t to the values t[Y]. If w[Y]={}, then t[Y]={} as well.

Example 2. Let the edge $(A,B) \in Y_2$. Let a traversing in M(L) be

BCADCACAEBDCBAEB.

We select the non-capitalized nodes enumerated in the subscript according to the sequence numbers of the visiting:

$$BCADCACa_1Eb_1DCBa_2Eb_2.$$

Definition 5 (Regular Functional Dependency). Let L be regular language and let M(L) be the graph representation of the finite state automaton for L. Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be two assignments over M(L) so, that $X_1 \subseteq Y_1$ and $V(X_2) \subseteq V(Y_2)$. A functional dependency defined over M(L) (regular FD) is an expression of the form $X \to Y$. The R (finite) database instance over L satisfies the $X \to Y$ functional dependency (denoted by $R \models X \to Y$), if for any two $t_1, t_2 \in R$ tuples t_1[X]=t_2[X] can be fulfilled only then, when t_1[Y]=t_2[Y] also comes true. We call the case Y=M(L) key dependency.

The main difference between the XFD (XML functional dependency) concepts and regular functional dependency is, that RFD can deal with only one element type declaration, it takes a "horizontal" snapshot from an XML tree, however, the XFD concepts represent a mainly "vertical" approach, using path expressions from the root to the scope of the XFD components, those components could belong to elements of different types on the XML tree.

4 FD on Extended Context Free Languages

We extend the definition of regular functional dependency to dependencies over extended context free languages. An extended context free language is generated by an extended context free grammar (ECFG). An ECFG is a tuple G=(N,T,S,P), where

N is the (finite) set of non-terminal symbols,
T is the (finite) set of terminal symbols, and $N \cap T = \emptyset$,
P is the set of production rules of the form $A \Rightarrow R_A$, where $A \in N$, R_A is a regular expression over $N \cup T$,
$S \in N$ is the start symbol.

In the following we restrict ECFG allowing production rules of two forms only:

1. $A \Rightarrow R_A$, where $A \in N$, R_A is a regular expression over N,
2. $A \Rightarrow u$, where $A \in N$, $u \in T$.

We assume that the two sets of non-terminals those are LHS in group 1. or 2. are disjoint.

For each production rule $A \Rightarrow R_A$ the regular expression R_A denotes a regular language $L_A \subseteq N^*$, the corresponding FSA (denoted by M_A) can be constructed according to Alg. 2. During derivation by the grammar G we substitute the non-terminal A either by a sentence from L_A (for group 1. rules), or by a terminal symbol (for group 2. rules).

There is another (not necessary regular) language generated from A: we denote $L(A) \subseteq T^*$ the strings derived from A by the rules in P.

Remark 1. We can assume without loss of generality, that each non-terminal symbol occurs once as LHS in a production rule, because the rules $A \Rightarrow R_A^1$ and $A \Rightarrow R_A^2$ can be replaced with the rule $A \Rightarrow \left(R_A^1 + R_A^2\right)$ so, that the obtained grammar is equivalent with the original one ([2]).

Let $t \in L(A)$, let $U \in N$ be a non-terminal symbol so, that $U \in R_A$. We can interpret U as attribute of A. U, as start symbol is the root of the ECFG $G_U = G(N,T,U,P)$, so when generating t by G, some sentences of L(U) will be generated by the way, let these sentences be $u_1,\ldots,u_k \,(k \geq 1)$. Then $t = \omega_1 u_1 \ldots \omega_k u_k \omega_{k+1}$, where $\omega_i \in T^*, 1 \leq i \leq k+1$. We interpret the projection of U to t as $t[U] = u_1 \ldots u_k$.

Definition 6. The symbol $Y \in N \cup T$ is reachable from the $X \in N$ non-terminal symbol (denoted by $X \overset{*}{\Rightarrow} Y$) if $X \Rightarrow^* \alpha Y \beta$ where $\alpha, \beta \in (N \cup T)^*$, and $\Rightarrow^*$ denotes a sequence of substitutions.

Definition 7. Let G=(N,T,S,P) be an ECFG, let $\alpha \in N^*$ be a non-terminal string, then the language $L(\alpha) \subseteq T^*$ is the set of all strings that can be derived from α by the production rules in P. Formally, let $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n, \alpha_i \in N, 1 \leq i \leq n$, then $L(\alpha) = \{\omega \in T^* | \omega = \omega_1 \omega_2 \ldots \omega_n\}$, $\alpha_i \Rightarrow^+ \omega_i$, where $\Rightarrow^+$ is the transitive closure of the derivating relation.

Theorem 1. *Let G=(N,T,S,P) be an ECFG, let $A \in N$ and let $\alpha \in L_A$ and let $\omega \in L(\alpha)$. Let $\alpha = \{B_1 B_2 \ldots B_n\}$ then $\omega = \{\omega_1 \omega_2 \ldots \omega_n\}$ so, that $\omega_i \in L(B_i), 1 \le i \le n$.*

Proof. We can see via a simple induction on the number of derivation steps, that each member of the derivation chain $\alpha \mapsto \alpha_1 \mapsto \alpha_2 \ldots \mapsto \alpha_m = \omega$ is a concatenation of n substrings $\beta_i \in (N \cup T)^*, 1 \le i \le n$, so, that β_i is a derivation from B_i. This is true for the first step. When in the step k the member β_i contains the non-terminal Q to be substituted in the step k+1 using the rule $Q \Rightarrow R_Q$, means, $Q \Rightarrow Q_1 Q_2 \ldots Q_s$, then let $\beta_i = \gamma Q \eta$, with some $\gamma \eta \in (N \cup T)^*$, if $\beta_i \mapsto \beta_i^{'}$ then $\beta_i^{'} = \gamma Q_1 Q_2 \ldots Q_s \eta$. Using the rule $Q \Rightarrow u$ we get $\beta_i^{'} = \gamma u \eta$.

We can assign values (taken from a non-empty domain set D) to the terminal symbols as follows. Let $u \in T$ be terminal symbol, then let $D_u \in D$ be a set so, that when $u, v \in T, u \neq v$ then $D_u \cap D_v = \emptyset$.

The mapping $val : u \in T \mapsto D_u$ assigns a domain value to a terminal symbol so, when the assignment will be made for a string of terminal symbols, then each assignment occurs autonomously, that is, $val(u_1) \neq val(u_2)$ can occur also when $u_1 = u_2$. Obviously, when $u_1 \neq u_2$ then $val(u_1) \neq val(u_2)$.

For $\omega \in L(A)$, let $\omega = \{u_1 u_2 \ldots u_n\}$ then $val(\omega) = \{v_1 v_2 \ldots v_n\}$ is a valuation of ω, where $v_i = val(u_i), 1 \le i \le n$.

Definition 8 (Complex valued tuple). Let G=(N,T,S,P) be an ECFG, let $A \in N$ be a non-terminal symbol and let $\alpha \in L_A, \alpha = \{B_1 B_2 \ldots B_n\}$ be a string of non-terminal symbols and let $\omega \in L(\alpha), \omega = \{\omega_1 \omega_2 \ldots \omega_n\}$ so, that $\omega_i \in L(B_i), 1 \le i \le n$ then we say, that α is a dual sentence in the scope A, the B_i-s are the attributes of α, ω_i is the extended attribute associated to B_i and a valuation of ω is a complex valued tuple t of A, denoted by $t_{CV} = val_{CV}(\omega)$.

We say that A is an ECFG-style schema to G, and if R is a finite set of tuples of A then we say that R is an instance of A.

If $\omega = \{\omega_1 \omega_2 \ldots \omega_n\}$ then $val(\omega) = \{val(\omega_1) val(\omega_2) \ldots val(\omega_n)\}$

We presented functional dependencies in Sect. 3 syntactical defined on the graph for the accepting FSA of a regular language, and we gave semantics for them on sentences of the language. We extend this definition to ECFG so, that the syntax of the FDs will be defined on a single regular expression (using one production step only), but for the semantics we use the whole derivation tree of the ECFG. With this restriction we can yet handle most real-life applications, meaning "horizontally" connected data, and it allows a quadratic complexity of implication.

We get the dual language according to Def. 2. For the syntax of the functional dependency we can select two subsequences on each dual sentence, as the left and right side of the dependency.

Let G=(N,T,S,P) be an ECFG, let $A \in N$ and let M_A be the corresponding automaton. Let $w = \{v_1\, v_2\, \ldots\, v_n\}$ be a traversing on M_A.

Definition 9 (Selection on Scope). Let $Y = (Y_1, Y_2)$ be an assignment on M_A for the scope A and let w be a traversing on M_A. The symbols in Y_1 will be selected in order of their exploration (when visited). For each edge $e \in Y_2$ when the edge will be closed on the shortest path between its endpoints during the traversing on w, these two endpoints will be selected in their succession order (when visited at all). That is, if the two endpoints of the closing path are A and B ($A = v_i, B = v_j$ for some $1 \leq i < j \leq n$) then that path will be selected which does not contain neither A nor B. The nodes in Y_2 will be selected by each visiting (if any) during the traversing on $walk(w)$. The selection will be processed for all edges and nodes in Y_2 autonomously. By the end of the selection the from w selected symbols build up the (possibly empty) array $w[Y] = \left\{v_{i_1}, \ldots, v_{i_k}\right\}$ $(1 \leq i_1 < i_2 < \ldots < i_k \leq n\,(k \geq 0))$.

Let w be a traversing on M_A, let $\omega \in L(w)$, and let $t = val(\omega)$ be a tuple of A. We interpret the w[Y] sequence of symbols as set of "attributes", that projects the tuple t to the values $t[Y] = val(\omega[Y])$, that is, let $w = \{v_1 v_2 \ldots v_n\}$, let $w[Y] = \left\{v_{i_1}, \ldots, v_{i_k}\right\}$ $(1 \leq i_1 < i_2 < \ldots < i_k \leq n\,(k \geq 0)$ and let $\omega = \{\omega_1 \omega_2 \ldots \omega_n\}$ then $t[Y] = val(\omega_{i_1}) val(\omega_{i_2}) \ldots val\left(\omega_{i_k}\right)$.
If w[Y]={}, then t[Y]={} as well.

Concerning the regular language L_A we can define functional dependency over M_A as we seen in Sec. 3, considering the non-terminal A as the scope for the functional dependency.

Definition 10 (Scoped Functional Dependency). Let A be a scope in the ECFG G and let M_A be the corresponding finite state automaton. Let $X = (X_1, X_2)$ and $Y = (Y_1, Y_2)$ be two assignments over M_A. A functional dependency defined over M_A (FD_A) is an expression of the form $X \rightarrow Y$. The R (finite) database instance of A satisfies the $X \rightarrow Y$ functional dependency (denoted by $R \models X \rightarrow Y$), if for any two $t_1, t_2 \in R$ tuples t_1[X]=t_2[X] can be fulfilled only then, when t_1[Y]=t_2[Y] also comes true. We call the case $Y = M_A$ key dependency.

Definition 11 (Scoped Functional Dependency Implication). Let A be a scope in the ECFG G and let M_A be the corresponding finite state automaton. Let Σ be a set of FD_A-s and let σ be an FD_A over M_A, then Σ implies σ (denoted by $\Sigma \models \sigma$) when for all R (finite) database instances of A those satisfy Σ, $R \models \sigma$ will also be fulfilled.

Algorithm 3. *Algorithm for checking implication.*
Input: graph $M_A = (V, E)$, a set Σ and $\sigma : X \rightarrow Y$ (where X=(X_1, X_2) and Y=(Y_1, Y_2)) functional dependencies over M_A
Output: true, if $\Sigma \models \sigma$, false otherwise

```
  1. Initialization
TC=(V',E') := transitive closure of M(L)
do color all nodes and edges in M(L) black,in TC blue,
do color all nodes and edges in X: both graphs green,
do color all nodes and edges in Y: for M(L) yellow,for TC red
2. $FDSET := \Sigma$;
3. $greene := X$ (means, that $greene_1 := X_1$ and $greene_2 := X_2$);
4. repeat until no more dependency applicable:
if $W = (W_1, W_2) \rightarrow Z = (Z_1, Z_2) \in \Sigma$ and $W \subseteq greene$ (means, that $W_1 \subseteq greene_1$
and $W_2 \subseteq greene_2$) then
```

i. $FDSET := FDSET - (W \to Z)$;
ii. greene $:= greene \cup Z$ (means, that $greene_1 := greene_1 \cup Z_1$ and $greene_2 := greene_2 \cup Z_2$);
iii. do color all nodes and edges in Z green(for M(L) and TC)

5. if count_nodes(V,yellow) = count_nodes(V',red) = 0 and count_edges(E,yellow) = count_edges(E',red) = 0 then output true else output false.

Proposition 1 (Scoped Functional Dependency Implication). *Let A be a scope in the ECFG G and let M_A be the corresponding finite state automaton and let Σ be a set of FDs and let $X \to Y$ be an FD over M_A, then $\Sigma \models X \to Y$ if and only if when the Algorithm 3 for input M_A, Σ and $X \to Y$ returns true.*

Proof. We only sketch here the proof: the chasing process guarantees the success, because the non-recursive parts could be viewed as relational attributes, the recursive parts are fixed on the graph. For details we refer to [7]

Remark 2 (Reasoning about Scoped Functional Dependencies). The Algorithm 3 runs in quadratic time measured in the number of extended attributes appearing in Σ and $\sigma : X \to Y$ respectively.

Definitions 9 and 10 based upon immediate components of a scope, represented by the states of the automaton for the scope. This relative simple case allows an easy handling and tractable decision on implication. We can get a more complex concept when we take into account sub-elements, reachable from the immediate elements, too.

Definition 12 (Single valued tuple). Let G=(N,T,S,P) be an ECFG, let $A \in N$ be a non-terminal symbol and let $\omega \in L(A), \omega = \{\omega_1\omega_2\ldots\omega_n\}, \omega_i \in T$ be a string of terminal symbols, then we say, that ω is a single valued dual sentence in the scope A, the ω_i-s are the attributes of ω, and a valuation of ω is a single valued tuple t of A, denoted by $t = val(\omega)$.

We say that A is an ECFG-style schema to G, and if R is a finite set of tuples of A then we say that R is an instance of A.

If $\omega = \{\omega_1\omega_2\ldots\omega_n\}$ then $val(\omega) = \{val(\omega_1)val(\omega_2)\ldots val(\omega_n)\}$

Let $A \in N$, let $\omega \in L(A)$, let $Y \subseteq T$ and let $t = val(\omega)$ be a tuple of A. We interpret the $\omega[Y]$ sequence of symbols as set of "attributes", that projects the tuple t to the values $t[Y] = val(\omega[Y])$, that is, let $\omega = \{v_1 v_2 \ldots v_n\}$, let $\omega[Y] = \{v_{i_1}, \ldots, v_{i_k}\}$ $(1 \le i_1 < i_2 < \ldots < i_k \le n (k \ge 0)$ iff, when $v_{i_k} \in Y$, then $t[Y] = val(v_{i_1})val(v_{i_2})\ldots val(v_{i_k})$.

If $\omega[Y] = \{\}$, then t[Y]={} as well.

Concerning the regular language L_A we can define functional dependency considering the non-terminal A as the scope for the functional dependency.

Definition 13 (TreeScoped Functional Dependency). Let A be a scope in the ECFG G. Let $X \subseteq T$ and $Y \subseteq T$ be set of attributes to A. A functional dependency defined over A (FD_A) is an expression of the form $X \rightarrow Y$. The R (finite) database instance of A satisfies the $X \rightarrow Y$ functional dependency (denoted by $R \models X \rightarrow Y$), if for any two $t_1, t_2 \in R$ tuples t_1[X]=t_2[X] can be fulfilled only then, when t_1[Y]=t_2[Y] also comes true. We call the case $Y = M_A$ key dependency.

4.1 Scoped FD on XML Schema Languages

We apply our definitions of functional dependency for XML schema languages. As presented in Section 1 elements of XML documents are sentences of regular languages and the corresponding dual language is the language generated by the element declaring regular expressions of DTD or XML Schema. We consider an example with DTD element description.

Let us see an example XML document (Fig. 2.), the corresponding DTD contains the following declaration:

```
<!ELEMENT course (Cid,(Stid,Stn)+)>
```

Figure 3 presents the corresponding automaton constructed from the regular expression above.

Based upon this automaton we can define the following key dependency:

$(\{Cid\},\{\}) \rightarrow (\{\},\{Stid,Stn\})$

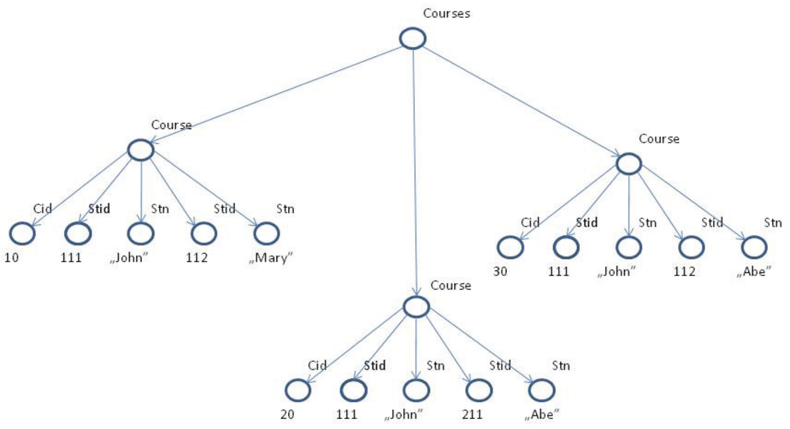


Fig. 2 Example XML data document (courses.xml)

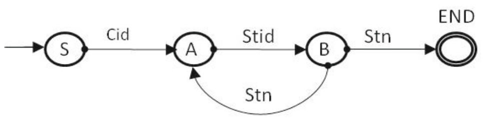


Fig. 3 Example FSA for course data

5 Conclusion and Future Works

Functional dependency is a constraint that can help to detect corrupt data in a data set. But functional dependency is also a tool for the database (document) administrator supporting him or her to create a consistent data structure. Functional dependency for relational data model is easy to understand and it can be used easily in database design. The known FD concepts for other data models (e.g. XML and semantic data models) are very intricate. We wanted to propose a general functional dependency concept, based upon extended context free languages, that works with the relational and other (e.g. XML) data models as well.

Our model can deal with the XML schema language DTD well, but it can not manage some properties of W3C XML Schema (*XSD Mixed Type*,*XSD Element Substitution*, *XSD Restriction* etc.). We would like to extend our model to deal with them too.

Another future research direction would be to use the concept of regular grammar for describing the complex value model (tuple and set constructor) and for defining dependencies on this model. We can specify the structure of the complex value with a regular expression, so that each constructor would have a distinct name. This differs from the traditional concept of complex values used by Abiteboul et al. [1], but no more than writing type names in state of constructor symbols. We can also say, that the notation will have an XML character. The name of the constructor corresponds to an XML tag.

There exists no finite axiomatization for the logical implication of XFD (XML functional dependency) as proven by Arenas and Libkin in [5]. Using our model we would like to investigate the reason of this hardness.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Albert, J., Giammarresi, D., Wood, D.: Normal form algorithms for extended context-free grammars. Theoretical Computer Science 267(1-2), 35–47 (2001)
3. Amano, S., Libkin, L., Murlak, F.: XML schema mappings. In: Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Providence, Rhode Island, USA, June 29-July 01, pp. 33–42 (2009)
4. Berry, G., Sethi, R.: From regular expressions to deterministic automata. Theoretical Computer Science 48, 117–126 (1986)

5. Arenas, M., Libkin, L.: A normal form for XML documents. ACM TODS 29, 195–232 (2004)
6. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. ACM Trans. Internet Technol. 5(4), 660–704 (2005)
7. Szabó, G.I., Benczúr, A.: Functional Dependencies on Extended Relations Defined by Regular Languages. In: Lukasiewicz, T., Sali, A. (eds.) FoIKS 2012. LNCS, vol. 7153, pp. 384–403. Springer, Heidelberg (2012)
8. Vincent, M.W., Liu, J., Liu, C.: Strong functional dependencies and their application. To Normal Forms in XML. ACM ToDS 29, 445–462 (2004)

Extending HQL with Plain Recursive Facilities

Aneta Szumowska, Marta Burzańska, Piotr Wiśniewski, and Krzysztof Stencel

Abstract. The mismatch between relational databases and object-oriented programming languages has been significantly mitigated by the use of object-relational mapping. However, the querying facilities available in such mapping systems are still inferior when compared to the features of a fully-fledged relational DBMS. In our research we aim at enriching object-relation mapping with advanced database concepts. An example of such an aspect is recursive querying. In prequel papers we have shown how to extend Hibernates mapping configurations with comprehendible recursive views that map to SQL common table expressions. In this paper we show how one can extend Hibernate Query Language (HQL) with plain recursive query facilities based on Oracles CONNECT BY phrase. Although, unfortunately it has not become a part of the SQL standard, its properties like cleanness and guaranteed stop make it worth exploiting. We propose adding CONNECT BY to HQL. We have implemented a prototype mapping of this phrase to recursive queries in major DBMSs and tested its efficiency. As the result we have got a simple, safe and fast way to pose recursive queries in HQL.

1 Introduction

Already for a noteworthy number of years software development has been object-oriented, while the data storage has been relational. That situation has been the cause of many problems implied by the mismatch of these two data models. These problems widely recognised as so called *impedance mismatch* concern security, maintainability, portability and data-type mismatch. They have become a severe obstruction. The software industry soon addressed that problems and proposed

Aneta Szumowska · Marta Burzańska · Piotr Wiśniewski · Krzysztof Stencel
Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland
e-mail: {iriz,quintria,pikonrad,stencel}@mat.umk.pl

object-relational mapping (ORM). ORMs significantly reduced problems troubling application programmers [7, 8].

With the advent of ORM systems, programmers can focus on the application code and at least to some extent they do not have to worry about advanced SQL aspects. Most of the complexity of joins, nested queries, aggregations and updates is separated from a programmers domain.

Currently most of object-oriented programming languages are supported with object-relational mapping. Some of them like LINQ are even built into the basic feature set of their host languages. Others are smoothly integrated add-ons. The most noteworthy of them are Hibernate and JDO for Java, ADO.NET for .NET platform, Django-models and SQLObject for Python. There were also some proposals of a standard for object relational mapping frameworks. One of them is the Oracle's Java Persistence API (JPA). It has quickly gained popularity among software developers. Nowadays many application servers implement JPA. Examples are JBoss, Apache Geronimo or Oracle's OC4J.

Although object-relational systems has been a major improvement to a software developers toolkit, they still lack numerous features that are offered by query engines of mature relational database systems. One of them is recursive querying. In business application there are numerous entities that are best modelled and stored as networks or hierarchies. Such data items are best queried by recursive facilities that have been implemented in relational DBMSs for already 25 years (Oracles CONNECT BY) and eventually they qualified into the SQL-1999 standard (recursive common table expressions—CTE). Currently recursive CTEs are implemented in all major database systems.

More about history of recursive queries, their availability and efficiency of evaluation in modern DBMSs may be found in [1, 2]. Despite the fact that recursive queries appeared quite a long time ago, intensive research is still being conducted on their optimization [3, 4]. The time that has passed allowed the field of recursive queries to mature and now they become increasingly popular among software developers.

SQL standard promotes recursive CTEs. However, the original Oracles recursive contribution, namely CONNECT BY, has various interesting properties. CONNECT BY is significantly simpler than CTEs and its stop is guaranteed. In contrast, one can easily code a recursive CTE that will never stop. Therefore, we have decided to propose extending HQL with CONNECT BY. In our opinion it is natural compared to CTE and safe (guaranteed termination). We have prepared a prototype implementation of this extension in Hibernate for Java [9, 11, 12] that currently is the most popular object-relational mapping system. The results prove that using the proposed facility we can achieve orders of magnitude of performance improvement.

The initial research on joining ORM systems and recursive queries has been made for a SQLObject - an ORM system for Python and PostgreSQL [10]. Initial work on recursive query system for Hibernate has been presented in [6, 13]. This paper makes the following contributions:

- we propose an extension to HQL with Oracle-like CONNECT BY phrase;
- we describe our prototype implementation of the mapping of this phrase to the query language of the underlying DBMS; our prototype is able to map queries to Oracle Database, IBM DB2 and PostgreSQL;
- we show experimental results that prove the robustness of our idea.

The paper is organized as follows. In Section 2 we describe the data sorts and problems that are best solved by recursive queries. Section 3 presents the proposed extension to HQL. Section 4 details performance evaluation of our prototype implementation. Section 5 concludes and portrays our plans for future work.

2 Data Structures

The research presented in this papers focuses on the problem of processing graph and hierarchical data structures. There is an abundance of real-life problems associated with such structures among which are: finding the communication links between two cities or finding routes based on information provided by GPS systems, processing championships' scoreboards, corporate hierarchy or bill-of-material. In this section we present two natural examples of recursive data. The first table called *Emp* contains data on corporate hierarchy with special focus on employees hierarchy. A sample of its rows is presented by Table 1. The second table is called *Conns*. It describes a network of flight connections between cities. Table 2 shows a small fragment of it.

SQL-1999 introduced common table expressions in order to query data structures having recursive or graph-like structure. Then, various database management systems implemented this construct. A query that selects Smith's direct and indirect subordinates based on the recursive CTE is shown on Listing 1. It starts from the rows with the data on all Smiths and then in subsequent iterations it collects their subordinates. Direct subordinates are gathered in the first iteration, while indirect subordinates are accumulated in all further iterations. In fact the forest with the tree for each Smith is being built.

Table 1 Hierarchical data, table Emp

empId	bossId	sname	fname	salary
7521	7698	Christie	Andrew	210
7566	7839	Jones	Brandon	360
7654	7698	Ford	Carl	210
7698	7839	Blake	Ernest	360
7782	7839	Bell	Gordon	360
7788	7839	Willis	James	360
7839		Smith	John	500
7844	7698	Turner	Johnathan	210
7902	7698	Adams	Trevor	210
7900	7566	Miller	Kyle	150

Table 2 Graph data, table Conns

departure	arrival	flightId	price	travelTime
Phoenix	Huston	PW 230	100	3h 10min
Huston	Chicago	RW 121	90	2h 45min
Huston	Dallas	RW 122	80	3h
Dallas	Chicago	DW 80	110	2h 30min
Chicago	Atlanta	CH 542	220	2h 45min
Chicago	Berlin	CH 543	360	7h 15min
Paris	Berlin	TW 118	300	1h 10min
Dallas	Berlin	DW 90	350	5h 45 min
Berlin	Boston	YW 421	100	6h
Chicago	Boston	CH 544	250	2h 15min

Listing 1 List of Smith's Subordinates

```
WITH RECURSIVE rcte (
   SELECT sname, fname, empId, False as isSub
      FROM Emp WHERE sname = 'Smith'
 UNION
   SELECT e.sname, e.fname, e.empId, True as isSub
      FROM Emp e JOIN rcte r ON (e.bossId = r.empId))
SELECT sname, fname FROM rcte WHERE rcte.isSub = True;
```

However the first DBMS that introduced recursive querying was Oracle. It had then an elegant approach based on the CONNECT BY clause. The abovementioned query in the form of the SQL statement with CONNECT BY phrase would have the following form:

Listing 2 List of Smith's Subordinates using CONNECT BY

```
SELECT sname, fname FROM Emp
  WHERE level > 1
  START WITH sname = 'Smith'
  CONNECT BY bossId = PRIOR empId;
```

Listing 3 Recursive data retrieval using traditional tools

```
List<Empl> first = session.createCriteria(Empl.class).
    add(Restrictions.eq("sname", "Smith")).list();
Empl firstEmp = first.get(0);
while(!stack.isEmpty()) {
  Empl emp = (Empl) stack.firstElement();
  List<Empl> emps = session.createCriteria(Empl.class).
    add(Restrictions.eq("bossId", emp.getId())).list();
  for(int i = 0; i < emps.size(); i++)
    stack.push(emps.get(i));
  stack.remove(0);
}
```

3 CONNECT BY in HQL

Currently, in order to perform a recursive query in Hibernate an application programmer has to use a non-declarative 3GL code. An example of such code is presented as Listing 3.

In Section 4 we call this approach the *Hibernate loop*. Performance evaluation shown in Section 4 proves that this approach is highly inefficient compared to the evaluation of recursive queries performed by a DBMS.

In papers [6, 13] we have presented proposals of the integration of recursive queries into Hibernate object-relational mapping system. This integration was done through the means of definition of the recursive queries in XML files [6] or via annotations for the classes specifically designed for this purpose [13]. However, observing the number of projects that utilize HQL language, we have decided to integrate the recursive queries into HQL. The main idea that guided the proposed solutions was the intuitiveness and the simplicity of use. We have abandoned the initial idea of the introduction of the common table expressions' equivalents in the HQL. These expressions provide ample opportunities, but would require major changes in the grammar. Also, programmers often view more complex CTEs as lacking readability and difficult to understand and maintain. Therefore, we decided to base the solution on the START WITH and CONNECT BY expressions proposed by the Oracle. These queries do not solve all of the problems covered by the recursive common table expressions. However, they are still very expressible and useful. Furthermore, their syntax is more readable that CTE.

With the usage of the extensions to the language HQL introduced in this paper, the query retrieving Smith's direct or indirect subordinates would take the form presented an the listing 4.

Listing 4 List of Smith's Subordinates using CONNECT BY

```
Query query = session.createQuery("FROM␣Emp
␣␣WHERE␣level␣!=␣1
␣␣START␣WITH␣sname␣=␣:bossname
␣␣CONNECT␣BY␣bossId␣=␣PRIOR␣empId");
query.setParameter("bossname", "Smith");
List list = query.list();
```

The introduction of the CONNECT BY phrase required a number of changes in the grammar of HQL. The original `query` rule have been extended with symbols `StartWithCondition` and `connectbyClause`:

```
queryRule : selectFrom (whereClause)?
                 ( (groupByClause)? (orderByClause)? )
               | ( (StartWithCondition)? (connectbyClause) );
```

We have also added two rules for new non-terminals `StartWithCondition` and `connectbyClause`:

```
StartWithCondition: STARTWITH^ logicalInitExpression;
connectbyClause   : CONNECTBY^ logicalRecursiveExpression;
```

Rules for other two non-terminal symbols `logicalInitExpression` and `logicalRecursiveExpression` are slight modifications of the existing symbol `logicalExpression` that disallow subqueries. Moreover, the rules for `logicalRecursiveExpression` include the possibility to use the `PRIOR` keyword.

Those modifications allow parsing recursive queries based on the `CONNECT BY` clause. Such query is then processed by the query generators developed by the authors of this paper. Those generators create a recursive CTE in the desired SQL dialect that represents the requested query. The CTE is then sent to the database. The query from Listing 4 is thus translated into the query from Listing 5.

Listing 5 List of Smith's Subordinates

```
WITH RECURSIVE rcte (
  SELECT empId, sname, fname, bossId
    FROM Emp WHERE sname = 'Smith'
 UNION
  SELECT e.empId, e.sname, e.fname, e.bossId
    FROM Emp e JOIN rcte r ON (e.bossId = r.empId)
)
SELECT empId, sname, fname, bossId FROM rcte
```

It is noteworthy that the structure of the records returned by this query corresponds to the structure of the *Emp* table. Moreover, because of this correspondence the resulting records are mapped directly to instances of the *Emp* class.

4 Performance

In this section we present experimental data that is intended to verify the efficiency of the prototype implementation. The feasibility of the whole approach is also assessed by a comparison to coding recursive queries using 3GL languages. Since our solution concerns modifications to HQL, obviously we use Hibernate as the test bed.

The problem of corporate hierarchy has been tested against five data sets: 900 records with 7 levels of hierarchy, 1800 records with 8 levels of hierarchy, 2700 records with 9 hierarchy levels, 3600 records with 10 hierarchy levels and 4500 records with 11 hierarchy levels. In order to test the native Hibernate's method (called in this paper *Hibernate loop*), we have prepared the source code based on the *while* loop. The main part of this code has been presented in Listing 3 in Section 3.

We traced the execution of the *Hibernate loop* and we detected that Hibernate generated and sent to the database as many queries as there are matching objects in the examined data set. Obviously, the recursive HQL query induces the execution of only one database query. It saves a significant amount of database (parsing queries, consulting the data dictionary etc.) and communication resources (network round-trips).

The results of those tests are presented in Tables 3, 4 and 5. Tests were performed on different machine configurations for different database systems. Therefore one should only judge the results for a single DMBS. The results are incomparable between different DBMSs. However, the goal of our performance tests was to compare the evaluation of Hibernate's solutions and the proposed extension of HQL. The *Hibernate loop* column presents the time needed to complete the execution of the native Hibernate code (Listing 3). The *Ratio* column presents the percentage of time needed for the HQL CONNECT BY to complete with respect to the native method. The results of the performance evaluation is obvious. An extended HQL query is significantly faster than corresponding 3GL code for all sizes of the data set and for all database systems. This shows potential improvement that an application programmer can exploit using our proposal.

Table 3 Comparison of average execution times on ORACLE

	Hibernate loop	HQL with CONNECT BY	Ratio
900 rec.	1385 ms	159 ms	**11.48 %**
1800 rec.	3475 ms	205 ms	**5.89 %**
2700 rec.	6152 ms	242 ms	**3.39 %**
3500 rec.	9534 ms	276 ms	**2.89 %**
4500 rec.	14912 ms	301 ms	**2.02** %

Table 4 Comparison of average execution times on IBM DB2

	Hibernate loop	HQL with CONNECT BY	Ratio
900 rec.	2045 ms	210 ms	**10.26 %**
1800 rec.	4678 ms	398 ms	**8.50 %**
2700 rec.	10255 ms	831 ms	**8.10 %**
3500 rec.	15121 ms	1301 ms	**8.61 %**
4500 rec.	22045 ms	1892 ms	**8.59** %

Table 5 Comparison of average execution times on PostgreSQL

	Hibernate loop	HQL with CONNECT BY	Ratio
900 rec.	3121 ms	153 ms	**4.90 %**
1800 rec.	8723 ms	272 ms	**3.10 %**
2700 rec.	18012 ms	287 ms	**1.59 %**
3500 rec.	27979 ms	413 ms	**1.48 %**
4500 rec.	41412 ms	434 ms	**1.05 %**

5 Conclusions and Future Work

In this paper we have presented a proposal to extend HQL with plain recursive facilities based on the Oracle's style CONNECT BY clause. In our opinion this option may be considered by most programmers as a brief, simple and readable alternative to Hibernate loops and SQL'c CTEs. In fact initially we considered extending HQL with recursive CTEs. However, the syntactical complexity of SQL queries using CTEs convinced us that it would be more profitable to application developers to use simpler means. Another advantage of CONNECT BY is its guaranteed stop.

In order to encourage potential users we implemented a prototype mapper module that serves HQL queries enriched with CONNECT BY. The result of performance tests conducted for this prototype emphasize the value of the proposed improvement. Compared to 3GL code we can achieve orders of magnitude improvement using our solution.

As the future research we plan to continue pushing recursive query facilitates closer to application programmers. Database engines have numerous features that are not used in practical development because of (1) the ignorance of potential users and (2) the complexity of these features. If one offers these facilities as a part of an object-relational mapper, they can eventually become useful. An example of such a feature is a support for logical constraints that would allow automatic generation of DBMS specific triggers. We also plan to develop corresponding extensions for other Database Management Systems with the special focus on SQL Server and SQL Anywhere.

References

1. Brandon, D.: Recursive database structures. J. Comput. Small Coll. 21(2), 295–304 (2005)
2. Przymus, P., Boniewicz, A., Burzańska, M., Stencel, K.: Recursive Query Facilities in Relational Databases: A Survey. In: Zhang, Y., Cuzzocrea, A., Ma, J., Chung, K.-i., Arslan, T., Song, X. (eds.) DTA and BSBT 2010. CCIS, vol. 118, pp. 89–99. Springer, Heidelberg (2010)
3. Ghazal, A., Crolotte, A., Seid, D.Y.: Recursive SQL Query Optimization with k-Iteration Lookahead. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 348–357. Springer, Heidelberg (2006)
4. Burzańska, M., Stencel, K., Wiśniewski, P.: Pushing Predicates into Recursive SQL Common Table Expressions. In: Grundspenkis, J., Morzy, T., Vossen, G. (eds.) ADBIS 2009. LNCS, vol. 5739, pp. 194–205. Springer, Heidelberg (2009)
5. django-models, https://docs.djangoproject.com/en/dev/topics/db/models/
6. Burzaska, M., Boniewicz, A., Szumowska, A., Winiewski, P.: Hibernate the Recursive Queries - Defining the Recursive Queries Using Hibernate ORM. In: ADBIS Research Communications, pp. 190–199 (2011)
7. Keller, W.: Mapping objects to tables: A pattern language. In: EuroPLoP (2007)
8. Wisniewski, P., Burzańska, M., Stencel, K.: The impedance mismatch in light of the Unified State Model. Fundam. Inform. (to appear, 2012)
9. Hibernate, http://www.hibernate.org
10. Burzańska, M., Stencel, K., Suchomska, P., Szumowska, A., Wiśniewski, P.: Recursive Queries Using Object Relational Mapping. In: Kim, T.-H., Lee, Y.-H., Kang, B.-H., Ślęzak, D. (eds.) FGIT 2010. LNCS, vol. 6485, pp. 42–50. Springer, Heidelberg (2010)
11. Bauer, C., King, G.: Java Persistence with Hibernate. Manning Publications Co., Greenwich (2006)
12. O'Neil, E.J.: Object/relational mapping 2008: hibernate and the entity data model (EDM). In: Proc. ACM SIGMOD, pp. 1351–1356 (2008)
13. Szumowska, A., Burzańska, M., Wiśniewski, P., Stencel, K.: Efficient Implementation of Recursive Queries in Major Object Relational Mapping Systems. In: Kim, T.-H., Adeli, H., Slezak, D., Sandnes, F.E., Song, X., Chung, K.-I., Arnett, K.P. (eds.) FGIT 2011. LNCS, vol. 7105, pp. 78–89. Springer, Heidelberg (2011)

Trust in RDF Graphs

Dominik Tomaszuk, Karol Pąk, and Henryk Rybiński

Abstract. It is well-known that pure Resource Description Framework (RDF) is not suitable to represent trust information. This work is to overcome these limitations. In our proposal RDF graphs are extended by trust metrics. The paper describes a mechanism for representing and reasoning with trust annotated RDF data. We present how with metric algebra such an annotation can be used for processing data on inferred RDF triples.

Keywords: Semantic Web, Resource Description Framework, trust, metric, annotation.

1 Introduction and Motivation

An RDF triple consists of a subject, a predicate, and an object. In [5] the meaning of subject, predicate and object is explained. The subject denotes the resource, the predicate means traits or aspects of the resource, and expresses a relationship between the subject and the object. A collection of RDF statements intrinsically represent a labeled, directed multigraph. The nodes are the subjects and objects of their triples.

Dominik Tomaszuk · Karol Pąk
Institute of Computer Science, University of Bialystok, Sosnowa 64,
15-887 Bialystok, Poland
e-mail: dtomaszuk@ii.uwb.edu.pl,pakkarol@uwb.edu.pl

Henryk Rybiński
Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19,
00-665 Warsaw, Poland
e-mail: h.rybinski@ii.pw.edu.pl

With the increasing interest in trust metric in classical systems, there have been a number of proposals for data description. One can distinguish two main categories of trust values: discrete [1] and continuous [17, 10]. In the context of Semantic Web, information providers do not have any explicit way to express any intention concerning trust information. The tools supporting storage of RDF triples have no support for the trust of knowledge written in RDF and it is not possible to store such metrics in RDF triples directly. In the paper we attempt to define the proper functions to equip the RDF information with trust means.

In this paper a new annotation for RDF is presented, which providing trust metrics. The approach makes possible sharing private and public resources between the sites, according to a trust information. Our proposal extends RDF graphs to fourth value, which symbolizes trust. We also introduce inference rules, annotation algebra, and mapping algorithms for this approach.

The paper is constructed as follows. Section 2 is devoted to related work. In Section 3 we introduce preliminaries concerning our proposal. In Section 4 we propose a solution for trust in RDF graphs. Section 5 presents algorithms for mapping our approach to the RDF model. The paper ends with conclusions.

2 Related Work

In Semantic Web, there have been proposals for trust metrics. They can be divided into categories: based on Web of Trust [6, 13], and based on ontologies [7, 8]. In [6] continuous trust metrics is defined as interval $\{x \in \mathbb{R} : 0 \leq x \leq 1\}$. Yet another proposal in this area is provided in [13], which describes metrics as interval $\{x \in \mathbb{R} : 1 \leq x \leq 10\}$. The authors of [7] extended the Friends of Friends (FOAF) ontology with a value of trust $\{1,2,...,9\}$, where 1 means complete distrust and 9 absolute trust. Another proposal is Hoonoh ontology [8], which in contrast to [7] allows to use a continuous scale.

On the other hand, there are also proposals for annotation algebras and deductive systems. One of the paper [11] presents a simple abstract fragment of RDF, easy to formalize and to reason about. Unfortunately, it do not support trust metrics. Yet another approach is shown in [15], which presents a generic framework for representing and reasoning with annotated Semantic Web data. Unfortunately, the authors do not demonstrate inference rules for RDF with trust. What is important is that, our approach differs from [11, 15] in that we support OWL properties. In [16], the authors discuss how to annotate the predicate, rather than the triple. Unfortunately, it needs specific algorithms, while we describe that a simple extension to the pure RDF inference rules is adequate. In [9] annotation of authority is presented, which is a boolean value. This approach uses Linked Data principles to conservatively determine whether or not some terminological information can be trusted.

3 Preliminaries

Following [5], let I be the set of all Internationalized Resource Identifiers (IRI) references, B an infinite set of blank nodes, L_s the set of RDF plain literals without language tag, L_l the set of RDF plain literals with language tag, and L_d the set of all RDF typed literals. Let $L = L_s \cup L_l \cup L_d$, $O = I \cup B \cup L$ and $S = I \cup B$, then $T \subseteq S \times I \times O$ is set of RDF triples. If $t = \langle s, p, o \rangle$ is a RDF triple, s is subject, p predicate and o object. An RDF graph G is a finite set of RDF triples $G \subseteq T$.

Let us denote by ρ a subset of the vocabularies of RDF [5], RDFS [3] and OWL [14] as follows: $\rho = \{rn, dm, tp, spo, sco, sa, df, io, ep, ec, pdw, dw\}$, where the elements of ρ are as in Table 1.

Table 1 ρ set elements

Set element	Description
rn	`range` predicate (defined in [3])
dm	`domain` predicate (defined in [3])
tp	`type` predicate (defined in [5] and [3])
spo	`subPropertyOf` predicate (defined in [3])
sco	`subClassOf` predicate (defined in [3])
sa	`sameAs` predicate(defined in [14])
df	`differentFrom` predicate(defined in [14])
io	`inverseOf` predicate (defined in [14])
ep	`equivalentProperty` predicate (defined in [14])
ec	`equivalentClass` predicate (defined in [14])
pdw	`propertyDisjointWith` predicate (defined in [14])
dw	`disjointWith` predicate (defined in [14])

Let V_1 be the RDF vocabulary, V_2 the RDFS vocabulary, and V_3 the OWL vocabulary, then $V = V_1 \cup V_2 \cup V_3$ is a vocabulary that includes the RDF, RDFS and OWL vocabularies. A simple interpretation over a vocabulary V is $\mathcal{N} = \langle R^{\mathcal{N}}, P^{\mathcal{N}}, EXT^{\mathcal{N}}, S^{\mathcal{N}}, L^{\mathcal{N}}, LV^{\mathcal{N}} \rangle$, where:

1. $R^{\mathcal{N}}$ is a nonempty set of named resources (the universe of $\mathcal{N}$) $R^{\mathcal{N}} \supseteq I \cup L_s \cup L_l \cup L_d$,
2. $P^{\mathcal{N}}$ is a set which elements are named properties, $P^{\mathcal{N}} \subseteq R^{\mathcal{N}}$,
3. $EXT^{\mathcal{N}}$ is an extension function used to associate properties with their property extension, $EXT^{\mathcal{N}} : P^{\mathcal{N}} \to 2^{R^{\mathcal{N}} \times R^{\mathcal{N}}}$,
4. $S^{\mathcal{N}}$ is a function used to assign the IRIs to resources or properties, $S^{\mathcal{N}} : I \to R^{\mathcal{N}} \cup P^{\mathcal{N}}$,
5. $L^{\mathcal{N}}$ is a function used to assign the typed literals to resources, $L^{\mathcal{N}} : L_d \to R^{\mathcal{N}}$,
6. $LV^{\mathcal{N}}$ is a subset of literal values, $LV^{\mathcal{N}} \subseteq L_s \cup L_l$.

Following [3, 14, 5] $S^{\mathcal{N}}(Property) \in C^{\mathcal{N}}$, $S^{\mathcal{N}}(Resource) \in C^{\mathcal{N}}$, $S^{\mathcal{N}}(Literal) \in C^{\mathcal{N}}$ and $S^{\mathcal{N}}(Class) \in C^{\mathcal{N}}$. The set of classes $C^{\mathcal{N}}$ is defined as $C^{\mathcal{N}} = \{x \in R^{\mathcal{N}} : \langle x, S^{\mathcal{N}}(Class) \rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(tp))\}$, and the mapping $CEXT^{\mathcal{N}} : C^{\mathcal{N}} \to 2^{R^{\mathcal{N}}}$ is

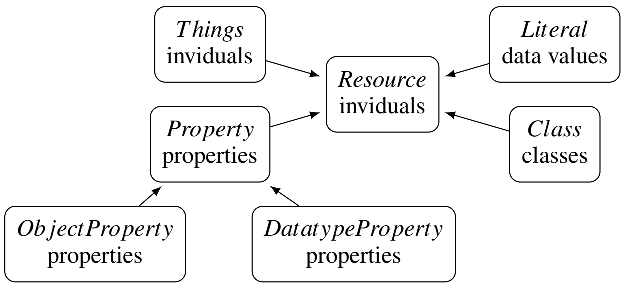


Fig. 1 Parts hierarchy of RDF semantics

defined as $CEXT^{\mathcal{N}}(c) = \{x \in R^{\mathcal{N}} : \langle x,c\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(tp))\}$. All relations between RDF parts are presented in Fig. 1. Let $R^{\mathcal{N}} = CEXT^{\mathcal{N}}(Resource)$, $P^{\mathcal{N}} = CEXT^{\mathcal{N}}(Property)$, $C^{\mathcal{N}} = CEXT^{\mathcal{N}}(Class)$ and $LV^{\mathcal{N}} = CEXT^{\mathcal{N}}(Literal)$, then a simple interpretation satisfies the following conditions:

1. Properties:

1. $x \in P^{\mathcal{N}} \Leftrightarrow \langle x, S^{\mathcal{N}}(Property)\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(tp))$,
2. $\big(\mathfrak{p} \in \{S^{\mathcal{N}}(dm), S^{\mathcal{N}}(rn)\} \wedge \langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \Rightarrow$ $\big(x \in P^{\mathcal{N}} \wedge (\langle u,v\rangle \in EXT^{\mathcal{N}}(x) \Rightarrow u \in CEXT^{\mathcal{N}}(y))\big)$,
3. $\mathfrak{p} \in \{S^{\mathcal{N}}(spo), S^{\mathcal{N}}(ep)\} \Rightarrow \Big(\big(x \in P^{\mathcal{N}} \Rightarrow \langle x,x\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \wedge$ $\big((\langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p}) \wedge \langle y,z\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})) \Rightarrow \langle x,z\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \wedge$ $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p}) \Rightarrow (x,y \in P^{\mathcal{N}} \wedge EXT^{\mathcal{N}}(x) \subseteq EXT^{\mathcal{N}}(y))\big) \Big)$,
4. $\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(ep)) \Rightarrow \langle y,x\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(ep))$,
5. $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(io)) \wedge \langle u,v\rangle \in EXT^{\mathcal{N}}(x)\big) \Rightarrow \langle v,u\rangle \in EXT^{\mathcal{N}}(y)$,
6. $\langle x, pdw, y\rangle \Leftrightarrow \langle x, ep, y\rangle$.

2. Class:

1. $x \in C^{\mathcal{N}} \Rightarrow \langle x, S^{\mathcal{N}}(Resource)\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sco))$,
2. $\big(\mathfrak{p} \in \{S^{\mathcal{N}}(dm), S^{\mathcal{N}}(rn)\} \wedge \langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(\mathfrak{p}))\big) \Rightarrow y \in C^{\mathcal{N}}$,
3. $\mathfrak{p} \in \{S^{\mathcal{N}}(sco), S^{\mathcal{N}}(ep)\} \Rightarrow \Big(\big(x \in C^{\mathcal{N}} \Rightarrow \langle x,x\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \wedge$ $\big((\langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p}) \wedge \langle y,z\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})) \Rightarrow \langle x,z\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \wedge$ $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p}) \Rightarrow (x,y \in C^{\mathcal{N}} \wedge CEXT^{\mathcal{N}}(x) \subseteq CEXT^{\mathcal{N}}(y))\big) \Big)$,
4. $\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(ec)) \Rightarrow \langle y,x\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(ec))$,
5. $\langle x, ec, y\rangle \Leftrightarrow \langle x, dw, y\rangle$.

3. Individuals:

1. $\big(\mathfrak{p} \in \{S^{\mathcal{N}}(sa), S^{\mathcal{N}}(df)\} \wedge \langle x,y\rangle \in EXT^{\mathcal{N}}(\mathfrak{p})\big) \Rightarrow$ $\big(x \in CEXT^{\mathcal{N}}(c) \Leftrightarrow y \in CEXT^{\mathcal{N}}(c)\big)$,
2. $x \in R^{\mathcal{N}} \Rightarrow \langle x,x\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa))$,
3. $\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa)) \Rightarrow \langle y,x\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa))$,
4. $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa)) \wedge \langle x,z\rangle \in EXT^{\mathcal{N}}(p)\big) \Rightarrow \langle y,z\rangle \in EXT^{\mathcal{N}}(p)$,

5. $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa)) \wedge \langle y,z\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa))\big) \Rightarrow$
$\langle x,z\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa))$,
6. $\big(\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(sa)) \wedge \langle x,z\rangle \in EXT^{\mathcal{N}}(p)\big) \Rightarrow \langle y,z\rangle \in EXT^{\mathcal{N}}(p)$,
7. $\langle x,y\rangle \in EXT^{\mathcal{N}}(S^{\mathcal{N}}(df)) \Rightarrow \big(\exists_{p\in P^{\mathcal{N}},z\in R^{\mathcal{N}}}(\langle x,z\rangle \in EXT^{\mathcal{N}}(p) \Longleftrightarrow$
$\langle y,z\rangle \notin EXT^{\mathcal{N}}(p)) \vee (\langle z,x\rangle \in EXT^{\mathcal{N}}(p) \Longleftrightarrow \langle z,y\rangle \notin EXT^{\mathcal{N}}(p))\big)$.

4 Annotated RDF with Trust Metrics

4.1 RDF Quads

An RDF quad Q is an RDF triple with a trust metric. Let M be the set of all trust metrics, then $Q \subseteq S \times I \times O \times M$. If $q = \langle s, p, o, m\rangle$ is a RDF quad, s is subject, p predicate, o object (see Section 3) and m trust metric. The listing in Table 2 presents the example of possible syntax of RDF quad The RDF quads grammar is defined in Appendix 6.

Table 2 Example of RDF quad

```
:me rdf:type :Teacher 0.9
```

An RDF graph with trust metrics (Fig. 2) is $TG = \langle S\cup O, ARC, f, g, I, M\rangle$, where $ARC \subseteq S\times O$, f is a function from ARC to the set of predicates I (see Section 3) $f : ARC \rightarrow I$ and g is a function from ARC to the partially ordered set of trust metrics $M = \{x \in \mathbb{R} : 0 \le x \le 1\}$, $g : ARC \rightarrow M$. An RDF graph with trust metrics TG is a finite set of RDF triples $TG \subseteq Q$.

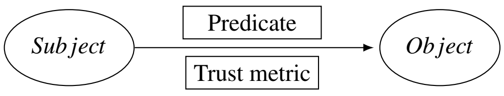


Fig. 2 An RDF graph with trust metrics

4.2 Interpretation and Inference Rules

A $\mathbb{M}$-annotated interpretation is $E = \langle R^{\mathcal{N}}, P^{\mathcal{N}}, EXT^{\mathcal{N}}, S^{\mathcal{N}}, L^{\mathcal{N}}, LV^{\mathcal{N}}, EXT^{\mathbb{M}}\rangle$, where $R^{\mathcal{N}}$, $P^{\mathcal{N}}$, $EXT^{\mathcal{N}}$, $S^{\mathcal{N}}$, $L^{\mathcal{N}}$ and $LV^{\mathcal{N}}$ are exactly the same as for simple interpretation (see Section 3). The $EXT^{\mathbb{M}}$ is a partial function used to trust metric for triple, which are built from property and each pair of resources. $EXT^{\mathbb{M}} : R^{N} \times P^{\mathcal{N}} \times R^{\mathcal{N}} \rightarrow M$, for which the triple $\langle x, p, y\rangle$ is in the domain of $EXT^{\mathbb{M}}$ if and only if $\langle x,y\rangle \in EXT^{\mathcal{N}}(p)$, with $p \in P^{\mathcal{N}}, x,y \in R^{\mathcal{N}}$. The set of classes

$C^{\mathscr{N}}$ is based on $EXT^{\mathscr{N}}$ and also defined exactly the same as for simple interpretation. In the same way, $CEXT^{\mathbb{M}}$ based on $EXT^{\mathbb{M}}$. $CEXT^{\mathbb{M}}$ is a partial function $CEXT^{\mathbb{M}} : C^{\mathscr{N}} \times R^{\mathscr{N}} \rightarrow M$. $\{\langle c,r\rangle : c \in C^{\mathscr{N}} \wedge r \in CEXT^{\mathscr{N}}(c)\}$ is domain of a partial function $CEXT^{\mathbb{M}}$ and $CEXT^{\mathbb{M}}(c,r) = EXT^{\mathbb{M}}(r, S^{\mathscr{N}}(tp), c)$, where $c \in C^{\mathscr{N}}, r \in CEXT^{\mathscr{N}}(c)$.

Let $\mathbb{M} = \langle M, \oplus, \otimes, \ominus, \perp, \top\rangle$ be a boolean algebra, where:

1. M is partially ordered set of trust metrics $M = \{x \in \mathbb{R} : 0 \leq x \leq 1\}$,
2. $\oplus$ is binary operation of addition,
3. $\otimes$ is binary operation of multiplication,
4. $\otimes$ is unary operation of complement,
5. $\perp$ is the smallest element,
6. $\top$ is the largest element.

This structure complies the following conditions:

- associativity: $(a \oplus b) \oplus c = a \oplus (b \oplus c) \wedge (a \otimes b) \otimes c = a \otimes (b \otimes c)$,
- commutativity: $a \oplus b = b \oplus a \wedge a \otimes b = b \otimes a$,
- absorption: $a \oplus (a \otimes b) = a \wedge a \otimes (a \oplus b = a)$,
- distributivity: $a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c) \wedge a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$,
- complements: $a \oplus \ominus a = \top \wedge a \otimes \ominus a = \perp$,

where a, b, c are elements of M. For every $a, b \in M$ we define $a \preceq b$ iff for certain $c \in M$ we have $a \oplus c = b$.

We assume simple interpretation $\overline{\mathscr{N}} = \langle \overline{R}^{\mathscr{N}}, \overline{P}^{\mathscr{N}}, \overline{EXT}^{\mathscr{N}}, \overline{S}^{\mathscr{N}}, \overline{L}^{\mathscr{N}}, \overline{LV}^{\mathscr{N}}\rangle$, which extends E that satisfies $R^{\mathscr{N}} \subseteq \overline{R}^{\mathscr{N}}$, $P^{\mathscr{N}} \subseteq \overline{P}^{\mathscr{N}}$, $p \in P^{\mathscr{N}} \Longrightarrow EXT^{\mathscr{N}}(p) \subseteq \overline{EXT}^{\mathscr{N}}(p)$, $\overline{S}^{\mathscr{N}}|_{\mathfrak{D}(S^{\mathscr{N}})} = S^{\mathscr{N}}$, $\overline{L}^{\mathscr{N}}|_{\mathfrak{D}(L^{\mathscr{N}})} = L^{\mathscr{N}}$ and $LV^{\mathscr{N}} \subseteq \overline{LV}^{\mathscr{N}}$, where $\mathfrak{D}(f)$ denotes the domain of function f. We also consider a partial function $\mathscr{E} : \overline{P}^{\mathscr{N}} \times \overline{R}^{\mathscr{N}} \rightarrow M$, which satisfies $\mathfrak{D}(EXT^{\mathbb{M}}) = \mathfrak{D}(\mathscr{E})$ and $EXT^{\mathbb{M}}(x,p,y) = \mathscr{E}(x,p,y)$, where $\langle x,p,y\rangle \in \mathfrak{D}(EXT^{\mathbb{M}})$.

We construct the trust function $\overline{EXT}^{\mathbb{M}}$, which is natural extension of $\mathscr{E}$, so that the $\overline{\mathscr{N}}$ structure enriched with $\overline{EXT}^{\mathbb{M}}$ is an interpretation. Additionally, we intend that it should comply condition $\mathscr{E}(x,p,y) \preceq \overline{EXT}^{\mathbb{M}}(x,p,y)$, where $\langle x,p,y\rangle \in \mathfrak{D}(\mathscr{E})$.

Let us consider to this end an auxiliary partial function $\mathscr{E}' : \overline{R}^{N} \times \overline{P}^{\mathscr{N}} \times \overline{R}^{\mathscr{N}} \rightarrow 2^{\mathbb{M}}$, such that $\mathfrak{D}(\mathscr{E}') = \{\langle x,p,y\rangle : p \in \overline{P}^{\mathscr{N}} \wedge \langle x,y\rangle \in \overline{EXT}^{\mathbb{M}}(p)\}$ and given dependence $\mathscr{E}'(x,p,y) = \{\mathscr{E}(x,p,y)\}$ if $\langle x,p,y\rangle \in \mathfrak{D}(\overline{EXT}^{\mathbb{M}})$ and $\emptyset$ otherwise.

We will denote by (x,p,y,m) the dependence $m \in (\overline{EXT^{\mathbb{M}}})(x,p,y)$. Using the inference rules (see Table 3) we enrich each sets $(\overline{EXT^{\mathbb{M}}})(x,p,y)$ [1]. Lastly, the partial function $\overline{EXT}^{\mathbb{M}}$ is determined by dependence $\mathfrak{D}(\overline{EXT}^{\mathbb{M}}) = \mathfrak{D}(\mathscr{E}')$ and $\overline{EXT}^{\mathbb{M}}(x,p,y) = \inf \mathscr{E}'(x,p,y)$, where $(x,p,y) \in \mathfrak{D}(\mathscr{E}')$ and $\inf \mathscr{E}'(x,p,y)$ is

[1] x, y, z represent properties, X, Y, Z represent classes, A, B, C represent individuals and m, n are trust metrics.

Table 3 Inference rules

$$\begin{array}{ccc}
\frac{(A,x,B,m)\,(A,x,B,n)}{(A,x,B,m\oplus n)}, & \frac{(x,\mathit{spo},y,m)\,(y,\mathit{spo},z,n)}{(x,\mathit{spo},z,m\otimes n)}, & \frac{(A,x,B,m)\,(x,\mathit{spo},y,n)}{(A,y,B,m\otimes n)}, \\
\frac{(x,\mathit{spo},y,m)}{(x,\mathit{spo},x,m)\,(y,\mathit{spo},y,m)}, & \frac{(A,x,B,m)}{(x,\mathit{spo},x,m)}, & \frac{p\in\rho}{(p,\mathit{spo},p,\top)}, \\
\frac{(X,\mathit{sco},Y,m)\,(Y,\mathit{sco},Z,n)}{(X,\mathit{sco},Z,m\otimes n)}, & \frac{(A,\mathit{tp},X,m)\,(X,\mathit{sco},Y,n)}{(A,\mathit{tp},Y,m\otimes n)}, & \frac{(X,\mathit{sco},Y,m)}{(X,\mathit{sco},X,m)\,(Y,\mathit{sco},Y,m)}, \\
\frac{(A,\mathit{tp},X,\top)}{(X,\mathit{sco},X,\top)}, & \frac{(A,x,B,m)\,(x,\mathit{dm},Y,n)}{(A,\mathit{tp},Y,m\otimes n)}, & \frac{(A,x,B,m)\,(x,\mathit{rn},Y,n)}{(B,\mathit{tp},Y,m\otimes n)}, \\
\frac{(x,q,Y,m)\;q\in\{\mathit{dm},\mathit{rn}\}}{(Y,\mathit{sco},Y,m)}, & \frac{(x,q,Y,m)\;q\in\{\mathit{dm},\mathit{rn}\}}{(x,\mathit{spo},x,m)}, & \frac{(x,\mathit{ep},y,m)\,(y,\mathit{ep},z,n)}{(x,\mathit{ep},z,m\otimes n)} \\
\frac{(A,x,B,m)\,(x,\mathit{ep},y,n)}{(A,y,B,m\otimes n)}, & \frac{(x,\mathit{ep},y,m)}{(x,\mathit{ep},x,m)\,(y,\mathit{ep},y,m)}, & \frac{(A,x,B,m)}{(x,\mathit{ep},x,m)}, \\
\frac{p\in\rho}{(p,\mathit{ep},p,\top)}, & \frac{(x,\mathit{ep},y,m)}{(y,\mathit{ep},x,m)}, & \frac{(X,\mathit{ec},Y,m)\,(Y,\mathit{ec},Z,n)}{(X,\mathit{ec},Z,m\otimes n)}, \\
\frac{(A,\mathit{tp},X,m)\,(X,\mathit{ec},Y,n)}{(A,\mathit{tp},Y,m\otimes n)}, & \frac{(X,\mathit{ec},Y,m)}{(X,\mathit{ec},X,m)\,(Y,\mathit{ec},Y,m)}, & \frac{(A,\mathit{tp},X,\top)}{(A,\mathit{ec},A,\top)}, \\
\frac{(X,\mathit{ep},Y,m)}{(Y,\mathit{ep},X,m)}, & \frac{(x,\mathit{pdw},y,m)\,(y,\mathit{pdw},z,n)}{(x,\mathit{pdw},z,m\otimes\ominus n)}, & \frac{(A,x,B,m)\,(x,\mathit{pdw},y,n)}{(A,y,B,m\otimes\ominus n)}, \\
\frac{(x,\mathit{pdw},y,m)}{(x,\mathit{pdw},x,\ominus m)\,(y,\mathit{pdw},y,\ominus m)}, & \frac{(A,x,B,m)}{(A,\mathit{pdw},A,\ominus m)}, & \frac{p\in\rho}{(p,\mathit{pdw},p,\bot)}, \\
\frac{(X,\mathit{dw},Y,m)\,(Y,\mathit{dw},Z,n)}{(X,\mathit{dw},Z,m\otimes\ominus n)}, & \frac{(A,\mathit{tp},X,m)\,(X,\mathit{dw},Y,n)}{(A,\mathit{tp},B,m\otimes\ominus n)}, & \frac{(X,\mathit{dw},Y,m)}{(X,\mathit{dw},X,m)\,(Y,\mathit{dw},Y,\ominus m)}, \\
\frac{(A,x,B,m)\,(x,\mathit{io},y,n)}{(A,y,B,m\otimes n)}, & \frac{(A,x,B,m)\,(A,\mathit{sa},C,n)}{(C,x,B,m\otimes n)}, & \frac{(A,x,B,m)\,(A,\mathit{df},C,n)}{(C,x,B,m\otimes\ominus n)}, \\
\frac{(A,x,B,m)\,(B,\mathit{sa},C,n)}{(C,x,A,m\otimes n)}, & \frac{(A,x,B,m)\,(B,\mathit{df},C,n)}{(C,x,A,m\otimes\ominus n)}, & \frac{(A,\mathit{sa},B,m)}{(B,\mathit{sa},A,m)}, \\
\frac{(A,\mathit{df},B,m)}{(B,\mathit{df},A,m)}, & \frac{(A,\mathit{tp},X,\top)}{(A,\mathit{sa},A,\top)}, & \frac{(A,\mathit{sa},B,m)\,(B,\mathit{sa},C,n)}{(A,\mathit{sa},C,m\otimes n)}.
\end{array}$$

evaluated $\preceq$ relation. Finally, $\overline{E} = \langle \overline{R}^{\mathscr{N}}, \overline{P}^{\mathscr{N}}, \overline{EXT}^{\mathscr{N}}, \overline{S}^{\mathscr{N}}, \overline{L}^{\mathscr{N}}, \overline{LV}^{\mathscr{N}}, \overline{EXT^{\mathbb{M}}} \rangle$ is extended interpretation of $\mathbb{M}$-annotation.

5 Mapping into RDF Model

In this Section the mapping from our approach to the RDF model is presented. We propose simple ontology for trust metric. We define `trustValue` property, where trust metric can be stored. A range of the property uses subset $\{x \in \mathbb{R} : 0 \leq x \leq 1\}$ as typed literal. Listing in Table 4 presents Simple Trust Ontology (STO) in Terse RDF Triple Language [12].

We present two methods for mapping our proposal to RDF model: reification of the statement and Named Graphs [4].

The first method is based on the built-in vocabulary intended for describing RDF statements. Our reification proposal consists of the type `Statement`, and the properties: `subject`, `predicate`, `object`, and `trustValue`. The algorithm in Table 5 presents the process of transforming our approach, which uses RDF

Table 4 Simple Trust Ontology

```
:trustValue rdf:type owl:DatatypeProperty;
            rdfs:range [rdf:type rdfs:Datatype;
                        owl:onDatatype xsd:float;
                        owl:withRestrictions (
                                  [xsd:minInclusive 0]
                                  [xsd:maxInclusive 1])
                       ].
```

Table 5 Algorithm of mapping to reification statements

```
input: TG
output: G
foreach quads do
  get triple from q = (s, p, o, m)
  create unique subject US
  create predicates "subject" PS, "predicate" PP, "object" PO
  insert triples (US, PS, s), (US, PP, p), (US, PO, o)
  get metric from q = (s, p, o, m)
  create predicate "trustValue" PV
  insert triple (US, PV, m)
end
```

Table 6 Example of reification statements

```
_:st rdf:type rdf:Statement;
     rdf:subject :me;
     rdf:predicate rdf:type;
     rdf:object :Teacher;
     sto:trustValue "0.9"^^xsd:float .
```

Table 7 Algorithm of mapping to Named Graphs

```
input: TG
output: NG, G
foreach quads do
  create unique named graph UNG
  get triple from q = (s, p, o, m)
  insert (s, p, o) into UNG
  create subject UNG
  create predicate "trustValue"
  get metric from q = (s, p, o, m)
  create object m
end
```

Table 8 Example of Named Graphs

```
metric:tg { :me rdf:type :Teacher }
metric:tg sto:trustValue "0.9"^^xsd:float .
```

reification. An example of mapping RDF quads, shown in Table 2, is presented in Table 6. This example is expressed in Terse RDF Triple Language [12].

The second method is based on Named Graphs [4]. Named Graphs NG is a set of pairs forming a partial function from I to T (see Section 1) and each pair $(n,g) \in NG$, where n is called *name* and g is called *graph*. Our proposal consists of the triple in named graph and `trustValue` property in default graph. The algorithm in Table 7 presents the process of transforming, which uses Named Graphs. An example of mapping RDF quads, shown in Table 2, is presented in Table 8. This example is provided in TriG [2].

However, the second method has two problems: many pure RDF providers would not be forwards-compatible with such quad-centric data representation and none of named graph syntaxes are standards at the moment. But the second approach allows for more compact annotation serialization than the first method.

6 Conclusions

The problem of how to support trust in RDF has produced many proposals. We assume that RDF, to be more functional, should provide a method to set the trust metrics. There is a need to be able to attach it to the RDF triples.

We have produced a simple and thought-out proposal for representing and reasoning with trust annotated RDF data. We propose how our annotation with metric algebra can be used for processing data on inferred RDF graphs. We believe that our idea is an interesting approach, because it can be transformed to pure RDF model. More importantly, we have provided algorithms for mapping our proposal. Our approach excellently extends the classical case of RDF with trust annotations. A crucial advantage of the proposal is that it can work either with graph stores or serialized RDF documents.

References

1. Adams, C., Lloyd, S.: Understanding PKI: Concepts, Standards, and Deployment Considerations. Sams (2002)
2. Bizer, C., Cyganiak, R.: The trig syntax. Tech. rep., Freie Universität Berlin (2007)
3. Brickley, D., Guha, R.V.: Rdf vocabulary description language 1.0: Rdf schema. Tech. rep., World Wide Web Consortium (2004)

4. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: Proceedings of the 14th International Conference on World Wide Web, pp. 613–622 (2005)
5. Cyganiak, R., Wood, D.: Resource description framework (rdf): Concepts and abstract syntaxrdfcas. Tech. rep., World Wide Web Consortium (2012)
6. Golbeck, J.: Combining provenance with trust in social networks for semantic web content filtering. Provenance and Annotation of Data, 101–108 (2006)
7. Golbeck, J., Parsia, B., Hendler, J.: Trust networks on the semantic web. Cooperative Information Agents VII, 238–249 (2003)
8. Heath, T., Motta, E.: The hoonoh ontology for describing trust relationships in information seeking. In: Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008), pp. 67–75 (2008)
9. Hogan, A., Bonatti, P., Polleres, A., Sauro, L.: Robust and scalable linked data reasoning incorporating provenance and trust annotations. Web Semantics: Science, Services and Agents on the World Wide Web (2011)
10. Marsh, S.: Formalising trust as a computational concept. Ph.D. thesis. University of Stirling (1994)
11. Munoz, S., Pérez, J., Gutierrez, C.: Minimal deductive systems for rdf. The Semantic Web: Research and Applications, 53–67 (2007)
12. Prud'hommeaux, E., Carothers, G.: Turtle - terse rdf triple language. Tech. rep., World Wide Web Consortium (2012)
13. Richardson, M., Agrawal, R., Domingos, P.: Trust Management for the Semantic Web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 351–368. Springer, Heidelberg (2003)
14. Schneider, M.: Owl 2 web ontology language rdf-based semantics. Tech. rep., World Wide Web Consortium (2009)
15. Straccia, U., Lopes, N., Lukácsy, G., Polleres, A.: A general framework for representing and reasoning with annotated semantic web data. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1437–1442 (2010)
16. Udrea, O., Recupero, D., Subrahmanian, V.: Annotated rdf. The Semantic Web: Research and Applications, 487–501 (2006)
17. Zimmermann, P.: PGP User's Guide. MIT Press (1994)

Appendix: RDF Quads Syntax

Listing in Table 9 presents the syntax of RDF quads. All possible terminals are taken from Terse RDF Triple Language [12].

Table 9 RDF quads EBNF

```
 1.  quadDoc            ::=  WS* statement? (EOL WS* statement)*
                             WS* EOL?
 2.  statement          ::=  subject WS+ predicate WS+ object
                             (WS+ metric) WS* "."
 3.  subject            ::=  IRI_REF | BLANK_NODE_LABEL
 4.  predicate          ::=  IRI_REF
 5.  object             ::=  IRI_REF | BLANK_NODE_LABEL |
                             literal
 6.  metric             ::=  0.[0-9]+ | 1
 7.  literal            ::=  STRING_LITERAL2 ("^^" IRI_REF |
                             ("@" LANG) )?
 8.  IRI_REF            ::=  "<" ([^<>"{}|^`\]-[#x00-#x20] |
                             UCHAR )* ">"
 9.  BLANK_NODE_LABEL   ::=  "_:" PN_LOCAL
10.  LANG               ::=  [a-zA-Z]+ ( "-" [a-zA-Z0-9]+ )*
11.  WS                 ::=  " " | "\t" | "\r" | "\n"
12.  EOL                ::=  [\r\n]+
13.  UCHAR              ::=  ( "\\u" HEX HEX HEX HEX ) |
                             ( "\\U" HEX HEX HEX HEX HEX HEX
                             HEX HEX )
14.  PN_LOCAL           ::=  ( PN_CHARS_U | [0-9] )
                             ( ( PN_CHARS | "." )* PN_CHARS )?
15.  STRING_LITERAL2    ::=  '"' ( ( [^\"\\\n\r] ) | ECHAR |
                             UCHAR )* '"'
16.  ECHAR              ::=  "\\" [tbnrf\\\"']
17.  HEX                ::=  [0-9] | [A-F] | [a-f]
18.  PN_CHARS_U         ::=  PN_CHARS_BASE | "_"
19.  PN_CHARS           ::=  PN_CHARS_U | "-"  | [0-9] |
                             #00B7 | [#0300-#036F] |
                             [#203F-#2040]
20.  PN_CHARS_BASE      ::= [A-Z] | [a-z] | [#00C0-#00D6] |
                            [#00D8-#00F6] | [#00F8-#02FF] |
                            [#0370-#037D] | [#037F-#1FFF] |
                            [#200C-#200D] | [#2070-#218F] |
                            [#2C00-#2FEF] | [#3001-#D7FF] |
                            [#F900-#FDCF] | [#FDF0-#FFFD] |
                            [#10000-#EFFFF] | UCHAR
```

Inference of XML Integrity Constraints*

Matej Vitásek and Irena Mlýnková

Abstract. In this paper we expand upon the previous efforts to infer schema information from existing XML documents. We focus on inference of *integrity constraints*, more specifically `ID`/`IDREF`/`IDREFS` attributes in DTD. Building on the research by Barbosa and Mendelzon (2003) we introduce a heuristic approach to the problem of finding an optimal ID set. The approach is evaluated and tuned in a wide range of experiments.

1 Introduction

The XML is one of the leading formats for representing structured data. However, even though languages such as DTD and XML Schema [4] to describe XML structure exist for a long time, most of the documents use outdated or have no schema at all [17]. To tackle this problem one may employ reverse-engineering techniques to infer the schema from existing documents [1, 3]. In particular, [12] introduces the *jInfer* schema inference framework, dealing primarily with the structural parts of the schema: how all the elements, attributes and text data are to be organized in an XML document conforming to that schema.

But the structural requirements of a schema is not the only constraint that can be imposed on an XML document. The concept of *keys* and *foreign keys*, well known from the relational database world, applies to schemas as well and will be the topic of this paper.

From all the constraints that can be applied to an XML document by means of its schema, this paper will focus on keys and foreign keys. The most important concepts in this field are introduced in [5] and formalized in the notions of `ID`/`IDREF`/

Department of Software Engineering, Charles University in Prague, Czech Republic
e-mail: vektor330@gmail.com, mlynkova@ksi.mff.cuni.cz

* The paper was supported by the GAČR grant no. P202/10/0573.

`IDREFS` attributes in DTD and XSD and `xs:key/xs:keyref` structures in XSD [4]. The scope of this paper is focussed on inference of `ID`/`IDREF`/`IDREFS` from existing XML documents. We discuss the equivalence of *ID set search* and *maximum weighted independent set*. We introduced the *mixed integer problem* and demonstrated how to solve it using external GLPK[1] solver in the environment of the *jInfer* framework [13].

The paper is structured as follows: In Section 2 we describe all necessary definitions and terms used in the rest of the text. In Section 3 we discuss the related papers and problems. In Section 4 we specify the MIP approach and in Section 5 we discuss related experiments. Finally in Section 6 we conclude.

2 Definitions

We use the representation introduced in [2], where an XML file is represented by a labeled tree consisting of nodes for elements, attributes and simple text data. Parent nodes are connected to child nodes with edges. This tree shall be called an *XML tree*. For a given node v of an XML tree we define $label(v)$ (name of the node in the document, only for elements and attributes), $id(v)$ (unique identifier across the document) and $value(v)$ (text content, only for attributes and simple text data). We denote $\mathscr{I}$ the set of all ids in the document. Without loss of generality we ignore the actual ordering of nodes in the tree.

From [4]: an XML attribute may have the type `ID`, `IDREF` or `IDREFS`. In short, `ID` attribute values must be unique across the document, element cannot have more than one `ID` attribute, `IDREF` and `IDREFS` values must reference an `ID` value.

Attribute Mapping

We will use a slightly different definition of *attribute mappings* (AMs) than [2] (without introducing keys from [5]) that will however give us the same result.

Definition 1 (Σ^E, Σ^A, Σ). Σ^E *is the* set of all element labels*,* Σ^A *is the* set of all attribute labels*.* $\Sigma = \Sigma^E \cup \Sigma^A$ *is their union and effectively the* set of all labels *in the document.*

Definition 2 (Attribute mapping). *For* $x \in \Sigma^E$ *and* $y \in \Sigma^A$ *we define the* attribute mapping *of* y *over* x*, denoted* M_x^y*, the* $\mathscr{I} \times \mathscr{I}$ *relation defined by*

$$M_x^y = \{(z,w) : label(z) = x, label(w) = y, parent(w) = z\}.$$

Thus the relation M_x^y contains edges in the XML tree connecting element nodes labeled x and attribute nodes labeled y. We can use a projection to retrieve all the unique *ids* of either elements or attributes from the relation, with notation $\pi_E(M_x^y)$ and $\pi_A(M_x^y)$.

[1] GNU Linear Programming Kit, `http://www.gnu.org/s/glpk/`

Definition 3 (Type of the attribute mapping). *AM M_x^y is of the* type $\tau(M_x^y) = x$.

Definition 4 (Image of attribute mapping). Image ι *of the attribute mapping M_x^y is defined as* $\iota(M_x^y) = \{z : z = value(w), w \in \pi_A(M_x^y)\}$.

So the image of an AM is a set of all the values of all attribute nodes contained in the mapping.

Definition 5 ($name()$). *Given an attribute mapping $m = M_x^y$, $name(m)$ shall be defined as the string $x - y$.*

ID Set

Based on the requirements for an `ID` attribute [4] we define ID set with the help of the following definition.

Definition 6 (Candidate attribute mapping). *An attribute mapping m is a* candidate attribute mapping *if it is an injective function, that is,* $|m| = |\pi_E(m)| = |\pi_A(m)| = |\iota(m)|$.

Definition 7 (ID set). *A set of candidate attribute mappings $I = \{m_1, \dots m_n\}$ is an* ID set *iff*

$$\bigcap_{m_i \in I} \tau(m_i) = \emptyset \; and \; \bigcap_{m_i \in I} \iota(m_i) = \emptyset.$$

That is, an ID set has images without repeating values and all the types are unique (an element cannot have more than one `ID` attribute).

Given an ID set I, the requirements from [4] give us the following condition for an attribute mapping m to be marked `IDREF` (`IDREFS` in case of multivalued attributes): $\iota(m) \subseteq \bigcup_{m_i \in I} \iota(m_i)$.

Attribute Mapping Weight

This definition of weight for AMs or AM sets comes from [2] again. Let $M = \{m_1, \dots m_i\}$ be the set of all non-empty AMs in the document.

Definition 8 (Support). Support *of AM m is defined as* $\phi(m) = \frac{|m|}{\sum_{p \in M} |p|}$.

Definition 9 (Coverage). Coverage *of AM m is defined as*

$$\chi(m) = \left(\sum_{p \in M, p \neq m} |\iota(m) \cap \iota(p)| \right) / \sum_{p \in M} |\iota(p)|.$$

Definition 10 (Weight). *For $\alpha, \beta \geq 0$ as relative priorities of support and coverage we define the AM* weight *as $weight(m) = \alpha.\phi(m) + \beta.\chi(m)$. For a set of AMs $S = \{m_1, \dots m_i\}$ we define $weight(S) = \sum_{m \in S} weight(m)$.*

Definition 11 (Independent set). *Given an undirected graph $G = (V,E)$, a set of vertices $I \subseteq V$ is an* independent set, *iff* $\forall v_1, v_2 \in I, v_1 \neq v_2 : (v_1, v_2) \notin E$.

Definition 12 (Maximum weighted independent set). *Given an undirected graph $G = (V,E)$ and a weight function $w : V \to \mathbb{R}$, an independent set I_{max} is the* maximum weighted independent set, *iff the following is satisfied:*

$$\forall I' \subseteq V, I' \textit{is an independent set} : \sum_{v \in I'} w(v) \leqslant \sum_{v \in I_{max}} w(v).$$

Finding the maximum weighted IS is an NP-hard optimization problem [15].

Linear Programming

The problem of *linear programming* is optimization of a linear function under a set of linear constraints. The formulation is usually called a *linear program* (LP). It can be written in the following form (a minimization version is possible, too).

$$\begin{aligned} \max_{x} z &= \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ s.t. A\mathbf{x} &\leqslant \mathbf{b} \\ \mathbf{x} &\geqslant 0 \end{aligned}$$

where **x** is the vector of variables (to be found by the optimization), **b** is the vector and A its accompanying matrix of constraints and **c** is the vector of coefficients for the objective function.

Definition 13 (Mixed integer problem). *A* mixed integer problem *(MIP) is an instance of LP in which some or all variables are limited to integral or boolean values.*

While solving LP is usually possible in polynomial time using the *simplex algorithm* (see [6]), solving MIP in general is *NP*-hard.

3 Related Work

According to the article [2, Chapter 4], the problem of finding an ID set with weight more than some given K (K-IDSET) is in NP. Furthermore, the independent set (IS) problem can be reduced to K-IDSET, meaning K-IDSET is NP-hard and thus NP-complete. The article continues by proving that finding the maximum weighted IS can be reduced to the problem of finding an ID set with maximum weight (MAX-IDSET). This again means that MAX-IDSET is NP-complete and, furthermore, unless $P = NP$, MAX-IDSET has no constant factor approximation algorithm. The transformation works in both ways: it is equivalently possible to create a maximum weighted IS instance for a given MAX-IDSET instance. The authors then suggest a heuristic algorithm, which is incorporated into the framework proposed by this paper. To the best of our knowledge, there are no other articles dealing with this problem.

Finding XML Keys

XML keys are a structure somewhat similar to `ID` attributes, but with a much larger expressive strength. They have been introduced in [5] and implemented in XML Schema[2].

Fajt in [8] summarizes several algorithms to help find XML keys in existing data, namely *Gordian*, *XML Primary Keys*, *SPIDER* and *DBA Companion*. Except for *XML Primary Keys*, they all are originally purposed to find keys in relational databases.

In our paper we find `ID` attributes directly. And even though we can always convert them to XML keys by the process mentioned above, we are unable to find more complex keys this way.

Maximum Weighted IS

Maximum weighted IS is a well researched topic with a lot of known direct or approximation algorithms, see e.g. [15] or [9]. According to [14], the best known approximation algorithm for weighted IS to-date achieves an approximation ratio of $3(\Delta+2)$, where Δ is the maximum degree of a vertex in the IS graph.

4 MIP Approach

In this section we introduce a new approach to finding maximum ID sets. First, we transform the problem formulation to maximum weighted IS problem formulation. Then we transform this into an MIP formulation and demonstrate how this can be solved using a solver such as GLPK. We will continue by applying heuristic approaches to improve the performance of the process.

4.1 ID Set to IS Formulation

Given $C = \{m_1, \ldots, m_n\}$ a set of all AMs in a document, we construct a graph $G = (V, E)$ as follows. For each AM $m_i \in C$ we create a vertex $v_{name(m_i)}$. Two vertices $v_{name(m_i)}$ and $v_{name(m_j)}$ shall be connected by an edge iff they cannot share the same ID set, either because they have the same type ($\tau(m_i) = \tau(m_j)$), or their images intersect ($\iota(m_i) \cap \iota(m_j) \neq \emptyset$). Weight of a vertex $v_{name(m_i)}$ is the weight of the attribute mapping: $w(v_{name(m_i)}) = weight(m_i)$.

[2] `http://www.w3.org/TR/xmlschema11-1/#Identity-constraint_Definition_details`

Now finding the maximum weighted IS in G finds the maximum (optimal) ID set in the original document.

4.2 IS to MIP Formulation

Given a graph $G = (V,E)$ with a weight function $w : V \to \mathbb{R}$, we introduce a binary variable x_i for each vertex $v_i \in V$ and an inequality constraint $x_i + x_j \leq 1$ for each edge $e = (v_i, v_j) \in E$. Furthermore we introduce an objective function in form $\sum_{x_i} x_i w(v_i)$.

It is obvious that the objective function and all the constraints consitute a MIP instance, and that solving it finds the maximum weigthed IS in G.

4.3 Finding ID Sets with GLPK

By chaining these two translations we can create a MIP formulation for a given set of AMs from a document. Solving this MIP instance will give us the optimal ID set for this document.

GLPK is a multi-platform, multi-purpose solver well suited for this task. This approach works and for any possible input we can let GLPK find the optimal solution. However, sometimes it takes too long to find the optimum, thus the aim of this paper is to improve this process.

A *heuristic* is an approach to problem solving based on prior experience, educated guess or common knowledge. A *heuristic algorithm* is one that, in a reasonably short time, generates a good, maybe even optimal solution to an optimization problem. However, it will not provide any formal guarantee about its quality.

While a heuristic algorithm can be seen as a tool designed to solve one specific problem, the notion of a *metaheuristic*[3] is a recipe to solve a whole family of problems.

Definition 14 (Solution Space, Solution Quality). Solution space *in general is the set of all permissible solutions (not violating any constraints). In the specific case of a MIP formulation it is the set of all* $\mathbf{x}$ *subject to* $A\mathbf{x} \leqslant \mathbf{b}$. *Every solution in the solution space has its* quality*, in case of MIP for solution* $\mathbf{x}$ *it is the value of the objective function in* $\mathbf{x}$.

Definition 15 (Solution Neighborhood). *A* neighborhood *of a solution* $\mathbf{x}$ *in the solution space are all the other solutions close to* $\mathbf{x}$ *according to some metric.*

The precise definition of the neighborhood is always adjusted according to specific needs. However, the neighborhood should be defined so that it is *continuous* (with respect to quality).

[3] A metaheuristic covers a wide range of topics in the field of heuristics, such as *Tabu Search* [10], *Ant Colony Optimization* [7] or *Genetic Algorithms* [11], to name a few.

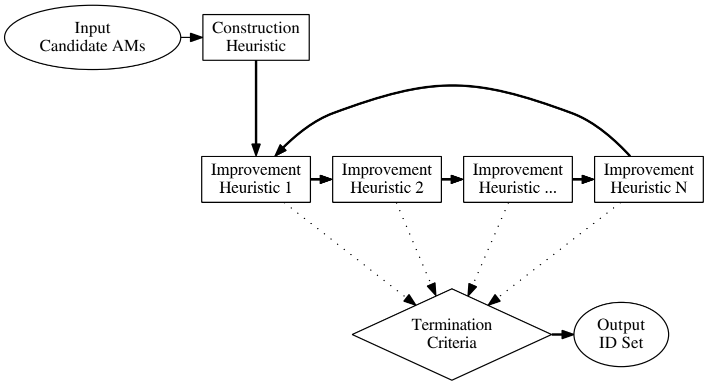


Fig. 1 Metaheuristic Schema

Definition 16 (Solution Pool, Incumbent Solution). Solution pool *(sometimes called* pool of feasible solutions *or* feasible pool*) is a set of solutions of different qualities that were found in one of the stages of the metaheuristic. The solution with the highest quality is called the* incumbent *solution.*

Our problem is the following: given a list of AMs with weights, find a non-conflicting subset maximizing the sum of weights in the subset. This sum will be the quality of the solution (subset). Our metaheuristic from Figure 1 works as follows: First we take the list of candidate AMs and ask a *construction heuristic* (CH) to provide us with a pool of solutions. Then, in a loop, we use this pool as input for *improvement heuristics* (IH) and in turns ask them to improve it. All the time we check whether *termination criteria* are met, and if so, we terminate the metaheuristic. The incumbent solution from the last pool is then the output ID set.

Constructions Heuristics

The purpose of a CH is to find at least one solution. Some IHs can profit from having several sub-optimal solutions, and some CHs can produce this pool from them. We implemented the following CHs:

- *FIDAX*: The deterministic algorithm described in [2] gives us one feasible solution.
- *Random*: Greedy algorithm, picks AMs at random and adds them if possible.
- *Fuzzy*: Greedy algorithm, similar to *Random*, but picks on weighted random.
- *Incremental*: Greedy algorithm, iterates AMs by descending weight, adds if possible.
- *Removal*: Greedy algorithm, adds all AMs, then iterates by ascending weight and removes until the ID set condition is not violated.

- `Glpk` (Truncated Branch & Bound): This is a time-constrained run of GLPK: limiting the run time gives us the best solution found so far. To be able to create a pool of solutions, the GLPK input is always shuffled. This causes the solver to explore the search tree in various orders, producing different solution in each run.

Improvement Heuristics

IHs start with a solution pool, attempt to improve one or more solutions in it and then return this improved pool. *Intensification* is the attempt to move the solution towards the nearby local optimum in the solution space. *Diversification* is the attempt to escape the local optimum, to be able to explore more of the solution space when the metaheuristic starts stagnating. A metaheuristic needs to combine both tendencies to explore the solution space and finally arrive at a local optimum.

We implemented the following IHs:

- `Remove Worst`: Removes the worst solution from the pool.
- `Random Remove`: Removes a random subset of specified size from each solution in the pool.
- `Hungry`: Intensification IH, tries to improve each solution in the pool by iteratively adding AMs sorted by weight, whenever possible.
- `Mutation`: Assumes that an incumbent solution already contains some AMs from the optimal solution. It takes a random guess and fixes some them, i.e. adds new constraints to the MIP formulation setting values of the respective variables to 1. This new formulation contains less free variables and is easier to solve, probably even to optimum. GLPK is run again with the constrained formulation. The solution found this way is a feasible solution of the original problem, but its optimum is not necessarily the same as in unconstrained formulation. It is intensification: we limit the search to the neighborhood of incumbent solution.
- `Crossover`: This IH looks for commonalities among the solutions in the pool. This is based on the idea that if more solutions agree on the same AMs, those probably are in the optimal solution, too. `Crossover` takes a parameter – fraction of solutions in the pool among which to look for commonalities. AMs found in every one of them are fixed in the modified MIP formulation the same way as in `Mutation`. This is again intensification.
- `Local Branching`: Another intensification IH, where the neighborhood being searched is defined by edit distance. The incumbent solution is represented by a vector of ones and zeroes, signifying which AMs belong to the ID set. A new constraint is added to the MIP formulation: for every other solution its edit distance, i.e. number of positions in which this solution and the incumbent solution differ, will have to be less than some threshold.

5 Experiments

Our aim in performing experiments is: firstly, to describe behavior of the system and its components. Secondly, to evaluate performance in terms of speed and results quality. And finally, to find the "best" heuristic configuration.

In our experiments, we are using *realistic*, *converted* and *artificial* XML documents. Converted documents were created taking some of the values in text nodes and converting them to attributes. Artificial documents are randomly generated and have the following structure:

```
<graph>
  <vertex0 attr="-296887"/>
  <vertex1 attr="1729745"/>
  <vertex2 attr="-902054/>
  ...
  <vertex99 attr="-75459"/>

  <vertex82 attr="0"/><vertex21 attr="0"/>
  <vertex64 attr="1"/><vertex21 attr="1"/>
  <vertex44 attr="2"/><vertex2 attr="2"/>
  ...
  <vertex96 attr="99"/><vertex40 attr="99"/>
</graph>
```

It is interesting to look at graphs from the IS interpretation of our problem. Two of them are in Figure 2, one for a realistic XML document, one for an artificial. A quick reminder: vertices are AMs, edges connect pairs of AMs that cannot be in the same ID set together.

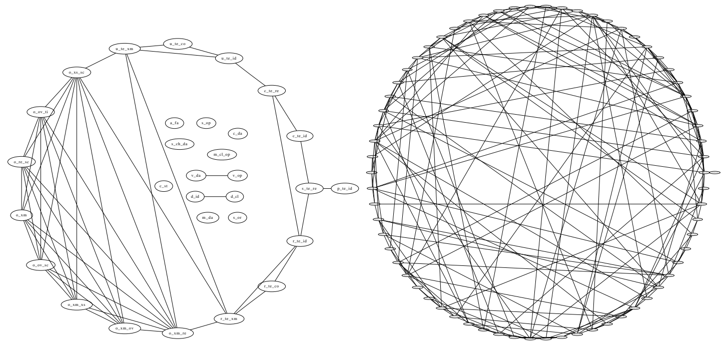

(a) `OVA3` (b) `70-245`

Fig. 2 IS Graphs for XML Documents

The size and density of these IS graphs determines how hard it is to solve the maximum ID set problem. It is obvious that the artificial document (`70-245`) is much harder – we created these artificial XML files to test our algorithms in the worst-case scenarios.

We performed a number experiments, but due to space limitations, we cannot possibly list them all here. Please refer to [16] for details. Only key findings will be mentioned here.

`Fuzzy` outperforms `FIDAX`

First, we compared `FIDAX` from the original article [2] to our trivial CHs `Random` and `Fuzzy`. Times for each of the algorithms were very similar. However, `FIDAX` was outperformed in terms of solution quality in most cases by our most trivial construction heuristics. Interestingly enough, adding our trivial IH `Hungry` after `FIDAX`, we were able to improve the quality of its results. This means, that `FIDAX` is not “greedy enough”.

Best standalone CH is `Glpk`

We tried to find the CH that, on its own, produces the best results. The only CH with a parameter is `Glpk`, where we set the time limit to 1 second. It became obvious that pure `Glpk` is best one for this setup.

Best IH for `Glpk` is `Mutation`

After we found `Glpk` to be the best CH, we tried to find the best IH to complement it. This means, our metaheuristic setup for this case was `Glpk`$(limit = 1s) \rightarrow IH$, with the selected IH constrained for 1 run only. We found that the best IH for this purpose is `Mutation`, with time limit of 1 second (this was a common limit) and $fraction = 0.1$.

Chaining IHs – strategy

Next we tried the full strength of our metaheuristic approach by chaining several IHs together in “strategies” and comparing them. Our chosen CH was `Random` with pool size 10. The best strategy was the following.

`RndRemove` $\rightarrow$ `Mutation` $\rightarrow$ `RndRemove` $\rightarrow$ `Crossover` $\rightarrow$ `RemoveWorst` $\rightarrow \ldots < repeat >$.

In all IHs that support it, the fraction was 0.1 and the time limit 1 second. Results of several runs of this strategy are visualized in Figure 3. Each broken line is a representation of one run. X axis is time, Y axis is quality. Each break in the line indicates an IH finishing its work and handing the pool over to the next one, in a specific time and with a specific incumbent quality. Some lines terminate soon – this happens when the known optimum is reached.

Chaining IHs – tuning

We then tuned fractions in IHs in our “best” strategy. We found the best values: 0.5 for `RandomRemove`, 0.2 for `Mutation` and 0.1 for `Crossover`[4].

[4] Interpretation: `RandomRemove` fraction of 0.5 means that a randomly chosen half of all AMs from every ID set in the pool will be discarded. This amounts to a very strong diversification tendency and keeps the strategy from stalling in local optima. `Mutation` fraction of 0.2 means around 1/5th of AMs in the incumbent solution will be fixed for the next GLPK optimization. `Crossover` fraction of 0.1 means that around 10% of ID sets in the pool (randomly chosen) will be scanned for common AMs.

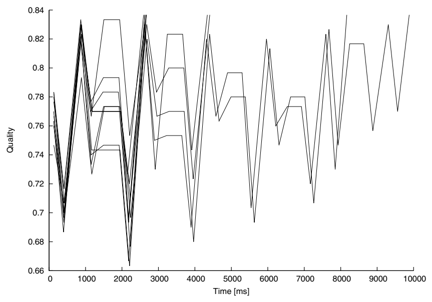


Fig. 3 Chained IHs – `100-100` Results

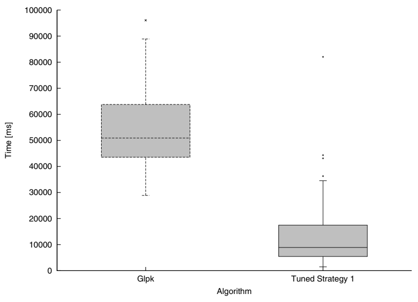


Fig. 4 Chained IHs – Pure `Glpk` vs. Tuned Strategy

Comparison with pure `Glpk`

Finally, we compared our tuned strategy to the performance of pure `Glpk`. We measured the time needed to find optimum in the hardest test XML document. The results are in Figure 4. Our tuned strategy found optimum in on average almost 4x shorter times than pure `Glpk` and under 10 seconds in more than a half of cases.

6 Conclusion

From all the integrity constraints in XML we chose the `ID`/`IDREF`/`IDREFS` attributes and decided to improve upon the search for them. We discussed the approach from [2] and the equivalence of ID set search and maximum weighted independent set. We introduced the MIP approach and demonstrated how to solve it using external GLPK solver in the environment of *jInfer* framework. Afterwards, we introduced a range of construction and improvement heuristics. We combined these algorithms to create a metaheuristic and performed a number of experiments to understand its behavior. Finally, we selected a promising metaheuristic strategy and tuned its parameters to find very good ID sets in short times.

A straightforward extension granting the ability to handle more than one input XML file has already been suggested in [2]. It is easy to add more CHs and IHs, as well as more metaheuristics using the existing IHs. A starting point is in [16]. It would be interesting to compare performance of our metaheuristic against two other interesting classes: Ant Colony Optimization and Genetic Programming.

References

1. Ahonen, H.: Generating Grammars for Structured Documents Using Grammatical Inference Methods. Ph.D. thesis, Department of Computer Science, University of Helsinki. Series of Publications A, Report A-1996-4 (1996)

2. Barbosa, D., Mendelzon, A.: Finding ID Attributes in XML Documents. In: Bellahsène, Z., Chaudhri, A.B., Rahm, E., Rys, M., Unland, R. (eds.) XSym 2003. LNCS, vol. 2824, pp. 180–194. Springer, Heidelberg (2003)
3. Bex, G.J., Neven, F., Vansummeren, S.: SchemaScope: a System for Inferring and Cleaning XML Schemas. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, pp. 1259–1262. ACM, New York (2008)
4. Bray, T., Paoli, J., Maler, E., Yergeau, F., Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0, 5th edn. W3C recommendation, W3C (2008)
5. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.C.: Keys for XML. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001, pp. 201–210. ACM, New York (2001)
6. Dantzig, G.: Linear Programming and Extensions. Landmarks in Physics and Mathematics. Princeton University Press (1998)
7. Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Books. MIT Press (2004)
8. Fajt, S.: Mining XML Integrity Constraints. Master's thesis, Charles University in Prague (2010), http://www.ksi.mff.cuni.cz/projects/infer/keyminer/Fajt.pdf
9. Fomin, F.V., Grandoni, F., Kratsch, D.: A Measure & Conquer Approach for the Analysis of Exact Algorithms. J. ACM 56, 25:1–25:32 (2009)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
11. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. In: Artificial Intelligence. Addison-Wesley Pub. Co. (1989)
12. Klempa, M., Mikula, M., Smetana, R., Švirec, M., Vitásek, M.: jInfer Architecture (2011), http://jinfer.sourceforge.net/modules/architecture.pdf
13. Klempa, M., Mikula, M., Smetana, R., Švirec, M., Vitásek, M.: jInfer: Java Framework for XML Schema Inference (2011), http://jinfer.sourceforge.net
14. Paschos, V.T.: A survey of Approximately Optimal Solutions to Some Covering and Packing Problems. ACM Comput. Surv. 29, 171–209 (1997)
15. Robson, J.: Algorithms for Maximum Independent Sets. Journal of Algorithms 7(3), 425–440 (1986)
16. Vitásek, M.: Inference of XML Integrity Constraints. Master's thesis, Charles University in Prague (2012), http://www.ksi.mff.cuni.cz/~mlynkova/dp/Vitasek.pdf
17. Vošta, O., Mlýnková, I., Pokorný, J.: Even an Ant Can Create an XSD. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 35–50. Springer, Heidelberg (2008)

Optimizing the Resource Allocation for Approximate Query Processing[*]

Anna Yarygina and Boris Novikov

Abstract. Query optimization techniques are a proven tool essential for high performance of the database management systems. However, in a context of data spaces or new querying paradigms, such as similarity based search, exact query evaluation is neither computationally feasible nor meaningful and approximate query evaluation is the only reasonable option. In this paper a problem of resource allocation for approximate evaluation of complex queries is considered and an approximate algorithm for an optimal resource allocation is presented, providing the best feasible quality of the output result subject to a limited total cost of a query.

1 Introduction

Query optimization techniques are considered as a cornerstone for high performance of the database management systems. Modern optimizers provide for efficient evaluation of queries expressed in high-level declarative languages, typically extensions of relational algebra and calculus.

However, in broader contexts of heterogeneous federations of autonomous information resources and in presence of alternative querying paradigms, such as probabilistic and similarity-based queries, the traditional exact queries are neither computationally feasible nor semantically meaningful. Hence an approximate query evaluation is the only option.

The query evaluation is approximate if the output is, in certain sense, incomplete or imprecise. Obviously, approximate evaluation makes sense only if it requires less

Anna Yarygina · Boris Novikov
Saint Petersburg University
e-mail: anya_safonova@mail.ru, b.novikov@spbu.ru

[*] This research is supported by HP Labs and Russian Foundation for Basic Research, grant 10-07-00156.

resource (e.g. time) than exact execution. Informally, the quality of the result is expected to be better if more resources are spent on the query evaluation.

In the context of approximate evaluation the optimization techniques should be re-considered. The problem of traditional query optimization, namely "find an execution plan with the minimal cost" is replaced with either "find the best plan yielding at least specified quality", or "provide the best possible quality for at most given amount of resources". The remaining part of this research considers the latter problem only.

More specifically, we assume that the query is formulated as an algebraic expression in terms of certain set of operations. Typically these operations constitute a variation or extension of the relational algebra; however, this is not essential for the algorithms presented in this research. For each operation the relative quality of its output depends on the amount of resources allocated to the operation. Usually the critical resource is time. Further we use terms resource, time, and cost interchangeably.

In this research a problem of resource allocation for approximate evaluation of complex queries is considered and an algorithm for an optimal resource allocation is presented, providing the best feasible quality of the output result subject to a limited total cost of a query.

The optimization problem for approximate query evaluation seems to be much harder than commonly known exact query optimization: in addition to selection of one of equivalent execution plans, the optimizer has to distribute available processing resources between operations of the query execution plan.

An approximate solution is to find the best execution plan for unlimited resource yielding the best possible quality and then allocate the limited resource for the selected plan.

Thus, the contribution of this research is an algorithm for near-optimal allocation of limited resources providing the best possible quality of the output.

We start with the definition of extended abstract cost model providing trade-offs between quality and cost for all operations and proceed with the formal statement of the resource allocation problem. We then provide a solution for the problem for some special cases and proceed with an algorithm for a general case.

2 Abstract Model

2.1 Query Execution Plans

The query processor operates with a query evaluation plan consisting of unary and binary operations and represented as a binary tree. In this query tree vertices are operations and edges connect operations with their arguments. Let P be an execution plan, that is, a set of operations $op(P)$ organized as a tree.

In this research we assume that data may be imprecise or uncertain and the operations may yield incomplete or imprecise output. Although the concept of data quality is complex, we assume that the quality of a data set is estimated with a single numeric value and the quality of different data sets is expressed in comparable way. Further, (most of) operations in our model are approximate and may produce different quality of the output depending on the quality of input and amount of allocated resources. We define the relative quality as the ratio of the achieved quality to the best one for given arguments. Subsequent paragraphs describe this more formally. The expected behavior of each operation from the query tree ($op \in op(P)$) depends on its relative quality ($q(op)$).

The quality of each subquery ($Q(op)$) in the execution plan rooted in $op \in op(P)$ depends on the quality of its child subqueries and properties of the root operation. Say, for a binary operation it depends on 3 parameters: the quality of the left subquery ($Q(l)$), the quality of the right subquery ($Q(r)$), and the quality which the root operation produces ($q(op)$).

For binary operations we assume that the impact of the worst argument (in terms of quality) dominates. More formally, we assume the following dependence $Q(op) = min(Q(l), Q(r))q(op)$.

For each operation the relative quality of its output depends on the amount of resources allocated to it. Therefore, the behavior of each operation from the query tree ($op \in op(P)$) can be expressed with the relative quality function $q(op) : Time \to Quality$.

The relative quality of the query ($Q(root)$) represents the quality of the whole query, where $root \in op(P)$ is a root operation of the query.

2.2 Problem Statement

For any given total resource specified, find the amount of resources $t_{op} \in Time$ for any operation $op \in op(P)$ such that the estimated quality of the final query result is maximized. The set $t_P = \{t_p, p \in op(P)\}$ is called a distribution of resources. The amount of resources allocated for subquery rooted in $op \in op(P)$ is denoted as $T_{op} \in Time$.

Let us state the problem in exact terms. We have the query tree organized as described above and fixed amount of time (T). The resource allocation process can be represented by an algorithm which for any operation $op \in op(P)$ allocates $t_{op} \in Time$ resource such that $\sum_{op \in op(P)} t_{op} \leq T$ and $Q(root)$ is maximized.

2.3 Assumptions

We assume that the following conditions hold.

For each operation $op \in op(P)$ a minimal amount of resources needed to complete the operation is known $t_{min}(op) \in Time$. This amount yields a certain level of the output quality. For each operation $op \in op(P)$ an amount of time $t_{max}(op) \in Time$ is known, such that any additional resource does not improve the quality.

For any amount of resources allocated between the minimal and maximal amount, the quality of the output is a non-decreasing function of the amount allocated $(\forall op \in op(P), t_1, t_2 \in Time : t_1 < t_2 \; q(op)(t_1) \leq q(op)(t_2))$.

In this research we assume that each operation quality function can be approximated by a continuous piecewise linear function. That is we assume that

$$\forall op \in op(P) \; q(op)(t) = q^i(op) + s^i(op)(t - t^i_{min})$$

where $t \in [t^i_{min}, t^i_{max}]$ and $q^i(op) + s^i(op)(t^i_{max} - t^i_{min}) = q^{i+1}(op)$.

Further we will work with resource increments and consider at each moment of time only one linear segment: $\forall op \in op(P) \; q(op)(\Delta t) = q(op) + s(op)\Delta t$, where $t \in [t^i_{min}, \; t^i_{max}], \; t_{max} = t^i_{max} - t, \; \Delta t \in [0, t_{max}]$, and $q(op) = q(op)(t) = q^i(op) + s^i(op)(t - t^i_{min})$.

2.4 Critical Subquery

As soon as minimal needed resource is allocated for each operation in a query, additional resource may be allocated to some operations to improve the quality of the final result. Obviously, the resource should be allocated only to operations which have an impact on the output quality. These operations constitute a critical sub-query C, which is formally defined as follows.

The critical subquery is a set of nodes in the query tree ($op(C) \subseteq op(P)$) which can be constructed recursively:

- $root \in op(C)$;
- if c is an only child of $op \in op(C)$ then $c \in op(C)$;
- if $Q(l) < Q(r)$ where l, r are children of $op \in op(C)$ then $l \in op(C)$;
- if $Q(l) > Q(r)$ where l, r are children of $op \in op(C)$ then $r \in op(C)$;
- if $Q(l) = Q(r)$ where l, r are children of $op \in op(C)$ then $l, r \in op(C)$.

Lemma 1. *For $op \in op(P) \setminus op(C)$ $t_{op} = 0$ according to the optimal resource allocation process.*

In order to meet page limits we omit all proofs in this version of the paper.

It is important to mention that the quality of the whole query is equal to the quality of its critical subquery.

2.5 Resource Allocation along Paths and between Siblings

In case when the critical subquery is a path the quality function of the query is a product of the linear quality functions of its operations. Now we face the problem of resource allocation along this path and have to solve corresponding optimization problem.

Lemma 2. *Let C be a critical path, T the amount of resources for allocation, for each $op \in op(C)$ the quality function is linear, that is $q(op)(t) = q(op) + s(op)t$, where t is within the limits for the quality function. According to the optimal resource allocation process $\exists op(C^+) \subseteq op(C) : t_p > 0$ iff $p \in op(C^+)$; and*

$$t_{op} = \max\left(\frac{T}{n} + \frac{1}{n}\sum_{p\in op(C^+)} \frac{q(p)}{s(p)} - \frac{q(op)}{s(op)}, 0\right),$$

where $op \in op(C)$, and $n = |op(C^+)|$.

The following lemma 3 suggests how to split resource between critical siblings.

Lemma 3. *Let C be a critical subtree; $l, r \in op(C)$ are two root nodes of sibling subtrees with linear quality functions, that is $Q(l)(t) = Q(l) + S(l)t$ and $Q(r)(t) = Q(r) + S(r)t$, where t is in the range for the quality function; $T \in Time$ is the amount of resources for allocation between this siblings. According to the optimal resource allocation process $T_l = \frac{S(r)}{S(l)+S(r)}T$ and $T_r = \frac{S(l)}{S(l)+S(r)}T$.*

3 Resources Allocation Algorithm

3.1 The Algorithm and Data Structures

This section describes an approximate algorithm for optimal resource allocation on a fixed query execution plan. To obtain an optimal plan for query evaluation under constrained resource, we first build a plan with conventional optimization techniques which do not account for amount of resources, and then apply the resource allocation algorithm to this plan. Of course, this plan is not necessarily optimal globally. However, our approach provides locally optimal plan for any fixed tree.

According to lemmas in section 2 we are able to efficiently distribute resource among operations for two types of the plan structure, naively, for plans consisting of a single path from root to leaf (vertical path) or between leaf siblings of a single parent node.

In order to exploit these lemmas, we build virtual hypernodes from certain parts of the query execution plan and then apply an appropriate lemma inside a hypernode.

A node is called a splitting point if it has two child nodes. The virtual hypernodes are constructed for the critical tree according to the following rules:

- If the critical plan contains sibling leaf (hyper-)nodes, these siblings a retracted into a hypernode, which becomes a single child of the parent node.
- Any path from a leaf to a splitting point is retracted into a vertical hypernode, which becomes a leaf child of the splitting point.

Obviously, at least one of two steps listed above is applicable if a plan contains at least two nodes. The process stops when the whole plan is retracted into a single hypernode which is always vertical as a root cannot have siblings.

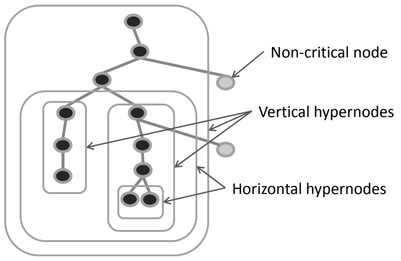


Fig. 1 An example of hypernodes

An example of hypernode structure is shown on figure 1.

The quality functions for original tree nodes are provided from cost models for corresponding operations. Quality functions for hypernodes are constructed when the hypernodes are created.

The lemmas require that the quality of each operation (node) be a linear function of the allocated resource. Unfortunately, the quality of a hypernode cannot be always expressed as a linear function and it is replaced with a liner approximation, making our algorithm approximate (except for few special cases). The quality functions for horizontal (sibling) hypernodes are linear by lemma 3. Functions for vertical nodes are not linear in general, and are replaced with linear approximations, valid on a certain range of the argument values.

The range for a hypernode is constructed in such a way that the corresponding amount of resources can always be distributed to the sub-nodes of this hypernode without violation of the constraints inside the hypernode.

The overall resource allocation control flow is displayed in the following pseudo code:

```
Input: Query tree $P$, resource for allocation $T$.
Output: For all operations $op \in op(P)$ allocated amount of resources $t_{op}$.
  Initialization($P,T$)
  while $T > 0$ & !isMaxQualityReached($P$) do
      QualityEstimation($P$)
      $C$=CriticalSubtreeConstruction($P$)
      $H$=HypergraphConstruction($C$)
      ResourceAllocation($H,T$)
  end while
```

The algorithm can now be outlined as follows.

During the initialization the resource equal to minimal needed is allocated for each operation in P and the quality functions are adjusted to accept increments. If the sum of minimal resources exceeds T, the algorithm fails as the execution of the plan is not possible with available resource.

After initial allocation of minimally required resources the remaining amount of resources is distributed in incremental amounts. The increments are subject to the following constraints:

- boundaries of linear segments are not exceeded for any node, and
- the quality of a critical sub-query should not exceed the quality of its non-critical siblings.

The incremental allocation is repeated until either all available resource will be exhausted or a maximal possible quality for the query is reached.

For each incremental step, the critical sub-tree and hypernode structure are recalculated and then allocation proceeds recursively into all hypernodes, using appropriate lemma for allocation inside each hypernode.

It is easy to prove that our algorithm has polinomial complexity on the number of nodes in the query evaluation plan.

3.2 Horizontal Hypernodes

According to lemma 3, the amount of resources allocated to the horizontal hypernode is distributed between siblings so that the increase of quality function is same for all siblings in the hypernode. Let H be a hypernode with operations l, r and s_l, s_r be the slopes for the quality functions. Let $S = s_l + s_r$ for brevity. If the total amount of resources allocated to H is T, then $t_l = \frac{s_r T}{S}$, $t_r = \frac{s_l T}{S}$.

The construction of the quality function for a horizontal hypernode is straightforward: the initial quality is the same as that of siblings, and the slope is calculated from the slopes of siblings and is equal to $\frac{s_l s_r}{S}$. Let t^l_{max}, t^r_{max} be the right boundaries for the linear segments of the quality functions of siblings, then the right boundary for linear segment of the hypernode is $T_{max} = \min(\frac{t^r_{max} S}{s_l}, \frac{t^l_{max} S}{s_r})$.

3.3 Vertical Hypernodes

Handling of vertical nodes is more complex.

Let $V = \{p\}$ be a vertical hypernode with nodes p as its elements, and let $r(p) = q(p)/s(p)$, where q and s are defined as in lemma 2. Also let $S = \sum_{p \in V} r(p)$. According to this lemma, an optimal allocation of resources T is reached when

$$t_p = \begin{cases} (T+S)/n - r(p), & \text{if } (T+S)/n - r(p) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where S and n are the sum and count, respectively, of those $r(p)$ for which the $(T+S)/n - r(p) > 0$. Obviously, this expression will be positive for minimal values of $r(p)$ as S/n is the arithmetic mean of $r(p)$.

To construct the quality function for a hypernode, the algorithm selects the subset of sub-nodes with minimal value of $r(p)$. The resource will be allocated to these nodes only. The allocated amounts cannot exceed the ranges for quality functions of respective nodes. In addition, the increased value of quality functions should not exceed that of the node with second minimal $r(p)$ and the quality of non-critical siblings also should not be exceeded.

The exact quality function for V is a product $\prod_{p \in V}(q(p)+s(p)t_p)$. This function is replaced with a linear tangent approximation. The range is calculated from the constraints. As soon as the amount of resources allocated to a hypernode is calculated, it is divided between selected sub-nodes evenly and this completes the incremental step for a hypernode.

Our algorithm is approximate because linear approximations are used instead of precise quality functions. However the smaller time increments are allocated at each iteration the smaller error we obtain and more iterations are spent for resource allocation. This fact enables us to balance cost with quality of the output.

4 Experiments

To analyze the behavior of the proposed approximate resource allocation algorithm we have performed the set of experiments. All experiments are based on Java implementation, running on i5 2.67 GHz CPU, 8 Gb RAM, 64 bit operating system.

We evaluated both the efficiency and the effectiveness of the developed technique. The performance is measured in terms of the amount of time needed for resource allocation and the number of iterations in the main loop of the algorithm.

To analyze the accuracy of the algorithm, a value of the optimal relative quality for a plan is needed.

The error of our algorithm is caused by use of linear approximations instead of precise quality functions. Obviously, this error becomes negligible for sufficiently small increments. To obtain an estimation of the optimal resource allocation, the algorithm was executed with additional restriction on the size of the increment at each iteration. The results were used as a substitute for the optimal solution.

All experiments were run on synthetic data. A number of random query trees were generated with N nodes, where $N \in \{5, 10, 15, 20, 25\}$. Quality functions for each operation in the query were generated with the number of linear segments varying from 1 to l, with $l \in \{1, 2, 3, 4, 5\}$. Minimum and maximum amounts of resources for each operation were also chosen randomly from the range [0, 100]. The topology of the query tree was defined randomly and then the number of splits was calculated in order to estimate how this parameter affects the approximate algorithm.

The most interesting findings from our algorithms are described below.

Figure 2(a) shows how the number of iterations needed to allocate all available resource depends on the query tree: number of operations and number of linear segments in operation quality functions. For each pair (N, l) 300 trees were constructed for the experiment. For each tree we allocated by the approximate algorithm $T = T_{min} + p(T_{max} - T_{min})$ resource, where $p \in \{0, 0.25, 0.5, 0.75, 1\}$. Similar results with absolute time measurements are presented on figure 2(b). One may see that the average number of iterations and average absolute time needed for resource

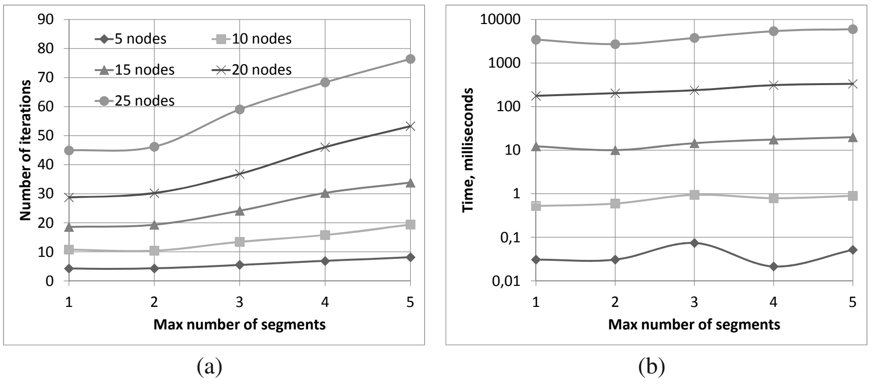


Fig. 2 Average number of iterations and absolute time for various queries

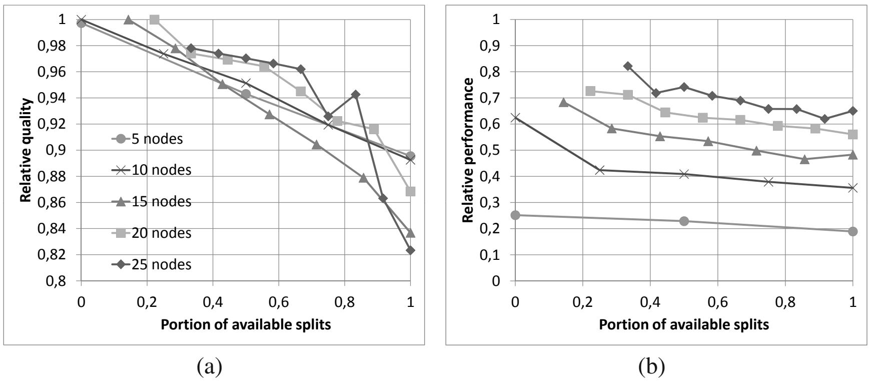


Fig. 3 Cost-quality trade-off

allocation goes up with increase of number of operations in the query and number of linear segments in each node.

The results of experiments on accuracy of the proposed approximate resource allocation algorithm are presented on figure 3. For each pair (N,l) 100 trees were randomly constructed for the experiment and for each tree 10 quality functions were generated in order to collect enough number of examples for trees with rare topologies. We allocated $T = T_{min} + 0.5 * (T_{max} - T_{min})$ resource. On the Y-axis figure 3(a) shows the relative quality produced by the proposed approximate algorithm compared to one with artificially restricted resource increments allocated on each iteration to be equal 5% of maximum amount of resources needed for the query. One may see the relative quality decreases when the number of splits grows, that is, a tree becomes bushy. In this case trees contain enough paths to use tangent approximation of the sub-query quality functions and enough splits to use them in the resource allocation process.

Figure 3(b) shows the ratio of the number of iterations used in approximate algorithm and in the nearly exact one. The number of iterations in the approximate algorithm decreases as the portion of available splits grows. The reason is that horizontal hypernodes accept larger increments of resources compared to the vertical ones.

The results of experiments have shown that approximate resource allocation algorithm is quite precise and efficient.

5 Related Work

The query optimization became both required and enabled since the advent of high-level declarative query languages, mostly in the context of the relational database model. A brief overview of classical query optimization techniques can be found in [9]. The optimization techniques for distributed systems are summarized in [8] An optimizer for distributed heterogeneous systems is proposed in [11]. The query optimization techniques based on hypergraphs are analyzed in [12].

The algorithm proposed in this research uses the output of traditional a optimizer as its starting point.

The approximate query evaluation techniques were considered in the context of very large data warehouses and mobile networks [1, 3, 2, 6]. The approximation is typically based on sampling, wavelets, or synopsis. However, in both cases the query optimization was not extensively studied. For data warehouses, the dominating cost is produced by accesses to a single huge table, hence the performance depends mostly on the performance of a single operation, namely, data extraction from this table. For mobile networks the queries are typically very simple and the optimization is not an issue. Note that the critical resource might depend on the context: in contrast with large database and data warehouses, where time is the most important, the energy might be more valuable for mobile or sensor devices. Our approach does not depend on the nature of the resources to be allocated and might be applicable in both contexts.

Handling of time constraints on complex SQL queries is proposed in [5]. The authors distinguish approximate (based on sampling) and partial (top k) query evaluation. The proposed approach is based on extendibility of an optimizer in the commercial DBMS and does not consider resource allocation problem.

The quality and performance trade-offs for stream processing are discussed in [14, 7].

Optimal resource allocation was studied in the context of distributed query evaluation and load balancing [15, 13, 4]. Although our algorithm resembles load balancing, it is not specifically related to a distributed system and allocates resource to operations rather than to processing units.

The approximate query evaluation is a must in contexts such as information retrieval, however, in these contexts the primary objective is the quality of result (usually measured as precision and recall, while resource are considered less important.

Further, complex queries and hence query optimization are not common in these contexts.

The extensive research on data and information quality is summarized in [10]. Although the nature of data quality is essential, in our research we rely on a quantitative estimation of quality expressed as a single number and abstract from any specific aspects of data quality. Further, our goal is to estimate relative quality, that is, how the operations and the whole query affects and depends on the quality of data sources and final output, rather than the absolute quality.

6 Conclusion

The high-level declarative querying facilities both require and enable sophisticated query optimization. In the context of data spaces and/or heterogeneous information resources an approximate query evaluation turns out to be a dominating paradigm.

To address the need in controlled trade-offs between quality and performance, the optimization techniques should be augmented with additional capabilities.

We presented an approximate polynomial algorithm for optimal resource allocation providing best possible quality of the result subject to the constrained total resource for a query and experimentally evaluated its performance.

References

1. Babcock, B., Chaudhuri, S., Das, G.: Dynamic sample selection for approximate query processing. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD 2003, pp. 539–550. ACM, New York (2003), doi: http://doi.acm.org/10.1145/872757.872822
2. Chaudhuri, S., Das, G., Narasayya, V.: Optimized stratified sampling for approximate query processing. ACM Trans. Database Syst. 32 (2007), doi: http://doi.acm.org/10.1145/1242524.1242526
3. Dell'Aquila, C., Di Tria, F., Lefons, E., Tangorra, F.: Accuracy estimation in approximate query processing. In: Proceedings of the 14th WSEAS International Conference on Computers: Part of the 14th WSEAS CSCC Multiconference, ICCOMP 2010, vol. II, pp. 452–458. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2010), http://dl.acm.org/citation.cfm?id=1984366.1984374
4. Epimakhov, I., Hameurlain, A., Dillon, T., Morvan, F.: Resource Scheduling Methods for Query Optimization in Data Grid Systems. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 185–199. Springer, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2041746.2041765
5. Hu, Y., Sundara, S., Srinivasan, J.: Supporting time-constrained sql queries in oracle. In: Proceedings of the 33rd International Conference on Very large Data Bases, VLDB 2007, pp. 1207–1218. VLDB Endowment (2007), http://dl.acm.org/citation.cfm?id=1325851.1325989
6. Jermaine, C., Arumugam, S., Pol, A., Dobra, A.: Scalable approximate query processing with the dbo engine. ACM Trans. Database Syst. 33, 23:1–23:54 (2008), doi: http://doi.acm.org/10.1145/1412331.1412335

7. Jiang, Q.: A framework for supporting quality of service requirements in a data stream management system. Ph.D. thesis, Arlington, TX, USA (2005) AAI3181900
8. Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. 32(4), 422–469 (2000), doi: http://doi.acm.org/10.1145/371578.371598
9. Kossmann, D., Stocker, K.: Iterative dynamic programming: a new class of query optimization algorithms. ACM Trans. Database Syst. 25(1), 43–82 (2000), doi: http://doi.acm.org/10.1145/352958.352982
10. Madnick, S.E., Wang, R.Y., Lee, Y.W., Zhu, H.: Overview and framework for data and information quality research. J. Data and Information Quality 1(1), 2:1–2:22 (2009), doi: http://doi.acm.org/10.1145/1515693.1516680
11. Pentaris, F., Ioannidis, Y.: Query optimization in distributed networks of autonomous database systems. ACM Trans. Database Syst. 31(2), 537–583 (2006), doi: http://doi.acm.org/10.1145/1138394.1138397
12. Scarcello, F., Greco, G., Leone, N.: Weighted hypertree decompositions and optimal query plans. J. Comput. Syst. Sci. 73(3), 475–506 (2007), doi: http://dx.doi.org/10.1016/j.jcss.2006.10.010
13. Yang, R., Bhulai, S., van der Mei, R., Seinstra, F.: Optimal resource allocation for time-reservation systems. Perform. Eval. 68, 414–428 (2011), doi: http://dx.doi.org/10.1016/j.peva.2011.01.003
14. Zhang, R., Koudas, N., Ooi, B.C., Srivastava, D., Zhou, P.: Streaming multiple aggregations using phantoms. The VLDB Journal 19, 557–583 (2010), doi: http://dx.doi.org/10.1007/s00778-010-0180-z
15. Zhao, H.C., Xia, C.H., Liu, Z., Towsley, D.: A unified modeling framework for distributed resource allocation of general fork and join processing networks. In: Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2010, pp. 299–310. ACM, New York (2010), doi:http://doi.acm.org/10.1145/1811039.1811073

Author Index

Amous, Ikram 207
Andruszkiewicz, Piotr 1
Ayad, Sarah 45

Barros, Samarony 217
Bazydło, Marcin 9
Bellatreche, Ladjel 149
Benczúr, András 253
Berkani, Nabila 149
Bleja, Michał 57
Boinski, Pawel 19
Brdjanin, Drazen 31
Bulánek, Jan 69
Burzańska, Marta 265

Cherfi Si-Said, Samira 45
Chirkova, Rada 115
Chrószcz, Aleksander 91
Comyn-Wattiau, Isabelle 45

Drozd, Marcin 57

Falt, Zbyněk 69
Fathi, Yahya 115
Forenc, Kamil 195
Francuzik, Szymon 9
Froelich, Wojciech 79

Gorawski, Marcin 91
Grunow, Sebastian 103

Heuser, Carlos Alberto 125
Huang, Rong 115

Kantorski, Gustavo Zanini 125
Keneshloo, Yaser 137
Khouri, Selma 149

Leszczyński, Konrad 161
Leszczyński, Paweł 173

Maric, Slavko 31
Matthes, Florian 103
Mikhaylova, Elena 185
Mlýnková, Irena 285
Moraes, Tiago Guimaraes 125
Moreira, Viviane Pereira 125
Morzy, Mikołaj 195

Novikov, Boris 185, 297

Pąk, Karol 273
Pokrzywa, Rafal 241

Quesado, Jefferson 217

Roth, Sascha 103
Rybiński, Henryk 273

Sampaio, Marcus 217
Sivogolovko, Elena 229
Skowron, Adam 241
Sobaniec, Cezary 9
Stencel, Krzysztof 57, 173, 265
Subieta, Kazimierz 57
Szabó, Gyula I. 253
Szumowska, Aneta 265

Thalheim, Bernhard 229
Tomaszuk, Dominik 273

Vitásek, Matej 285
Volokhov, Anton 185

Wawrzyniak, Dariusz 9
Wiśniewski, Piotr 265

Yaghob, Jakub 69
Yarygina, Anna 297
Yazdani, Sasan 137

Zakrzewicz, Maciej 19, 161
Zayani, Corinne Amel 207
Zghal Rebai, Rim 207