Advances in Intelligent Systems and Computing 312

Nick Bassiliades · Mirjana Ivanovic
Margita Kon-Popovska · Yannis Manolopoulos
Themis Palpanas · Goce Trajcevski
Athena Vakali *Editors*

# New Trends in Database and Information Systems II

Selected Papers of the 18th East European Conference on Advances in Databases and Information Systems and Associated Satellite Events, ADBIS 2014 Ohrid, Macedonia, September 7–10, 2014, Proceedings II

Springer

**Advances in Intelligent Systems and Computing**

Volume 312

**Series editor**

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

Nick Bassiliades · Mirjana Ivanovic
Margita Kon-Popovska · Yannis Manolopoulos
Themis Palpanas · Goce Trajcevski
Athena Vakali
Editors

# New Trends in Database and Information Systems II

Selected Papers of the 18th East European Conference on Advances in Databases and Information Systems and Associated Satellite Events, ADBIS 2014 Ohrid, Macedonia, September 7-10, 2014
Proceedings II

Springer

*Editors*
Nick Bassiliades
Aristotle University of Thessaloniki
Thessaloniki
Greece

Mirjana Ivanovic
Department of Mathematics and Informatics
Faculty of Sciences
University of Novi Sad
Novi Sad
Serbia

Margita Kon-Popovska
Faculty of Natural Science and Mathematics
Institute for Informatics
Ss Cyril and Methodious University
Skopje
Macedonia

Yannis Manolopoulos
Department of Informatics
Aristotle Univ of Thessaloniki
University Campus
Thessaloniki
Greece

Themis Palpanas
Paris Descartes University
Paris
France

Goce Trajcevski
EECS Department
Northwestern University
Evanston Illinois
USA

Athena Vakali
Department of Informatics
Aristotle University of Thessalonik
Thessaloniki
Greece

ISSN 2194-5357 ISSN 2194-5365 (electronic)
ISBN 978-3-319-10517-8 ISBN 978-3-319-10518-5 (eBook)
DOI 10.1007/978-3-319-10518-5

Library of Congress Control Number: 2014947874

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

This volume contains a selection of papers presented at the 18th East-European Conference on Advances in Databases and Information Systems (ADBIS 2014), held on September 7–10, 2014, in Ohrid, Republic of Macedonia.

Database and information systems technologies have been rapidly evolving in several directions in the recent years. New types of data, new kinds of emerging applications and information systems to support them, raise diverse challenges to be addressed. The so-called big data challenge, streaming data management and processing, social networks and other complex data analysis, including semantic reasoning into information systems supporting for instance trading, negotiations, and bidding mechanisms are but a few examples of such emerging research areas.

The ADBIS series of conferences aims to provide a forum for the presentation and dissemination of research on database theory, development of advanced DBMS technologies, and their advanced applications. ADBIS 2014 continued the ADBIS series held every year in different countries of Europe, beginning in St. Petersburg (1997), Poznan (1998), Maribor (1999), Prague (2000), Vilnius (2001), Bratislava (2002), Dresden (2003), Budapest (2004), Tallinn (2005), Thessaloniki (2006), Varna (2007), Pori (2008), Riga (2009), Novi Sad (2010), Vienna (2011), Poznan (2012) and Genoa (2013). The conferences are initiated and supervised by an international Steering Committee consisting of representatives from Armenia, Austria, Bulgaria, Czech Republic, Estonia, Finland, Germany, Greece, Hungary, Israel, Italy, Latvia, Lithuania, Poland, Russia, Serbia, Slovakia, Slovenia and Ukraine.

The Programme of ADBIS 2014 includes keynotes, research papers, a tutorial session entitled “Online Social networks Analytics - Communities and Sentiment Detection" by Athena Vakali, a Doctoral Consortium, and thematic workshops. The general idea behind each satellite event was to collect specialized sub-domains of the broad research areas of databases and information systems, providing forums that will enable more focused discussions encapsulating new trends in these two important areas.

This volume contains 26 papers (15 papers as short contributions, 3 papers from the Doctoral Symposium, and 8 papers from the 3 workshops) selected as contributions presented at the 18th East-European Conference on Advances in Databases and

Information Systems ADBIS 2014. 15 short papers were selected out of 82 submitted to the conference from 34 different countries representing all the continents with 210 authors. In a rigorous reviewing process done by the international Programme Committee consisting of 115 reviewers from 34 countries and 27 additional reviewers supporting workload, 26 papers were selected as full contributions for inclusion in LNCS proceeding and 16 papers as short contributions of which 15 are included in this volume. All papers were evaluated by at least three reviewers.

Each of the satellite events complementing the main ADBIS conference had its own international program committee, whose members served as the reviewers of papers included in this volume. The volume is divided into five parts, one devoted to ADBIS 2014 short contributions and each other to a single satellite event. The short papers from this volume consider a wide variety of data, ranging from classical relational, row and column store layouts, graph structure, spatio-temporal, unstructured, XML to workflows and streams of (real time) data, from the points of view of specialized architectures, physical structure and indexing, cleansing, query processing and benchmarking.

ADBIS 2014 aimed to create conditions for experienced researchers to share their knowledge and experience with the young researchers participating in the Doctoral Consortium. The ADBIS Doctoral Consortium is a forum for Ph.D. students to present their research ideas, confront them with the scientific community, receive feedback from senior mentors, socialize and tie cooperation bounds. This year 6 papers have been submitted to the Doctoral Consortium and 3 papers have been selected for presentation and are included in this volume. They cover three different topics, all quite relevant for emerging applications in the database and information system field: querying and managing complex data, generalized relational algebraic operations, data warehouse schema evolution.

The following 3 workshops associated with the ADBIS conference were organized:

- 3rd Workshop on GPUs in Databases (GID), organized by Witold Andrzejewski (Poznan University of Technology), Krzysztof Kaczmarski (Warsaw University of Technology) and Tobias Lauer (Jedox).
- 3rd Workshop on Ontologies Meet Advanced Information Systems (OAIS) organized by Ladjel Bellatreche (LIAS/ENSMA, Poitiers) and Yamine Aït Ameur (IRIT/ENSEIHT, Toulouse), and
- 1st Workshop on Technologies for Quality Management in Challenging Applications (TQMCA) organized by Isabelle Comyn-Wattiau (CNAM, Paris), Ajantha Dahanayake (Prince Sultan University, Saudi Arabia), and Bernhard Thalheim (Christian Albrechts University).

Each workshop had its own international Program Committee.

The 3rd International Workshop on GPUs in Databases (GID 2014) is devoted to all subjects related to utilization of Graphics Processing Units in database environments. The concept of using GPUs in databases is relatively young and has not yet received enough attention from the database community. The intention of the GID workshop is to provide a discussion forum for industry and science, with emphasis on popularizing the GPUs and providing a forum for discussion with respect to the GID's research ideas and their potential to achieve high speedups in many applications domains of databases.

Presentation of practical and theoretical research creates chances for fruitful cooperation between the two communities. Three papers have been selected for presentation at GID 2014 and are included in this volume.

The 3rd International Workshop on Ontologies Meet Advanced Information Systems (OAIS 2014) has a twofold objective to present: new and challenging issues in the contribution of ontologies for designing high quality information systems, and new research and technological developments which use ontologies all over the life cycle of information systems. The Workshop addresses scientists, engineers, educators, industry, policy makers, decision makers, and others to share their insight, vision, and understanding of the ontologies challenges in Advanced Information Systems. Three papers have been selected for presentation at OAIS 2014 and are included in this volume.

The 1st International Workshop on Technologies for Quality Management in Challenging Applications (TQMCA 2014) focuses on quality management and its importance in new fields such as big data, crowd-sourcing, and stream databases. The Workshop has addressed the need to develop novel approaches and technologies, and to entirely integrate quality management into information system management. TQMCA builds on the fact that the technologies for quality management offer a growing research domain specifically within large scale complex applications and their companion database management system development as well as in the information system development and in general in system development disciplines. Four papers have been selected for presentation at TQMCA 2014 and are included in this volume.

Conference is supported by the President of the Republic of Macedonia, H.E. Dr. Gjorge Ivanov. We would like to express our gratitude to every individual who contributed to the success of ADBIS 2014. Firstly, we thank the authors who submitted papers to the conference – the geographical statistic indicates that there were authors from 34 different countries, including submissions from non-European countries (Algeria, Lebanon, Tunisia, Australia, Brazil, US, China, Japan, Singapore, Vietnam). However, we are also indebted to the members of the community who offered their time and expertise in performing various roles, ranging from organizational to reviewing ones – the efforts, energy and degree of professionalism deserve highest commendations. Special thanks go to the Program Committee members as well as to the external reviewers for their support in evaluating the papers submitted to ADBIS 2014, ensuring the quality of the scientific program. Thanks also to all the colleagues involved in the conference organization, as well as the workshop organizers. A special thank you is due for the members of the Steering Committee and, in particular, its Chair, Leonid Kalinichenko, for all their help and guidance. Finally, we thank Springer for publishing the proceedings containing research and satellite events papers in AISC series. The Program Committee work relied on EasyChair, and we thank its development team for creating and maintaining it – it offered great support throughout the different phases of the reviewing process. The conference would not have been possible without our supporters and sponsors: Ministry of Information Society and Administration, University Ss Cyril and Methodius in Skopje, Faculty of Computer Sciences and Engineering, ICT-ACT Association, Municipality Ohrid.

Last, but not least, we thank all the participants of ADBIS 2014 for having sharing their works, providing a lively, fruitful and constructive forum, and giving us the pleasure of knowing that our work was purposeful.

September 2014

Nick Bassiliades
Mirjana Ivanovic
Margita Kon-Popovska
Yannis Manolopoulos
Goce Trajcevski
Themis Palpanas
Athena Vakali (Eds.)

# Organization

## General Chair

| | |
|---|---|
| Margita Kon-Popovska | Ss Cyril and Methodious University in Skopje |

## Program Committee Co-chairs

| | |
|---|---|
| Yannis Manolopoulos | Aristotle University of Thessaloniki |
| Goce Trajcevski | Northwestern University, IL |

## Workshop Co-chairs

| | |
|---|---|
| Themis Palpanas | Paris Descartes University |
| Athena Vakali | Aristotle University of Thessaloniki |

## PhD Consortium Co-chairs

| | |
|---|---|
| Nick Bassiliades | Aristotle University of Thessaloniki |
| Mirjana Ivanovic | University of Novi Sad |

## Publicity Chair

| | |
|---|---|
| Goran Velinov | Ss Cyril and Methodious University in Skopje |

## Website Chair

| | |
|---|---|
| Vangel Ajanovski | Ss Cyril and Methodious University in Skopje |

## Proceedings Technical Editor

| | |
|---|---|
| Ioannis Karydis | Department of Informatics, Ionian University Corfu |

## Local Organizing Committee Chair

| | |
|---|---|
| Goran Velinov | Ss Cyril and Methodious University in Skopje |

## Local Organizing Committee

| | |
|---|---|
| Anastas Mishev | Ss Cyril and Methodious University in Skopje |
| Boro Jakimovski | Ss Cyril and Methodious University in Skopje |
| Ivan Chorbev | Ss Cyril and Methodious University in Skopje |

## Supporters

Ministry of Information Society and Administration
University Ss Cyril and Methodius in Skopje
Faculty of Computer Sciences and Engineering,
ICT-ACT Association
Municipality Ohrid

# Steering Committee

## Program Committee

| | |
|---|---|
| Marko Bajec | University of Ljubljana, Slovenia |
| Mirta Baranovic | University of Zagreb, Croatia |
| Guntis Barzdins | University of Latvia, Latvia |
| Andreas Behrend | University of Bonn, Germany |
| Ladjel Bellatreche | Ecole Nationale Supérieure de Mécanique et d'Aérotechnique, France |
| Maria Bielikova | Slovak University of Technology in Bratislava, Slovakia |
| Iovka Boneva | University Lille 1, France |
| Omar Boucelma | LSIS, Aix-Marseille University, France |
| Stephane Bressan | National University of Singapore, Singapore |
| Davide Buscaldi | LIPN, Université Paris 13, France |
| Albertas Caplinskas | Vilnius University, Lithuania |
| Barbara Catania | University of Genoa, Italy |
| Wojciech Cellary | Poznan University of Economics, Poland |
| Richard Chbeir | Université de Pau et des Pays de l'Adour, France |
| Dickson K.W. Chiu | University of Hong Kong, China |
| Ricardo Ciferri | Federal University of São Carlos, Brazil |
| Alfredo Cuzzocrea | University of Calabria, Italy |
| Danco Davcev | University Ss Cyril and Methodius in Skopje, Macedonia |
| Vladimir Dimitrov | Sofia University, Bulgaria |
| Eduard Dragut | Temple University, USA |
| Schahram Dustdar | Vienna University of Technology, Austria |
| Todd Eavis | Concordia University, Canada |
| Johann Eder | University of Klagenfurt, Austria |
| Tobias Emrich | Ludwig-Maximilians-Universität München, Germany |
| Markus Endres | University of Augsburg, Germany |
| Victor Felea | University Alexandru Ioan Cusa, Iasi, Romania |
| Pedro Furtado | University of Coimbra, Portugal |
| Zdravko Galic | University of Zagreb, Croatia |
| Johann Gamper | Free University of Bozen-Bolzano, Italy |
| Minos Garofalakis | Technical University of Crete, Greece |
| Jan Genci | Technical University of Kosice, Slovakia |
| Matteo Golfarelli | University of Bologna, Italy |
| Katarina Grigorova | Ruse University, Bulgaria |
| Giovanna Guerrini | University of Genova, Italy |
| Ralf Hartmut Güting | Fernuniversität Hagen, Germany |
| Theo Härder | Tehnical University Kaiserslautern, Germany |
| Stephen Hegner | Umeå University, Sweden |
| Ali Inan | Isik University, Turkey |
| Mirjana Ivanovic | University of Novi Sad, Serbia |
| Hannu Jaakkola | Tampere University of Technology, Finland |

| | |
|---|---|
| Manfred Jeusfeld | Tilburg University, Netherlands |
| Slobodan Kalajdziski | University Ss Cyril and Methodius in Skopje, Macedonia |
| Leonid Kalinichenko | Russian Academy of Science, Russia |
| Kalinka Kaloyanova | Sofia University St. Kliment Ohridski, Bulgaria |
| Mehmed Kantardzic | University of Louisville, USA |
| Marite Kirikova | Riga Technical University, Latvia |
| Harald Kosch | University of Passau, Germany |
| Georgia Koutrika | HP Labs, USA |
| Andrea Kulakov | University Ss Cyril and Methodius in Skopje, Macedonia |
| Lars Kulik | The University of Melbourne, Australia |
| Wolfgang Lehner | Technical University Dresden, Germany |
| Jan Lindström | IBM Helsinki, Finland |
| Yun Lu | Florida International University, USA |
| Ivan Luković | University of Novi Sad, Serbia |
| Federica Mandreoli | University of Modena, Italy |
| Rainer Manthey | University of Bonn, Germany |
| Manuk Manukyan | Yerevan State University, Armenia |
| Giansalvatore Mecca | University Basilicata, Italy |
| Marco Mesiti | University of Milano, Italy |
| Irena Mlynkova | Charles University in Prague, Czech Republic |
| Alexandros Nanopoulos | University of Eichstaett-Ingolstadt, Germany |
| Pavol Navrat | Slovak University of Technology, Slovakia |
| Daniel C. Neagu | University of Bradford, UK |
| Anisoara Nica | SAP, Canada |
| Nikolaj Nikitchenko | Kiev State University, Ukraine |
| Kjetil Nørvåg | Norwegian University of Science and Technology, Norway |
| Boris Novikov | University of St.Petersburg, Russia |
| Gultekin Ozsoyoglu | Case Western Reserve University, USA |
| Euthimios Panagos | Applied Communication Sciences, USA |
| Gordana Pavlovic-Lazetic | University of Belgrade, Serbia |
| Torben Bach Pedersen | Aalborg University, Denmark |
| Dana Petcu | West University of Timisoara, Romania |
| Evaggelia Pitoura | University of Ioannina, Greece |
| Elisa Quintarelli | Politecnico di Milano, Italy |
| Paolo Rosso | Polytechnic University Valencia, Spain |
| Viera Rozinajova | Slovak University of Technology, Slovakia |
| Ismael Sanz | Universitat Jaume I, Spain |
| Klaus-Dieter Schewe | Software Competence Center, Austria |
| Marc H. Scholl | University of Konstanz, Germany |
| Holger Schwarz | Universität Stuttgart, Germany |
| Timos Sellis | National Technical University of Athens, Greece |

| | |
|---|---|
| Bela Stantic | Griffith University, Australia |
| Predrag Stanisic | University of Montenegro, Montenegro |
| Yannis Stavrakas | Institute for the Management of Information Systems, Greece |
| Krzysztof Stencel | University of Warsaw, Poland |
| Leonid Stoimenov | University of Nis, Serbia |
| Panagiotis Symeonidis | Aristotle University of Thessaloniki, Greece |
| Amirreza Tahamtan | Vienna University of Technology, Austria |
| Ernest Teniente | Unversitat Politècnica de Catalunya, Spain |
| Manolis Terrovitis | Institute for the Management of Information Systems, Greece |
| Bernhard Thalheim | Christian Albrechts University Kiel, Germany |
| A Min Tjoa | Vienna University of Technology, Austria |
| Ismail Toroslu | Middle East Technical University, Turkey |
| Juan Trujillo | University of Alicante, Spain |
| Traian Marius Truta | Northern Kentucky University, USA |
| Ozgur Ulusoy | Bilkent University, Turkey |
| Maurice Van Keulen | University of Twente, Netherlands |
| Olegas Vasilecas | Vilnius Gediminas Technical University, Lithuania |
| Panos Vassiliadis | University of Ioannina, Greece |
| Jari Veijalainen | University of Jyvaskyla, Finland |
| Goran Velinov | University Ss Cyril and Methodius in Skopje, Macedonia |
| Gottfried Vossen | Universität Münster, Germany |
| Boris Vrdoljak | University of Zagreb, Croatia |
| Fan Wang | Microsoft, USA |
| Gerhard Weikum | Max Planck Institute for Informatics, Germany |
| Tatjana Welzer | University of Maribor, Slovenia |
| Marek Wojciechowski | Poznan University of Technology, Poland |
| Robert Wrembel | Poznan University of Technology, Poland |
| Vladimir Zadorozhny | University of Pittsburgh, USA |
| Jaroslav Zendulka | Brno University of Technology, Czech Republic |
| Andreas Zuefle | Ludwig-Maximilians-Universität München, Germany |

## Additional Reviewers

| | |
|---|---|
| Selma Bouarar | LIAS/ISAE-ENSMA, France |
| Kamel Boukhalfa | LSI/USTHB, Algiers |
| Ljiljana Brkić | University of Zagreb, Croatia |
| Jacek Chmielewski | Poznań University of Economics, Poland |
| Armin Felbermayr | Catholic University of Eichstätt Ingolstadt, Germany |
| Flavio Ferrarotti | Software Competence Center Hagenberg (SCCH) |

# Workshop on GPUs In Databases (GID 2014)

## Chairs

| | |
|---|---|
| Witold Andrzejewski | Poznan University of Technology, Poland |
| Krzysztof Kaczmarski | Warsaw University of Technology, Poland |
| Tobias Lauer | Jedox AG, Germany |

## Program Committee

| | |
|---|---|
| Artur Gramacki | University of Zielona Góra, Poland |
| Bingsheng He | Nanyang Technological University, Singapore |
| Ming Ouyang | University of Massachusets, Boston, USA |
| Gunter Saake | University of Magdeburg, Germany |
| Peter Sestoft | IT University of Copenhagen, Denmark |
| Krzysztof Stencel | University of Warsaw, Poland |
| Jens Teubner | Technische Universitat Dortmund, Germany |
| Paweł Wojciechowski | Poznan University of Technology, Poland |

## Additional Reviewers

| | |
|---|---|
| Sebastian Breß | University of Magdeburg, Germany |

# Workshop on Ontologies Meet Advanced Information Systems (OAIS 2014)

## Chairs

| | |
|---|---|
| Ladjel Bellatreche | National Engineering School for Mechanics and Aerotechnics, France |
| Yamine Aït Ameur | Institut de Recherche en Informatique de Toulouse, France |

## Program Committee

| | |
|---|---|
| Yamine Ait-Ameur | ENSEEIHT/IRIT, Toulouse, France |
| Idir Ait-Sadoune | Supelec, Paris, France |
| Mahmoud Barhamgi | LIRIS, Lyon, France |
| Ladjel Bellatreche | ISAE-ENSMA, France |
| Karim Benouaret | LIRIS, Lyon, France |
| Djamal Benslimane | LIRIS, Lyon, France |
| Sadok Benyahia | Faculty of Sciences of Tunis, Tunisia |
| Brice Chardin | ISAE-ENSMA, France |
| Alfredo Cuzzocrea | ICAR-CNR/University of Calabria, Italy |
| Stéphane Jean | Poitiers University, France |
| Selma Khouri | ESI, Algeria |
| Leandro Krug-Wives | Federal University of Rio Grande do Sul, Brazil |
| Sofian Maabout | Labri, Bordeaux, France |
| Farhi Marir | College of Technological Innovation, Dubai |
| Brahim Medjahed | Michigan University, USA |
| Oscar Romero Moral | Universitat Politècnica de Catalunya, Spain |
| Boris Vrdoljak | University of Zagreb, Croatia |
| Robert Wrembel | Poznań University of Technology, Poland |

# Workshop on Technologies for Quality Management in Challenging Applications (TQCMA 2014)

## Chairs

| | |
|---|---|
| Isabelle Comyn-Wattiau | Conservatoire National des Arts et Métiers, Paris, France |
| Ajantha Dahanayake | Prince Sultan University, Saudi Arabia |
| Bernhard Thalheim | Christian Albrechts University, Germany |

## Program Committee

| | |
|---|---|
| Jacky Akoka | Conservatoire National des Arts et Métiers and TMSP, France |
| Tiziana Catarci | Università di Roma La Sapienza, Italy |
| Corine Cauvet | Université Aix-Marseille 3, France |
| Virginie Goasdoue-Thion | Université Dauphine, Paris, France |
| Paul Johannesson | Stockholm University, Sweden |
| Zoubida Kedad | Université de Versailles, France |
| Oscar Pastor | Valencia University of Technology, Spain |
| Geert Poels | University of Ghent, Belgium |
| Farida Semmak | Université Paris XII, IUT Sénart Fontainebleau |
| Samira Si-Saïd | Conservatoire National des Arts et Métiers, France |
| Carlo Batini | University of Milan, Italy |

## Doctoral Consortium – Program Committee

| | |
|---|---|
| Ladjel Bellatreche | LIAS/ENSMA, France |
| Christos Berberidis | International Hellenic University, Greece |
| Maria Bielikova | Slovak University of Technology in Bratislava, Slovakia |
| Zoran Bosnić | University of Ljubljana, Slovenia |
| Dražen Brdjanin | University of Banja Luka, Republic of Srpska |
| Zoran Budimac | University of Novi Sad, Serbia |
| Barbara Catania | DISI-University of Genoa, Italy |
| Schahram Dustdar | TU Wien, Austria |
| Georgios Evangelidis | University of Macedonia, Greece |
| Minos Garofalakis | Technical University of Crete, Greece |
| Giovanna Guerrini | DISI- University of Genova, Italy |
| Gordan Ježić | University of Zagreb, Croatia |
| Ioannis Katakis | University of Athens, Greece |
| Marite Kirikova | Riga Technical University, Latvia |
| Federica Mandreoli | DII - University of Modena, Italy |
| Alexandros Nanopoulos | University of Eichstaett-Ingolstadt, Germany |
| Pavol Navrat | Slovak University of Technology |
| Kjetil Nørvåg | Norwegian University of Science and Technology, Norway |
| Boris Novikov | St. Pegersburg University, Russia |
| George Pallis | University of Cyprus |
| Thimios Panagos | Applied Communication Sciences |
| Apostolos N. Papadopoulos | Aristotle University of Thessaloniki, Greece |
| Miloš Radovanović | University of Novi Sad, Serbia |
| Klaus-Dieter Schewe | Software Competence Center Hagenberg, Austria |
| Bela Stantic | Griffith University, Australia |
| Yannis Stavrakas | Athena - Research and Innovation Center in Information, Communication and Knowledge Technologies, Greece |
| Panagiotis Symeonidis | Aristotle University of Thessaloniki, Greece |
| Bernhard Thalheim | Christian Albrechts University Kiel, Germany |
| A Min Tjoa | Vienna University of Technology, Austria |
| Grigorios Tsoumakas | Aristotle University of Thessaloniki, Greece |
| Panos Vassiliadis | University of Ioannina, Greece |
| Robert Wrembel | Poznan University of Technology, Poland |
| Jaroslav Zendulka | Brno University of Technology, Czech Republic |

# Contents

# Part I
# Data Mining

# Benchmarking Graph Databases on the Problem of Community Detection

Sotirios Beis, Symeon Papadopoulos, and Yiannis Kompatsiaris

Information Technologies Institute, CERTH, 57001, Thermi, Greece
`{sotbeis,papadop,ikom}@iti.gr`

**Abstract.** Thanks to the proliferation of Online Social Networks (OSNs) and Linked Data, graph data have been constantly increasing, reaching massive scales and complexity. Thus, tools to store and manage such data efficiently are absolutely essential. To address this problem, various technologies have been employed, such as relational, object and graph databases. In this paper we present a benchmark that evaluates graph databases with a set of workloads, inspired from OSN mining use case scenarios. In addition to standard network operations, the paper focuses on the problem of community detection and we propose the adaptation of the Louvain method on top of graph databases. The paper reports a comprehensive comparative evaluation between three popular graph databases, Titan, OrientDB and Neo4j. Our experimental results show that in the current development status Neo4j is the most efficient graph database for most of the employed workloads, while Titan handles better single insertion operations.

## 1 Introduction

Over the past few years there has been vivid research interest in the study of networks (graphs) arising from various social, technological and scientific activities. Typical examples of social networks are graphs constructed with data from Online Social Networks (OSNs), one of the most famous and widespread Web 2.0 application categories. The rapid growth of OSNs contributes to the creation of high-volume and velocity data, which are modeled with the use of graph structures. The increasing demand for massive graph data management and processing systems has been addressed by the researchers proposing new methods and technologies, such as RDBMS, OODBMS, graph databases, etc. Every solution has its pros and cons so benchmarks to evaluate candidate solutions with respect to specific applications are considered necessary.

Relational databases have been widely used for the storage of a variety of data, including social data, and have proven their reliability. On the other hand RDBMS lack operations to efficiently analyze the relationships among the data points. This led to the development of new systems, such as object and graph databases. More specifically, graph databases are designed to store and manage effectively big graph data and constitute a powerful tool for graph-like queries, such as “find the friends of a person”.

In this paper we address the problem of comparing graph databases in terms of performance, focusing on the problem of community detection. We implement a clustering workload, which consists of a well-known community detection algorithm for modularity optimization, the Louvain method [1]. We employ cache techniques to take advantage of both graph database capabilities and in-memory execution speed. The use of the clustering workload is the main contribution of this paper, because to our knowledge other existing benchmark frameworks evaluate graph databases in terms of loading time, node creation/deletion or traversal operations, such as "find the friends of a person" or "find the shortest path between two persons". Furthermore, the benchmark comprises three supplementary workloads that simulate frequently occurring operations in real-world applications, such as the the creation and traversal of the graph. The benchmark implementation is available online as an open-source project[1].

We use the proposed benchmark to evaluate three popular graph databases, *Titan*[2], *OrientDB*[3] and *Neo4j*[4]. For our experiments we used both synthetic and real networks and the comparative evaluation is held with respect to the execution time. Our experimental results show that Neo4j is the most efficient graph database to apply community detection algorithms, in our case the Louvain method. Concerning the supplementary workloads, Neo4j is also the fastest alternative, although Titan performs the incremental creation of the graph faster.

The paper is organized as follows. We begin in Section 2 by providing a survey in the area of benchmarks between database systems oriented to store and manage big graph data. In Section 3 we describe the workloads that compose the benchmark. In Section 4 we list some important aspects of the benchmark. Section 5 presents our experimental study, where we describe the datasets used for the evaluation and report the obtained experimental results. Finally, Section 6 concludes the paper and delineates our future work ideas.

## 2 Related Work

Until now many benchmarks have been proposed, comparing the performance of different databases for graph data. Giatsoglou et al. [2], present a survey of existing solutions to the problem of storing and managing massive graph data. Focusing on the Social Tagging System (STS) use case scenario, they report a comparative study between the Neo4j graph database and two custom storages (H1 and Lucene). Angles et al. [3], considering the category of an OSN as an example of Web 2.0 applications, propose and implement a generator that produces synthetic graphs with OSN characteristics. Using this data and a set of queries that simulate common activities in a social network application, the authors compare two graph databases, one RDF and two relational data management systems. Similarly, in LinkBench [4] a Facebook-like data generator is employed and the performance of a MySQL database system is evaluated. The

[1] `https://github.com/socialsensor/graphdb-benchmarks`
[2] `http://thinkaurelius.github.io/titan/`
[3] `http://www.orientechnologies.com/`
[4] `http://www.neo4j.org/`

authors claim that under certain circumstances any database system could be evaluated with LinkBench.

In a recent effort, Grossniklaus et al. [5] define and classify a workload of nine queries, that together cover a wide variety of graph data use cases. Besides graph databases they include RDBMS and OODBMS in their evaluation. Vicknair et al. [6] also present a benchmark that combines different technologies. They implemented a query workload that simulates typical operations performed in provenance systems and they evaluate a graph (Neo4j) and a relational (MySQL) database. Furthermore, the authors describe some objective measures to compare the database systems, such as security, flexibility, etc.

In contrast with the above works, we argue that the most suitable solution to the problem of massive graph storage and management are graph databases, so our research focuses on them. In this direction Bader et al. [7] describe a benchmark that consists of four kernels (operations): (a) bulk load of the data; (b) retrieval of a set of edges that verify a condition (e.g. weight $> 3$); (c) execution of a $k$-hops operation; and (d) retrieval of the set of nodes with maximum betweenness centrality. Dominguez et al. [8] report the implementation of this benchmark and a comparative evaluation of four graph database systems (Neo4j, HypergraphDB, Jena and DEX).

Ciglan et al. [9] are based on the ideas proposed in [8] and [10], and extend the discussion focusing primarily on graph traversal operations. They compare five graph databases (Neo4j, DEX, OrientDB, NativeSail and SGDB) by executing some demanding queries, such as "find the most connected component". Jouili et al. [11] propose a set of workloads similar to [7] and evaluate Neo4j, Titan, OrientDB and DEX. Unlike, previous works they conduct experiments with multiple concurrent users and emphasize the effects of increasing users. Dayarathna et al. [12] implement traversal operation-based workloads to compare four graph databases (Allegrograph, Neo4j, OrientDB and Fuseki). The key difference with other frameworks is that their interest is focused mostly on graph database server and cloud environments.

## 3 Workload Description

The proposed benchmark is composed of four workloads, Clustering, Massive Insertion, Single Insertion and Query Workload. Every workload has been designed to simulate common operations in graph database systems. Our main contribution is the Clustering workload (CW), however supplementary workloads are employed to achieve a comprehensive comparative evaluation. In this section we describe in more detail the workloads and emphasize their importance by giving some real-world examples.

### 3.1 Clustering Workload

Until now most community detection algorithms used the main memory to store the graph and perform the required computations. Although, keeping data in memory leads to fast executions times, these implementations have a major drawback: they cannot manage big graph data reliably, which nowadays is a key requirement for big graph processing applications. This motivated this work and more specifically the implementation of the Louvain method on top of three graph databases. We used the Gephi Tookit[5] Java implementation of the algorithm as a starting point and applied all necessary modifications to adapt the algorithm to graph databases.

In a first implementation, all the required values for the computations were read directly from the database. The fact that the access of any database (including graph databases) compared to memory is very slow, soon made us realize that the use of cache techniques is necessary. For this purpose we employed the cache implementation of the Guava project[6]. The Guava Cache is configured to evict entries automatically, in order to constrain its memory footprint. Guava provides three basic types of eviction: size-based eviction, time-based eviction, and reference-based eviction. To precisely control the maximum cache size, we utilize the first type of eviction, size-based, and the evaluation was held both between different systems and among different cache sizes. The measurements concern the required time for the algorithm to be completed.

As the authors of the Louvain method mention[7], the algorithm is a greedy optimization method that attempts to optimize the modularity of a partition of the network. The optimization is performed in two steps. First, the method looks for "small" communities by optimizing modularity locally. Second, it aggregates nodes belonging to the same community and builds a new network whose nodes are the communities. We call those *communities* and *nodeCommunities* respectively. The above steps are repeated in an iterative manner until a maximum of modularity is attained.

We keep the community and nodeCommunity values stored in the graph database as a property of each node. The implementation is based on three functions that retrieve the required information either by accessing the cache or the database directly. We store this information employing the LoadingCache structure from the Guava Project, which is similar to a ConcurrentMap[8]. More specifically we use the following functions and structures:

- *getNodeNeighours*: gets the neighbours of a node and stores them to a LoadingCache structure, where the key is the node id and the value is the set of neighbours.
- *getNodesFromCommunity*: gets the nodes from a specific community and stores them to a LoadingCache structure, where the key is the community id and the value is the the set of nodes that the community contains.
- *getNodesFromNodeCommunity*: gets the nodes from a specific nodeCommunity and stores them to a LoadingCache structure, where the key is the nodeCommunity id and the value is the the set of nodes that the nodeCommunity contains.

[5] `https://gephi.org/toolkit/`

[6] `https://code.google.com/p/guava-libraries/`

[7] `http://perso.uclouvain.be/vincent.blondel/research/louvain.html`

[8] `http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ConcurrentMap.html`

We use the above information to compute values such as, the degree of a node, the amount of connections a node or a nodeCommunity has with a particular community, the size of a community or a nodeCommunity.

The Clustering Workload is very important due to its numerous applications in OSNs [13]. Some of the most representative examples include topic detection in collaborative tagging systems, such as Flickr or Delicious, tag disambiguation, user profiling, photo clustering, and event detection.

### 3.2 Supplementary Workloads

In addition to CW, we recognize that a reliable and comprehensive benchmark should contain some supplementary workloads. Here, we list and describe the three additional workloads that constitute the proposed benchmark.

- *Massive Insertion Workload (MIW)*: we create the graph database and configure it for massive loading, then we populate it with a particular dataset. We measure the time for the creation of the whole graph.
- *Single Insertion Workload (SIW)*: we create the graph database and load it with a particular dataset. Every object insertion (node or edge) is committed directly and the graph is constructed incrementally. We measure the insertion time per block, which consists of one thousand nodes and the edges that appear during the insertion of these nodes.
- *Query Workload (QW)*: we execute three common queries:
    - FindNeighbours (FN): finds the neighbours of all nodes.
    - FindAdjacentNodes (FA): finds the adjacent nodes of all edges.
    - FindShortestPath (FS): finds the shortest path between the first node and 100 randomly picked nodes.

  Here we measure the execution time of each query.

It is obvious that MIW simulates the case study in which graph data are available and we want to load them in batch mode. On the other hand, SIW models a more real-time scenario in which the graph is created progressively. We could claim that the growth of an OSN follows the steps of SIW, by adding more users (nodes) and relationships (edges) between them.

The QW is very important as it applies in most of the existing OSNs. For example with the FN query we can find the friends or followers of a person in Facebook or Twitter respectively, with the FA query we can find whether two users joined a particular Facebook group and with the FS query we can find at which level two users connect with each other in Linkedin. It is critical for every OSN that these queries can be executed efficiently and in minimal time.

## 4 Benchmark Description

In this section we discuss some important aspects of the benchmark implementation. The graph database systems selected for the evaluation are Titan (v0.4.1), OrientDB (v1.7-RC2) and Neo4j (v2.0.1). The benchmark was implemented in

Java 1.7 using the Java API of each database. In order to configure each database, we used the default configuration and the recommendations found in the documentation of the web sites.

For Titan we implement MIW with the BatchGraph interface that enables batch loading of a large number of edges and vertices, while for OrientDB and Neo4j we employ the OrientGraphNoTx and BatchInserter interface respectively, which drop the support for transactions in favor of insertion speed. For all graph databases we implement SIW without using any special configuration. The operations for the QW and CW for the OrientDB and Titan were implemented using the Gremlin API, whereas for the Neo4j we employed its respective API.

To ensure that a benchmark provides meaningful and trustworthy results, it is necessary to guarantee its fairness and accuracy. There are many aspects that can influence the measurements, such as the system overhead. It is really important that the results do not come from time periods with different system status (e.g. different number of processes in the background), so we execute MIW, SIW and QW sequentially for each database. In addition to this, we execute them in every possible combination for each database, in order to minimize the possibility that the results are affected by the order of execution. We report the mean value of all measurements.

Regarding the CW, in order to eliminate the cold cache effects we execute it twice and keep always the second value. Moreover, as we described in the previous section to get an acceptable execution time, cache techniques are necessary. The cache size is defined as a percentage of total nodes. For our experiments we use six different cache sizes (5%, 10%, 15%, 20%, 25%, 30%) and we report the respective improvements.

## 5 Experimental Study

In this section we present the experimental study. At first we describe the datasets used for the evaluation. We include a table with some important statistics of each dataset. Then we report and discuss the results.

### 5.1 Datasets

The right choice of datasets that will be used for running database benchmarks is important to obtain representative and meaningful results. It is necessary to test the databases on a sufficient number of datasets of different sizes and complexity to get an approximation of the database scaling properties.

For our evaluation we use both synthetic and real data. More specifically, we execute MIW, SIW and QW with real networks derived from the SNAP dataset collection[9]. On the other hand, with the CW we use synthetic data generated with the LFR-Benchmark generator [1] that produces networks with power-law degree distribution and implanted communities within the network. The Table 1 presents the summary statistics of the datasets.

[9] `http://snap.stanford.edu/data/index.html`

**Table 1.** Datasets used in the experiments

| Dataset | Nodes | Edges | max. $\kappa$ | $\langle \kappa \rangle$ | $\langle cc \rangle$ |
|---|---|---|---|---|---|
| Graph500 | 500 | 1,975 | 25 | 7.904 | 0.267 |
| Graph1000 | 1,000 | 7,745 | 50 | 15.490 | 0.296 |
| Graph2000 | 2,000 | 29,720 | 100 | 29.721 | 0.291 |
| Graph3000 | 3,000 | 65,839 | 150 | 43.893 | 0.303 |
| Graph4000 | 4,000 | 120,814 | 200 | 60.407 | 0.301 |
| Graph5000 | 5,000 | 184,711 | 249 | 73.884 | 0.313 |
| Graph10000 | 10,000 | 748,105 | 500 | 149.622 | 0.291 |
| Enron (EN) | 36,692 | 367,662 | 1,383 | 20.041 | 0.497 |
| Amazon (AM) | 334,863 | 925,872 | 168 | 5.530 | 0.398 |
| Youtube (YT) | 1,134,890 | 2,987,624 | 28,576 | 5.265 | 0.081 |
| Livejournal (LJ) | 3,997,962 | 34,681,189 | 14,703 | 17.349 | 0.045 |

## 5.2 Benchmark Results

In this section we report and discuss the performance of Titan, OrientDB and Neo4j employing the proposed benchmark. Table 2 lists the required time for the execution of MIW and QW, while Figure 1 illustrates the experimental results of SIW. Table 3 and Figure 2 depict the measurements of CW. Note that in every table we mark the best performance with bold. All experiments were run on a 2×Intel Xeon 6-core at 2.1Ghz with 128GB of main memory and a 2.8 TB hard disk, the OS being Ubuntu Linux 12.04 (64bit).

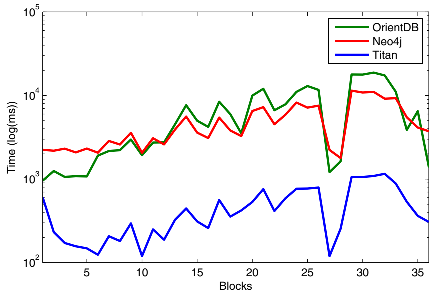
Table 2 summarizes the measurements of the MIW and QW for all the benchmarked graph databases with respect to each real dataset. According to the benchmark results, we observe that Neo4j handles the massive insertion of the data more efficiently from its competitors. Titan is also an effective alternative, while OrientDB could not load the data in a comparable time.

Concerning the QW, Table 2 indicates that Neo4j performs queries more effectively than the other candidates. More specifically, although Neo4j has slightly smaller execution times comparing to OrientDB in the FN query load, Neo4j is considerably faster in the FA and FS query loads. It is worth mentioning that the shortest path search is limited to paths of depth 6, because with larger depth the FS query workload in Titan and OrientDB cannot be executed in a reasonable amount of time.

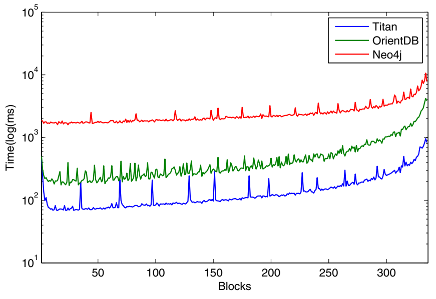
The results for SIW with each real dataset are illustrated in Figure 1. Each sub-figure includes three diagrams, one for every graph database, that plot the required time for the insertion of a block. As we described in Section 3, a block consists of 1,000 nodes and the edges that appear during the insertion of these nodes. In order to present more readable diagrams for the three technologies we used a logarithmic scale for the time axis. It appears that Titan is the most efficient solution for single insertion of data. Moreover, we observe that the performance of OrientDB and Neo4j is comparable, however OrientDB seems to perform slightly better.

**Table 2.** MIW and QW results (sec)

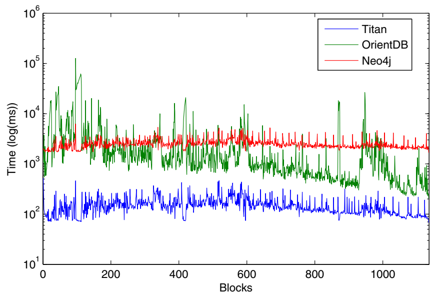
| Graph | Workload | Titan | OrientDB | Neo4j |
|---|---|---|---|---|
| EN | MIW | 7.020 | 378.192 | **1.091** |
| AM | MIW | 28.997 | 298.560 | **4.635** |
| YT | MIW | 80.064 | 5963.333 | **16.830** |
| LJ | MIW | 720.179 | 2485.529 | **229.465** |
| EN | QW-FN | 4.707 | **0.976** | 1.144 |
| AM | QW-FN | 12.243 | 4.75 | **3.224** |
| YT | QW-FN | 37.912 | 14.804 | **10.532** |
| LJ | QW-FN | 352.968 | 68.176 | **109.494** |
| EN | QW-FA | 6.132 | 3.449 | **0.478** |
| AM | QW-FA | 18.143 | 17.877 | **1.446** |
| YT | QW-FA | 58.921 | 44.013 | **3.486** |
| LJ | QW-FA | 504.909 | 341.306 | **44.510** |
| EN | QW-FS | 10.652 | 16.554 | **0.483** |
| AM | QW-FS | 0.149 | 11.382 | **0.293** |
| YT | QW-FS | 6.598 | 6.927 | **0.223** |
| LJ | QW-FS | 31.01 | 47.183 | **0.479** |



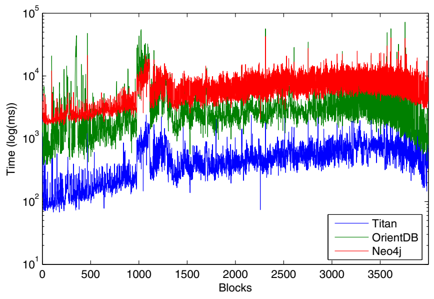
(a) Enron



(b) Amazon



(c) Youtube



(d) Livejournal

**Fig. 1.** SIW benchmark results

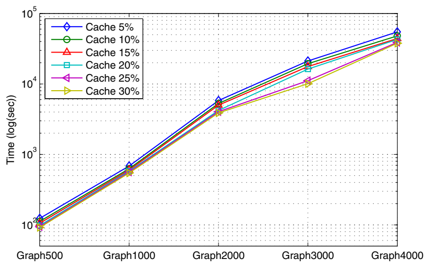
**Table 3.** CW results (sec)

| Graph-Cache | Titan | OrientDB | Neo4j |
|---|---|---|---|
| Graph500-5% | 123.774 | 9.525 | **3.786** |
| Graph500-10% | 111.943 | 9.019 | **3.15** |
| Graph500-15% | 105.425 | 8.879 | **3.01** |
| Graph500-20% | 98.084 | 8.126 | **2.808** |
| Graph500-25% | 94.183 | 6.776 | **2.408** |
| Graph500-30% | 92.021 | 6.566 | **2.212** |
| Graph1000-5% | 683.572 | 44.449 | **8.841** |
| Graph1000-10% | 626.777 | 36.651 | **8.591** |
| Graph1000-15% | 599.873 | 33.896 | **7.312** |
| Graph1000-20% | 579.948 | 29.797 | **6.912** |
| Graph1000-25% | 562.162 | 27.227 | **6.94** |
| Graph1000-30% | 544.43 | 25.448 | **6.629** |
| Graph2000-5% | 5822.678 | 254.05 | **38.29** |
| Graph2000-10% | 5283.674 | 224.65 | **35.824** |
| Graph2000-15% | 5004.3 | 200.074 | **32.747** |
| Graph2000-20% | 4202.316 | 182.04 | **31.111** |
| Graph2000-25% | 4041.574 | 168.995 | **30.317** |
| Graph2000-30% | 3884.714 | 152.193 | **29.833** |
| Graph3000-5% | 21191.741 | 870.295 | **95.355** |
| Graph3000-10% | 19364.449 | 755.737 | **89.931** |
| Graph3000-15% | 17663.532 | 711.586 | **86.228** |
| Graph3000-20% | 16250.283 | 649.357 | **83.474** |
| Graph3000-25% | 1137.774 | 614.796 | **79.753** |
| Graph3000-30% | 1081.654 | 564.096 | **77.595** |
| Graph4000-5% | 54758.397 | 1806.228 | **175.89** |
| Graph4000-10% | 47998.816 | 1533.492 | **153.151** |
| Graph4000-15% | 44158.921 | 1408.739 | **146.413** |
| Graph4000-20% | 42834.153 | 1309.597 | **136.671** |
| Graph4000-25% | 38979.079 | 1237.009 | **130.942** |
| Graph4000-30% | 37809.707 | 1145.699 | **120.792** |
| Graph5000-5% | - | 3308.99 | **284.463** |
| Graph5000-10% | - | 2846.417 | **248.167** |
| Graph5000-15% | - | 2547.742 | **244.285** |
| Graph5000-20% | - | 2327.123 | **231.639** |
| Graph5000-25% | - | 2114.561 | **213.956** |
| Graph5000-30% | - | 1944.151 | **199.226** |
| Graph10000-5% | - | 23351.878 | **1514.992** |
| Graph10000-10% | - | 21250.209 | **1378.643** |
| Graph10000-15% | - | 16292.998 | **1106.668** |
| Graph10000-20% | - | 14866.024 | **1017.766** |
| Graph10000-25% | - | 14042.227 | **927.603** |
| Graph10000-30% | - | 13183.961 | **852.232** |

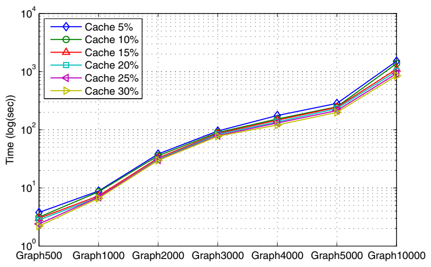
The experimental results of CW are reported in Table 3. Both the OrientDB and the Neo4j are much faster than the Titan graph database. The serious delay of the Titan did not allow us to complete the experiments with larger graphs (Graph5000 and Graph10000). Furthermore, Table 3 reveals that while OrientDB has comparable execution times with Neo4j for the small graphs, it does not scale as good as Neo4j. Thus, for graphs with >1,000 nodes, Neo4j is much faster. Given that the Louvain method consists of a set queries, such as "get the neighbours of a node", the above results are expected, if we take into consideration the results of the QW.

Additionally, Table 3 points out the positive impact of increasing the cache size. We observe that for all graph databases regardless of the graph size, as the cache size increases the execution time decreases. We wrap up the comparative evaluation, including Figure 2 which depict the scalability of each database when the CW is executed. Every sub-figure contains six diagrams, one for each cache value, that plot the required time for the convergence of the algorithm for the respective synthetic graph. For better representation we used a logarithmic scale for the time axis. We can deduce that since the diagrams with the logarithmic scale increase linearly, the execution time follows an exponential distribution.

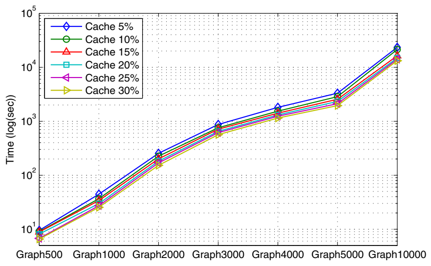
In summary, we found that despite the fact that Neo4j falls behind in SIW compared to Titan and OrientDB, for the MIW, QW and CW, is clearly the most efficient solution, especially when we deal with big graph data. On the



(a) Titan



(b) Neo4j



(c) OrientDB

**Fig. 2.** CW benchmark results

other hand, Titan is the fastest alternative for the incremental creation of a graph database (SIW). Titan also has competitive performance in MIW, but does not scale very well compared to its two competitors.

## 6 Conclusions and Future Work

In this paper we proposed a benchmark framework for the comparative evaluation of database systems oriented to store and manage graph data. The benchmark consists of four workloads, Massive Insertion, Single Insertion, Query and Clustering Workload. For the Clustering Workload we implemented a well-known community detection algorithm, the Louvain method, on top of three graph databases. Employing the proposed benchmark we evaluated the selected graph databases, Titan, OrientDB and Neo4j using both synthetic and real networks.

The experimental results demonstrate that in most cases the measurements are comparable when processing small graphs. But when the size of the datasets grows significantly, Neo4j appears to be the most efficient solution for storing and managing graph data. On the other hand, Titan seems to be the best alternative for single insertion operations.

In the future we hope to investigate the performance gain if we parallelize the operations of the graph databases. Moreover, it would be interesting to run the benchmark employing the distributed implementations of Titan and OrientDB in order to examine their horizontal and vertical scalability properties. Also, we intend to improve the performance of the implemented community detection algorithm and test it on graphs of much larger size.

**Acknowledgments.** This work was supported by the SocialSensor FP7 project, partially funded by the EC under grant agreement 287975.

## References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10), P10008 (2008)
2. Giatsoglou, M., Papadopoulos, S., Vakali, A.: Massive graph management for the web and web 2.0. In: Vakali, A., Jain, L.C. (eds.) New Directions in Web Data Management 1. SCI, vol. 331, pp. 19–58. Springer, Heidelberg (2011)
3. Angles, R., Prat-Pérez, A., Dominguez-Sal, D., Larriba-Pey, J.L.: Benchmarking database systems for social network applications. In: First International Workshop on Graph Data Management Experiences and Systems, GRADES 2013, pp. 15:1–15:7. ACM, New York (2013)
4. Armstrong, T.G., Ponnekanti, V., Borthakur, D., Callaghan, M.: Linkbench: a database benchmark based on the facebook social graph (2013)
5. Grossniklaus, M., Leone, S., Zäschke, T.: Towards a benchmark for graph data management and processing (2013)

6. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkins, D.: A comparison of a graph database and a relational database: A data provenance perspective. In: Proceedings of the 48th Annual Southeast Regional Conference, ACM SE 2010, pp. 42:1–42:6. ACM, New York (2010)
7. Bader, D.A., Feo, J., Gilbert, J., Kepner, J., Koester, D., Loh, E., Madduri, K., Mann, B., Meuse, T., Robinson, E.: HPC scalable graph analysis benchmark (2009)
8. Dominguez-Sal, D., Urbón-Bayes, P., Giménez-Vañó, A., Gómez-Villamor, S., Martínez-Bazán, N., Larriba-Pey, J.L.: Survey of graph database performance on the hpc scalable graph analysis benchmark. In: Shen, H.T., et al. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 37–48. Springer, Heidelberg (2010)
9. Ciglan, M., Averbuch, A., Hluchy, L.: Benchmarking traversal operations over graph databases. In: 2012 IEEE 28th International Conference on Data Engineering Workshops (ICDEW), pp. 186–189 (April 2012)
10. Dominguez-Sal, D., Martinez-Bazan, N., Muntes-Mulero, V., Baleta, P., Larriba-Pey, J.: A discussion on the design of graph database benchmarks. In: Nambiar, R., Poess, M. (eds.) TPCTC 2010. LNCS, vol. 6417, pp. 25–40. Springer, Heidelberg (2011)
11. Jouili, S., Vansteenberghe, V.: An empirical comparison of graph databases. In: 2013 International Conference on Social Computing (SocialCom), pp. 708–715 (September 2013)
12. Dayarathna, M., Suzumura, T.: Xgdbench: A benchmarking platform for graph stores in exascale clouds. In: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 363–370 (December 2012)
13. Papadopoulos, S., Kompatsiaris, Y., Vakali, A., Spyridonos, P.: Community detection in social media. Data Mining and Knowledge Discovery 24(3), 515–554 (2012)

# Efficient Processing of Streams of Frequent Itemset Queries[*]

Monika Rokosik and Marek Wojciechowski

Institute of Computing Science,
Poznan University of Technology,
Piotrowo 2, 60-965 Poznan, Poland
Marek.Wojciechowski@cs.put.poznan.pl

**Abstract.** Frequent itemset mining is one of fundamental data mining problems that shares many similarities with traditional database querying. Hence, several query optimization techniques known from database systems have been successfully applied to frequent itemset queries, including reusing results of previous queries and multi-query optimization. In this paper, we consider a new problem of processing of streams of incoming frequent itemset queries, where like in multi-query optimization a number of queries are executed together and share some of their operations, but unlike in previously considered scenarios, new queries are dynamically being added to the currently processed set of queries.

**Keywords:** data mining, frequent itemsets, data mining queries.

## 1 Introduction

Frequent itemset mining is one of fundamental data mining problems, where the goal is to discover subsets frequently occurring in a collection of sets of items. The problem was introduced in the context of market basket analysis as the initial step in association rule mining [1] but quickly became the main focus of research on frequent pattern discovery. While generating association rules from discovered frequent itemsets is a relatively straightforward task, numerous frequent itemset mining algorithms have been proposed, of which Apriori [3] is the most widely implemented in practice. Apriori starts with the discovery of frequent items and then iteratively finds larger frequent itemsets using a generate-and-test strategy, exploiting the property that all subsets of a frequent itemset must also be frequent. In order to facilitate efficient counting of potentially frequent itemsets (called candidates), Apriori maintains a specialized in-memory data structure called hash tree.

Frequent itemset mining can be regarded as advanced database querying [7], and hence may benefit from optimization strategies that have previously been considered and successfully applied in the context of database management systems. A frequent itemset query contains predicates for selection of source data

[*] This work was partially supported by the Polish National Science Center (NCN), Grant No. 2011/01/B/ST6/05169.

N. Bassiliades et al. (eds.), *New Trends in Database and Information Systems II*,
Advances in Intelligent Systems and Computing 312, DOI: 10.1007/978-3-319-10518-5_2

to be mined and a minimum support threshold. Optionally, it may also contain predicates concerning frequent itemsets to be discovered.

First solutions addressing efficient processing of frequent itemset queries focused on incorporating pattern constraints into the mining process, rather than verifying them in the post-processing step, to reduce the query execution time. Various types of patterns constraints were identified and strategies of handling them within existing pattern mining methodologies were proposed [13].

The next step in the area of frequent itemset query optimization was reusing materialized results of previous queries. It was observed that data mining is often an interactive and iterative process where users adjust constraints of their queries, and as a result, a sequence of similar data mining queries may be submitted to the system. Several result reusing schemes were proposed, exploiting various classes of differences between the queries [6][10][12].

Finally, the problem of efficient processing of sets of frequent itemset queries was considered, borrowing general ideas of computation sharing from the area of multi-query optimization in database systems. This time the motivation was to speed up execution of batches of queries that may occur mainly in data mining systems working in a batch mode. Several processing schemes were proposed for concurrent execution of sets of frequent itemset queries, broadly divided into techniques depending on (e.g. [15]) and independent of (e.g., [16]) a particular frequent itemset mining algorithm.

In this paper we shift the focus back on interactive data mining systems. Reusing results of previous queries, which is targeted at such systems, has several important limitations. Firstly, any given query may benefit from the results of only these queries which have completed earlier and whose results have been materialized. Secondly, specific relationships between the source datasets and pattern constraints of the queries must hold for one query to be able to consume the results of another query.

On the other hand, techniques of processing of sets of frequent itemset queries try to exploit any overlapping between the queries' datasets but their application to streams of queries is problematic. It was postulated that an interactive system could group queries from a given time window in order to process them together but clearly such a strategy, while possibly beneficial from the point of view of utilizing the system's resources, may lead to postponing some queries with no actual benefit.

Motivated by the shortcomings of existing solutions, we propose to handle streams of frequent itemset queries in a similar manner to sets of such queries, i.e., trying to benefit from any overlapping among the datasets of the queries currently available to the data mining system, but allowing new queries to join the batch currently being executed without waiting for it to complete. Obviously, within such an approach we are going to look for solutions by adapting existing techniques for sets of frequent itemset queries to handle "dynamic" batches of queries. In this paper, we focus on the adaptation of one of the simplest but at the same time efficient and easy to implement technique called Common Counting [15], dedicated to Apriori.

## 2 Related Work

Apart from the approaches to optimizing execution of frequent itemset queries already mentioned in the introduction, the most related to the problem considered in this paper are works concerning multi-query optimization in data mining and other research domains.

Multiple-query optimization has been extensively studied in the context of database systems (see [14] for an overview). The idea was to identify common subexpressions and construct a global execution plan minimizing the overall processing time by executing the common subexpressions only once for the set of queries [4]. In data warehousing, multiple-query optimization has been applied to speed up maintenance of the set of materialized views by exploiting common subexpressions between different view maintenance expressions [11].

To the best of our knowledge, apart from the problem considered in this paper, multiple-query optimization for frequent pattern queries has been considered only in the context of frequent pattern mining on multiple datasets [9]. The idea was to reduce the common computations appearing in different complex queries, each of which compared the support of patterns in several disjoint datasets. This is fundamentally different from our problem, where each query refers to only one dataset and the queries' datasets overlap.

The need for multiple-query optimization has also been postulated in the somewhat related research area of inductive logic programming, where a technique based on similar ideas as Common Counting was proposed, consisting in combining similar queries into query packs [5].

## 3 Background and Common Counting Technique

### 3.1 Basic Definitions

**Definition 1.** Let $I$ be a set of literals called *items*. An *itemset* $X$ is a set of items from $I$ ($X \subseteq I$). The *size* of the itemset is the number of items in it. An itemset of size $k$ is called a $k$-itemset.

A *transaction* over $I$ is a couple $T = \langle tid, X \rangle$, where $tid$ is a transaction identifier and $X$ is an itemset. A database $\mathcal{D}$ over $I$ is a set of transactions over $I$ such that each transaction has a unique identifier.

A transaction $T = \langle tid, X \rangle$ *supports* an itemset $Y$ if $Y \subseteq X$. The *support* of an itemset $Y$ in $\mathcal{D}$ is the number of transactions in $\mathcal{D}$ that support $Y$. An itemset is called frequent in $\mathcal{D}$ if its support is no less than a user-specified minimum support threshold.

Given a database $\mathcal{D}$ and a minimum support threshold *minsup*, the problem of *frequent itemset mining* consists in discovering all frequent itemsets in $\mathcal{D}$ together with their supports.

**Definition 2.** A *frequent itemset query* is a tuple $dmq = (\mathcal{R}, a, \Sigma, \Phi, minsup)$, where $\mathcal{R}$ is a database relation, $a$ is a set-valued attribute of $\mathcal{R}$, $\Sigma$ is a condition involving the attributes of $\mathcal{R}$ called a *data selection predicate*, $\Phi$ is a condition involving discovered itemsets called a *pattern constraint*, and *minsup* is the minimum support threshold. The result of *dmq* is a set of itemsets discovered in $\pi_a \sigma_\Sigma \mathcal{R}$, satisfying $\Phi$, and having support $\geq$ *minsup* ($\pi$ and $\sigma$ denote relational projection and selection operations respectively).

**Definition 3.** The *set of elementary data selection predicates* for a set of frequent itemset queries $DMQ = \{dmq_1, dmq_2, ..., dmq_n\}$ is the smallest set $S = \{s_1, s_2, ..., s_k\}$ of data selection predicates over the relation $\mathcal{R}$ such that for each $u$, $v$ $(u \neq v)$ we have $\sigma_{s_u} \mathcal{R} \cap \sigma_{s_v} \mathcal{R} = \emptyset$ and for each $dmq_i$ there exist integers $a, b, ..., m$ such that $\sigma_{\Sigma_i} \mathcal{R} = \sigma_{s_a} \mathcal{R} \cup \sigma_{s_b} \mathcal{R} \cup .. \cup \sigma_{s_m} \mathcal{R}$. The set of elementary data selection predicates represents the partitioning of the database determined by overlapping of queries' datasets.

### 3.2 Common Counting

Common Counting reduces the data retrieval costs for a batch of frequent itemset queries with respect to sequential processing by concurrent execution of a set of frequent itemset queries using Apriori and integration of scans of the shared parts of the database. The method iteratively generates and counts candidates for all frequent itemset queries. The candidates are generated separately for each query using the original procedure from the Apriori algorithm and then stored in separate hash trees. Occurrences of candidates for all the queries are counted during one integrated database scan so that if a database partition is shared by several queries, it is read only once during each candidate counting phase. Common Counting does not incorporate pattern constraints into the actual mining process, leaving them for post-processing.

Common Counting is a simple technique, optimizing only one aspect of frequent itemset query execution, i.e., data retrieval, but it has several desired properties important from the point of view of its practical applications. Firstly, it has a negligible overhead and therefore practically guarantees reduction of the overall processing time if any overlapping between the queries' datasets occurs. Secondly, it can be applied to a large number queries even if their hash trees do not fit together in memory thanks to the possibility of partitioning the set of queries and dividing candidate counting into phases [17]. Finally, it has been shown to work well regardless of the availability of efficient access paths to data partitions determined by query overlapping [8].

## 4 Common Counting Stream

The key to adapting Common Counting to streams of frequent itemset queries is the observation that a Common Counting iteration does not rely on the fact that all the processed queries are at the same Apriori iteration. Hence, we can actually add a new query to the currently processed batch even if the queries previously added to it already performed one or more Apriori iterations. We formalize this idea as the Common Counting Stream technique that maintains a dynamic batch of queries and integrates their data retrieval phases. Similarly to Common Counting, Common Counting Stream works in iterations but each query has to control its own iteration counter because Common Counting Stream iterations are not aligned with the queries' Apriori iterations. The pseudo-code of Common Counting Stream is presented in Fig. 1.

```
Input: $DMQ = \{dmq_1, dmq_2, ..., dmq_n\}$,
where $dmq_i = (\mathcal{R}, a, \Sigma_i, \Phi_i, minsup_i)$
(1)  while $true$ do
(2)    update $DMQ$
(3)    $S$ = set of elementary data selection predicates for $DMQ$
(4)    for ($i$=1; $i \leq n$; $i$++) do
(5)      if $ki = 1$ then
(6)        $\mathcal{C}_{ki,i}$ = all possible 1-itemsets
(7)      else
(8)        $\mathcal{C}_{ki,i} = apriori\_gen(\mathcal{F}_{ki-1,i})$
(9)      end if
(10)     if $\mathcal{C}_{ki,i} = \emptyset$ then $Answer_i = \sigma_{\Phi_i} \bigcup_k \mathcal{F}_{k,i}$
(11)   end for
(12)   for each $s_j \in S$ do
(13)     $\mathcal{CC} = \{\mathcal{C}_{ki,i} : \sigma_{s_j}\mathcal{R} \subseteq \sigma_{\Sigma_i}\mathcal{R}\}$
(14)     if $\mathcal{CC} \neq \emptyset$ then $count(\mathcal{CC}, \sigma_{s_j}\mathcal{R})$
(15)   end for
(16)   for ($i$=1; $i \leq n$; $i$++) do
(17)     $\mathcal{F}_{ki,i} = \{C \in \mathcal{C}_{ki,i} : C.counter \geq minsup_i\}$
(18) end while
```

**Fig. 1.** Common Counting Stream

Common Counting Stream works in an infinite loop (line 1). At the beginning of the loop (line 2) it updates the current batch of queries by adding new queries and removing the ones that completed in the previous iteration, and then updates the set of elementary data selection predicates (line 3). Next, the algorithm generates candidates for each query from the batch (lines 4-11) taking into account that for some queries it may be the first iteration whereas for others a later one. Generation of candidates of size greater than one (represented in the pseudo-code by the *apriori_gen*() function) is performed exactly as in the original Apriori algorithm. Current Apriori iterations are tracked individually for each query and denoted by $ki$ in the algorithm. $\mathcal{C}_{ki,i}$ and $\mathcal{F}_{ki,i}$ denote candidates and frequent itemsets of size $ki$ for the query $dmq_i$. If for a given query no further candidates can be generated, the query completes and its final results are collected (line 10).

The counting of candidates is performed exactly as in Common Counting (lines 12-17). For each elementary data selection predicate, the transactions from its corresponding database partition are read one by one. For each transaction the candidates of the queries referring to the database partition being read are considered, and the counters of candidates contained in the transaction are incremented (lines 12-15). The inclusion test is performed by confronting the transaction with hash trees of all the queries referring to the database partition containing the transaction. Candidate counting is represented in the pseudo-code as the *count*() function.

Analogously to Common Counting, in our formulation of Common Counting Stream we assumed that hash trees of all the currently processed queries fit into main memory. However, a practical implementation of Common Counting Stream should apply the same strategy of dividing the counting into phases if the queries' data structures cannot be accommodated together in memory as developed for Common Counting.

## 5 Experimental Results

In order to evaluate efficiency of the proposed new technique of processing streams of frequent itemset queries we performed a series of experiments on synthetic data on a PC with Intel Core 2 Duo 2.4GHz processor and 3.5GB RAM, running Windows 7 32-bit. The compared algorithms were implemented in Java. The test dataset was prepared using the following procedure. First, we generated a small dataset using the GEN [2] generator with the following settings: number of transactions in the database = 100000, average number of items in a transactions = 8, number of different items = 1000, number of patterns = 500, average pattern length = 4. Then, we multiplied the resulting dataset 10 times, thus producing a dataset containing 1000000 transactions. formed of 10 partitions having exactly the same data distribution. Thanks to the applied procedure, when we later considered only queries selecting a number of identical partitions, we eliminated the possible impact of irregular data distribution on the obtained results. The total size of the prepared test dataset was 71MB. The dataset was stored on a hard disk as a flat file accompanied by an index facilitating selective access to data partitions.

As the goal of the proposed technique was to reduce the overall processing time of a stream of queries with respect to sequential execution of the queries, the sequential execution was chosen as a primary reference query execution method. As a secondary reference method we decided to include execution of the set of queries using the original Common Counting technique, which can be regarded as the optimal scenario where all the queries to be processed are submitted to the system at once.

In all the experiments we measured total execution times but to provide a better insight into performance gains due to sharing data reading operations between the queries we also counted the total number of transactions retrieved from the database. The problem with relying solely on execution times in assessment of the compared query processing techniques is that the observed differences in execution times are dependent on the ratio of CPU costs to I/O costs, and the latter depend on the location of the dataset (local disk or remote database server) and its size as well as the possibility of caching the data. In our test bed the whole dataset easily fit into disk cache, while for real-life scenarios that would be unlikely. In fact, we can regard our testing environment as the worst-case scenario to observe reduction of execution times due to sharing data retrieval operations.

In the first series of experiments we tested the effect of the overlapping between the queries' datasets on efficiency of the three compared techniques: sequential execution (seq), Common Counting (cc), and Common Counting Stream (cc-s). We considered the case of two queries, each retrieving half of the dataset. With the way our input database had been generated, for the tested overlapping levels we could formulate the queries so that they always operated on datasets identical in terms of their size and contents, thus eliminating the possible effect of different data distributions on observed results. The minimum support threshold of both the queries was set to 2.1% so that they performed five Apriori iterations. The second query was added after the first one completed its second Apriori iteration.

The results of this first series of experiments are shown in Fig. 2. As expected, both the number of retrieved transactions and the total execution time of Common Counting Stream decrease linearly with the increase of overlapping. However, since the queries were processed by Common Counting Stream together only for three out of their five Apriori iterations, the performance gains are smaller than these of the reference Common Counting method, which was provided with both the queries from the beginning of its operation (this is how Common Counting was designed to operate, of course).

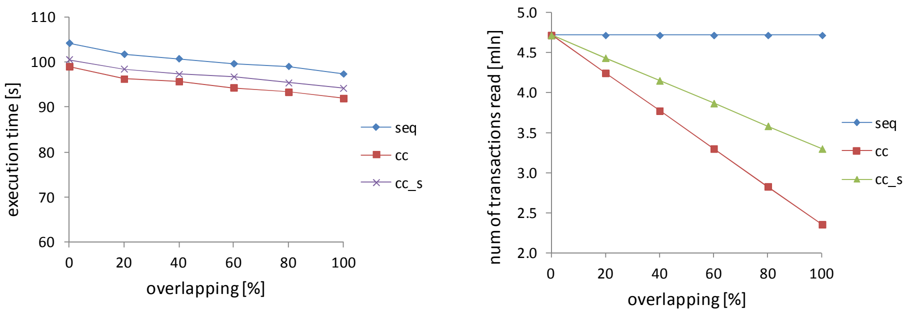


**Fig. 2.** Execution times (left) and numbers of transactions read (right) for different levels of overlapping between two frequent itemset queries with minsup=2.1%

The goal of the second series of experiments was to observe the effect of iteration offset between the queries (i.e., the number of Apriori iterations performed by the first query after which the second query was added). In this series of experiments the queries always shared 60% of their datasets. The experiments were repeated for two minimum support thresholds: 2.1% and 3%. With the increased support threshold both the queries required only four iterations to complete, i.e., one iteration less than for the threshold of 2.1%. The results are presented in Figures 3 and 4.

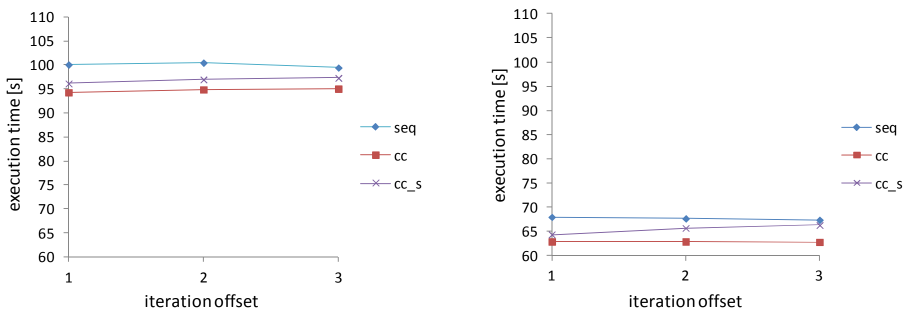


**Fig. 3.** Execution times for two frequent itemset queries for different iteration offsets between the two queries with minsup=2.1% (left) and minsup=3% (right)

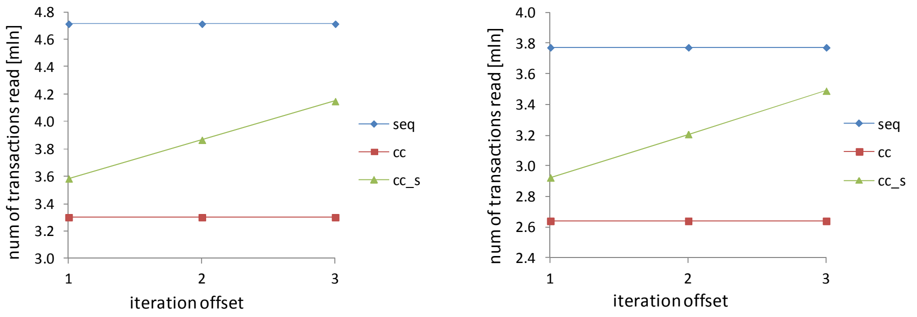


**Fig. 4.** Numbers of transactions read for two frequent itemset queries for different iteration offsets between the two queries with minsup=2.1% (left) and minsup=3% (right)

Obviously, the iteration offset matters only for Common Counting Stream. In sequential execution any subsequent query is postponed until the previous one completes. On the other hand, Common Counting was applied assuming that both the queries were available from the beginning. The experiments show that the smaller the iteration offset the better for the efficiency of Common Counting Stream. This is certainly not surprising as the smaller this offset the more iterations can have their data scanning phases integrated for the two queries. Since we expressed the iteration offset as the number of Apriori iterations (not the percentage), its impact was more visible for the higher of considered support thresholds, for which the total number of iterations, and consequently the number of iterations where I/O integration took place, was smaller than for the lower support threshold.

In the last series of experiments we tested the three frequent itemset processing schemes on streams of two to five queries in order to evaluate scalability with the number of queries of Common Counting Stream. The streams of queries were prepared using the following set of rules: 1) Each query had the same minimum support threshold (2.1% or 3%) and equal size and contents of the source dataset by referring to 5 consecutive partitions of the database. 2) The first query's dataset started from the first partition, and each subsequent query had its dataset shifted by one partition. 3) Each subsequent query (beginning with the second one) was added after the previous one completed its second iteration.

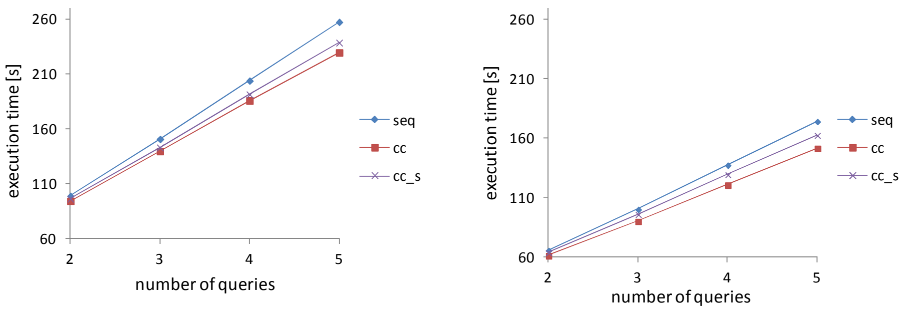


**Fig. 5.** Execution times for streams of two to five frequent itemset queries with minsup=2.1% (left) and minsup=3% (right)

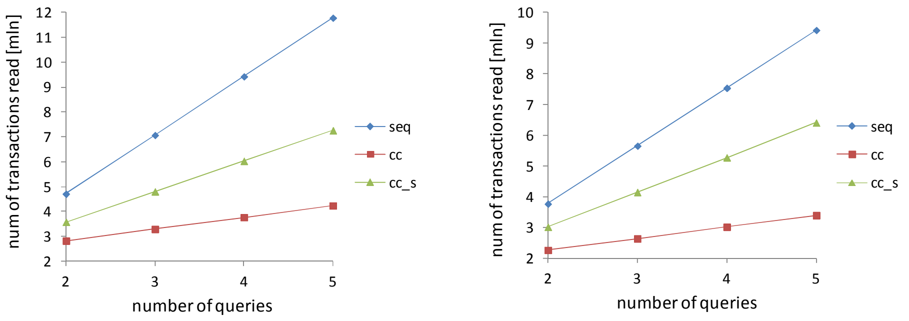


**Fig. 6.** Numbers of transactions read for streams of two to five frequent itemset queries with minsup=2.1% (left) and minsup=3% (right)

Execution times and numbers of transactions retrieved from the database for the last series of experiments are shown in Figures 5 and 6, respectively. It can be seen that all the compared methods scale linearly with the number of queries (under the assumption that the queries are identical in terms of their support thresholds and contents of their source datasets, and additionally for Common Counting Stream the time intervals between subsequent queries are uniform). For the smaller of the considered support thresholds performance of Common Counting Stream is relatively closer to that of Common Counting than for the higher threshold for the same reason as in the second series of experiments.

The general conclusion is that Common Counting Stream is an efficient technique of processing streams of frequent itemset queries. Even in our test environment where the database resided on a local disk and its size was small enough to fit into disk cache, Common Counting Stream noticeably outperformed sequential execution in all the conducted experiments. Moreover, detailed analysis of numbers of data transactions processed by Common Counting Stream is an indication of even more significant benefits in terms of overall processing time in production data mining systems where the source data is often remote and/or too big to fit into disk cache.

On the other hand, in all our experiments Common Counting Stream took longer to complete than Common Counting. Nevertheless, such a behavior was expected as the execution of a stream of frequent itemset queries cannot be easier than the execution of a set of the same queries. In fact, we can regard Common Counting as a specific case of Common Counting Stream where all the queries are available from the beginning, and hence the chances of sharing data retrieval operations between the queries are maximized.

Several parameters influence efficiency of Common Counting Stream. Similarly to Common Counting (and other techniques of processing sets of frequent itemset queries), Common Counting Stream's performance gains with respect to sequential processing are proportional to the level of overlapping between the queries' dataset. Other important factors contributing to the efficiency of Common Counting Stream are the minimum support threshold and time intervals between the queries (which translates to iteration offset for Common Counting Stream). In general, the lower the support threshold and the smaller the interval between query submissions, the better Common Counting Stream performs due to sharing a greater fraction of Apriori iterations between the queries.

## 6 Conclusions and Future Work

In this paper we addressed interactive data mining systems supporting frequent itemset discovery by means of frequent itemset queries. We claimed that existing solutions were not fully adequate for streams of queries occurring in such systems. As desired features of a processing scheme for streams of frequent itemset queries, we listed the ability to exploit any overlapping among the queries' datasets and the possibility to add new queries to the currently processed batch of queries when some of the previous queries are still being processed.

We postulated that a natural direction in search for such processing schemes should be adaptation of existing techniques for sets of frequent itemset queries to the scenario where new queries are continually added. In this paper we presented an extension of Common Counting, a method falling into the aforementioned category, dedicated to the most popular frequent itemset mining algorithm, Apriori. The resulting technique, Common Counting Stream, applies the same data retrieval optimization method as Common Counting but maintains a dynamic batch of queries which can be at different Apriori iterations. Our experimental analysis showed that the new technique noticeably outperforms sequential execution, and may significantly reduce the I/O costs depending on the characteristics of the query stream and of the queries themselves.

A natural direction of future research is adaptation of other techniques of processing batches of frequent itemset queries to handle a stream of incoming queries dynamically added to the processed batch.

## References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.) Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216. ACM Press (1993)
2. Agrawal, R., Mehta, M., Shafer, J.C., Srikant, R., Arning, A., Bollinger, T.: The quest data mining system. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 244–249. AAAI Press (1996)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann (1994)
4. Alsabbagh, J.R., Raghavan, V.V.: Analysis of common subexpression exploitation models in multiple-query processing. In: Proceedings of the Tenth International Conference on Data Engineering, pp. 488–497. IEEE Computer Society (1994)
5. Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J., Vandecasteele, H.: Improving the efficiency of inductive logic programming through the use of query packs. Journal of Artificial Intelligence Research 16, 135–166 (2002)
6. Cheung, D.W.L., Han, J., Ng, V.T.Y., Wong, C.Y.: Maintenance of discovered association rules in large databases: An incremental updating technique. In: Su, S.Y.W. (ed.) Proceedings of the Twelfth International Conference on Data Engineering, pp. 106–114. IEEE Computer Society (1996)
7. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM 39(11), 58–64 (1996)
8. Jedrzejczak, P., Wojciechowski, M.: Data access paths in processing of sets of frequent itemset queries. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 376–385. Springer, Heidelberg (2011)
9. Jin, R., Sinha, K., Agrawal, G.: Simultaneous optimization of complex mining tasks with a knowledgeable cache. In: Grossman, R., Bayardo, R.J., Bennett, K.P. (eds.) Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 600–605. ACM (2005)
10. Meo, R.: Optimization of a language for data mining. In: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 437–444. ACM (2003)
11. Mistry, H., Roy, P., Sudarshan, S., Ramamritham, K.: Materialized view selection and maintenance using multi-query optimization. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, pp. 307–318 (2001)
12. Morzy, T., Wojciechowski, M., Zakrzewicz, M.: Materialized data mining views. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 65–74. Springer, Heidelberg (2000)
13. Pei, J., Han, J.: Can we push more constraints into frequent pattern mining? In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 350–354 (2000)

14. Sellis, T.K.: Multiple-query optimization. ACM Transactions on Database Systems 13(1), 23–52 (1988)
15. Wojciechowski, M., Zakrzewicz, M.: Evaluation of common counting method for concurrent data mining queries. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 76–87. Springer, Heidelberg (2003)
16. Wojciechowski, M., Zakrzewicz, M.: Evaluation of the mine-merge method for data mining query processing. In: Proceedings of the 8th East European Conference on Advances in Databases and Information Systems (2004)
17. Wojciechowski, M., Zakrzewicz, M., Boinski, P.: Integration of dataset scans in processing sets of frequent itemset queries. In: Holmes, D., Jain, L. (eds.) Data Mining: Foundations and Intelligent Paradigms, vol. 1: Clustering, Association and Classification, pp. 223–266. Springer (2012)

# Part II
# Data Warehouses

# A Content-Driven ETL Processes for Open Data

Alain Berro[1], Imen Megdiche[2], and Olivier Teste[3]

[1] Manufacture Tabacs, Université Toulouse I, France
[2] IRIT, Université Toulouse III, France
[3] IUT Blagnac, Université Toulouse II, France
{Berro,Megdiche,Teste}@irit.fr

**Abstract.** The emergent statistical Open Data (OD) seems very promising to generate various analysis scenarios for decision-making systems. Nevertheless, OD has problematic characteristics such as semantic and structural heterogeneousness, lack of schemas, autonomy and dispersion. These characteristics shakes the traditional Extract-Transform-Load (ETL) processes since these latter generally deal with well structured schemas. We propose in this paper a content-driven ETL processes which automates "as far as possible" the extraction phase based only on the content of flat Open Data sources. Our processes rely on data annotations and data mining techniques to discover hierarchical relationships. Processed data are then transformed into instance-schema graphs to facilitate the structural data integration and the definition of the multidimensional schemas of the data warehouse.

**Keywords:** Open Data, ETL, Graphs, Self-Service BI, Hierarchical classification, Data warehouse.

## 1 Introduction

Open Data (OD) is an emergent trend which consists on freely available data provided by public organisations. OD are very rich in statistical and informative data which make them precious to generate interesting data analytic processes [7]. Whatever, they have several problematic characteristics such as structural and semantic heterogeneity, scattering across multiple providers, lack of sources' schemas.

Recently, several works in Linked Open Data (LOD) [5] [2] have sprung up. These works focus on the semantic web and use OD in RDF' format. Nevertheless, this format requires precise knowledges, some manual efforts to generate and link data and has not yet been largely adopted by governmental providers. According to the Open Data benchmark[1] which compares the French (FR), American (US) and British (UK) Open Data, we notice that the percentage of flat OD sources (such as CSV format) is very high (82% FR, 65% US, and 66% UK). These sources are poorly-structured regarding to (i) the data structure (i.e we need advanced techniques and/or human efforts to identify them), (ii) the

[1] `http://fr.slideshare.net/cvincey/opendata-benchmark-fr-vs-uk-vs-us`

flattened and implicit relationships between structural data (i.e bi-dimensional tables can represent more than two indexing structures).

Recently, both scientific [6] [17] and industrial partners [13][19] are interested by these poorly-structured OD. Even though these proposals tackle interesting aspects, none of them addresses the problem of integrating poorly-structured OD in decision systems. These later are based on multidimensional organisation of data warehouses [12] which collect analytical data. The data collection is performed by specialized processes known as Extract-Transform-Load (ETL) processes [20]. Nowadays, these processes are challenged by the mushrooming and scattered poorely-structured data in the web in particular OD. Urged by this observation, new approaches such as [18], [3] and [9] address more suitable ETL solutions to the web of data.

The novelty of our approach is to provide users the possibility to realize automatically "as far as possible" a self-service Business Intelligence (BI)[16] [11] process. Users just need to provide the input resources (flat datasets) without additional external resources (such as ontologies) and/or specific knowledges on BI. Our proposal consists on transforming flat OD into instance-schema graphs which make them exploitable in decision-support systems as well as extensible for the semantic web. Graphs ensure flexibility to cope with semantic and structural heterogeneity in the integration step [16] in particular we provide a holistic optimal integrated graph.

The remainder of this paper is organized as follows. In section 2, we present our process for self-service BI on OD. In Section 3, we detail the different steps of the content-driven ETL processes on OD. In section 4, we present the OD extraction tool (ODET) and some experimentations. Section 5 concludes by drawing the perspectives of this work.

## 2 A Process for Self-service BI on Open Data

The process aims at supporting users to design the multidimensional schema in order to generate a data warehouse. It takes as input poorly-structured flat OD and generates as output multidimensional schemas [14] and a data warehouse. The process, as shown in Fig.1, involves three main phases:

- ☐ Phase 1: The content-driven ETL phase aims at automatically extracting and transforming poorly-structured OD to a structured and flexible representation. The extraction task separates data from structural parts. We proceed on an automatic annotation of the flat OD with three overlapping types: inherent, topological and semantic. Users ensure the validity of automatic annotations. Furthermore, we use data mining techniques, in particular conceptual classification techniques, to identify potential hierarchical relationships in the annotated structural parts. The transformation task transposes identified parts into a common instance-schema graph.
- ☐ Phase 2: The holistic integration takes as input structural parts of several OD graphs and generates as output an integrated graph and the underlying

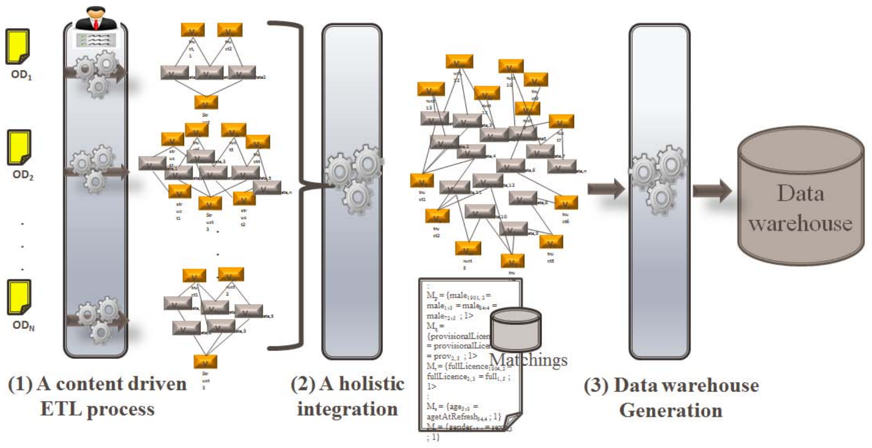

OD1
OD2
ODN
(1) A content driven ETL process
(2) A holistic integration
Matchings
(3) Data warehouse Generation
Data warehouse

**Fig. 1.** A process for self-service BI on Open Data

matchings. We perform a two steps holistic integration. First, we automatically generate the matchings from semantic and syntactic similarity between graphs nodes' labels. Second, we use the graphs' structures to generate the integrated graph and to enhance matchings while ensuring strict and covering hierarchies [8].

☐ Phase 3: From the integrated graph of OD, we propose to users some multidimensional components namely dimensions, dimensions' parameters and hierarchies [12]. These latter correspond to trees, trees' levels and trees' paths in the graph retrieved based on graphs' algorithms. Users can incrementally refine/validate these components and define additional components such as facts and analysis measures [12]. According to the multidimensional schema, we feed the data warehouse with corresponding data. At this level, we resolve the very recurrent problem of differences between granularity levels in OD. For this purpose, we will use simulation techniques to generate artificial data for missing data.

In the remainder of this paper we will detail the first phase of our process.

## 3 Extracting and Transforming Poorly-Structured Open Data

In this section, we propose generic algorithms which are able to extract, enrich and transform structures from flat OD regardless to the complexity of sources. Our algorithms can process complex flat files which contain multiple multi-dimensional matrices disposed randomly in the same source. The input and output of this phase are as following:

– Input: poorly-structured OD. We refer to the basic table anatomy described by [21]. A table contains a body (which contains numerical data) indexed

by structural data which are StubHead (row headers) and BoxHead (column headers). Tables have also associated label regions (titles, footnotes,...).
– Output: instance-schema graphs, denoted as $G = (V, E)$, represents relationships between numerical data (in the body) and structural data (in BoxHead and/or StubHead and/or enrichment data).

## 3.1 Extracting Poorly-Structured Open Data

The "Extraction" step in traditional ETL processes is considered as the simplest task [20]. However, for flat OD, the extraction phase is not really the easiest since we do not have sources' schemas. Hence, we adopt a strategy of annotations and enrichment of sources in the extraction to guide schema discovery using only the content of input sources. We distinguish three overlapping annotation types as follow :

– Inherent annotation (IA) describes the low level data cell types namely *Numeric*, *Label*, *Formula*, *Date* and *Empty.*
– Topological annotation (TA) describes name and location of tables' parts such as numerical blocks (or Body), Boxhead, Stubhead.
– Semantic annotation (SA) describes the semantic class of data, we focused on Temporal (Year, Month,..) and Spatial (Region, GPS coordinates,..) classes.

**Inherent Annotations.** To automatically discover and annotate data, we convert flat OD into a matrix $M$ of size $nbLine \times nbCol$. The matrix represents encoding of low level cells' types namely Numeric, Label, Formula, Date, Empty. The matrix $M$ is defined as:

$M = (a_{i,j})_{1 \leq i \leq nbLine, 1 \leq j \leq nbCol}\ such\ as\ a_{i,j} \in \{-1, 0, 1, 2, 3\}$

In order to enhance the formula data detection which was not correctly encoded as formula, we apply some algorithms on the numeric cells in the matrix $M$. The algorithms check if the numeric content of some cells is the (sum, avg, min,...) of its adjacent cells by calculating the numeric contents of these cells.

**Topological Annotations.** We propose different algorithms to identify the different topological types as follows:

*Unit Numerical Block Identification.* The numerical blocks contain numerical data. These blocks are interesting seeing that they probably form the analytical data which will feed fact' tables [12] in data warehouses. We denote a numerical block as $UnitNumBlock$ (1). It is a sub-matrix of $M$ and we landmark it by four indexes : First Line $FL$, Last Line $LL$, First Column $FC$ and Last Column $LC$.

$$\begin{aligned} &UnitNumBlock(k) = (a_{i,j})_{FL_k \leq i \leq LL_k, FC_k \leq j \leq LC_k} \\ &\text{such as} \\ &a_{i,j} = 1; 1 \leq FL_k \leq LL_k \leq nbLin; 1 \leq FC_k \leq LC_k \leq nbCol \end{aligned} \tag{1}$$

The algorithm 1 searches all the $UnitNumBlock$ contained in the sources. It performs a first pass in the matrix $M$ to identify an horizontal succession of lines containing numerical, this correspond to line 3 in algorithm 1. **FirstLastBlockLines** function returns the index $FL$ and $LL$ of the first identified numerical block from the line $cntLin$. The identified block may encompasses several $UnitNumBlock$, hence we call the **FirstLastBlockColumns** function that proceeds to a vertical search of a succession of numerical columns in $M$ starting from the column $cntCol$. Thereby, the **FirstLastBlockColumns** function returns the index of the $FC$ and $LC$ of the $UnitNumBlock$ and we add it to the $listUnitNumBlock$ of identified data.

**Algorithm 1.** Unit Numerical Block Identification

```
 1: $listUnitNumBlock \leftarrow \emptyset$
 2: while $cntLin \leq nbLin$ do
 3:     $UnitNumBlock(k) \leftarrow$ FirstLastBlockLines$(cntLin)$
 4:     $cntCol \leftarrow 0$
 5:     $cntLin \leftarrow UnitNumBlock(k).FL_k + 1$
 6:     while $cntCol \leq nbCol$ do
 7:         $UnitNumBlock(k) \leftarrow$ FirstLastBlockColumns$(cntCol, UnitNumBlock(k))$
 8:         $cntCol \leftarrow UnitNumBlock(k).FC_k + 1$
 9:         $listUnitNumBlock \leftarrow listUnitNumBlock \cup \{UnitNumBlock(k)\}$
10:     end while
11:     $k \leftarrow k + 1$
12: end while
```

*StubHead and BoxHead Identification.* StubHead and BoxHead are included under structural parts of the sources. They will play a substantial role in the OD integration. StubHead and BoxHead index unit numerical blocks. We use the positions of the different identified $UnitNumBlock$(s) to identify the landmark of this latter. We note that the handled sources can have only one head (BoxHead or StubHead) or both of them.

The BoxHead (2) is a line vector contained in the matrix $M$ situated at the line denoted by $LBH$.

$$BoxHead(k) = (a_{LBH,j})_{UnitNumBlock(k).FC_k \leq j \leq UnitNumBlock(k).LC_k} \tag{2}$$

The StubHead (3) is a column vector contained in the matrix $M$ situated at the column denoted by $CSH$.

$$StubHead(k) = (a_{i,CSH})_{UnitNumBlock(k).FL_k \leq i \leq UnitNumBlock(k).LL_k} \tag{3}$$

The algorithms we use to identify the $BoxHead(k)$(respectively $StubHead(k)$) for each $UnitNumBlock(k)$ consist on a backtracking search of the first line (respectively column) of labels (i.e encoded by 0) situated above (respectively on the left) of the $UnitNumBlock(k)$ beginning the search from the $UnitNumBlock(k).FL_k$ (respectively $UnitNumBlock(k).FC_k$).

*Similar Numerical Block Identification.* Flat OD, as observed, present complex views such as several *UnitNumBlock* from the same source. To cope with this observation, we gather similar *UnitNumBlock*. Similar block is a set of disjoint *UnitNumBlock* having either the same BoxHead block denoted as *SimBlockC* or the same StubHead Block denoted as *SimBlockL*.

$$SimBlockC(resp.SimBlockL) = \cup_{1 \leq k \leq nbSimBlock} UnitNumBlock(k)$$

such as

$$- FC(resp.FL) = \min_k \{UnitNumBlock(k).FC_k(resp.FL_k),$$
$$StubHead.CSH(resp.StubHead.LBH) + 1\}$$
$$- LC(resp.LL) = \max_k \{UnitNumBlock(k).LC_k(resp.LL_k)\}$$
$$- StubHead(k) = StubHead(l)(resp.BoxHead(k) = BoxHead(l))$$
$$\forall k \neq l \; 1 \leq k, l \leq nbSimBlock$$

**Semantic Annotations.** The semantic annotations that concerns us are spatio-temporal data. We propose a straightforward solution which consist on defining two generic graphs for spatial and temporal data (this solution akin to ontology mechanisms). The generic temporal data graph is a covered oriented graph. The vertices represent temporal entities such as (Year...), the edges are directed from the least detailed granularity level vertex to the most detailed granularity level vertex. We refer to the generic temporal graph defined by [10] in which we complete missing edges between levels. The goal is to find paths to link temporal entities. We use regular expressions to identify instances of each temporal entity. The generic spatial data graph represent relations betweens spatial entities such as geographic organisation of a country. The graph must be complete and directed from the low level granularity to the high level granularity. We use predefined lists[2] to feed up our graphs with related instances.

## 3.2 Transforming Open Data into Instance-Schema Graphs

To cope with poorly-structured Open Data, we propose to transform them into simple instance-schema graphs. The choice of graphs is motivated by different reasons : (1) graph data models [1] and graph databases have found increasing interest last years in research communities (semantic web ..) and BI platforms such as SAP HANA, (2) graphs are flexible as they hold objects and relationships which may vary from the most generic and simple to the more specific and complex. The graph we define is simple in order to facilitate its reuse in different scenarios. For instance, we can transform our graphs into RDF by specifying hierarchies as a "subclassOf" property, structural data may be specified by searching the corresponding vocabulary in the Linked Open Vocabulary (LOV). Moreover, the graphs we define can be stored into graph databases (such as Neo4j) or transformed into relational or R-OLAP schemas to enrich enterprises databases. In

[2] `www.geonames.org`

this section, we explain how we transform annotated structural parts such as BoxHead and StubHead into hierarchical trees under some constraints as detailed below. Then, we define two complementary alternatives: (1) algorithms on the annotated data and (2) data mining techniques to derive hierarchical trees. Finally, we show the structure of the instance-schema graphs.

**Constraints for Hierarchical Concept Classification.** Regarding the impact of complex hierarchies [8] in the summarizability issues, we have chosen to tackle the problem since the advanced stages of our approach and particularly when we define hierarchical structure of OD. The different types of complex hierarchies are defined as follow:

– A non-strict hierarchy [8] has parameters that belong to more than one parent. The hierarchy "Film $\rightarrow$ CategoryOfFilm" is non-strict if a film belongs to more than one category, for instance the "Thor" film belongs to categories fantastic and action.
– A non-covering hierarchy [8] has some parameters' instances which skip parameters in a higher level. For instance, in the hierarchy "Office-bank $\rightarrow$ City $\rightarrow$ Region $\rightarrow$ Country", an office may be directly linked to a country without being linked to a city.
– A non-ontological or unbalanced hierarchies is a particular case of the non-covering hierarchies. It has skipped parameters' instances which occur in leaf level. For example, in the hierarchy "Office-bank $\rightarrow$ City $\rightarrow$ Region $\rightarrow$ Country", we can find a city which does not have any office-bank.

We define three constraints to prohibit the generation of complex hierarchies in hierarchical concept classification,we assume that hierarchies are trees :

– **C1**: We must find a unique path between each leaf node and root node. This implies that each node (expect root) has only one parent. The satisfaction of this constraint guarantees strict hierarchies.
– **C2**: In a tree of height $k$, missing intermediate nodes must be duplicated by the value of the parent or generated with a node "Other". The satisfaction of this constraint guarantees covering hierarchies.
– **C3**: The height of the tree must be the same when we traversal tree from all leaf nodes to root node. The satisfaction of this constraint guarantees ontological hierarchies.

**Hierarchical Concept Classification Using Annotations.** We propose three strategies of classification using annotations. The inputs are BoxHead or StubHead and the outputs are hierarchical trees of concepts taking into account the constraints C1,C2 and C3. The strategies are:

1. Numerical blocks arrangement can determine hierarchical relationships between StubHead concepts. The main idea is encoded in algorithm 2. It consists on identifying the hierarchical level of the SimBlockL' StubHead concepts based on the disposition of all UnitNumBlock' StubHead concepts.

2. The annotated formula data can be used to identify hierarchical relationships for StubHead or BoxHead concepts. For instance, if we have a StubHead vector composed of (C1, C2, C3, C4). This later indexes a column of data composed of (numData1, numData2, numData3, numData4) and numData1 is a sum(numData2, numData3, numData4) so we can deduct that C2, C3 , C4 are sub-categories of C1. Based on this reasoning, we can build hierarchical trees for the concepts of BoxHead and StubHead
3. Merged Cells located above BoxHead or in the left of StubHead can be used to deduct a tree representing a generalisation of the concepts in BoxHead or StubHead. The merged cells become roots of the trees and the cells below the BoxHead or in the right of StubHead become leaves of the tree.

**Algorithm 2.** StubHead Conceptual Classification

```
1: $k \leftarrow 1$
2: $crtBlock \leftarrow UnitNumBlock(k)$
3: while $i < SImBlockL.LLandk < nbrBlock$ do
4:     if $i = crtBlock.FL$ then
5:         Assign all concepts in $StubHead$ between $crtBlock.FL$ (resp. $.LL$) to $level_1$
6:         $k \leftarrow k + 1$
7:         $crtBlock \leftarrow UnitNumBlock(k)$
8:         $i \leftarrow crtBlock.LL + 1$
9:     end if
10:    if $i < crtBlock.FL$ then
11:        Count the $nbrConcepts$ between $i$ and $crtBlock.FL$
12:        for $jfromitocrtBlock.FL$ do
13:            Assign each concept to $level_{nbrConcepts+1}$
14:            $nbrConcepts \leftarrow nbrConcepts - 1$
15:        end for
16:    end if
17: end while
```

**Hierarchical Concept Classification Using Data Mining Techniques.** We propose the use of data mining techniques to complement the first category of techniques for hierarchical concept classifications. For this purpose, we apply two techniques of conceptual classification and we select the relevant classes resulting from the cross of the two techniques under the constraints C1, C2 and C3. The first technique is lattices [4] which does not consider semantic aspects and generate formal contexts formed by concepts and attributes (the stem of words composing concepts). The second technique is RELEVANT [3]. It clusters a set of concepts and provide relevant attributes for each cluster. RELEVANT offers two clustering techniques: (1) hierarchical clustering technique which produces disjoint clusters presented by several attribute names, (2) overlapping clustering technique which produce non-disjoint clusters presented by a single attribute name. Seeing that we envision strict hierarchies (constraint C1), we have chosen to apply the hierarchical clustering technique to obtain disjoint clusters. In the following, we will explain how we combine the underlying techniques to obtain a relevant hierarchical classification for a set of concepts.

– Step 1: we perform a tokenisation and stemming to the set of input concepts (StubHead or BoxHead) to generate the list of concepts' attributes.
– Step 2: we apply in parallel to the set of preprocessed concepts the lattice technique and the RELEVANT technique;
– Step 3: we cross the mono-attribute formal contexts obtained from the lattice with the clusters obtained by RELEVANT. We keep the intersection between them which represent relevant mono-attribute formal contexts. After that, we transform the lattice into hierarchical trees. Indeed, the relevant mono-attribute contexts will form the trees roots, then we select from the lattice all their successors. Thereafter, we resolve the problem of strict-hierarchies for each context related to more than one parent, we compute the similarity measure (we use the WUP [22] metric) between the context and its parent and we connect it to the best parent. In case of equal measure we merge the parents into a single node.

**An Instance-Schema Graph of Open Data.** Our instance-schema graphs akin to a property graph model [15]. The OD graph $G = (V, E)$ is described as follows :

***Vertices,*** denoted as $V$, is composed of two types:

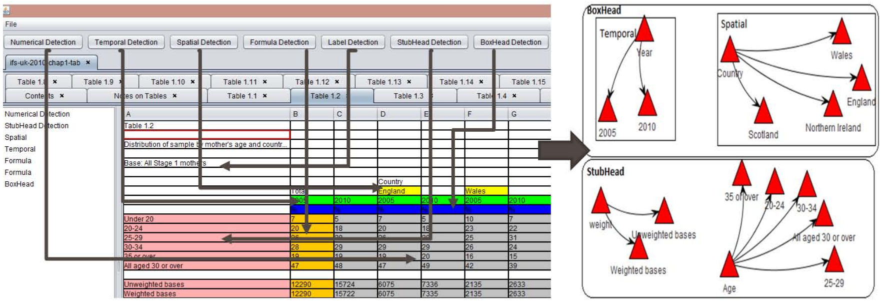
– Structural vertices $V_{Struct}$ which are composed of the StubHead, BoxHead and enrichment data (semantic annotations or concepts of hierarchical trees). The structural vertices are complex objects described by: (1) id, (2) the displayed value,(3) inherent annotation, (4) topological annotation, (5) semantic annotation, (6) position(nbLin,nbCol) or landmarks and (7) identifier of original source.
– Data vertices $V_{NumData}$ which are composed of the numerical data. Data vertices are complex objects described by : (1) id, (2) the displayed value, (3) inherent annotation, (4) topological annotation, (5) semantic annotation, (6) position(nbLin,nbCol) and (7) identifier of original source.

***Edges,*** denoted as $E$, describe two types of relationships: (1) the relationships between two different structural vertices denoted as $E_{Struct,Struct}$, (2) the relationships between structural and data vertices denoted as $E_{Struct,NumData}$.

## 4 Experimetation

An Open Data Extraction Tool (ODET) has been implemented to validate our approach. It enables users to perform automatic and/or manual data extraction on flat OD through different functionalities (StubHead detection, Numerical detection ...). For each functionality, we have attributed a distinct color and we have established an order for the displayed colors in order to keep a closure in the composed results when users carry out several detections with overlapping types. The color of semantic detections is greater than the color of typological annotations which takes the colors of inherent types of cells composing them.

In the left side of Fig. 2, we show the ODET detection results on the dataset[3]. This later presents several tables of the infant feeding survey on 2010 in UK. The example we take shows the distribution of taken samples by mother's age and country in UK in 2005 and 2010. In the right side of the Fig. 2, we observe identified structural parts which has been transformed to hierarchical trees. In the graph, we can distinguish the different analysis axes which have been flattened in a bi-dimensional table in the OD source. We mention that all annotations and graphs structures are saved as graphML files which consists on the input of the integration step.



| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| Table 1.2 | | | | | | |
| | | | | | | |
| Distribution of sample by mother's age and countr... | | | | | | |
| | | | | | | |
| Base: All Stage 1 mothers | | | | | | |
| | | | | | | |
| | | | Country | | | |
| | Total | | England | | Wales | |
| | 2005 | 2010 | 2005 | 2010 | 2005 | 2010 |
| | % | % | % | % | % | % |
| Under 20 | 7 | 5 | 7 | 5 | 10 | 7 |
| 20-24 | 20 | 18 | 20 | 18 | 23 | 22 |
| 25-29 | [illegible] | [illegible] | [illegible] | [illegible] | 25 | 31 |
| 30-34 | 28 | 29 | 29 | 29 | 26 | 24 |
| 35 or over | 19 | 19 | 19 | 20 | 16 | 15 |
| All aged 30 or over | 47 | 48 | 47 | 49 | 42 | 39 |
| | | | | | | |
| Unweighted bases | 12290 | 15724 | 6075 | 7336 | 2135 | 2633 |
| Weighted bases | 12290 | 15722 | 6075 | 7335 | 2135 | 2633 |

**Fig. 2.** Open Data Extraction Tool

To experiment ODET, we have selected 100 flat OD from the data.gouv.fr provider. Tested sources falls within nine topics (Agriculture, culture, employment & economy, research & education, accommodations, international & Europe, society, transport, health). For each source, we have manually evaluated the existence or not of each type $i$ (label (StubHead and/or BoxHead), numerical, spatial, temporal, formula). Then we have computed the number of relevant sources denoted as $md_i$ that contain the type $i$. Thereafter, for each source we execute each automatic function and we observe if the automatic detection returns the same result as the manual annotations. We compute the total number of automatic relevant results denoted as $ad_i$. Hence, the precision value for each type $i$, as shown in table 1, is defined as follow : $Precision(Type_i) = \frac{ad_i}{md_i}$ Table 1 also shows the data proportion of each type (according to the sources' number) among the experimented sources.

As shown in table 1 numerical, temporal and label detections are very satisfactory with precision rates greater than 90%. However, formula and spatial

[3] http://data.gov.uk/dataset/infant-feeding-survey-2010/resource/10b56653-3242-40bf-bc81-3adad559eaa7

**Table 1.** Quality of automatic detection functions

| | Numeric Detection | Label Detection | Formula Detection | Temporal Detection | Spatial Detection |
|---|---|---|---|---|---|
| Precision | 98,88% | 92,22% | 57,89% | 95,23% | 53,84% |
| Data proportion | 100% | 100% | 21,11% | 46,66% | 43,33% |

detection ensure only half detections with precision rate around 50%. These latter deserve to be enhanced especially we have to cope with the spelling error in spatial data and imperfection in numerical formula data.

## 5 Conclusion

In this paper we have defined a content-driven ETL processes for poorly-structured flat Open Data. We have provided automatic algorithms which use only the content of sources to extract and annotate data using three overlapping types. Then we discover hierarchical trees using data mining techniques and data annotations. Thereafter, we transform the processed data into instance-schema graphs. We have presented the OD Extraction Tool and some results according to the detection quality. In our future experimentations, we will evaluate the ODET in other OD providers with a higher number of sources and with different users. Our future works will detail the remaining steps in our self-service BI process.

## References

1. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Comput. Surv. 40(1), 1:1–1:39 (2008)
2. Balakrishnan, S., Chu, V., Hernández, M.A., Ho, H., Krishnamurthy, R., Liu, S., Pieper, J., Pierce, J.S., Popa, L., Robson, C., Shi, L., Stanoi, I.R., Ting, E.L., Vaithyanathan, S., Yang, H.: Midas: integrating public financial data. In: SIGMOD, pp. 1187–1190. ACM (2010)
3. Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C., Vincini, M.: A semantic approach to etl technologies. Data and Knowledge Engineering 70(8), 717–731 (2011)
4. Birkhoff, G.: Lattice Theory, 3rd edn. American Mathematical Society (1967)
5. Böhm, C., Freitag, M., Heise, A., Lehmann, C., Mascher, A., Naumann, F., Ercegovac, V., Hernandez, M., Haase, P., Schmidt, M.: Govwild: integrating open government data for transparency. In: WWW 2012 Companion, pp. 321–324. ACM (2012)
6. Coletta, R., Castanier, E., Valduriez, P., Frisch, C., Ngo, D., Bellahsene, Z.: Public data integration with websmatch. In: WOD, pp. 5–12. ACM (2012)
7. Ghozzi, F., Ravat, F., Teste, O., Zurfluh, G.: Constraints and multidimensional databases. In: 5th International Conference on Enterprise Information Systems, ICEIS 2003, Angers (France), Iceis, pp. 104–111 (2003)
8. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: From conceptual modeling to logical representation. Data Knowl. Eng. 59(2), 348–377 (2006)

9. Mansmann, S., Rehman, N.U., Weiler, A., Scholl, M.H.: Discovering olap dimensions in semi-structured data. Information Systems (2013)
10. Mansmann, S., Scholl, M.H.: Empowering the olap technology to support complex dimension hierarchies. IJDWM 3(4), 31–50 (2007)
11. Mazón, J.N., Zubcoff, J.J., Garrigós, I., Espinosa, R., Rodríguez, R.: Open business intelligence: On the importance of data quality awareness in user-friendly data mining. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops, pp. 144–147 (2012)
12. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Algebraic and graphic languages for OLAP manipulations. International Journal of Data Warehousing and Mining 4(1), 17–46 (2008)
13. Refine, G.: (2014), `http://code.google.com/p/google-refine`
14. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: Dead or alive? In: DOLAP 2006, pp. 3–10. ACM (2006)
15. Rodriguez, M.A., Neubauer, P.: Constructions from dots and lines. Bulletin of the American Society for Information Science and Technology 36(6), 35–41 (2010)
16. Schneider, M., Vossen, G., Zimányi, E.: Data warehousing: from occasional olap to real-time business intelligence (dagstuhl seminar 11361). Dagstuhl Reports 1(9), 1–25 (2011)
17. Seligman, L., Mork, P., Halevy, A.Y., Smith, K., Carey, M.J., Chen, K., Wolf, C., Madhavan, J., Kannan, A., Burdick, D.: Openii: an open source information integration toolkit. In: SIGMOD Conference, pp. 1057–1060. ACM (2010)
18. Skoutas, D., Simitsis, A., Sellis, T.: Ontology-driven conceptual design of ETL processes using graph transformations. In: Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.) Journal on Data Semantics XIII. LNCS, vol. 5530, pp. 120–146. Springer, Heidelberg (2009)
19. Tables, F.: (2014), `http://www.google.com/drive/apps.htmlfusiontables`
20. Vassiliadis, P.: A survey of extract-transform-load technology. IJDWM 5(3), 1–27 (2009)
21. Wang, X.: Tabular abstraction, editing, and formatting. Technical report, University of Waretloo, Waterloo, Ontaria, Canada (1996)
22. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: 32nd Annual Meeting of the Association for Computational Linguistics, pp. 133–138. New Mexico State University, Las Cruces (1994)

# Data Integration Patterns for Data Warehouse Automation

Kalle Tomingas[1], Margus Kliimask[2], and Tanel Tammet[1]

[1] Tallinn University of Technology, Ehitajate tee 5, Tallinn 19086 Estonia
[2] Eliko Competence Center, Teaduspargi 6/2, Tallinn 12618 Estonia

**Abstract.** The paper presents a mapping-based and metadata-driven modular data transformation framework designed to solve extract-transform-load (ETL) automation, impact analysis, data quality and integration problems in data warehouse environments. We introduce a declarative mapping formalization technique, an abstract expression pattern concept and a related template engine technology for flexible ETL code generation and execution. The feasibility and efficiency of the approach is demonstrated on the pattern detection and data lineage analysis case studies using large real life SQL corpuses.

**Keywords:** data warehouse, etl, data mappings, template based sql generation, abstract syntax patterns, metadata management.

## 1 Introduction

The delivery of a successful Data Warehouse (DW) project in a heterogeneous landscape of various data sources, limited resources, and lack of requirements, an unstable focus and budgeting constraints is always challenging and risky. Many long-term DW project failures are related to the requirements and reality mismatch between available data, defined needs and information requirements for decision making [5]. Extract, transform and load (ETL) is a database usage process widely used in the data warehouse field. ETL involves extracting data from outside sources, transforming it to fit operational needs and loading it into the end target: a database or a Data Warehouse.

Mappings between source and target data structures or schemas are the basic specifications of data transformations. Mappings can be viewed as metadata capturing the relationships between information sources and targets. Mappings document the decisions for information structuring and modeling [12]. They are used for several different goals in DW processes: writing a specification for ETL programmers, generating a transformation query or program that uses the semantics of the mapping specification (e.g., a SQL query that populates target tables from source tables), providing metadata about relationships between structures or schemas, providing metadata about data flows and origin sources [4].

Programming mappings in the ETL environment involves writing special database loading scripts (e.g. Oracle Sql*Loader, MSSQL Bulkload, Teradata Fastload, Postgresql Copy etc.) and SQL queries (i.e. select, insert, update and delete statements) which are incremental, iterative, time consuming and routine activities. The manual programming of the data loadings is test-and-error based and not too efficient in case there is no support from the environment and no methodology. Manual scripting and coding of SQL gives high flexibility but backfires in terms of efficiency, complexity, reusability and maintenance of data loadings [7]. The execution and optimization of existing loading programs can be a very complex and challenging task without access to the full dependencies and intelligent machinery to generate optimized workflows [3], [1].

The processes of creation, integration, management, change, reuse, and discovery of data integration programs are not especially efficient without the dependencies and semantics of data structures, mappings and data flows. The creation and management of human- and machine readable documentation, impact analysis (IA) and data lineage (DL) capabilities has become critical for maintaining complex sequences of data transformations. We can control the risks and reduce the costs of dynamic DW processes by making the data flows and dependencies available to developers, managers and end-users.

The paper describes a methodology for formalizing data transformations to an extent that allows us to decouple unique mapping instances from reusable transformation patterns. We demonstrate handling and storing declarative column mappings join and filter predicates in a reusable expression pattern form. We use the Apache Velocity template engine and predefined scenario templates to handle reusable procedural parts of data transformations. We show how the combination of those techniques allows us to effectively construct executable SQL queries and generate utility loading scripts. We also take a look at what has been previously done in the ETL and DW automation field.

In the third chapter we present our open-source architecture of knowledge and metadata repository (MMX[1]) with the related data transformation language and runtime environment (XDTL[2]) which is used as the technology stack for our metadata-driven Data Warehousing process. In the fourth chapter we describe the decoupling of procedural and declarative parts of data mappings and the template-based SQL construction technique. We introduce a case study of Abstract Syntax Pattern (ASP) discovery from a real life DW environment in the fifth chapter and two data lineage analysis case studies in the sixth chapter.

## 2 Related Work

The roles and functions of general programming and ETL tools as well as the relations between manual scripting and script generation are discussed in [7]. The first generation ETL tools were similar to procedural programming or scripting tools, allowing a user to program specific data transformations. The concept of mapping was used for initial specification purposes only.

[1] `www.mmxframework.org`

[2] `www.xdtl.org`

Modern ETL tools - e.g. Informatica PowerCentre, IBM WebSphere DataStage or Oracle Data Integrator - exploit the internal mapping structure for transformation design and script or query generation purposes. In addition to specific ETL technologies there exist the general purpose schema mapping tools that allow discovery and support documenting the transformations or generating transformation scripts (XSLT transformation between XML-schemas, XQuery or SQL DML statements etc.). The meaning and purpose of mappings and general application areas, tools and technologies is discussed by Roth et al. with the generic usage scenarios in different enterprise architecture environments [12]. The declarative mappings designed for ETL program generation and vice versa are discussed in [4] and [6].

In addition to mappings with program generation instructions and data transformation semantics, there exist relations and dependencies between mappings and source or target schema objects. The declarative representation of the dependencies allows us to generate, optimize and execute data transformations workflows effectively. We can find different optimization approaches described in papers [1],[2] and [3]. An extensive study of common models for ETL and ETL job optimization is published by Simitsis et al. [13], [14] and Patil et al. [9]. The dynamic changes of data structures, connections between mapping and jobs and a rule-based ETL graph optimization approach is discussed in [8].

An effective ETL job optimization is related to data mappings, dependencies between mappings, dynamics and changes of source structures and the data quality (DQ). All of those aspects can be formalized and taken into account by ETL job automation where timing, the right order and data volumes are always important issues. Estimation and evaluation of data structures, quality of data and discovery of rules can be automated and integrated into ETL processes [11]. By adding rule-based DQ into ETL process, we can automate the mapping generation and improve the success rate of data loadings. Rodiç et al. demonstrated that most of the integration rules can be generated automatically using the source and target schema descriptions [11].

## 3 System Architecture

The modern enterprise data transformation systems are built according to the model-driven architecture principles, including internal metadata about the source and target models, mappings, transformations and dependencies between models. We introduce a new architectural concept, based on open source java and xml technologies that can be used in lightweight scripting configuration, mixed configuration with partial mapping formalization and full model- and metadata driven knowledge base implementations. When the first lightweight configuration gives you a quick start, low-cost and small technology track and metadata-driven approach gives a knowledge base with different new possibilities (e.g. mapping generation and management, dynamic dependency management and job automation, impact analysis, data quality integration etc.), then both exploit the template-based code construction and automation principles. Our design goals of the new ETL architecture were an open and flexible environment, extensible and reusable programming techniques with moderate formalization and decoupling of declarative knowledge from procedural parts of executable code.

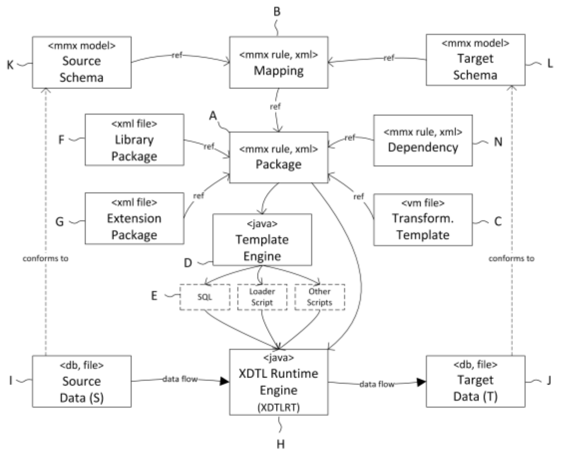

B
<mmx model> Source Schema
K
<mmx rule, xml> Mapping
<mmx model> Target Schema
L
ref
A
<xml file> Library Package
F
<mmx rule, xml> Package
<mmx rule, xml> Dependency
N
<xml file> Extension Package
G
<java> Template Engine
D
<vm file> Transform. Template
C
conforms to
E
SQL
Loader Script
Other Scripts
<db, file> Source Data (S)
I
data flow
<java> XDTL Runtime Engine (XDTLRT)
H
<db, file> Target Data (T)
J

**Fig. 1.** System architecture components

The general system architecture with main building blocks is drawn in Figure 1. The main system components are ETL Package (A) that can be written in XDTL language or represented as Tasks and Rules and Dependencies (N) in MMX repository. The mapping (B) is a formalized representation of source schema objects (K), target (L), column transformations and patterns, join and filter conditions that can be again be a part of an XDTL package or stored in MMX repository. The transformation Template (C) is a reusable and repeating part of SQL query patterns or some other scripting executable language scenarios that are written in the Apache Velocity macro language (or any other language of template engine). The template Engine (D) is a configurable java code that is responsible for runtime code construction using Mappings (B) and Templates (C). Examples of mappings and templates are discussed in more detail level in Chapter 4. The tasks in Package (A) can be created using previously prepared Library Packages (F) or managed modularly, reused and published as Extension (G) modules. The XDTL Runtime Engine (H) is a preconfigured environment and java package, able to interpret packages written in the XDTL language, execute those and deliver actual data transformations from the source (I) to the target (J).

## 3.1 Data Transformation Language (XDTL)

The Extensible Data Transformation Language (XDTL) is an XML based descriptive language designed for specifying data transformations between different data formats, locations and storage mechanisms. XDTL is created as a Domain Specific Language (DSL) for the ETL domain and is designed by focusing on the following principles: modular and extensible, re-usable, decoupled declarative (unique) and procedural (repeated) patterns. The XDTL syntax is defined in an XML Schema document. The wildcard elements of an XML Schema enable extending the syntax of core language with a new functionality implemented in other programming languages or in XDTL itself. The XDTL scripts are built as reusable components with the clearly defined interfaces via parameter sets. The components can be serialized and deserilized between the XML and database representations, thus making XDTL scripts suitable for storing and managing in a data repository. XDTL provides the functionality to use data mappings stored independently of the scripts, being efficiently decoupled from the scripts. Therefore the mappings stored in a repository can exist as objects independent from the transformation process and be reused by several different processes. XDTL acts as a container for a process that often has to use facilities not present in XDTL itself (e.g. SQL, SAS language etc.).

## 3.2 Knowledge Repository Structure (MMX)

The MMX metadata framework is a general purpose integrative metadata repository built on the relational database technology for different knowledge management (KM) and rule-based analytical applications. The MMX repository is designed according to the OMG Metadata Object Facility (MOF) idea with separate abstraction and modeling layers (M0-M3). The MMX physical data model (schema) is based on principles and guidelines of EAV (Entity-Attribute-Value) or EAV/CR (Entity-Attribute-Value with Classes and Relationships) modeling technique suitable for modeling highly heterogeneous data with very dynamic nature. The metadata model and schema definition in EAV is separated from physical storage and therefore it is easy to modifications to schema on 'data' without changing the DB structures: by just modifying the corresponding metadata. The approach chosen is suitable for open-schema implementations (similar to key-value stores) where the model is dynamic and semantics is applied in query time, as well as model-driven implementations with a formal, well defined schema, structure and semantics.

The MMX physical schema (Figure 2) provides a storage mechanism for various knowledge- or meta-models (M2) and corresponding data or metadata (M1). Three physical tables - object, property and relation - follow the subject-predicate-object or object-property-value representation schemes, where object_type, property_type and relation_type tables are like advanced coding or dictionary tables for object, property and value types. The separate dictionary tables give us an advanced schema representation functionality using special attributes and relational database foreign keys (FK) mechanism. The formalized schema description and relations between different schemas make our metadata understandable and exchangeable between the other system components or external agents. The URI reference mechanism used and the resource storage schema makes an MMX repository a semantic data store, comparable to Resource Description Framework (RDF), serializable in different semantic formats or notations (e.g. RDF/XML , N3 , N-Triples , XMI  etc.) using XML or RDF APIs.

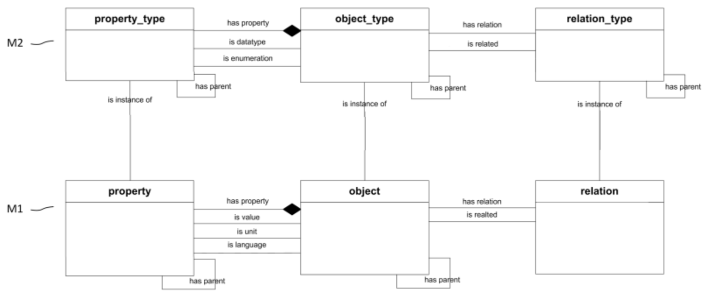


**Fig. 2.** MMX physical schema design

The MMX physical schema can be seen as a general-purpose, multi-level and hierarchical storage mechanism for different knowledge models, but also as communication medium or information integration and exchange platform for different software agents or applications (e.g. metadata scanners, metadata consumers etc.). Built in limited reasoning capability based on recursive SQL technique and it is captured to data and metadata APIs to implement inheritance and model validation functions. Semantic representation of data allows extend functionality with predicate calculus or apply other external rule-based reasoners (e.g. Jena) for more complicated reasoning tasks, like deduction of new knowledge.

Repository contains integrated object level security mechanism and different data access APIs (e.g. data API, metadata API, XML API, RDF API etc.) that are implemented as relational database procedures or functions. We have live implementations on PostgeSql, Oracle and MsSql platforms and differences in database SQL dialects and functionality are hidden and captured into API packages. Using common and documented API-s or an Object Relational Mapper (ORM) technology (e.g. Hibernate) we can choose and change repository DB technology without touching related applications.

An arbitrary number of different data models can exist inside MMX Metadata Model simultaneously with relationships between them. Each of these data models constitutes a hierarchy of classes where the hierarchy might denote an instance relationship, a whole-part relationship or some other form of generic relationship between hierarchy members. We have several predefined metadata models in MMX repository, e.g.

- terminology (ontology) and classification (based on ISO/IEC11179 [18]);
- relational database (based on Eclipse SQL Model [19]);
- abstract mappings and general ETL models;
- role-based access control model (based on NIST RBAC [20]).

In addition to existing models we can implement any other type of data mode for specific needs, like business process management (business rules, mappings, transformations, computational methods); data processing events (schedule, batch and task); data demographics, statistics and quality measures, etc.

The purpose of MMX repository depends on system configuration and desired functionality. In current paper we handle MMX repository as persistent storage mechanism for ETL metadata and we discuss about relational database and abstracted mapping knowledge models (KM) to store required data and relations. In addition to repository storage and access technologies MMX Framework has web-based navigation and administration tools, semantic-wiki like content management application and different scanner agents written in XDTL (e.g. DB dictionary scanner) to feel and detect surrounding environment and context. Due to space limitation we do not discuss all those topics in this paper.

## 4 Template Based SQL Construction

SQL is and probably remains the main workforce behind any ETL (and especially ELT flavor of ETL) tool. Automating SQL generation has arguably always been the biggest obstacle in building an ideal ETL tool (i.e. completely metadata-driven), with small foot-print, multiple platform support on single code base. While SQL stands for Structured Query Language, ironically the language itself is not too well 'structured', and the abundance of vendor dialects and extensions does not help either. Attempts to build an SQL generator supporting full feature list of SQL language have generally fallen into one of the two camps: one of them trying to create a graphical click-and-pick interface that would encompass the syntax of every single SQL construct, another one designing an even more high-level language or model to describe SQL itself, a kind of meta-SQL. The first approach would usually be limited to simple SQL statements, be appropriate mostly for SELECT statements only and struggle with UPDATEs and INSERTs, and be limited to a single vendor dialect.

### 4.1 Mappings, Patterns and Templates

Based on our experience we have extracted a set of SQL 'patterns' common to practical ETL (ELT) tasks. The patterns are converted into templates for processing by a template engine (e.g. Apache Velocity), each one realizing a separate SQL fragment, a full SQL statement or a complete sequence of commands implementing a complex process. Template engine merges patterns and mappings into executable SQL statements so instead of going as deep as full decomposition we only separate and extract mappings (structure) and template (process) parts of SQL. This limits us to only a set of predefined templates, but anyone can add new or customize the existing ones. Templates are generic and can be used with multiple different mappings/data structures. The mappings are generic as well and can be used in multiple different patterns/templates. Template engine instantiates mappings and templates to create executable SQL code which brings us closer to OO mind-set. The number of tables joined, the number of columns selected, the number of WHERE conditions etc. is arbitrary and is affected by and driven by the contents of the mappings only, i.e. well-designed templates are transparent to the level of complexity of the mappings. The same template would produce quite different SQL statements driven by minor changes in mappings. We have built a series of template libraries to capture the syntax of basic SQL constructs that are used to build complex statements. XDTL Basic SQL Template Library is a set of Apache Velocity templates that implements 'atomic' SQL constructs (INSERT, SELECT, UPDATE, FROM, WHERE etc.) as a series of Velocity macros. Each macro is built to expand into a single SQL construct utilizing the mappings in the form of predefined collections (targets, sources, columns, conditions). On top of Basic SQL library one or more higher-level layers can be built to realize more specific or more complex concepts, e.g. loading patterns, scenarios or process flows, as well as specifics of various SQL dialects.

It appears that, by use of Abstract Syntax Patterns, the same principle of reducing a disparate and seemingly diffuse set of all possible transformations in SQL statements to a limited set of patterns applies here as well. Abstract Syntax Pattern (ASP) is a reappearing code fragment that, similarly to Abstract Syntax Tree, has all the references to concrete data items removed. Thus, mappings between different data domains can be reduced to ASPs to be later processed synchronously with the process template by the same template engine turning them into executable code. Identifying and building a library of common syntax patterns enables creation of a user interface to generate a focused (limited) set of SQL statements without coding or even automatic SQL generation, validation of existing SQL statements [10]. More detailed ASP discovery case study can be found on chapter 5.

Various ETL metamodels discussed in previous works of [15],[16] and [17], but we decided to use pragmatic approach with ASP idea instead of complex and expressive modeling. We had modeling goals like: minimum footprint and complexity, effective code generation for different languages (e.g. SQL, SAS, R etc.), efficient storage and serialization, decomposition to the level where interesting parts would be identified and exposed with clear semantics (i.e. database objects, vendor specific terms and keywords, generic and reusable expressions etc.). In Figure 3 we have implemented mappings knowledge model with four basic classes which are designed as derivation type of rules in MMX repository. Mapping Group (B) is collection of Mappings (C) which used on multiple mapping cascade definition that produces single SQL statement with sub-queries or temporary table implementation. Each Transformation class (D) instance represents one column transformation in SQL select, insert-select or update statement with required source, target and pattern attribute definitions. Condition class (E) represents join and filter predicate conditions that required constructing data set from defined source variables (tables).

Mapping model (Figure 3) implementation in MMX physical schema (Figure 2) is straightforward transformation where each class implemented as one row in object_type table, each class attribute implemented as one row in property_type table, and each association implemented as one row in relation_type table. Instances of mapping model stored as corresponding rows in object, property and relation tables.

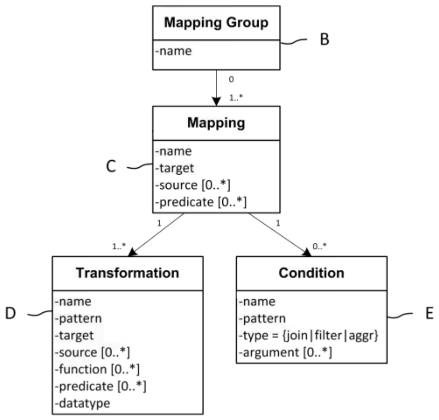


**Fig. 3.** Knowledge model (schema) for mappings representation and storage

Described method and model for constructing SQL statements complies with the following criteria:

- construction of all significant DML statements (INSERT, UPDATE, DELETE) based on a single mapping;
- construction of SQL statements from a single mapping for several different SQL dialects;
- construction of SQL statements covering different loading scenarios and performance considerations;
- construction of SQL statements based on multiple mappings (mapping groups);
- minimum footprint and complexity of the processing environment.

The next chapter example gives the basic idea of one mapping implementation and usage scenarios for SQL code generation in ETL process.

## Mapping Example

Following simple and generic insert-select SQL statement represents one very basic transformation code for everyday data transformation inside DW from multiple staging tables to target table.

*Example 1.* Insert-select SQL DML statement:

```
INSERT INTO person t0 (id, name, sex, age)
SELECT t1.cust_id,
       t1.firstname || ‘ ‘ || t1.lastname,
       DECODE(t1.sex, ‘M’, 1, ‘N’, 2),
       ssn_to_age(t2.ssn)
FROM customer t1
JOIN document t2 ON t2.cust_id = t1.cust_id
WHERE t1.cust_id IS NOT NULL;
```

Same query contains declarative mapping part formalized and represented by following objects and collections in MMX repository tables:

**Table 1.** Mapping objects source and target properties

| Target | isVirtual | Source | isQuery |
|---|---|---|---|
| t0:person | f (false) | t1:customer | f (false) |
| | | t2:document | f (false) |

**Table 2.** Column Transformation object(s) properties and values

| Pattern | Target | Source | Function | Key | Upd |
|---|---|---|---|---|---|
| %c1 | %c0:t0.id | %c1:t1.cust_id | null | t | f |
| %c1\|\|' '\|\|%c2 | %c0:t0.name | %c1:t1.firstname; %c2:t1.lastname | null | f | t |
| decode(%c1,'M',1,'F',2) | %c0:t0.sex | %c1:t1.sex | null | f | f |
| %f1(%c1) | %c0:t0.age | %c1:t2.ssn | %f1:ssn_to_age | f | t |

**Table 3.** Condition objects properties(s) and their values

| Pattern | Source | Function | Condition Type | Join Type |
|---|---|---|---|---|
| %a2 = %a1 | %a2:t2.cust_id;<br>%a1:t1.cust_id | null | join | inner |
| %a1 IS NOT NULL | %a1:t1.cust_id | null | filter | null |

Simple insert-select template produces initial SQL statement from given mapping (Table 4). When using the same mapping with the different template(s) we can generate update statement or series of different statements.

**Table 4.** Insert-Select template and result query

| Template | SQL Statement |
|---|---|
| `#foreach($tgt in $Targets)` | `INSERT INTO person (id, name, sex, age)` |
| `#if("$tgtmap" == "0")` | `SELECT` |
| `#INSERT($tgt $Columns)` | `  t1.cust_id` |
| `#SELECT($Columns)` | `, t1.firstname \|\| ' ' \|\| t1.lastname` |
| `#FROM($Sources $Conditions)` | `, DECODE(t1.sex, 'M', 1, 'F', 2)` |
| `#WHERE($Conditions)` | `, ssn_to_age(t2.ssn)` |
| `#GROUPBY($Columns $Conditions)` | `FROM customer t1` |
| `#HAVING($Conditions)` | `INNER JOIN document t2` |
| `#end` | `ON t2.cust_id = t1.cust_id` |
| `#end;` | `WHERE t1.cust_id IS NOT NULL;` |

The described example gives very basic idea and method how to formalize column mappings with reusable patterns and how to construct source and target data sets applying join and filter conditions that described again with reusable expression patterns. Using one single mapping together with limited number of scenario templates (e.g. full load, initial load, incremental load, insert only, 'upsert' (with or without deletion), versioned insert (history tables), slowly changing dimensions etc.) we can generate long SQL statement batches, that are adapted for specific dialect (when needed), validated, robust and working with expected performance. The main idea here is to formulate and describe as less as possible and reuse and generate as much as possible.

By using described set of methods we have effectively decomposed SQL statement into mappings and patterns. The same method can be applied to any SQL data manipulation statement (e.g. insert, update and delete) of reasonable complexity and we have used same approach, same mappings and different templates to generate code for different execution engines (e.g. SAS script). More expressive examples can be presented when to take transformations with more than 5-10 source tables and targets with 20-100 columns (common in analytical DW environment) then propositions of generated, defined and reused code parts change dramatically.

To conclude the mapping example we can say that presented methodical approach allows us to:

- migrate and translate vendor-specific (SQL) pattern dialects between different platforms or use same mapping code with different transformation scenarios or generate code for different execution engines;
- construct data transformation flows from sources to targets with column level transformation semantics for Impact Analysis and Audit Trail applications;
- construct system component dependency graphs for better management and automation of development and operation processes;
- automate change management and deployment of new functionality between different environments (e.g. development, test, production).

## 5 Experimental Abstract Syntax Pattern Case Study

Abstract Syntax Pattern (ASP) is the practical idea to narrow down expressiveness of SQL Data Manipulation Language (DML) to allow formalized descriptions of reusable patterns, decoupled data structures and functions. Decomposition of patterns and data structure instances are the central idea of the XDTL environment and the code construction capability, which gives additional flexibility and ergonomics in data transformation design and allows impact analysis capability for maintenance of complex Data Warehouse environment (described in chapter 4). We used existing SQL statements corpus (used for real life data transformations) containing about 26 thousand SQL statements to find hard evidences for existing patterns and we narrowed down the used corpus to 12 thousand DML statement to find specific column, join and where expressions.

We used open source GoldParser[3] library and developed our own custom SQL grammar in EBNF format for SQL text corpus parsing. We developed custom parser program for ASP pattern extraction from SQL corpus, we imported all parsed patterns to database and evaluated and analyzed SQL patterns and "life forms" writing new SQL queries. Implemented parsing program is tuned to recognize column construction patterns, join, where and having condition predicate patterns from SQL DML statements, replacing specific database structure identifiers and constants with `%a` and functions with `%f` pattern.

[3] `www.goldparser.org`

*Example 3.* The parsing program detects pattern `%f(%a,%a)` from the original column expression `COALESCE(Table1.Column1,0)` and assigns operator and operand values to replaced variables: `%f = {'COALESCE'}` and `%a = {'Table1.Column1','0'}`

Very general metrics about SQL parsing work can be described with total figures of 8,380 input SQL DML statements (select, insert, update) and 92,347 parsed expressions that group to 2,671 abstract patterns. It gives us 97/3 percentage division between expressions and patterns. Those figures can be improved by hand tuning of pattern detection technique and SQL grammar that is not currently covering all the aspects of used SQL dialect.

**Table 5.** Insert-Select template and result query

| Statement Type | Patterns Count | Statements Count | Expression Count | Expressions in Top Pattern | Top Pattern Coverage % |
|---|---|---|---|---|---|
| Insert | 1 890 | 4 965 | 61 899 | 37 924 | 61 |
| Update | 836 | 2 240 | 25 361 | 12 997 | 51 |
| Select | 224 | 1 175 | 5 087 | 3 175 | 62 |
| All | 2 950 | 8 380 | 92 347 | 54 096 | 59 |

**Table 6.** Discovered patterns by pattern types

| Pattern Type | Pattern Count | Statement Count | Expression Count | Expressions in Top Pattern | Top Patten Coverage % |
|---|---|---|---|---|---|
| Column | 1 897 | 10 631 | 78 264 | 55 921 | 71 |
| Join | 526 | 4 172 | 11 684 | 2 273 | 19 |
| Filter | 323 | 1 505 | 2 399 | 359 | 14 |

Based on current results we can conclude that top 10 patterns will cover 83% and top 100 patterns will cover 93% of all expressions that used in SQL DML corpus. Those metrics does not count the fact that most of the patterns that are not in top list are constructed from patterns that are in patterns top list.

To conclude this case-study we can say that actual expressiveness of formalized patterns and mappings will be comparable with expressiveness of real life usage of SQL DML in data transformations. A small set of meaningful patterns (about 100 different patterns) with defined semantics and experimental impact weights will direct us to automated and probabilistic impact analysis calculations that are one of the main applications for SQL formalization technique. We also got the confirmation that SQL parsing technique can be used for data transformation extraction, mapping formalization and future analysis.

## 6 Case Studies for Automating Data Lineage Analysis

The previously described architecture and algorithms form a basis for an integrated data lineage analysis toolset dLineage (http://dlineage.com). dLineage has been tested in large real-life projects and environments supporting several popular DW database platforms (e.g. Oracle, Greenplum, Teradata, Vertica, PostgreSQL, MsSQL, Sybase) and BI tools (e.g. SAP Business Objects, Microstrategy).

We have conducted two main case studies involving a thorough analysis of large international companies in the financial and the energy sectors. Both case studies involved an automated analysis of thousands of database tables and views, tens of thousands of data loading scripts and BI reports. Those figures are far over the capacity limits of human analysts not assisted by the special tools and technologies. The automation tools described in the paper enabled us to set up and conduct the analysis project in a few days by just two developers.

The following example graph from the case study maps DW tables to views and user reports: it is generated automatically from about 5 000 nodes (tables, views, reports) and 20 000 links (data transformations mappings form views and queries).

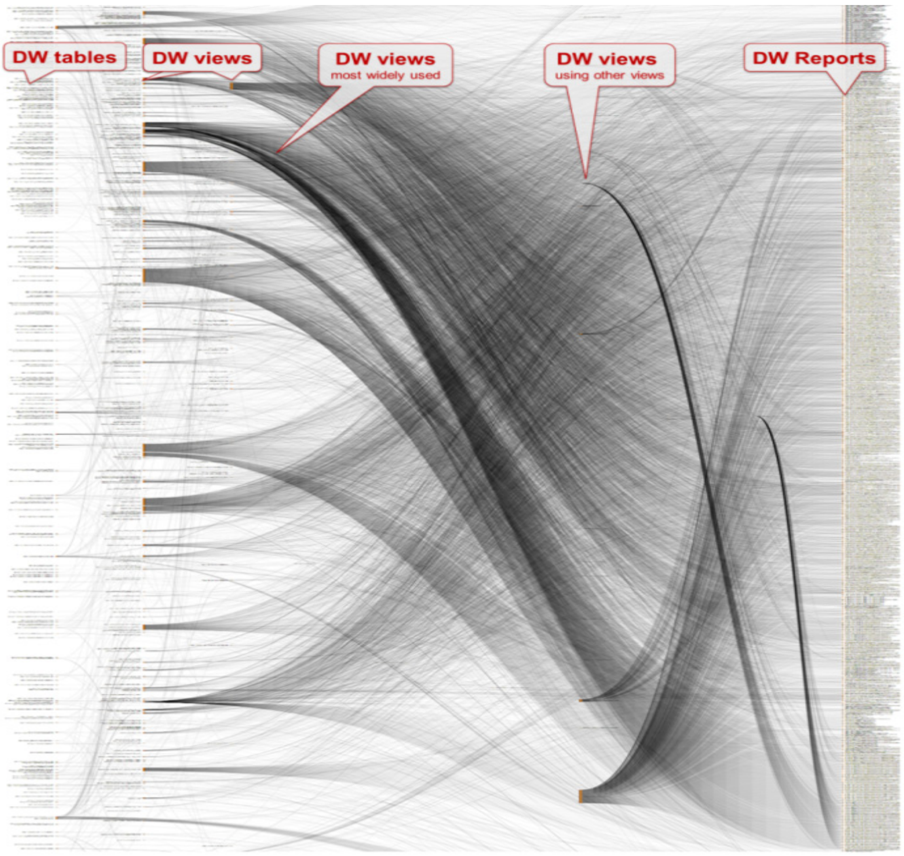


**Fig. 4.** Data lineage graph with dependencies between DW tables, views and reports

## 7 Conclusions and Future Work

We have presented a formalized mapping and abstract pattern methodology supporting template-based program construction. The technique is a development upon the ETL language runtime environment (XDTL) and metadata repository (MMX) designed earlier by the authors. We have introduced the technology and presented working samples motivated by real-life challenges and problems discussed in the first chapter. The described architecture and mapping concept have been used to implement an integrated toolset dLineage (http://dlineage.com) to solve data integration and dataflow visualization problems.

We have used our metadata-based ETL technology in the Department of Statistics of Estonian state to implement a system for automated, data-driven statistics production for the whole country. We have also tested our mapping methods and technology for data flow analysis and visualization in large international companies in the financial and the energy sectors. Both case studies contained thousands of database tables and views along with tens of thousands of data loading scripts and BI reports. The analysis of the large SQL data transformation corpus (see chapter 5) gave us taxonomy of reusable transformation patterns and demonstrated the two-way methodology approach from code to mappings and patterns.

The future work involves refining current implementation details, adding semantics to mappings and patterns, constructing dependency graphs of mappings, data structures and data flows and developing aggregation algorithms for different personalized user profiles and their interests (e.g. business user interest in data structures, flows and availability is different from that of a developer or system operator) as well as using those techniques for solving problems described in the first chapter.

**Acknowledgments.** This research has been supported by European Union through European Regional Development Fund.

## References

[1] Behrend, A., Jörg, T.: Optimized Incremental ETL Jobs for Maintaining Data Warehouses (2010)

[2] Boehm, M., Habich, D., Lehner, W., Wloka, U.: GCIP: Exploiting the Generation and Optimization of Integration Processes (2009)

[3] Böhm, M., Habich, D., Lehner, W., Wloka, U.: Model-driven generation and optimization of complex integration processes. In: ICEIS (2008)

[4] Dessloch, S., Hernández, M.A., Wisnesky, R., Radwan, A., Zhou, J.: Orchid: Integrating Schema Mapping and ETL. In: IEEE 24th International Conference on Data Engineering (2008)

[5] Giorgini, P., Rizzi, S., Garzetti, M.: GRAnD: A Goal-Oriented Approach to Requirement Analysis in Data Warehouses. DSS 45(1), 4–21 (2008)

[6] Haas, L.M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio Grows Up: From Research Prototype to Industrial Tool. In: SIGMOD, pp. 805–810 (2005)

[7] Jun, T., Kai, C., Yu, F., Gang, T.: The Research & Application of ETL Tool in Business Intelligence Project, International Forum on Information Technology and Applications. In: FITA 2009, pp. 620–623 (2009)
[8] Papastefanatos, G., Vassiliadis, P., Simitsis, A., Sellis, T., Vassiliou, Y.: Rule-based Management of Schema Changes at ETL sources. In: Grundspenkis, J., Kirikova, M., Manolopoulos, Y., Novickis, L. (eds.) ADBIS 2009. LNCS, vol. 5968, pp. 55–62. Springer, Heidelberg (2010)
[9] Patil, P.S., Rao, S., Patil, S.B.: Data Integration Problem of structural and semantic heterogeneity: Data Warehousing Framework models for the optimization of the ETL processes (2011)
[10] Reiss, S.P.: Finding Unusual Code. In: 2007 IEEE International Conference on Software Maintenance, pp. 34–43 (2007)
[11] Rodiç, J., Baranoviç, M.: Generating Data Quality Rules and Integration into ETL Process (2009)
[12] Roth, M., Hernández, M.A., Coulthard, P., Yan, L., Popa, L., Ho, H.C.T., Salter, C.C.: XML mapping technology: Making connections in an XML-centric world. IBM Systems Journal (2006)
[13] Simitsis, A., Vassiliadis, P., Sellis, T.K.: Optimizing ETL Processes in Data Warehouses. In: ICDE, pp. 564–575 (2005)
[14] Simitsis, A., Wilkinson, K., Dayal, U., Castellanos, M.: Optimizing ETL workflows for fault-tolerance. In: International Conference on Data Engineering (ICDE), pp. 385–396 (2010)
[15] Song, X., Yan, X., Yang, L.: Design ETL Metamodel Based on UML Profile, Knowledge Acquisition and Modeling. In: KAM 2009, pp. 69–72 (2009)
[16] Stöhr, T., Müller, R., Rahm, E.: An Integrative and Uniform Model for Metadata Management in Data Warehousing Environment. In: Workshop on Design and Management of Data Warehouses (DMDW) (1999)
[17] Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M.: A Framework for the Design of ETL Scenarios. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, Springer, Heidelberg (2003)
[18] ISO/IEC 11179 Metadata Registry (MDR) standard, `http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35343`
[19] Eclipse DB Definition Model, `http://www.eclipse.org/webtools/wst/components/rdb/WebPublishedDBDefinitionModel/DBDefinition.htm`
[20] NIST Role Based Access Control (RBAC) Standard, `http://csrc.nist.gov/groups/SNS/rbac`

# Part III
# Issues of Information Systems

# Secure Data Storage and Exchange with a Private Wallet

Oliver Jäger[1], Frank Kramer[2], and Bernhard Thalheim[2]

[1] NTT DATA Deutschland GmbH, Hammerbrookstrasse 89, 20097 Hamburg, Germany
[2] Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany

**Abstract.** Sharing private data between various users is a daily scene. The owner has both rights at the same time, the right of a self-determined and the right of a policed usage of his private data by foreign holders. We develop an approach that allows an owner to store and share his private data and gain full control about the usage of his private data. Therefore, we develop and implement a flexible system that uses a peer-to-peer architecture and modern encryption standards to realize data privacy. We call this system *private wallet*.

**Keywords:** privacy, communication, security, data storage, data management.

## 1 Introduction

The self-determination of privacy is a highly-valued asset and must be protected. The Universal Declaration of Human Rights of the United Nations says in article 12: *No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks* [1]. Also, other national laws see the self-determination of privacy as a high priority. For example, the Basic Law for the Federal Republic of Germany, the German constitution, defines in article 2 *that every person has the right to free development of his personality insofar as he does not violate the rights of others or offend against the constitutional order or the moral law* [2]. Today, the web makes it possible, to get information about a person very fast. Hence, the self-determination of personal data must be taken under the consideration of privacy. For example, the scientific article [3] shows what happens, if we abandon our data privacy. It describes, how easy it is for somebody to get information about a person only by scanning an image of this person.

To evolve a system to secure private data, it is important to distinguish between the owner and the holder of data. A person is owner of data, if he has created this data. Therefore, an owner can read, change and delete his data at will. Furthermore, only with his permission other persons can use his data and the owner is informed about how his data is used and changed from other persons. A person who uses the data of an owner is a holder of data. He can only use the data with the permission the owner has given him for the data. Moreover, the owner must inform the holder, if he has changed something on his data. Hence, a system is needed that allows an owner permanent control of his private data against a holder. The control of the private data covers, on the one hand, the encryption of the data, and on the other hand, the best possible logging of all usage of the data.

This paper presents an approach to reach such a data control. Therefore, we develop and implement a concept called *private wallet*. Every owner gets his own wallet to store his private data inside. To secure the data in the wallet, all data is only stored encrypted and solely the owner knows the key. Furthermore, the *private wallets* are connected within a network. The owner of the data can exchange his data with other users within the network. For every exchanged data the owner defines the permissions a holder can get. Additionally, all usage of the holder is logged for the owner. Consequently, an owner is informed permanently which holder uses his data in what way. In our approach, section 2 will first present the functionality of our *private wallet*. We will present our six main processes that are realized in our approach. Section 3 will then describe, how we realize our privacy wallet. This covers solutions for ground functionality, user management, encryption and data storage. In section 4, related work will be presented. We will take a look on other solutions that exist to secure private data. Finally, a conclusion and a short outlook on future research will be given in section 5.

## 2 The Privacy Wallet Functionality

This section presents the functionality of our *private wallet* approach. There are six different business use cases for the *private wallet*. Each business use case can be regarded as an independent process. All transfers between pares within a process are asynchronous transfers. We will describe each business use case. All these use cases can be modelled with a business process modelling language, for example BPMN 2.0[4]. Due to the limitations of this paper, we present only one process model as an example. We refer to the Bachelor thesis of [5] that presents the other process models too.

### 2.1 Request-a-Key

The first functionality we want to present is *Request-a-Key*. With *Request-a-Key*, the key for decrypting a specific document is requested by a user to the system of the key owner. Figure 1 presents this use case as BPMN model.

In the owner's system, a check is carried out in the keystore for whether the specific document and the specific user, who request the key, exist. If no key exists in the keystore, the fact that there was an incorrect request from this user for this document is logged. Reasons could be that the user has been denied rights to the document or the user received this document without authorization.

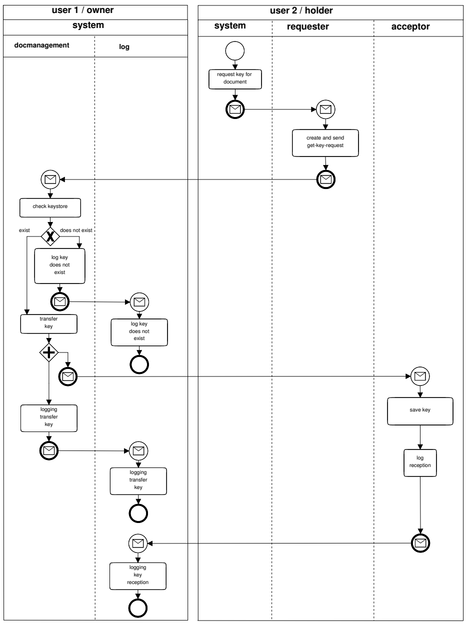


**Fig. 1.** BPMN request-a-key

If a key exists in the keystore, this key is transferred and the transfer of the key is logged in the system of the owner. The user's wallet receives the key and stores the key to the document. The reception of the key is confirmed. The wallet of the owner stores the receipt confirmation in a log.

## 2.2 Request-a-Document

With *Request-a-Document*, a particular document in the owner's system is requested. First, a connection is established to the desired participant. This connects both *private wallets.* The requester identifies itself on the basis of authorization features. The *private wallet* receiving the request now shows the requesting *private wallet* all documents to which the requesting *private wallet* is permitted to have access. It does not matter whether the requesting *private wallet* has full access or read-only access to the document. The requester can now select from this document list the documents that are to be transferred into his own *private wallet.* These documents are then sent by the owner to the requester. Each receipt must be confirmed. This receipt confirmation is then stored by the wallet which sent the documents.

### 2.3 Response-a-Document

With *Response-a-Document*, a particular document is sent to a particular person from the owner's system. To do this, the owner chooses the document to be sent in his own wallet. The document is encrypted for the selected person and sent to him. The key is also stored in the local keystore of the owner. The document sent is automatically written to the wallet of the recipient. Once this has happened, the document is displayed in the document list and a receipt confirmation is sent to the sender. The sender, that is, the owner of the document, receives this receipt confirmation and the system stores it automatically.

### 2.4 Synchronize

With *Synchronize*, updated data are sent to other users. These data cover new files, people, person objects, or altered document objects. Therefore, the wallet of the user that has changed the data starts the synchronization. The other wallets receive these changes and update the data in the wallet in the background on a fully automatic basis. Thereby all wallets only synchronize from the wallet that has started the synchronization. The user himself does not need to take any action. If a wallet cannot be reached, synchronization is repeated later. Thus, the synchronizing wallet looks in a periodical time, if the missed wallet is online, and then sends the changes. This ensures that each participant of the group receives all updates needed.

### 2.5 Ask-for-Extended-Rights

With *Ask-for-Extended-Rights*, a holder of a document requests from the owner an extension of the rights to this document. In our case, extended rights mean an extension of existing rights. For example, if a holder has only read rights on a document, he can ask for edit rights for this document. The owner must actively decide whether to grant the requested rights or not. If the owner does not grant the requested rights, the request is logged with the negative decision. If the owner grants the requested rights, the document is re-encrypted and transferred to the holder. In addition, the file transfer is logged and the new key is stored in the keystore. The wallet of the holder automatically receives the file and stores it in the filestore. In the process, the old file is overwritten and the old key, if it exists, is erased. After successful receipt of the document, a receipt confirmation is sent to the owner. The wallet of the owner saves the receipt confirmation in the form of a log.

### 2.6 Request-a-Person

If a new user joins a group, it is necessary that all members of the group know this new user. In the peer-to-peer network, the new user is known in the form of his email address if he is online, but other information is missing, in particular the public key. If a new user now participates in the group, his complete data is automatically distributed by the system to all participants who are online. To reduce administrative effort, that is, the management of the information regarding which users have been informed and which have not, the principle of an obligation to provide information is replaced by an obligation to retrieve information in this case. If a *private wallet* sees that an 'unknown' user is in the group, that is, in the group online, then this *private wallet* automatically requests the data of the 'unknown' user from the *private wallet* in the background. This data is then sent back to the requesting *private wallet* on a fully automatic basis.

In contrast to the other stories, a check for whether the data actually arrived is not performed with this process. This is not necessary, because no security and control mechanisms are disrupted or undermined if the data is not present. Sending of the documents can only take place if the personal data is locally available. If problems occur with the transmission of personal data, the wallet simply tries again. To make sure that there was really a receipt confirmation in the operations *Request-a-Document, Response-a-Document, Request-a-Key* and *Ask-for-Extended-Rights*, a check is carried out. After an appropriate time, the system checks whether there has been a receipt confirmation. If this is the case, then everything is fine. If this is not the case, this document is blocked for the receiver. This happens when a delete command is sent to the receiver wallet and at the same time the entry is removed from the keystore. This process is logged.

After presenting the functionality, the next section will describe the concrete realization of the *private wallet.*

## 3 Privacy Wallet Realization

This section describes the realization of the *private wallet.* The *private wallet* serves as an intelligent interface. This interface can be placed over a document management system (DMS), for example, but it can also work on a standalone basis. This makes it possible to use it in a very flexible manner. A set of wallets communicate in a peer-to-peer network. The whole communication is secured with a hybrid encryption approach and an additional usage of steganographic functions. Figure 2 shows this general structure of such a *private wallet* peer-to-peer network.

The next sections will take a closer look into the construction of such a *private wallet.* To see the implementation, we refer again to the Bachelor thesis of [5].

### 3.1 Modules and Task Areas

The implementation of the *private wallet* is modular. This allows easy replacement of individual components, such as another encryption or another communication framework. The following basic functional components are implemented for the *private wallet.*

**Importing Documents.** The *private wallet* is a closed system. Only within this closed system can the built-in safety mechanisms be effective. Therefore,

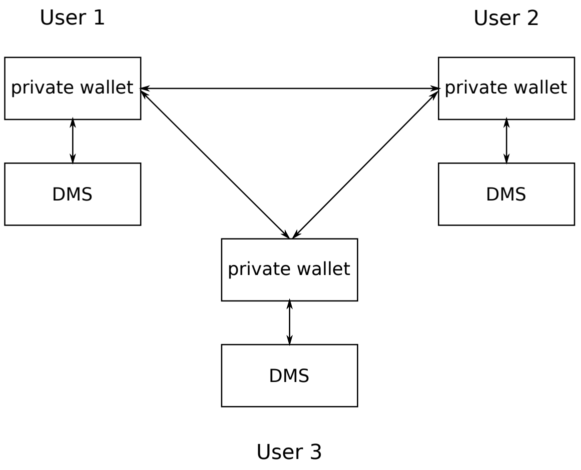


**Fig. 2.** Peer-To-Peer *private wallet* Model

a function must be provided that allows you to import data into this system. For this purpose, the *private wallet* makes the *Import* function available. Data such as document ID, creation date, and author are generated by the system and automatically added. Other data must be entered by the user. The file is encrypted immediately and stored in the filestore. The associated metadata is stored in the database. From this point on, the document is available in the *private wallet*.

**Communication.** A major component of the *private wallet* is exchange among the participants. In this approach, the decision was made in favour of a peer-to-peer network. In [6] a peer-to-peer system is described as a self-organizing system that is composed of autonomous units called peers. All peers have equal rights in the system and all peers use their resources and services completely decentralized. This short description can be extended by a set of characteristics of a peer-to-peer network as described in [7]. An ideal system has all these characteristics. But in most cases only a subset of these characteristics can be found in a peer-to-peer system. Using a peer-to-peer system brings some advantages over a client-server system. Through the omission of a server, the vulnerability of the entire system is reduced. Another advantage of a peer-to-peer network is simple distribution of information. It offers the possibility of sending a message to individual members of the group or to all members at the same time. Communication is always encrypted. This ensures that even if an outsider infiltrates the group, the information is secure.

**Logging.** Any action on or with a document triggers notification of the owner and author. To ensure this, the information about such an action is kept in a log object and sent to the *private wallet* of the owner and author. The owner and author stores this log object in his overall log. If one of the two participating *private wallets* is offline or unreachable, this log object is stored in the sender wallet and marked as not sent. As long as the sender wallet is turned on, the system attempts again and again to transfer non-transferred log objects at appropriate intervals until a transfer is successful. A log entry contains the document ID in your own *private wallet*, timestamps of when an action took place and when this log entry reached your own *private wallet*, the type of use of the document, the status, how many times this action was tried, and the user and his IP address. But our logging is limited. If a holder copies a document or takes a screenshot, we have no logging about what he is doing with the document any more. So at this point, we lose the control over an owners document.

**Distributing.** The aim of the *private wallet* is to exchange data securely and to keep as much control as possible over the exchanged data. This makes it necessary to distribute data. This data are documents or document data, keys, and log entries.

### 3.2 User Management

The *private wallet* requires its own user management, because only if users are registered in the *private wallet* network, they can exchange data. Therefore, in this user management, it is defined for each user who he is, in what way to communicate with him, what type of encryption is used, and to which user group he belongs. Allocation to user groups offers the possibility of easily assigning keys and rights to documents to several users. In this way you have the option to automate certain approvals for the use of documents.

### 3.3 Encryption

As described in [8] there are two different approaches for secure communication, steganography or cryptography. In our implementation a hybrid of two types of encryption is selected for cryptography. Symmetrical encryption is chosen for encryption of the document. In this way, it is possible to benefit from the speed advantage over asymmetrical encryption. The downside in terms of the effort and lower security with the key exchange is balanced out by asymmetrical encryption of the key of the symmetrical document encryption. A 128-bit key with the AES cryptosystem [9] is used for document encryption. A 192-bit or 256-bit key is also possible. For the RSA encryption [10], 2048 bit is used. Adequate security is already provided with a smaller number of bits. Because RSA encryption is only used to encrypt the AES key, meaning the text to be encrypted is very small, there are no performance drawbacks, but much more security.

Furthermore, this *private wallet* offers the possibility of increasing the security of the encryption through steganography. The encrypted key is hidden in an image. Hence, an owner can hide the key by using a special function from the private wallet. A holder can only reveal the key from the image by using the same function in his private wallet. Due to the combination of RSA encryption and steganography, the security is classified as very high. The method used in this *private wallet* for steganography is very simple. The method can be exchanged by every other known steganography algorithm. Here, every image is composed of colour points or pixels. A pixel has a red value, a green value, and a blue value. These values are between 0 and 255. To hide a message in the image, in each pixel each colour channel is changed in the last bit. The decision whether the last bit is set to 1 or 0 depends on the message to be hidden. For this, the letters of the message are converted into the associated ASCII values. These ASCII values are converted into binary form. Take as an example that DB is hidden. The ASCII value for D is 68, and the value for B is 66. In binary terms, it is the values 01000100 for decimal 68 and 01000010 for decimal 66. With the binary values, it is necessary to ensure that the leading zeros (always eight digits) are taken into account. Taking into account that it is always only the last bit of a colour channel that is changed and a pixel has three colour channels, eight pixels are needed to hide two letters and the message end character (ASCII 0). The number of pixels required is given by the following formula:

$$\lceil numberofpixels \rceil = (numberofcharacters + 1) * 8/3$$

The bit sequence of the letter combination DB results in 0100010001000010, which is 16 bit values. Now the the bit values are entered in sequence into the most recent bit of a colour channel in each case. If the last bit of a colour channel is identical to the bit entered, no change takes place. In this example, the last bit of the first colour channel of the first pixel receives the value 0, the last bit of the second colour channel receives the value 1, and the last bit of the third colour channel receives the value 0. Since all three colour channels have now been used, the next pixel is used. This is repeated until all bits of the bit sequence and the message end character have been entered. Since only the last bit of a colour channel is manipulated, the overall colour of a pixel is changed by 1/256. This is only 0.39 percent. As a result, the colour change for each pixel is from 0 to 0.39 percent. On average the change is about 0.2 percent. This change is not discernible to the human eye. This means the message has become invisible.

## 3.4 Data Transfer

There are various possibilities for transferring the data between the *private wallets* within the network. Using TCP and UDP, data can be transferred from both computers and mobile devices such as smart phones or tablets. This works both in a local network and over the Internet. The advantage of this transfer option is that all systems have this possibility of use.

Transfer via Bluetooth or NFC is suitable only for mobile devices such as smart phones or tablets. However, this type of transfer offers great advantages in terms of security. Since NFC[1] or Bluetooth[2] only work over short to very short

[1] range: up to 10 cm.

[2] range: class 3 approx. 10 m outdoors, class 2 approx. 20 m outdoors, class 1 approx. 100 m outdoors.

distances, the probability that such a connection will be 'overheard' is close to 0, since it is immediately apparent if someone is hanging around in the vicinity.

Transfers over a local network or the Internet are suitable due to the high transfer rates both for the exchange of documents and for the exchange of keys. Networks have transfer rates up to 1 gigabit/s, and the Internet provides transfer rates of between 1 and 100 Mbit/s per second for private households.

Bluetooth, on the other hand, only has transfer rates of 57.6 kbit/s to 732.2 kbit/s. Transfer of documents is possible here, but the advantages of Bluetooth come with smaller data packets, such as a key, because the security is significantly higher due to the proximity to the exchange partner. The same applies to NFC. With NFC the data transfer rate is only up to 424 kbit/s.

## 3.5 Data Management

To store the data, every *private wallet* gets its own small database. Every database get the same database schema to store the data. Figure 3 shows this database schema as Higher-Order Entity-Relationship Model (HERM)[11].

The central element is the document. A document can have one owner and multiple holders. The owner is the person who owns the document. This is in most cases the author of the document. The holder is the person who has power or control over the document. A document can have each number of keys with the corresponding rights. Behind each key is a holder with the appropriate rights. For each document, there is metadata. Information is stored for every person who participates in the *private wallet*. The log contains any number of log entries. A log entry provides information about the affected document, the point in time of the activity, the actual process, the status (the action was successful or not successful), the number of attempts, and the IP address of the device on which this action was performed. In the *private wallet*, users can define the document types in any way required. This ensures a high degree of flexibility.

Throughout the application, work is carried out only on data objects that are already stored in the database. What made the decision for such an approach was the objective of not losing any data. This is particularly important for guaranteed logging of actions on documents. For some operations, it is necessary to update existing data in the database. Only the following personal data can be changed: first name, last name, the marker, whether it is a local user or a group member, the password for local login to the *private wallet*, the public and private key, the document data rights to a document, the document key, and the expiration date. All other data, for example the email or the salutation, may not be altered under any circumstances.

Data which is stored in the database is deleted only by setting a marker which indicates that it is a deleted data record. The only exception is temporary data. Temporary data in this *private wallet* consists of log entries that have not yet been transmitted to the author of the document. They are removed from the database. Deleting data records only logically offers the advantage that no data inconsistency can occur. For documents marked as deleted, the physical file is deleted in the file system.

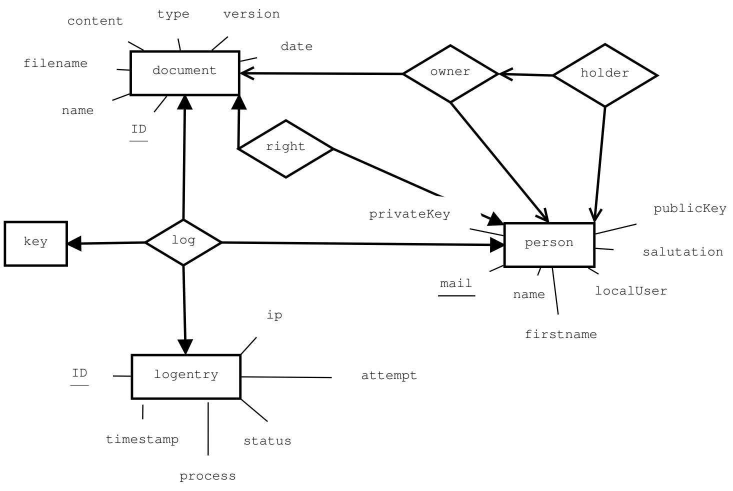


**Fig. 3.** HERM-Schema of the *private wallet*

## 4 Related Work

There exists a great set of implementations in the area of secure data storage and exchange. Here, we want to describe some of these implementations that track concepts similar to our *private wallet* approach. First, we look at cloud services like Dropbox[12] or Apple iCloud [13]. There, a user can store data on a foreign server and can share these data with other users of the cloud. The main problem is that all data that is saved in these cloud services are stored on an third party foreign server. At this point an owner has no full control over the usage of his data by this third party. He must trust in the service that there is no illegal access to the data by the third party. Furthermore, an owner is dependent on the security of the cloud service system. In our approach, only holders who have the owner's permission can get access to data. If an owner shares data, this data is transferred from the owner system to the holder system. Thus, no third party server system for storage and transfer is used. Additionally, in most of the cloud services our data is not stored encrypted within the cloud. We must install special programs for encryption before storing the data into the cloud. In our private wallet approach every document is encrypted automatically when it is send to a holder. Only if the owner gives the holder the key permission, the holder can decrypt and use a document.

Next, we take a look at an application that is close to our approach. The *Deutsche Post AG* provides a software called *DocWallet* [14]. This software can be used for secure personal document management. Therefore, a user can import documents into the *DocWallet* system. Within the system, the document is end-to-end encrypted with AES 256 and synchronized with every system where the *DocWallet* software is installed. Hence, a user has access to his documents on all his systems, including mobile devices like smart phones or tablets. All this is free of charge for the user. Additionally, one can pay for a premium account. Then the data within the *DocWallet* is synchronized with a cloud server in Germany that backups all imported files in the cloud. Unlike our *private wallet*, there is no data exchange between various *DocWallet* users. The system is only for single user data management.

Another mobile application for smart phones and tablets that can be used for a secure exchange of chat massages with attached videos or images between users is Snapchat [15]. Before a user can exchange a message with another one, he must define an expiration date for the message he wants to send. The receiver of the message can look at it on his device until the expiration date is reached. After this, the message is deleted from the receiver device. This should guarantee the securtiy of the videos and images that are send via Snapchat. But there are some problems with Snapchat. On the one hand, the receiver of a Snapchat message can take a screenshot of the received object. Then he has permanent access to the object without the control of the owner. On the other hand, user have to trust Snapchat that they fully delete the message. There are hints that this is not done by Snapchat [16]. Despite everything Snapchat is a first approach for a secure data exchange were the user has the control over how long another user has access to his data.

The last service we want to present is the data exchange service Mega [17]. Mega can be used to exchange data with other users through the internet. Therefore, a user can import any digital information to the Mega server. The information is end-to-end encrypted with AES 128 before it is stored on the server. Only the user gets the key to decode the data. Hence, no other user or Mega itself can decode the private data. If a user wants to share the information with another user, he has to exchange the key with this user, so that the other user can get access on the data too. How the user exchange the keys is not the task of Mega. Other than our *private wallet* the focus of the encryption of information is not the self-determination about the own data. The encryption of the data is a legal protection for Mega. Thus, they can assert that they never know what users send over their filesharing system.

## 5 Conclusion and Outlook

Exchanging data and protecting the privacy of data for an owner is not a contradiction. The approach outlined in this paper presents a system by which an owner of data can store and exchange his data with other users without loss of control. Therefore, we implement a system that is based on a peer-to-peer network to exchange data between users of this network. All exchanged data is encrypted with modern algorithms and only the owner of the data can decontrol the usage for different holders. Furthermore, the usage of steganographic algorithms and the storage of private data within a database increase the security of the private data once again.

We have only outlined a first approach of our *private wallet.* In a subsequent step, we have to finalize our prototype to a stable version for release. After this, we can extend our system with additional features. One could be a user management that has not only user but also user groups in the system . Furthermore, we have only a version for personal computers at the moment. Hence, we have to implement a mobile version to ensure secure data exchange also on mobile devices such as smart phones and tablets. Finally, we have no recovery strategies for media break while using the *private wallet*.

## References

1. UN: The Universal Declaration of Human Rigths, `http://www.un.org`
2. Bundesrepublik, D., Dürig, G.: Grundgesetz. 44. Auflage edn. Dtv (2013)
3. Acquisti, A., Gross, R., Stutzman, F.: Faces of facebook: Privacy in the age of augmented reality. BlackHat, USA (2011)
4. OMG: Business Process Model and Notation (BPMN) Version 2.0, `http://www.omg.org/spec/BPMN/2.0/PDF`
5. Jäger, O.: Technologien für das Privacy Wallet (April 2013)
6. Oram, A.: Peer-to-Peer - Harnessing the power of disruptive technologies. OŔeillt (2001)
7. Steinmetz, R., Wehrle, K.: Peer-to-peer-networking & -computing. Informatik Spektrum 27(1), 51–54 (2004)
8. Singh, S.: Geheime Botschaften Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet. Carl Hanser Verlag München Wien (2000)
9. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
10. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 120–126 (1978)
11. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer (2000)
12. Dropbox, `https://www.dropbox.com/`
13. iCloude, `https://www.icloud.com/`
14. Deutsche Post: Docwallet, `https://docwallet.de/`
15. Snapchat, `http://www.snapchat.com/`
16. CHIP: Snapchat: Sexting-App löscht Videos nicht, `http://www.chip.de/news/Snapchat-Sexting-App-loescht-Videos-nicht_63702600.html`
17. Schmitz, K.: Mega, `https://mega.co.nz/`

# Live Objects - Collaborative Window in the Corporate Documents

Riste Stojanov, Marjan Georgiev, Vladimir Zdraveski,
Milos Jovanovik, and Dimitar Trajanov

Faculty of Computer Science and Engineering, Ss. Cyril and Methodius in Skopje, Macedonia
{name.lastname}@finki.ukim.mk

**Abstract.** In this paper, a document collaboration platform for enterprise environments is presented. It incorporates the collaboration, security, auditing and reuse features into the document editor, as a tool that has well known interface to the end users. The platform enables template definition with annotation of the collaboration and protection units, called live objects. Through the live objects it allows multiple employees with different privileges to work on the same document, without having to violate the enterprise policies and processes. This work aims to increase the employees' productivity by providing a collaboration windows in corporate documents.

**Keywords:** enterprise, document, collaboration, security, auditing, reuse.

## 1 Introduction

In the digitalized and globalized environment of today, the amount and quality of knowledge makes the difference between the corporations. The enterprise knowledge is obtained from its employees and stored in various systems and formats [18]. In the process of knowledge management, teamwork and collaboration is required for task accomplishment[14][25], and the tools need to provide multiple employees with efficient collaboration, enabling knowledge distribution, synchronization and conflict management. Employees from different departments and various privileges can often work together[28], and their knowledge should be protected from unauthorized access and manipulation. The trust in the knowledge management systems is crucial[13], and the process of auditing is used to preserve it. Also, there is a need for managed knowledge reuse, with references of the reused parts for modification synchronization.

Corporate knowledge is represented in various formats, from structured information, stored in traditional databases[11], to unstructured information stored in documents.

The knowledge should be organized according to corporation policies, processes and standards, that are designed to increase the efficiency of their employees and to minimize the operation costs. In order to do so, tools that optimize the collaboration, security, auditing and reusing of the knowledge should be provided.

In most enterprises, all these aspects are achieved mainly through customized applications, leveraging the advantages of the database technologies [11]. These systems work with structured data and use the table records as fine-grained units for collaboration, protection, auditing and reuse. However, their main disadvantage is that they impose strict rules for data entry, and the employees may not be able to transform their knowledge to that format.

Beside the structured knowledge in database systems, the documents contain the major portion of the organizational information utilized in the corporate processes and used by different users and applications [20]. The documents are designed to store unstructured data in the form of natural language text and drawings, and provide a medium that can accept all available employee knowledge. Knowledge management in documents imposes greater conflict possibility when multiple employees collaborate on the same document. The document is the unit of management, which prevents collaboration of employees with different roles, makes it difficult to detect the exact change in the auditing process, and is too coarse for reuse.

In this paper, we present a platform that optimizes the collaboration, security, auditing and reuse of the document content, with a part of a document as a knowledge granularity unit.

The paper is organized as follows: in Section 2 the document collaboration approaches and applications are summarized, after which, in Section 3 the general overview of the implemented platform is presented. In Section 4, the implementation of the system is presented, and a comparison with the other solutions and methodologies is elaborated in Section 5. A conclusion about the presented work is given in the last section of the paper.

## 2 Related Work

Corporate knowledge can be managed in various systems and formats, including the Database Systems (DB)[11], Document Management Systems (DMS)[24], Content Management Systems (CMS)[22] and Online Document Collaboration Systems (ODCS)[27]. All these systems enable efficient collaboration, security, auditing and knowledge reuse on a different granularity units. The database and the Content Management systems work with records and content units correspondingly, which provide data structuring based on the corporate needs, and the employees should adopt their knowledge to the system requirements. According to [20], the major amount of the enterprise knowledge is stored in documents. Even though the DBS and CMS can produce documents, they are not intended to manage them.

Currently, there are two general document collaboration patterns. The first pattern is represented by the online document collaboration systems[27], where the employees edit the document in that system. The second pattern is when the documents are protected and distributed by one system, and edited by another, represented by document management systems.

The online document collaboration systems[27], such as Google Drive[1] and Microsoft Office 365[17], provide real-time collaboration among the employees and are deployed in public cloud infrastructures. These systems provide similar set of features as most of the modern document editors, and there is simple or no training required for employees to use them[27]. The protection granularity of these systems is the document, and there is no mechanism to protect part of content from employees that do not have the rights on it. Additionally, their infrastructure may violates the security policies of many corporations that do not allow storing their documents outside their infrastructure. Even if they do allow this, the corporate accounts should be connected with the platform accounts, which is not that simple process.

The Please Review[3] is another online document collaboration system that can be installed in the corporation. The primary goal of this system is collaborative review, but it also offers a co-authoring option. In the process of document collaboration, the document owner shares the document with the collaborators and assigns them in a so called editing zone. These zones are locked until the user is done with his work. The zone should be downloaded, edited and uploaded back to the system, since it does not provide web editing. Additionally, the user can see the whole document, and there is no option to protect only a part of the document with this system.

The DMS, that represent the second pattern are more often used in the corporations. The collaboration unit of these systems is the document. The document management systems, such as SharePoint[21], Alfresco[23] and many other[19][8][15][16], provide collaboration by document distribution and versioning. Conflicts are prevented by document locking, which disables concurrent modification and decreases the efficiency. Such systems provide document level security and auditing with corporate account integration. These systems does not manipulate with the content of the document, and are combined with document editing applications for content manipulation.

Similar approach is utilized in the file synchronization systems such as Dropbox[9], OneDrive[2] and Ubuntu One[4]. These systems provide document level security and client applications for document synchronization. However, all these solutions are cloud based and may be against the corporate rules.

In [6][5][10] document collaboration systems and methods are defined, where the document is split in workable elements that can be edited by multiple collaborators and the changes are synchronized with a database system in real-time using messaging system. These systems use document level protection using single passwords, and perhaps niter of them provide annotation of the workable elements according to their permissions or meaning in a document template. Also, they do not provide any tracking of the reused parts and are web based solutions.

## 3 Semantic Sky Platform

In corporate environments, the documents frequently contain parts that should not be accessible to some employees. Since none of the systems reviewed in Section 2 protect part of a document, they enforce document splitting based

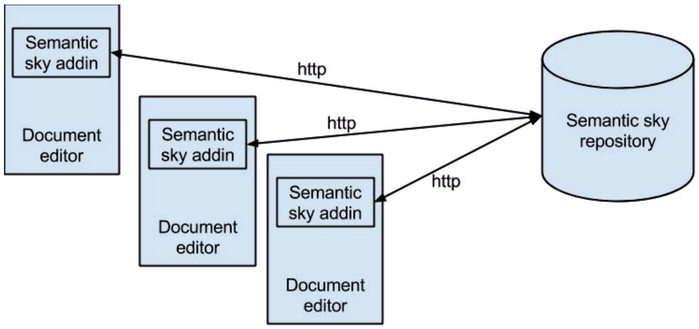


**Fig. 1.** Semantic Sky platform architecture

on the user privileges, collaborating on each of them and merging back the modifications.

The platform presented in this paper, called Semantic Sky, extends our previous work [26], by providing collaboration, security, auditing and managed reuse of document parts. The document parts managed by the Semantic Sky platform are referred to as live objects.

## 3.1 Semantic Sky Architecture

The Semantic Sky platform is implemented using client server architecture (Figure 1). The clients are implemented as document editor addins, while the server is RESTful live object repository. The platform is designed to coexist with the document management systems[24], and the current implementation of the addin is integrated with SharePoint[21] DMS, but not bounded to it.

The Semantic Sky addin is used to handle the document templates and live objects, and currently is implemented for Microsoft Office Word, as one of the most common document editing applications in the corporate world. Moreover, almost all corporate employees are trained to use document editors and they can use the addin with short or no additional training. This way, the employees can use the tools they know best, and simultaneously modify the document parts required to accomplish their tasks. The Semantic Sky addin is used to detect and manipulate the document's live objects, synchronize their content among the collaborators, and protect them from unauthorized access and modification.

All Semantic Sky client addins communicate with the central repository to provide synchronization of the live object content among the users. The content of the live object is versioned, and multiple versions for each live object are persisted into the repository. The Semantic Sky addin displays the content from the last version of the live object in the document. The live object versions provide a better insight to the editors about the evolution of the document. The users have an option to preview the versions of a selected live object, and a possibility to select the content of some previous version as current.

### 3.2 Live Objects

The enterprises use documents with standardized structure in a form of document templates. The templates define the structure of the document, where each part has position and meaning. The Semantic Sky platform is designed to coexist with the templates, where the dynamic and protected parts can be annotated as live objects.

The live objects wrap parts of a document that represents a logical unit and allow collaborative editing. The best practice is to wrap self-contained logical units for collaboration, such as paragraphs, sentences or chapters in live object, but the platform does not impose any limitation.

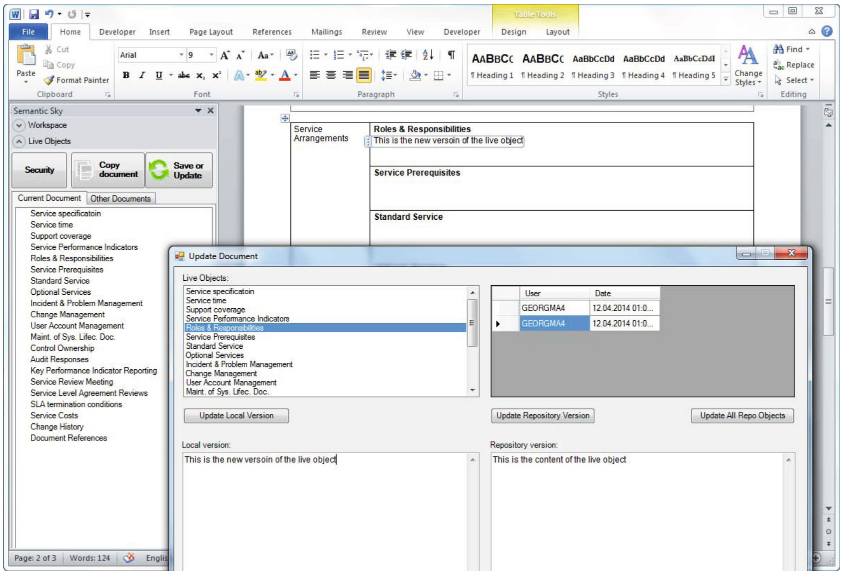
The live objects are the main units for collaboration and they represent the meaning of the element in the template, while their versions store the content, the modifier and the modification time. The Semantic Sky system uses an adaptation of the centralized version control[7] collaboration model for synchronization and conflict resolution with the live objects as granularity unit. The system automatically invokes the update and commit procedures on document open and save correspondingly.

The live objects are designed to enable managed reuse, where the reusing live object references to the reused one. This provides another level of document linking, and gives information about their content sharing. Additionally, when the reused live object is modified in some document, the documents that have reused it will obtain a notification for the modification, with option to accept it or not.

In order to provide collaboration of employees with different privileges in a same document, the Semantic Sky platform has separate access permissions for each live object. The system uses the RBAC model[12], where the users and their roles are obtained from theirs' computer account. Since the documents can be opened without the Semantic Sky addin, there are both public and private protection modes for the live objects. The public live objects are visible to everybody and their content is embedded into the document, while the content of the private live objects is synchronized on each document opening. For the private protection mode, the Semantic Sky system defines the following privileges: no access, read only, read & write, and manage. The higher privileges include and extends the rights from the lower one. The user that creates the document has modify permission, and can manage the protection mode and the privileges of the other collaborators. Live objects have public protection mode by default, and every user can read and modify their content.

### 3.3 Semantic Sky Document Editor Addin

The user interface of MS Word instance with the Semantic Sky addin is shown in Figure 2. All live objects are accessible through the panel at the left side of the editor, providing easy navigation to the required parts. There is also an option to preview the live objects from the other documents and to reuse their content in the current document (the right tab of the panel). The addin provides managed



Service specificatoin
Service time
Support coverage
Service Performance Indicators
Roles & Responsibilities
Service Prerequisites
Standard Service
Optional Services
Incident & Problem Management
Change Management
User Account Management
Maint. of Sys. Lifec. Doc.
Control Ownership
Audit Responses
Key Performance Indicator Reporting
Service Review Meeting
Service Level Agreement Reviews
SLA termination conditions
Service Costs
Change History
Document References

| Service Arrangements | Roles & Responsibilities |
|---|---|
| | This is the new versoin of the live object |
| | Service Prerequisites |
| | Standard Service |

| User | Date |
|---|---|
| GEORGMA4 | 12.04.2014 01:0... |
| GEORGMA4 | 12.04.2014 01:0... |

Local version: This is the new versoin of the live object

Repository version: This is the content of the live object

**Fig. 2.** Document with live objects and their versions

reuse, where the reusing live object stores a link to the reused one. The auditing and the conflict resolution is done through the popup window from Figure 2. It displays a list of all live object changes and enables to roll back the modifications to a previous version. The Semantic Sky platform provides access to the content even if the addin is absent, so that external collaborators can be included. The modifications from these collaborators are synchronized on the next document opening in document editor with installed Semantic Sky addin.

The platform does not provide real-time synchronization because the document may be opened in parallel by document editor instances (or other applications) that do not have the addin. The addin provides on demand live object synchronization and mechanism for conflict resolution. This way, the platform ensures the consistency of the live objects content among the different collaborators.

## 4 Semantic Sky Implementation

The live objects are implemented as MS Word native Rich Text Content Controls[1], handled by the semantic sky addin. The addin supports template creation through annotation of the live objects that are used for content protection and collaboration. The semantic sky addin uses the Tag property[2] of the Rich Text

[1] http://msdn.microsoft.com/en-us/library/microsoft.office.tools.word.richtextcontentcontrol.aspx
[2] http://msdn.microsoft.com/en-us/library/microsoft.office.tools.word.richtextcontentcontrol.tag.aspx

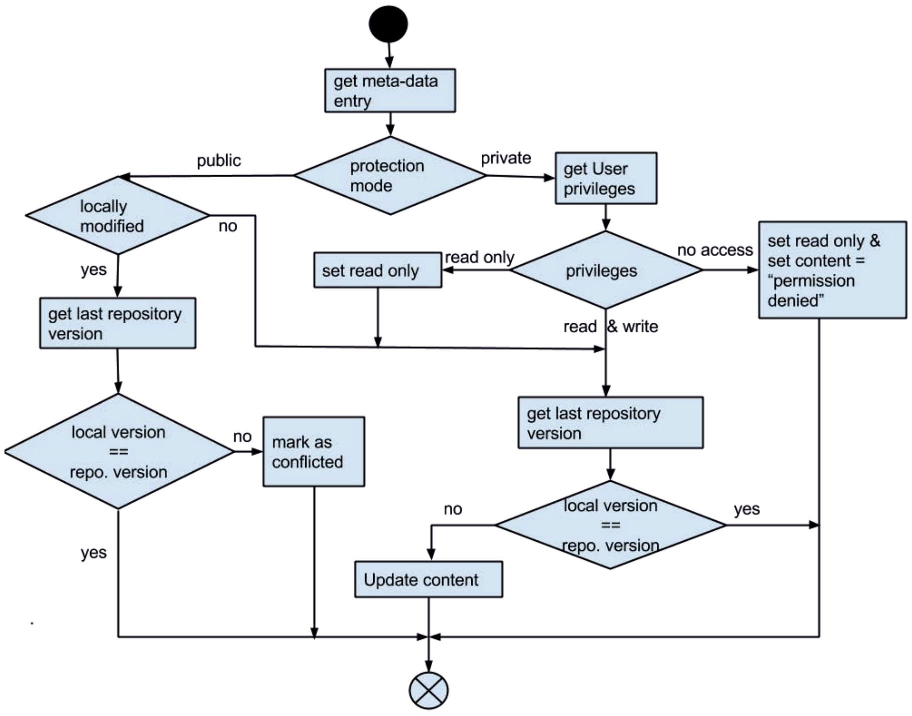


**Fig. 3.** Live Objects Initialization

Content Control to identify the live object controls, and the CustomXMLPart[3] to store their meta-data. The CustomXMLPart is defined as part of the Office Open XML standard[4], and it is not visible to the collaborators. The Semantic Sky addin uses it to decide how to protect and handle the live objects.

## 4.1 Live Object Initialization

When the document is opened in MS Word with Semantic Sky addin, the live objects mechanism iterates over all Reach Text Content Controls in the document, identifying the one that has Tag property starting with "#ssky|#lo|". For each live object control, the addin executes the initialization procedure from Figure 3 in a separate thread, since there are no correlations between them. This speeds up the document opening process, and each live object is initialized independently.

During the initialization process, the addin first obtains the live object's meta-data entity from the document's CustomXMLPart. Since the live objects with public protection mode can be modified outside of the system, there may be a

[3] http://msdn.microsoft.com/en-us/library/bb608618.aspx
[4] http://www.ecma-international.org/publications/standards/Ecma-376.htm

conflict in their content, and the addin checks whether they are modified in the document and in the repository. When there is conflict for the live object, it is marked as conflicted by setting this flag in its meta-data entry. The public live objects that are not locally modified, are update their content with the repository version.

The modification of the private live object is managed by the addin and there is no need for conflict resolution in their initialization phase. The access permissions for the user are first obtained for them. When the user does not have access for the live object, the read only property for its control is set and its content is not loaded nor displayed. Otherwise, the content is loaded from the repository only if the repository version is greater than the local version of the live object.

At the end of the initialization process, the addin registers a handler for the Exiting Event[5] of the live object control, which updates the live object's meta-data after each user interaction.

### 4.2 Live Object Synchronization

The content of the live objects can be synchronized on demand and on document save operations. In this process, the addin tries to store only the modified (dirty) live objects and tries to store their content in the repository. The system then checks for conflicts, and when there is none, a new version of the live object is created. Then the addin updates the live object's content and meta-data. In case the local version is smaller than the repository version, the repository stores the user live object version as conflicted, and the addin also marks the live object control as conflicted.

The popup dialog from Figure 2 is used for conflict resolution. On the upper left side all live objects are displayed, with the conflicted one colored red. In the lower left text box, the local version of the live object is displayed and the right text box is the latest repository version from the live object. The user can accept either the local or the repository version. If a combination from the both versions is needed, the user should manually change the content of its local version, and then accept it. The repository content (the right text area) is not editable. This dialog should be improved in order to display where the two versions differ, and it is planned to be improved as a part of the future work.

When the document is closed and all live objects are synchronized, the semantic sky addin removes the content from the live object controls with private protection mode. This way, when the document is opened without the addin, the protected live objects are not displayed and their information is protected.

## 5 Evaluation

The main difference of the Semantic Sky system is that it supports managed content reuse with referencing and protects the document parts from

[5] `http://msdn.microsoft.com/en-us/library/microsoft.office.tools.word.contentcontrolbase.exiting.aspx`

unauthorized access. All other systems provide content reuse with copy-paste, and do not provide tracking of the reused parts. Also, none of these systems provides protection of document parts based on employee privileges, and these employees must use workarounds that decrease their productivity in order to collaborate with these systems.

The on demand synchronization of the Semantic Sky system is not as efficient as the real-time synchronization, but it allows external collaborators to contribute in document editing tasks on a same document, without exposing protected information. All other systems either protect the whole document with password or do not provide protection at all.

Additionally, the Semantic Sky platform is designed to coexist with the DMS[24] and to use them for document distribution among the employees. The collaboration environment is integrated in Microsoft Office Word document editor, and the employees can use the platform with minimal training, and increases their efficiency.

Table 1 summarizes the features of the document collaboration systems. The collaboration feature describes the collaboration unit of each of the systems together with the synchronization mechanism. The security and auditing columns display the protection and versioning unit of each of the systems, while the content reuse column describes how content can be reused in them. The last column describes whether and how the documents are protected when they are being open out of the system.

**Table 1.** Document collaboration systems comparison

| | Collaboration | Security | Auditing | Content reuse | Template support | Out of system protection |
|---|---|---|---|---|---|---|
| Semantic Sky | live object, on demand | live object | live object | referencing (managed) | yes | private live objects |
| ODCS[27] | character, real-time | document | character | copy-paste | no | no |
| DMS[24] with editing app. | document, lock&versions | document | document | copy-paste | editor templates | document (password) |
| [3][5][6][10] | document part, lock or real-time | document | document part | copy-paste | no | document (password) |

## 6 Conclusion

The Semantic Sky platform provides document collaboration for enterprise environments, handling security with document content protection, even when the documents leave the controlled environment. The template mechanism enables structuring of the corporate documents through annotation of the collaborative and protected elements, called live objects.

The platform extends the document editors, with the Semantic Sky addin. The integration with the document editor significantly improves the employees efficiency because they can focus on their tasks, leaving the addin to handle the document synchronization and management. The versions of the live objects provide document evolution auditing, while the managed live object reuse provides notifications for the documents with reused content when the base live object is changed.

The use of the live object controls enables collaborators without the semantic sky addin to be involved in the editing process of the public content, without violating the corporate policies.

## References

1. Google drive, `http://drive.google.com/` (accessed April 10, 2014)
2. Onedrive, `https://onedrive.live.com/about/en-us/` (accessed April 10, 2014)
3. Please review, `http://www.pleasetech.com/pleasereview.aspx/` (accessed April 10, 2014)
4. Ubuntu one, `https://one.ubuntu.com/` (accessed April 10, 2014)
5. Bailor, J., Bernstein, E., Knight, M., Antos, C., Simonds, A., Jones, B., Clarke, S., Sunderland, E., Robins, D., Bose, M.: Collaborative authoring. US Patent 7,941,399 (May 10, 2011)
6. Chin, R., Lee, J.: Document collaboration system and method. US Patent App. 11/836,087 (March 6, 2008)
7. Collins-Sussman, B., Fitzpatrick, B., Pilato, M.: Version control with subversion. O'Reilly Media, Inc. (2004)
8. Cullen, J., Peairs, M.: Document management system. US Patent 5,893,908 (April 13, 1999)
9. Drago, I., Mellia, M., Munafo, M.M., Sperotto, A., Sadre, R., Pras, A.: Inside dropbox: understanding personal cloud storage services. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, pp. 481–494. ACM (2012)
10. Dutta, K.: Document collaboration system. US Patent App. 10/900,807 (February 2, 2006)
11. Elmasri, R.: Fundamentals of database systems. Pearson Education India (2008)
12. Ferraiolo, D., Cugini, J., Kuhn, D.R.: Role-based access control (RBAC): Features and motivations. In: Proceedings of 11th Annual Computer Security Application Conference, pp. 241–248 (1995)
13. Ford, D.P.: Trust and knowledge management: the seeds of success. In: Handbook on Knowledge Management 1, pp. 553–575. Springer (2004)
14. Goh, S.C.: Managing effective knowledge transfer: an integrative framework and some practice implications. Journal of Knowledge Management 6(1), 23–30 (2002)
15. Hajmiragha, M. Document management system. US Patent 6,289,460 (September 11, 2001)
16. Jeffery, S., O'Gwen, G., Hornsby, B., McBryde, K., Powell, W., Rizk, T.: Document management system. US Patent 6,957,384 (October 18, 2005)
17. Katzer, M., Crawford, D.: Office 365: Moving to the cloud. In: Office 365, pp. 1–23. Springer (2013)
18. O'Leary, D.E.: Enterprise knowledge management. Computer 31(3), 54–61 (1998)

19. Oliszewski, M.: Document management system. US Patent App. 10/208,062 (July 3, 2003)
20. Päivärinta, T., Tyrväinen, P.: Documents in information management: Diverging connotations of'a document'in digital era. In: Proceedings of IRMA 1998, pp. 163–173 (1998)
21. Pattison, T., Connell, A., Hillier, S., Mann, D.: Inside Microsoft SharePoint 2010. Microsoft Press (2011)
22. Rockley, A., Kostur, P., Manning, S.: Managing enterprise content: A unified content strategy. New Riders (2003)
23. Shariff, M.: Alfresco enterprise content management implementation. Packt Publishing Ltd. (2007)
24. Sutton, M.J.D.: Document Management for the Enterprise: Principles, Techniques, and Applications. John Wiley & Sons, Inc., New York (1996)
25. Syed-Ikhsan, S.O.S., Rowland, F.: Knowledge management in a public organization: a study on the relationship between organizational elements and the performance of knowledge transfer. Journal of Knowledge Management 8(2), 95–111 (2004)
26. Trajanov, D., Stojanov, R., Jovanovik, M., Zdraveski, V., Ristoski, P., Georgiev, M., Filiposka, S.: Semantic sky: a platform for cloud service integration based on semantic web technologies. In: Proceedings of the 8th International Conference on Semantic Systems, pp. 109–116. ACM (2012)
27. Vallance, M., Towndrow, P.A., Wiz, C.: Conditions for successful online document collaboration. TechTrends 54(1), 20–24 (2010)
28. Wong, K.Y., Aspinwall, E.: Development of a knowledge management initiative and system: A case study. Expert Systems with Applications 30(4), 633–641 (2006)

# Part IV
# Physical Level

# Flexs – A Logical Model for Physical Data Layout

Hannes Voigt, Alfred Hanisch, and Wolfgang Lehner

Database Technology Group,
Technische Universität Dresden,
01062 Dresden, Germany
{firstname.lastname}@tu-dresden.de
http://wwwdb.inf.tu-dresden.de/

**Abstract.** Driven by novel application domains and hardware trends database research and development set off to many novel and specialized architectures. Particularly in the area of physical data layout, specialized solutions have shown exceptional performance for specific applications. This trend is great for research and development and for those in need of top-level performance first and foremost. For those with moderate performance needs, however, a universal but flexible database system has the benefit of lower TCO. Regarding physical data layout, the more general systems are fairly inflexible compared to the variety of physical data layouts available in specialized systems. Particularly, the macroscopic characteristics, i.e., how the data is grouped and clustered, are generally hard-coded and cannot be changed by configuration. We present Flexs, a declarative storage description language for the macroscopic characteristics of physical data layouts. Flexs allows describing physical data layouts ranging from the row and column store layouts to data layouts for irregular data such as vertical schema. Using Flexs, a storage engine can be configured to use a specific physical data layout. Flexs contributes to make specialized physical data layouts available to the broad majority of universal database systems.

## 1 Introduction

The drastic expansion of the database ecosystem to novel application domains as well as new hardware trends sparked a new exciting age of database technology. Widely challenging the traditional database system architecture in new fields [28], the database community created the wide and diverse range of database systems available today. Specialized systems provide exceptional performance and features for specific applications that have not been seen before.

Among other techniques, these systems typically build on a physical data layout different from the traditional row-orientation. Column-oriented data layouts [15,9,27] showed to be advantageous for most analytic applications. In multi-tenancy databases, clustering data along versions and schema extensions is a preferable strategy [4]. Efficient processing of geo-spatial data often benefits from a grid representation [11]. Other layouts such as interpreted record [7] or vertical schema [2,12,1] are favored for applications with flexible schemas or sparse tables such as product catalogs or clinical information systems.

Specialization is great for bringing database technology to new frontiers and boosting its performance. However, it is not always affordable to every extent and for every customer. No matter if they run 10 database systems or 10 000 systems, companies require standardization, unification, and consolidation to keep total cost of ownership (TCO) under control [26]. Any additional system increases the overall complexity of an IT landscape. It requires additional maintenance and additional attention by staff trained on the details of that system. Any additional system adds additional interactions with other systems and by that it implies further need for supervision and further potential cause for failures. With all that additional complexity and cost, specialized systems are only worthwhile where absolute top performance is needed. In most cases, a single system with a configurable physical data layout comes at a much lower TCO, but would still be able to serve a diverse range of workloads well enough. Configurability also simplifies the adaption of a system to evolving workloads and requirements.

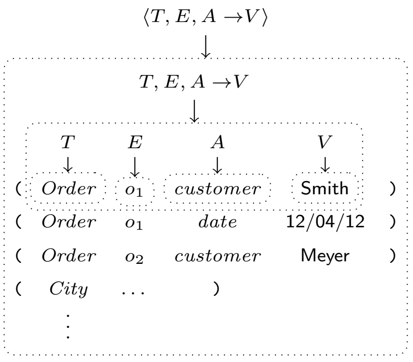
Physical data layouts differ in (1) microscopic characteristics, such as, used data structures, applied compression techniques, etc., and in (2) macroscopic characteristics, i.e. how the data is grouped and clustered. To some extent, most systems allow influencing the microscopic characteristics of their physical data layout. For instance, compression can be tuned or different data structures can be configured. Macroscopic characteristics, however, are generally hard-coded and cannot be changed by configuration.

In this paper, we present Flexs. Flexs is a declarative storage description language for the macroscopic characteristics of the physical data layout. It provides a generic way to configure the grouping and clustering of data. In contrast to other modeling approaches such as list comprehension, Flexs explicitly includes the schema elements entity types and attributes into the layout description. Flexs supports a wide range of layouts, such as row-oriented, column-oriented, interpreted record, or vertical schema. We introduce the Flexs notation with the help of three examples and present the grammar that specifies the Flexs language.

The remainder of the paper is structured as follows. Section 2 introduces the Flexs notation. Section 3 details an adaptive materilization strategy necessary to implement a storage engine that is configurable with Flexs. Finally, we discuss related work in Section 4 and conclude the paper in Section 5.

## 2 Flexs Notation

Flexs is a modeling language that allows describing various physical data layouts for structured data, specifically the macroscopic aspects of the physical data layouts. Commonly, structured data models organize values with the user-given specifiers entity types, entities, and attributes. The physical data layout determines how a database management system organizes the elements of structured data (entity types, entities, attributes, and values) on the physical storage to preserve their logical structure.



⟨T, E, A →V⟩
T, E, A →V
T
E
A
V
( Order o1 customer Smith )
( Order o1 date 12/04/12 )
( Order o2 customer Meyer )
( City ... )

**Fig. 1.** Vertical schema layout

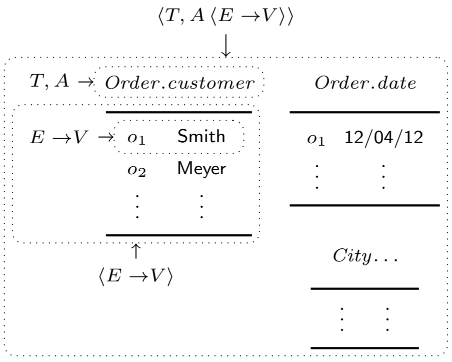


**Fig. 2.** BATs layout

The first and most simple way to preserve logical structure is to store logically related elements physically next to each other. For instance, the vertical schema as illustrated in Figure 1 stores an entity type, an entity, an attribute, and the value that belongs to the specific combination in one coherent physical block. In Figure 1, the first block lists Smith as the value of the attribute *customer* for the entity $o_1$ with type *Order*. In Flexs, we describe such a block by listing the domains of its elements. $T$ denotes the domain of entity types, $E$ denotes the domain of entities, $A$ denotes the domain of attributes, and $V$ denotes the domain of values Consequently, $T, E, A \rightarrow V$ describes a physical block in the vertical schema layout. The arrow $\rightarrow$ indicates that $T, E, A$ uniquely identify a block and functionally determine $V$ in this mental model. To represent more than a single value, the database system uses a whole set of similar blocks to represent a complete data set. In Flexs, we denote such a block set as $\langle T, E, A \rightarrow V \rangle$. Chevrons $\langle X \rangle$ indicate the repetition of the embedded block structure $X$. The order of the blocks in a block set is insignificant. The commas in the Flexs notation do not have a particular meaning but serve better visual separation of the domain symbols.

A second way to preserve logical structure is nesting. For nesting, let's have a look at the binary association tables (BATs) [9] layout. Figure 2 shows the same data represented in BATs. A single BAT consists of an entity type, an attribute, and a set of entity–value pairs. Hence, we can denote the header of a BAT in Flexs as $T, A$ and the body of a BAT – set of entity–value pairs – can be denoted as the block set $\langle E \rightarrow V \rangle$. In all, a BAT forms a block $T, A \langle E \rightarrow V \rangle$. This block nests block set of entity–value pairs to represent the fact these entity–value pairs are logically related to the entity type–attribute pair in BAT's header. Again, the complete physical data layout consists of multiple blocks like these and is denoted as $\langle T, A \langle E \rightarrow V \rangle \rangle$.

A third way to preserve logical structure is the physical order. Normally, the physical order of blocks within a block set is insignificant. Netherless, some layouts utilize the physical order, e.g. the traditional row store store layout, as

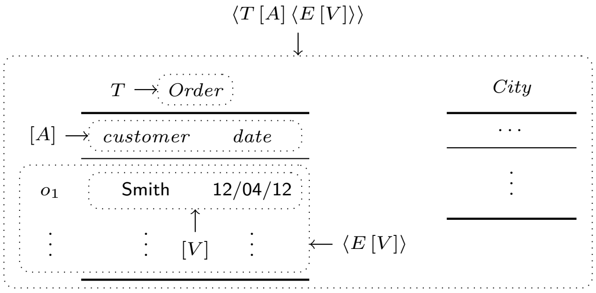


**Fig. 3.** Row store layout

shown in Figure 3. For each entity type, the row store maintains an ordered set of attributes. In Flexs, we denote such an ordered set with brackets: $[A]$. All values of an entity are also stored in order, namely in the order of the attributes of the corresponding entity type. Hence, the row store layout $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$ represents the logical relations between attributes and values exclusively by their order.

Next to three discussed layouts, Flexs can also describe various other layouts as shown in Table 1. The table shows each layout with its respective Flexs expression and an example of the resulting block set. Note that Flexs explicitly considers the role of the schema elements entity types and attributes within the physical data layout. The inclusion of schema elements allows Flexs to support physical data layout for irregular data, e.g., interpreted record and vertical schema. Further it allows Flexs to support also layouts that have not been widely considered so far. For instance, the tagging layout shown in Table 1 allows to physically represent multifaceted entities as they exist in the Freebase data model [8]. This clearly distinguishes Flexs from other modeling approaches such as list comprehension.

Flexs is formally defined by a grammar as shown in Figure 5. To be valid, a Flexs expression has to follow the grammar. The central element is the block set definition (<blockset>). A block set can be defined as ordered (<ordered>) or as a block set without particular order (<unordered>). Ordered block sets are defined on single domains. Further, ordered block sets are only allowed in pairs in Flexs expressions, where one of the two ordered block sets has to be defined on the domain of values $V$. Normal block set definitions consist of header domains (<header>), included domains (<include>), and nested block sets (<nesting>). There must be at least one header domain; included domains and nested block sets are optional. Among the four domains, we distinguish between the specifier domains – entity types $T$, entities $E$, and attributes $A$ – and the values domain $V$. We are not considering values to have an identity of their own; value identity derives from the entity type, the entity, and the attribute a value belongs to. Consequently, $V$ is only allowed in ordered block sets or as included domain in unordered block sets.

**Table 1.** Various physical data layouts

| | | |
|---|---|---|
| Vertical schema | $\langle T, E, A \rightarrow V\rangle$ | $\langle Order, o_1, customer \rightarrow \textsf{Smith} \mid Order, o_1, date \rightarrow \textsf{12/04/12} \mid Order, o_2, customer \rightarrow \textsf{Meyer} \mid \ldots\rangle$ |
| Row store | $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$ | $\langle Order, [customer \mid date]\,\langle o_1, [\textsf{Smith} \mid \textsf{12/04/12}] \mid o_2, [\textsf{Meyer} \mid \textsf{12/05/12}]\rangle\rangle$ |
| Column store | $\langle T\,[E]\,\langle A\,[V]\rangle\rangle$ | $\langle Order, [o_1 \mid o_2]\,\langle customer, [\textsf{Smith} \mid \textsf{Meyer}] \mid date, [\textsf{12/04/12} \mid \textsf{12/05/12}]\rangle\rangle$ |
| BATs | $\langle T, A\,\langle E \rightarrow V\rangle\rangle$ | $\langle Order, customer\,\langle o_1 \rightarrow \textsf{Smith} \mid o_2 \rightarrow \textsf{Meyer}\rangle \mid Order, date\,\langle o_1 \rightarrow \textsf{12/04/12} \mid o_2 \rightarrow \textsf{12/05/12}\rangle\rangle$ |
| Interpreted record | $\langle T\,\langle E\,\langle A \rightarrow V\rangle\rangle\rangle$ | $\langle Order\,\langle o_1, \langle customer \rightarrow \textsf{Smith} \mid date \rightarrow \textsf{12/04/12}\rangle \mid o_2, \langle customer \rightarrow \textsf{Meyer} \mid date \rightarrow \textsf{12/05/12}\rangle\rangle\rangle$ |
| BigTable | $\langle T\,\langle E, A \rightarrow V\rangle\rangle$ | $\langle Order\,\langle o_1, customer \rightarrow \textsf{Smith} \mid o_1, date, \rightarrow \textsf{12/04/12} \mid o_2, customer \rightarrow \textsf{Meyer} \mid \ldots\rangle\rangle$ |
| XML-like | $\langle E \rightarrow T\,\langle A \rightarrow V\rangle\rangle$ | $\langle o_1 \rightarrow Order\,\langle customer \rightarrow \textsf{Smith} \mid date \rightarrow \textsf{12/04/12}\rangle \mid o_2 \rightarrow Order\,\langle customer \rightarrow \textsf{Meyer} \mid \ldots\rangle\rangle$ |
| Tagging | $\langle E\,\langle T\rangle\,\langle A \rightarrow V\rangle\rangle$ | $\langle o_1\,\langle Order\rangle\,\langle customer \rightarrow \textsf{Smith} \mid date \rightarrow \textsf{12/04/12}\rangle \mid o_2\,\langle Order\rangle\,\langle customer \rightarrow \textsf{Meyer} \mid \ldots\rangle\rangle$ |

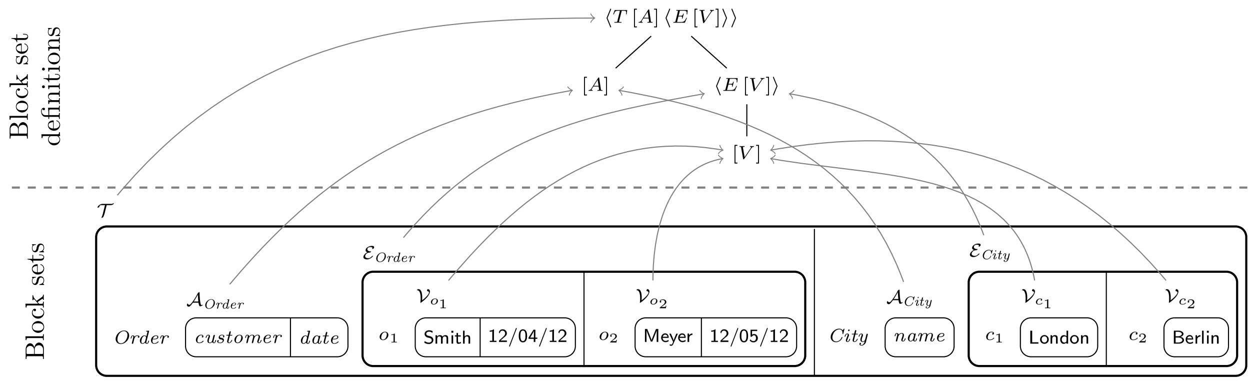


**Fig. 4.** Block set with the row store layout $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$

```
  <blockset> ::= <ordered> | <unordered>
   <ordered> ::= '[' <domain> ']'
<unordered> ::= '⟨' <header> <include>? <nesting>? '⟩'
    <header> ::= <domain>+
   <include> ::= '→' <domain>+
   <nesting> ::= <blockset>+
    <domain> ::= <specifiers> | <values>
<specifiers> ::= 'T' | 'E' | 'A'
    <values> ::= 'V'
```

**Fig. 5.** Flexs grammar

When parsed, a Flexs expression results in an abstract syntax tree of block set definitions as shown in the upper half of Figure 4. These block set definitions provide the necessary information to physically arrange and retrieve data on storage. Like the block set definitions, the physical layout described by the expression is hierarchical. On each hierarchy level the block set definition is instantiated with one or more block sets, so that each block set is typed by a particular block set definition from the abstract syntax tree. The topmost block set definition is only instantiated once, while lower-level block set definitions are likely to be instantiated multiple times, depending on the data.

Block sets, the instances of block set definitions, contain one or multiple blocks. The blocks contain data elements and may nest lower-level block sets. The definition of a block set defines the structure of its blocks. For instance, in a block set of the definition $\langle E\,[V]\rangle$, all blocks take the form $(e, \mathcal{X})$, where $e$ is an entity and $\mathcal{X}$ is a block set of the definition $[V]$. All blocks in a block set are uniquely identified by their headers, i.e. their elements from the header domains in the definition. In our example, all blocks are uniquely identified by the data element $e$. A block's data elements from the included domains in the block set definition form the block include.

As an example, Figure 4 illustrates the block sets resulting from the row store layout $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$ given our example data. In the figure, the rounded rectangles mark block sets, while the vertical lines separate blocks within block sets. The top most block set $\mathcal{T}$ is an instance of $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$ as indicated by the arrow pointing from the block set to its block set definition. Consequently, it contains blocks consisting of an entity type as header and two nested block sets of the form $[A]$ and $\langle E\,[V]\rangle$. $\mathcal{T}$ contains two of such blocks: $(\mathit{Order}, \mathcal{A}_{\mathit{Order}}, \mathcal{E}_{\mathit{Order}})$ and $(\mathit{City}, \mathcal{A}_{\mathit{City}}, \mathcal{E}_{\mathit{City}})$. The order of these two blocks as given in the figure is insignificant. Further down in the hierarchy, for instance, the block set $\mathcal{E}_{\mathit{Order}}$ is of the form $\langle E\,[V]\rangle$ and contains the blocks $(o_1, \mathcal{V}_{o_1})$ and $(o_2, \mathcal{V}_{o_2})$. $\mathcal{V}_{o_1}$ and $\mathcal{V}_{o_2}$ are both ordered block sets of the form $[V]$, each containing a pair of blocks, (Smith), (12/04/12) and (Meyer), (12/05/12), respectively. Here, the order of the blocks is essential since it allows connecting the values to their attributes stored in the ordered block set $\mathcal{A}_{\mathit{Order}}$.

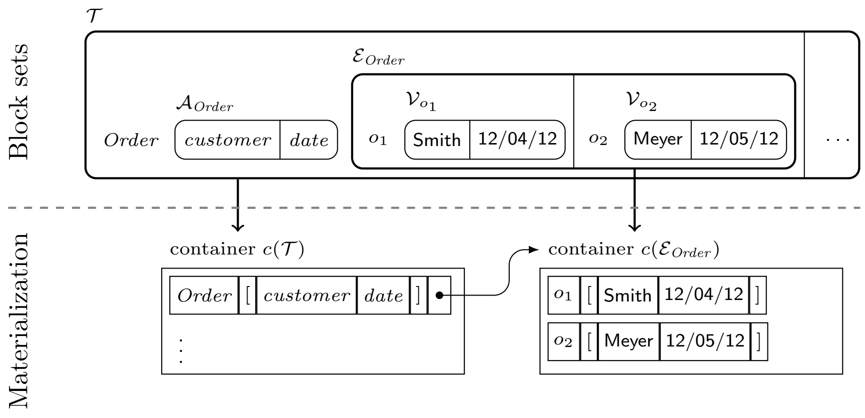


**Fig. 6.** Block set materialization for $\langle T\,[A]\,\langle E\,[V]\rangle\rangle$

## 3 Block Set Materialization

With Flexs a storage engine can be configured to a specific physical data layout. The storage engine then has to arrange the data accordingly in storage. Therefore the storage engine simply can materialize the blocks of the topmost block set consecutively in storage. The materialization of a block embeds the materialization of all nested block set. For the majority of layouts, however, this simple materialization strategy results in very large physical blocks. These large physical blocks and particularly the nested block sets are hard to maintain and cannot be efficiently queried. Selective queries would have to read a lot of unnecessary data.

Efficient retrieval and maintenance of blocks require indirection steps in the materialization. An indirection step separates nested block sets into a dedicated materialization. In the materialization of the nesting block a reference points to the dedicated materialization of the nested block set. To have an indirection step for each nested block set is not ideal either. Too many indirection steps lower the retrieval performance as well; they reduce locality and increase the number of references to resolve. The optimal set of indirection steps depends on the data set and the configured layout.

For instance, the materialization strategy used in Figure 6 is absolutely reasonable for regular relational data, where we have orders of magnitude more entities than attributes and entities are only a few hundred bytes of size. However, with a very large number of attributes or large blob values the embedded block sets $\mathcal{A}_{Order}$, $\mathcal{V}_{o_1}$, $\mathcal{V}_{o_2}$, etc. would be significantly larger. If so, a different materialization strategy may be more efficient.

We propose an adaptive strategy to avoid the embedding of very large block sets. When a database is created, the adaptive strategy starts with a single physical container for the topmost block set. All other block sets are materialized in an embedded way. With more data being inserted, the block sets grow. For every block set where the embedded materialization size exceeds a configured threshold, e.g., the size of a memory page or a disk block, the adaptive strategy introduces an indirection step and moves the block set to a dedicate materialization. Note that this reorganization is applied to relatively small block sets only (just above the threshold), so that operational overhead is small. Regardless of the configured layout, the adaptive strategy finds the optimal set of indirection steps.

## 4 Related Work

Flexs aims at supporting a wider range of applications in a single system, by offering a configurable specialization of the physical data layout. Most prominently, OLTP workloads favor a row-oriented data layout while OLAP workloads profit from a column-oriented layout. Supporting these two workloads in a single system has been the aim of multiple works in recent years.

Index-only plans try to emulate column-oriented processing in a row-based system. Additionally to the base relational, it adds an unclustered B+Tree index on every column. With the help of index intersection, the database system can then answer queries without reading the base relation [21,23]. The efficiency of index-only plans can be increased by exploiting the parallel processing capabilities of modern hardware [13]. Microsoft further exploits this idea by introducing a dedicated column store index [19]. Trojan Columns [18] follows the same line but omits the base relation completely. Still, the index-only plans remain an OLAP-focused add-on to a row-oriented database system. They do not offer the generality of Flexs.

A very early approach of combining OLTP and OLAP is Fractured Mirrors [22]. Fractured Mirrors leverages the fact, that disk-based databases usually replicate data to multiple disks. If the replicates hold the data in different physical layouts, queries can access the data in their preferred physical layouts. As presented, Fractured Mirrors supports exactly the two representations, row store and column store. Both are implemented as different scan operators in the database engine. Generally, the idea is orthogonal to Flexs. Both would make an appealing combination.

HyPer [20] is a main memory database system with OLTP and OLAP support. It runs OLAP queries concurrently and isolated from the OLTP queries, by utilizing hardware-supported page shadowing. In addition to the strictly sequential execution of OLTP write operations, this eliminates the need for locking or any other kind of currency control, resulting in outstanding transaction throughput and query response times. The physical data layout, though, is not the authors' main concern. HyPer uses the same physical layout for all data; globally configured to either row-oriented or column-oriented. Other physical data layouts are not supported.

SAP HANA [14,25] is another hybrid main memory database system for OLTP and OLAP. In its hybrid relational engine, it combines SAP's row store database engine P*Time [10] and SAP's column store engine TREX. HANA's academic sidekick [24] uses MaxDB as row-store engine. The setup, though, is the same: Two relational engines wired to the same query processor. During table creation it is decided which engine manages the table. The well-known open source database system MySQL also follows the multiple engine concept. Here, an interface separates data storage from the query processor including SQL parser and query optimizer. Over the years, many storage engines have been developed each implementing its own physical data layout. Instead of supporting a hard-coded set of physical data layouts, Flexs aims at a freely configurable physical data layout implemented in a single engine.

HYRISE [16] is also a hybrid main memory storage engine. It partitions tables vertically into configurable column groups, where each column group is stored directory compressed as if it is a single column. Hence, in its heart, HYRISE is a main memory column store that can be configured to mimic a row store or mixed layouts. The idea of bundling columns together that are typically accessed was already used earlier in the Data Morphing approach [17]. It is an extension of PAX [3]. PAX organizes pages like a column store for better cache performance. Data Morphing improves that by exploiting column groups. All these approaches focuses on OLTP and OLAP. Other workloads favoring different physical data layouts are not considered.

While the aforementioned approaches focus on the combination of OLTP and OLAP workloads RodentStore [11] takes a broader hold of the topic. Like Flexs, RodentStore envisions a freely configurable data layout, but with a different focus than Flexs. RodentStore describes the physical data layout as nested lists of values. How the lists are nested and which values they contain can be configured with a storage algebra building on list comprehensions. This allows a wide range of physical data layouts including row store, column store, and grid layout. With the focus rather on spatial data, RodentStore includes some representations not covered by Flexs, such as the utilization of space filling curves. The RodentStore concept assumes strictly regular relational data, though, and does not consider the role of specifiers in the data representation. Hence, it lacks support for physical data layouts most suitable for irregular data. It would be interesting to see how both concepts, RodentStore and Flexs could be combined.

Another notable approach on increasing physical data independence is GMAP [29]. GMAP offers a generalized definition language for access paths. Where traditional access paths are fixed to the logical data model, GMAP defines the data stored in an access path with query-like statements. This provides great flexibility in the physical design for the DBA. As RodentStore, GMAP assumes regular data and does not consider the role of specifiers in the data layout. GMAP's focus is to allow the consolidation of associated entities from different domains in a single access path. It also allows the replication of entities over multiple access paths. GMAP offers a great way to define data subsets which require different physical data layouts. In that respect, GMAP and Flexs complement each other perfectly.

Finally, GENESIS [5,6] is a project from the mid-eighties that developed detailed storage models for database systems including aspects of the physical data layout. The aim, however, was not to strengthen the physical data independence of database systems but to speedup their development. Instead of providing an exhaustive description of a database system's storage layer, it focuses on capturing its macroscopic characteristics.

## 5 Conclusion

In this paper, we presented Flexs. Flexs allows describing the macroscopic characteristics of physical data layouts, i.e. how data elements are grouped on the physical storage. It can express common layouts such as row store, column store or BATs. Flexs also can serve as a foundation for mixed physical data layouts – a direction we plan to explore next. In contrast to list comprehensions, Flexs is not limited to regular data; it supports also layouts for irregularly structured data, such as interpreted record and vertical schema. A Flexs expression describes a hierarchical structure of nested block sets, which contain blocks of data elements and thereby determine how data elements can be selected and scanned. We also proposed an adaptive materialization strategy that finds the optimial set of indirection steps to allow for efficient retrieval and maintanence of the data. Flexs is not meant to provide top-level performance but allow combining the characteristics of various physical data layouts in a single system.

## References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB 2007 (2007)
2. Agrawal, R., Somani, A., Xu, Y.: Storage and Querying of E-Commerce Data. In: VLDB 2001 (2001)
3. Ailamaki, A., DeWitt, D.J., Hill, M.D., Skounakis, M.: Weaving Relations for Cache Performance. In: VLDB 2001 (2001)
4. Aulbach, S., Seibold, M., Jacobs, D., Kemper, A.: Extensibility and Data Sharing in evolving multi-tenant databases. In: ICDE 2011 (2011)
5. Batory, D.S.: Modeling the Storage Architectures of Commercial Database Systems. ACM Transactions on Database Systems 10(4) (1985)
6. Batory, D.S., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., Wise, T.E.: GENESIS: An Extensible Database Management System. IEEE Transactions on Software Engineering 14(11) (1988)
7. Beckmann, J.L., Halverson, A., Krishnamurthy, R., Naughton, J.F.: Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format. In: ICDE 2006 (2006)
8. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In: SIGMOD 2008 (2008)
9. Boncz, P.A., Kersten, M.L.: MIL Primitives for Querying a Fragmented World. The VLDB Journal – The International Journal on Very Large Data Bases 8(2) (1999)
10. Cha, S.K., Song, C.: P*TIME: Highly Scalable OLTP DBMS for Managing Update-Intensive Stream Workload. In: VLDB 2004 (2004)

11. Cudré-Mauroux, P., Wu, E., Madden, S.: The Case for RodentStore: An Adaptive, Declarative Storage System. In: CIDR 2009 (2009)
12. Cunningham, C., Graefe, G., Galindo-Legaria, C.A.: PIVOT and UNPIVOT: Optimization and Execution Strategies in an RDBMS. In: VLDB 2004 (2004)
13. El-Helw, A., Ross, K.A., Bhattacharjee, B., Lang, C.A., Mihaila, G.A.: Column-Oriented Query Processing for Row Stores. In: DOLAP 2011 (2011)
14. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA Database - Data Management for Modern Business Applications. SIGMOD Record 40(4) (2011)
15. French, C.D.: "One Size Fits All" Database Architectures Do Not Work for DDS. In: SIGMOD 1995 (1995)
16. Grund, M., Krüger, J., Plattner, H., Zeier, A., Cudré-Mauroux, P., Madden, S.: HYRISE - A Main Memory Hybrid Storage Engine. The Proceedings of the VLDB Endowment 4(2) (2010)
17. Hankins, R.A., Patel, J.M.: Data Morphing: An Adaptive, Cache-Conscious Storage Technique. In: VLDB 2003 (2003)
18. Jindal, A., Schuhknecht, F., Dittrich, J., Khachatryan, K., Bunte, A.: How Achaeans Would Construct Columns in Troy. In: CIDR 2013 (2013)
19. Larson, P.Å., Clinciu, C., Hanson, E.N., Oks, A., Price, S.L., Rangarajan, S., Surna, A., Zhou, Q.: SQL Server Column Store Indexes. In: SIGMOD 2011 (2011)
20. Kemper, A., Neumann, T.: HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: ICDE 2011 (2011)
21. Mohan, C., Haderle, D.J., Wang, Y., Cheng, J.M.: Single Table Access Using Multiple Indexes: Optimization, Execution, and Concurrency Control Techniques. In: Bancilhon, F., Zhang, J., Thanos, C. (eds.) EDBT 1990. LNCS, vol. 416, pp. 29–43. Springer, Heidelberg (1990)
22. Ramamurthy, R., DeWitt, D.J., Su, Q.: A Case for Fractured Mirrors. In: VLDB 2002 (2002)
23. Raman, V., Qiao, L., Han, W., Narang, I., Chen, Y.L., Yang, K.H., Ling, F.L.: Lazy, Adaptive RID-List Intersection, and Its Application to Index Anding. In: SIGMOD 2007 (2007)
24. Schaffner, J., Bog, A., Krüger, J., Zeier, A.: A Hybrid Row-Column OLTP Database Architecture for Operational Reporting. In: Castellanos, M., Dayal, U., Sellis, T. (eds.) BIRTE 2008. LNBIP, vol. 27, pp. 61–74. Springer, Heidelberg (2009)
25. Sikka, V., Färber, F., Lehner, W., Cha, S.K., Peh, T., Bornhövd, C.: Efficient Transaction Processing in SAP HANA Database – The End of a Column Store Myth. In: SIGMOD 2012 (2012)
26. Stonebraker, M.: Stonebraker on NoSQL and enterprises. Communications of the ACM 54(8) (2011)
27. Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E.J., O'Neil, P.E., Rasin, A., Tran, N., Zdonik, S.B.: C-Store: A Column-oriented DBMS. In: VLDB 2005 (2005)
28. Stonebraker, M., Çetintemel, U.: "One Size Fits All": An Idea Whose Time Has Come and Gone. In: ICDE 2005 (2005)
29. Tsatalos, O.G., Solomon, M.H., Ioannidis, Y.E.: The GMAP: A Versatile Tool for Physical Data Independence. The VLDB Journal – The International Journal on Very Large Data Bases 5(2) (1996)

# Storing Long-Lived Concurrent Schema and Data Versions in Relational Databases

Bob Wall[1] and Rafal Angryk[2]

[1] Department of Computer Science – Montana State University
Bozeman, MT 59717-3880, USA
bwall@cs.montana.edu

[2] Department of Computer Science – Georgia State University
34 Peachtree Street, Atlanta, GA 30302-3994, USA
angryk@cs.gsu.edu

**Abstract.** Although there is a strong focus on NoSQL databases for cloud computing environments, traditional relational data bases are still an integral part of many computing services in the cloud. Two significant issues in managing a relational database in a cloud environment are handling the inevitable evolution of the database schema and managing changes to system configuration and other data stored in the database as the system evolves over time. Techniques for handling these issues in on-premise databases are much less feasible in cloud computing environments, which demand efficiency, elasticity, and scalability. We propose a versioning system that can be used in relational databases to allow new versions of the database schema and data to be maintained within the same database as the production data. Past research on versioning either handles data versioning but not schema changes, or handles both but is focused on OLAP or XML databases. In this paper, we describe a mechanism for storing concurrent versions of data in an OLTP database. We explore two different implementation alternatives for versioned data storage and evaluate their relative merits given different workloads. We provide a concrete description of how this can be implemented within the InnoDB storage engine, which is the default data store for MySQL databases, and we present a quantitative comparison of the two implementations in InnoDB.

**Keywords:** schema evolution, data versioning, Database as a Service.

## 1 Introduction

The explosive growth of cloud computing has emphasized the need for greater flexibility in installation, maintenance and management of systems that run in Software as a Service (SaaS) or Platform as a Service (PaaS) environments. Although NoSQL databases are very popular in cloud environments, they are primarily used for data analysis purposes. There is a significant demand for traditional SQL-based relational database systems, especially among companies with SaaS offerings. Many SaaS applications are built around relational databases, and there is demand for relational Database as a Service (DBaaS) systems such as the one described by Curino et al. [1].

On-premise software systems have historically had difficulty handling the inevitable evolution of their database schema. A related issue is management of changes to system configuration and other data stored in the database when the system is upgraded to a newer version. These problems are multiplied in a multi-tenant environment. Traditionally, dedicated software systems handle these issues by creating development and staging copies of the system, where changes are made and tested. After successful tests, the changes are applied to the production environment. These techniques impose significant excess compute and storage capacity requirements that are even more detrimental in a cloud environment, where they decrease the efficiency, elasticity, and scalability of the environment. There is significant motivation for the service provider to find ways to improve these aspects of system operation.

Previously, we proposed a system for versioning the schema and data in an online transaction processing (OLTP) database [2]. This work described extensions to a Relational Database Management System (RDBMS) to allow the data and schema of a database instance to be versioned, so that the schema and the configuration data stored in the instance can be changed and the changes can be tested while the instance is online, without creating a copy of the database. Once updates and testing are complete and ready to be placed into the production environment, the RDBMS should facilitate migration of the changes into the system without downtime. We are expanding this initial idea to a paradigm where versions of the production schema and data (which we call branches) are created. A branch consists of some set of schema and/or data deltas, or differences, from the parent version. A branch can be committed into the version of the database from which the branch was created when work on the branch is complete; this applies the deltas from the branch to the parent version. Branches can also be used for testing or other purposes and then be abandoned if their changes are not meant to be applied to the production system.

We refer to the production version as the trunk. A single major version branch, which we call the head, can be created from the trunk; this branch should be used to install and test version upgrades of the system. It is also possible to create minor version branches from the trunk or the head branches. Minor versions should be used to create and test customizations that are independent of the system software.

Each branch provides isolation of the deltas contained in that version; that is, a user that is not connected to a branch will not see any of the changes made in that branch. The exception is that changes made in any version are visible in all branches created from that version, unless there are changes made in the branch that mask the changes from the parent version.

With this versioning system in place, it will be possible to create a branch of a production database instance and make schema and data changes while queries are being run against the production instance, a very important factor in real-time SaaS operations. A key requirement of the system is that it must add little or no overhead to queries against the production instance. Another requirement is that data that is not changed in a branch should be retrieved from the version from which the branch was created. A key use case for this method of versioning is to prepare a system for a new software release. Any schema changes required by the new software version can be made in the branch, and configuration data can be updated as appropriate. Regression tests can be run, and then the branch can be committed into the trunk version at the same time as the software is updated to the new version.

A similar use case is to allow customizations of a production site. A branch can be created, schema changes and code changes can be made, and the changes can be tested without impacting users of the trunk database. Once the tests have been completed, the branch can be merged into the trunk, so that all schema and data changes can be applied seamlessly to the production environment in an atomic operation. Branches can also be used for additional testing and abandoned when they are no longer needed. This allows potentially destructive changes to be made to the data in an environment where the application can commit them to the branch using normal operations, but the changes can be isolated and discarded easily, without adversely affecting the production environment. All of these scenarios allow the branch version to be tested adequately without requiring that the full production database be copied or that sampling techniques be used to copy a subset of the data.

**Contribution:** We are providing a detailed exploration of one aspect of the versioning system we previously proposed, the concurrent storage of multiple versions of data in relational tables. We believe that such a versioning system will be particularly advantageous in cloud computing environments such as SaaS or DBaaS systems, to improve operational efficiency, scalability, and elasticity of the system, especially in multi-tenant environments. It will allows changes to be made and tested in isolation, while still accessing the production data. These changes do not require a copy of the database to be made; for a multi-tenant SaaS system, this can be a significant savings of storage, server resources, and time.

**Scope:** We explore two different mechanisms for storing the data for the branch versions of a multi-version database. We also describe how these mechanisms can be implemented in the InnoDB storage engine. We concentrate on the storage system for data versions and do not explore other aspects of the versioning system, including versioning schema changes. We present the results of experiments that compare the two storage techniques on a variety of workloads, using the metric of pages read by queries to evaluate their relative efficiency.

**Organization:** First, we will provide some background on previous research, and then briefly describe how table data is stored in InnoDB. We then describe two different options for storing multiple versions of row data for a table. Next, we compare the performance of these two different options for some different workloads to assess which of the options might be a better choice. Finally, we offer conclusions.

## 2 Background

Previous research has suggested solutions to similar problems, mostly in the realm of Computer-Aided Design (CAD) and Geospatial Information Systems (GIS). One suggested approach was to use long-running transactions to isolate changes. A paper by Agarwal et al. describes the Oracle Workspace Manager, which is a framework to support long-running transactions within the database [3]. The authors adopted similar terminology borrowed from source code version control systems. The underlying implementation augments tables with version information and defines views to mask the versions and provide SQL independence for system applications, so the applications do not need to be modified to use the versioning system.

Subsequent work by the same group (Chatterjee et al. [4]) provides more detail on their approach. When versioning is enabled for a table, the table is renamed, a version number column is added, and a view is created with the original table name to hide the new column. The system they describe provides comprehensive support for data versioning, but it does not allow for changes to the database schema in versions. Some aspects of the Workspace Manager are similar to the idea of minimal data sets that we introduced in a previous paper [5], in that a version can read data from any of the parents in its version hierarchy. However, the Workspace Manager does not provide support for schema versioning, and changes to data in parent versions are not automatically available in the child. Instead, the user of a version must request a refresh of the versions data. Another significant drawback is that when versioning is enabled for a table, the version number column must be added. In the InnoDB storage engine, this can be a fairly disruptive process if the table is large. The new column must also be made part of the primary key, which will increase the size of every index on the table (and require rebuilding every index).

There has been significant research on the topic of schema evolution in relational databases. Much of this work focused on object oriented databases (OODB), including the work by Rundensteiner et al. [6] [7] and the temporal versioning research by Grandi et al. [8] and Galante et al. [9]. There were a number of papers by Zaniolo, Moon, Curino, et al. on quantifying and supporting schema evolution in RDBMSes [10] [11] [12] [13]. Most of this research is based on temporal evolution of the schema and on mapping queries that use older versions of the schema into the current data store. This is somewhat different from the problem we are considering, where multiple versions are simultaneously active.

Recent research reinforces our assertion that relational databases are in use in cloud environments, and that the problem we examine is still very relevant. In addition to the recent work by Curino, et al. [1][12], there is work on accommodating multiple tenants in a RDBMS by Schiller et al. [14][15], and on scaling and load balancing for multi-tenant databases by Das, et al. [16], Moon, et al. [17] and Quamar et al. [18]. There was specific work on schema mapping techniques for multi-tenant SaaS by Aulbach et al. [19].

### 2.1 InnoDB Basics

The InnoDB storage engine provides full transactional support, implementing all the ACID properties required for reliable transactions (atomicity, consistency, isolation, and durability). InnoDB provides row-level locking, and it uses a Multi-Version Concurrency Control (MVCC) mechanism to prevent the writer of a row from blocking all readers until the writers transaction commits.

Jeremy Cole has developed an excellent description of the InnoDB data structures and internals, from which much of this summary information was gleaned [20]. InnoDB uses a standard B+-tree data structure to index data stored in database tables. The engine stores the actual table data using a clustered index; this same storage technique is used by Sybase and SQL Server. All of the row data in an InnoDB table is stored in the leaf entries of the index on the tables primary key. If no such key is defined on the table, InnoDB generates a unique six-byte ID for each row and uses this as the primary key.

This organization of data in InnoDB is the basis for the experiments we will present in the paper. Other databases and other storage engines will likely organize the data differently, and will handle MVCC and indexing differently. For our initial problem domain, we were constrained to use the MySQL database and InnoDB storage engine. In addition, the tools provided by Jeremy Cole [20] enabled detailed analysis of the impact of various changes to the data and to the storage engine.

## 3 Data Version Storage

In our initial examination of this problem, it was tempting to consider just handling the activity in a branch as a single long-lived transaction, as suggested by Agarwal [3]. However, with this approach interactions with the branch would not provide the normal transactional semantics, which would be undesirable. Also, as explained we were initially constrained to MySQL and InnoDB. With an InnoDB data store, it is not feasible to have activity in a single long-running transaction, due to InnoDBs implementation of MVCC. A transaction maintained over a long period of time causes queries against that transaction to scan many undo log records. It could also lead to exhaustion of the space available in the rollback segment, since undo log records must be retained until they are no longer needed to provide the data at the start of the long-running transaction. Also, updates to rows in the transaction will lock them until the transaction commits. Finally, MVCC does not accommodate versioning of schema changes.

To provide a workable versioning system that will support long-lived data versions, normal transaction semantics within each version, and eventually allow schema changes within versions, we are implementing a system somewhat similar to the one described by Chatterjee et al. [4]; rows of the table are augmented with a version, and we provide a simple version hierarchy where changes in versions above a branchs in the hierarchy are visible, but changes in versions in other branches of the hierarchy are not. Changes in a branch are committed into the parent version. At this time, we are concentrating on storing data changes in versions and to push the version handling further into the storage engine, so versioning will have less performance impact and can be better optimized than the view-based implementation described by Chatterjee et al. [4].

A key aspect of the data storage for our versioning system is the goal of storing only the modified data in a version. This increases the work required to answer queries on a branch, but it eliminates the need to copy data from versions higher in the version hierarchy. This helps to satisfy the requirement that the versioning system introduce as little impact on the trunk database as possible. To support this, we introduce the idea of a new unspecified value for a column in a row. If a column has a value of unspecified, that means that the value must be retrieved from its parent row in the version hierarchy. This step is applied recursively until the root (the trunk version) is reached; there are never any unspecified values in the trunk. Implementing this unspecified value concept in InnoDB is relatively simple. Each data record in the clustered index is prefixed by a variable length header which includes a bitmap indicating which of the nullable columns in the row actually contain NULL values, and it includes a list of lengths for the variable-length columns in the row. For non-trunk versions, we simply add a bitmap indicating which of the columns in the row contain unspecified values.

We propose creating a separate copy of the table associated with each version and providing a separate rollback segment for each version. This is our initial implementation approach. This provides good separation of activity in a version from other versions (particularly the trunk). However, fetches from a branch table typically require subsequent fetches from the parents of the table in the version hierarchy to retrieve unspecified values and rows that have not been modified in the branch.

For example, suppose a table is created and populated in the trunk using this SQL:

```
CREATE TABLE t
(   I INTEGER NOT NULL PRIMARY KEY,
    Col1 CHAR(5) NOT NULL,    Col2 CHAR(5) NOT NULL,
    Col3 CHAR(5) NOT NULL );
INSERT INTO t (I, Col1, Col2, Col3) VALUES
    ( 1, A, B, C), ( 5, D, E, F), (23, G, H, I);
```

Then suppose a branch is created, and the following SQL is executed:

```
UPDATE t SET Col2 = H1 WHERE I = 23;
```

This would produce the pair of tables shown on the left side of Fig. 1. Because InnoDB does support a somewhat variable format for data rows, we have also explored storing the data for all versions of a row in the same table. We add a field to the row header to encode the version number of the row. Records for the same key are linked in ascending order of version, with the trunk record first. The table on the right side of Fig. 1 shows a page containing data for a key A in the trunk version, a branch of the trunk, the head version, and a branch of the head.

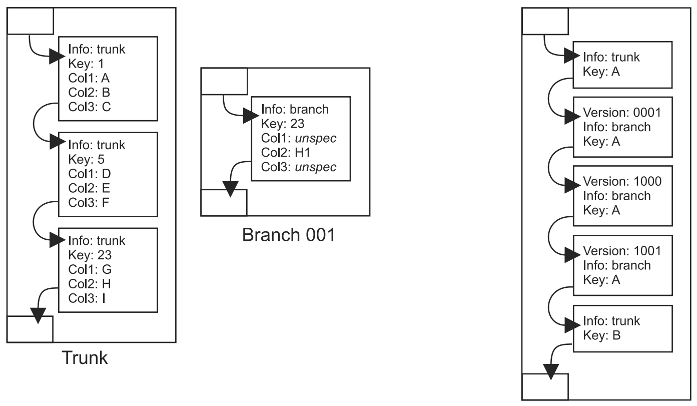


**Fig. 1.** Separate tables to store versions, and consolidation of multiple data versions

When a key has data in multiple versions, we will attempt to keep the rows for that key collocated on a single page. If we split a page, we will attempt to move all records for the same key to the new page. However, if all the records for that key will not fit on a single page, it will be necessary to split the records across pages.

Due to InnoDB's use of clustered indices, this implementation approach requires that we disallow changes to the values of any columns that make up a rows primary key in any branch, or in the trunk if any branches are present. However, even with our initial implementation, we need to impose the same restriction, so that we can retrieve the correct corresponding row from each parent version in the version hierarchy. Given this restriction, we expect this consolidated implementation to provide some advantages. Most importantly, it preserves locality of reference across versions, so that queries that must retrieve data from parents in the version hierarchy will typically have the required data in the current page of memory.

## 4 Comparative Analysis

In this section, we compare the performance of storing version data in separate tables vs. consolidating versions and embedding them in the same table. To perform a quantitative evaluation, we performed experiments using the InnoDB storage engine that is included with the MySQL database, version 5.6.12.

**Hypothesis:** storing the data for the trunk and the branch version consolidated into the same table will be more efficient, when measured in terms of the number of pages required to answer a range scan query on the branch, than storing the data in separate tables. This is expected because the query on the branch will still need to retrieve a significant amount of data from the trunk, and having the data consolidated on the same set of pages should provide a distinct advantage.

### 4.1 Experimental Data

We considered two different configurations of the initial data in a table:

– Populated in sequential order of the cluster key
– Populated in random order of the cluster key

For these two configurations, we considered that all non-key columns in the row were updated. Without modifications to the InnoDB storage engine, it requires the same space per row whether data is specified or not, so modifying only a subset of the non-key columns would not consume less space.

The key values for all rows to be updated in the branch were distributed uniformly through the table. That is, if 10% of the rows were updated, a row was updated every 10 rows. The experiments that follow were simulated using a standard version of MySQL by creating the following table:

```
CREATE TABLE t
(   a INTEGER NOT NULL AUTO_INCREMENT, b INTEGER NOT NULL DEFAULT 0,
    c TIMESTAMP NOT NULL,   d DATETIME NOT NULL,
    e SMALLINT NOT NULL,    f TINYINT NOT NULL,
    g CHAR(26) CHARACTER SET latin1 NOT NULL, PRIMARY KEY t$a$b(a,b) );
```

The *b* column is used to simulate the version number that we would add in the multi-version implementation. This table has a row size of 64 bytes in InnoDB (four bytes for each INTEGER, four for the TIMESTAMP, five for the DATETIME, two for the SMALLINT, one for the TINYINT, 26 for the CHAR, and 18 bytes of overhead per row.)

InnoDB by default has a page size of 16 KB, and there are 128 bytes of overhead per page, plus a byte for every two rows. This gives a theoretical limit of ((16384 - 128) / 64.5), or 252 rows per page. InnoDB prefers to limit the page fill factor (the portion of this space that is actually used for rows) to 15/16, so we would expect to see 236 rows in each full page of the table.

We configured InnoDB in file per table mode and inserted one million rows into this table, using sequentially increasing values for the primary key, a, and 0 for b. We used the *innodb_space* utility (described in [20]) to measure the page usage of the table. This showed that the table occupied 4,238 leaf nodes, five internal nodes, and the root node.

We also considered the case where one million rows were inserted into the table in random order of key values. The page size and row size remain the same, but InnoDB limits the page fill factor more for inserts in random order of the cluster key. After populating the table, *innodb_space* reported 6,041 leaf nodes, 8 internal nodes, and the root node.

### 4.2 Experimental Scenarios

For each combination of the table populated sequentially or randomly, we create a branch, modify some percentage of the tables rows in that branch, and then measure the average number of pages required to process a range scan of 1% of the rows in the table (10,000 rows). We compute this number for both storage cases:

- Storing branch data in a separate table
- Storing trunk and branch data consolidated in the same table

We varied the number of rows modified in the branch between one and ten percent, and we also modified 15% and 20% of the rows.

For example, evaluating the consolidated storage option and modifying 10% of the rows can be simulated using a standard version of MySQL by creating the table and then executing the following SQL:

```
INSERT INTO t SELECT a, 1, c, d, e, f, g FROM t WHERE a MOD 10 = 0;
```

As expected, the resulting table contained 1,100,000 rows, 1,000,000 with b=0 and 100,000 with b=1. Due to InnoDBs page splitting algorithm, *innodb_space* reported 8,474 leaf pages, with an average of 129.8 rows per page. If we considered just trunk rows (those with b=0), this would be equivalent to 118 rows per page.

For the separate table storage option, we would have a second table containing at least 424 leaf pages. This is the best case assuming that the rows were modified in strictly ascending or descending order of the key value, so InnoDB would pack them into the pages with the same efficiency it exhibited in the trunk table.

We evaluate these storage alternatives by measuring the number of pages that are required to satisfy a SELECT query on the table. For this work, we consider only queries that result in range scans of 10% of the rows in the table (10,000 rows).

If one of these queries is submitted in the trunk, it would need to read approximately the following number of pages:

| | |
|---|---|
| **Consolidated** | 10,000 rows / 236 rows/page = 43 pages |
| **Separate** | 10,000 rows / 236 rows/page = 43 pages |

The same query in the branch would scan approximately the following number of pages:

| | |
|---|---|
| **Consolidated** | 10,000 rows / 236 rows/page = 43 pages |
| **Separate** | 43 + (100 / 236) = 44 pages |

In this case, the differences between the two storage alternatives are negligible. The number of pages scanned to answer the range query on the trunk and on the branch is the same, within one page.

## 4.3 Experimental Results

Table 1 shows the resulting number of leaf pages occupied by the table for each method of populating the table and various percentages of rows modified. Looking across the first row, you can see that for the table populated with sequential index values, InnoDB was able to accommodate changes to up to 6% of the rows in the table without splitting any pages. However, after this, the number of pages allocated nearly doubled.

**Table 1.** Number of leaf pages occupied in consolidated storage

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq | 4238 | 4238 | 4238 | 4238 | 4238 | 4238 | 4238 | 8474 | 8474 | 8474 | 8474 | 8474 | 8474 |
| Rand | 6041 | 6100 | 6161 | 6225 | 6268 | 6326 | 6386 | 6460 | 6520 | 6571 | 6621 | 6891 | 7174 |

The behavior when the table was populated in random order of key values was significantly different. This is due to the much less homogeneous packing of rows into individual pages and the typically lower page fill factor. Row two of the table demonstrates very similar trends of monotonically increasing page allocation as the number of rows modified increases.

Tables 2 and 3 show the number of pages read to answer range scan queries over 1% of the rows in the table. Note that in the consolidated storage case, the numbers are the same for queries in the trunk and in the branch, since the rows are stored in the same pages. In the separate storage case, the number of rows required to answer the query in the trunk is unchanged regardless of the number of rows modified in the branch – 43 pages in the sequential insert case, 61 pages in the random insert case.

**Table 2.** Number of pages read to answer range scan query for consolidated storage

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 85 | 85 | 85 | 85 | 85 | 85 |
| Rand | 61 | 61 | 62 | 63 | 63 | 64 | 64 | 65 | 66 | 66 | 67 | 69 | 72 |

**Table 3.** Number of pages read to answer branch range scan query for separate storage

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq | 43 | 44 | 44 | 45 | 45 | 46 | 46 | 46 | 47 | 47 | 48 | 50 | 52 |
| Rand | 61 | 62 | 63 | 63 | 64 | 65 | 65 | 66 | 67 | 67 | 68 | 72 | 76 |

For the separate storage case, the number of pages required to answer a range scan of 1% of the rows in a branch is equal to the number of pages required to answer that question in the trunk, plus a number of pages computed by the following formula:

$$\left\lceil \frac{\%Mod \times NumTrunkRows}{\frac{NumTrunkRows}{NumLeafPages}} \times 0.01 \right\rceil = \lceil \%Mod \times NumLeafPages \times 0.01 \rceil$$

For example, for 5% of the rows modified in the random insert all columns case, the number of rows would be $61 + \lceil .05 \times 6326 \times .01 \rceil = 65$

These results show that one storage mechanism does not provide a clear advantage over the other in all cases. Consolidating storage is very sensitive to

InnoDBs page split algorithm, which eliminates much of the anticipated benefit of maintaining locality of reference for trunk and branch data. The data indicates that for large page fill factors, separate storage is considerably more beneficial, while for smaller fill factors, consolidated storage is somewhat better.

## 5 Conclusions

We believe that the versioning system we have proposed can provide a strong foundation for a MySQL-based schema and data versioning system. This system will allow the simultaneous existence of multiple versions of schema and data in an OLTP database. We believe that this system can significantly simplify the operation of cloud-based systems that use an OLTP database as a central component; upgrades to new versions and customizations of the schema and data can be made and tested in branches of the database, then the changes can be committed and merged into the production version without downtime.

In this work, we have considered a small subset of this versioning system, the storage of multiple versions of the data in a table. We have considered two storage alternatives for data and indices for the versions, one that stores each versions changes in separate tables and another that stores the changes in the same table with the base data. In comparing the two storage alternatives, while there are some situations where the consolidated storage alternative may offer better performance, these situations are not strongly compelling. We did not find that consolidating the branch storage in the same table as the trunk provided the expected significant benefits because it retained locality of reference; this was offset by the increased page splitting. This behavior is sensitive to the storage systems page management algorithm; in future work, we will evaluate these scenarios with other database engines to measure the variability.

## References

1. Curino, C., Jones, E., Popa, R., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N.: RelationalCloud: a Database Service for the Cloud. In: Proc. of Fifth Biennial Conference on Innovative Data Systems Research (CIDR 2011), Asilomar, CA, pp. 235–240 (2011)
2. Wall, B.: Research on a Schema and Data Versioning System. In: Proc. of the VLDB 2011 PhD Workshop, Seattle, WA, USA, pp. 43–48 (September 1, 2011)
3. Agarwal, S., Arun, G., Chatterjee, R., Speckhard, B., Vasudevan, R.: Long Transactions in an RDBMS. In: 26th Annual Conference on Geospatial Information & Technology Association (GITA), San Antonio, TX (2003)
4. Chatterjee, R., Arun, G., Agarwal, S., Speckhard, B., Vasudevan, R.: Using Data Versioning in Database Application Development. In: Proc. of the 26th Intl. Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, pp. 315–325 (2004)
5. Wall, B., Angryk, R.: Minimal Data Sets vs. Synchronized Data Copies in a Schema and Data Versioning System. In: Proc. of PIKM 2011: The 4th ACM Workshop for Ph.D. Students in Information and Knowledge Management, in Association with 20th ACM Conference on Information and Knowledge Management (CIKM 2011), Glasgow, Scotland, pp. 67–73 (October 28, 2011)

6. Ra, Y.-G., Rundensteiner, E.: A Transparent Schema-Evolution System Based on Object-Oriented View Technology. IEEE Transactions on Knowledge and Data Engineering 9(4), 600–624 (1997)
7. Zhang, X., Rundensteiner, E.: The SDCC Framework For Integrating Existing Algorithms for Diverse Data Warehouse Maintenance Tasks. In: 1999 Intl. Database Engineering and Applications Symposium, pp. 206–214 (1999)
8. Grandi, F., Mandreoli, F.: A Formal Model for Temporal Schema Versioning in Object-Oriented Databases. Data & Knowledge Engineering 46(2), 123–167 (2003)
9. Galante, R., Santos, C., Edelweiss, N., Moreira, Á.: Temporal and Versioning Model for Schema Evolution in Object-Oriented Databases. Data and Knowledge Engineering 53(2), 99–128 (2005)
10. Curino, C., Moon, H., Tanca, L., Zaniolo, C.: Schema Evolution in Wikipedia: toward a Web Information System Benchmark. In: 10th Intl. Conference on Enterprise Information Systems (ICEIS 2008) (2008)
11. Moon, H., Curino, C., Zaniolo, C.: Scalable Architecture and Query Optimization for Transaction-Time Databases with Evolving Schemas. In: Proc. of the 2010 Intl. Conference on Management of Data (SIGMOD 2010), pp. 207–218 (2010)
12. Curino, C., Difalla, D., Pavlo, A., Cudre-Maroux, P.: Benchmarking OLTP/Web Databases in the Cloud: the OLTP-bench Framework. In: Proc. of the 4th Intl. Workshop on Cloud Data Management (CloudDB 2012), New York, NY, pp. 17–20 (2012)
13. Curino, C., Moon, H., Deutsch, A., Zaniolo, C.: Automating the Database Schema Evolution Process. VLDB Journal 22(1), 73–98 (2013)
14. Schiller, O., Schiller, B., Brodt, A., Mitschang, B.: Native Support of Multi-tenancy in RDBMS for Software as a Service. In: Proc. of the 14th Intl. Conference on Extending Database Technology (EDBT 2011), New York, NY, pp. 117–128 (2011)
15. Schiller, O., Cipriani, N., Mitschang, B.: ProRea: Live Database Migration for Multi-tenant RDBMS with Snapshot Isolation. In: Proc. of the 16th Intl. Conference on Extending Database Technology (EDBT 2013), New York, NY, pp. 53–64 (2013)
16. Das, S., Agrawal, D., El Abbadi, A.: ElasTraS: An Elastic, Scalable, and Self-managing Transactional Database for the Cloud. ACM Transactions on Database Systems (TODS) 38(1), Article 5 (April 2013)
17. Moon, H., Hacígümüş, H., Chi, Y., Hsiung, W.-P.: SWAT: a Lightweight Load Balancing Method for Multitenant Databases. In: Proc. of the 16th Intl. Conference on Extending Database Technology (EDBT 2013), pp. 65–76 (2013)
18. Quamar, A., Kumar, K.A., Deshpande, A.: SWORD: Scalable Workload-Aware Data Placement for Transactional Workloads. In: Proc. of the 16th Intl. Conference on Extending Database Technology (EDBT 2013), pp. 430–441 (2013)
19. Aulbach, S., Grust, T., Jacobs, D., Kemper, A., Rittinger, J.: Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques. In: Proc. of the 2008 ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 2008), Vancouver, BC, Canada, pp. 1195–1206 (2008)
20. Cole, J.: On learning InnoDB: A journey to the core. Blog (2013), http://blog.jcole.us/2013/01/02/on-learning-innodb-a-journey-to-the-core/

# An Empirical Approach to Query-Subquery Nets with Tail-Recursion Elimination

Son Thanh Cao[1,2] and Linh Anh Nguyen[2,3]

[1] Faculty of Information Technology, Vinh University
182 Le Duan Street, Vinh, Nghe An, Vietnam
soncto@vinhuni.edu.vn

[2] Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

[3] Faculty of Information Technology
VNU University of Engineering and Technology
144 Xuan Thuy, Hanoi, Vietnam

**Abstract.** We propose a method called QSQN-TRE for evaluating queries to Horn knowledge bases by integrating Query-Subquery Nets with a form of tail-recursion elimination. The aim is to improve the QSQN method by avoiding materialization of intermediate results during the processing. We illustrate the efficiency of our method by empirical results, especially for tail-recursive cases.

## 1 Introduction

Query optimization has received much attention from researchers in the database community. Several optimization methods and techniques have been developed to improve performance of query evaluation. One of them is to reduce the number of materialized intermediate results during the processing by using the tail-recursion elimination. The general form of recursion requires the compiler to allocate storage on the stack at run-time. Such a memory consumption may be very costly. A call is tail-recursive if no work remains to be done after the call returns. Tail recursion is a special case of recursion that is semantically equivalent to the iteration construct, so a tail-recursive program can be compiled as efficiently as iterative programs.

This work studies optimizing query evaluation for Horn knowledge bases.

### 1.1 Related Work

Horn knowledge bases are a generalization of Datalog deductive databases as they allow function symbols and do not require the range-restrictedness condition. Researchers have developed a number of evaluation methods for Datalog deductive databases such as the top-down methods QSQ [15,1], QSQR [15,1,9], QoSaQ [16], QSQN [10] and the bottom-up method Magic-Set [1,2]. By Magic-Set we mean the evaluation method that combines the magic-set transformation with the improved semi-naive evaluation method. In [9], Madalińska-Bugaj and Nguyen generalized the QSQR method for Horn knowledge bases. Some authors also extended the magic-set technique together with the breadth-first approach for Horn knowledge bases [12,8]. One can also try to adapt computational procedures of logic programming that use tabled SLD-resolution [14,16,17] for evaluating queries to Horn knowledge bases.

In [10], we formulated the Query-Subquery Nets (QSQN) framework for evaluating queries to Horn knowledge bases. The aim was to increase efficiency of query processing by eliminating redundant computation, increasing flexibility and reducing the number of accesses to the secondary storage. The preliminary comparison between QSQN, QSQR and Magic-Set reported in [3] justifies the usefulness of QSQN.

In [13], Ross proposed an optimization technique that integrates Magic-Set with a form of tail-recursion elimination. It improves the performance by not representing intermediate results, as can be seen in the following example.

*Example 1.* This example shows the inefficiency of a logic program without a tail-recursive evaluation. It is a modified version of Example 1.1 from [13]. Consider the following positive logic program $P$:

$$\begin{array}{l} p(x, y) \leftarrow e(x, z), p(z, y) \\ p(n, x) \leftarrow t(x). \end{array}$$

where $p$ is an *intensional* predicate, $e$ and $t$ are *extensional* predicates, $n$ is a natural number (constant) and $x$, $y$, $z$ are variables. Let $p(1, x)$ be the query, $m$ a natural number, and let the *extensional* instance $I$ for $e$ and $t$ be as follows:

$$\begin{array}{l} I(t) = \{(1), (2), \ldots, (m-1), (m)\}, \\ I(e) = \{(1, 2), (2, 3), \ldots, (n-1, n), (n, 1)\}. \end{array}$$

In order to answer the query, a method such as QSQR, QSQN, Magic-Set would evaluate every tuple of the form $p(i, j)$, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Thus, it stores a set of $n \times m$ tuples, but many of them are not closely related to the query in question. As we shall see, for answering the query $p(1, x)$, we do not need to evaluate $p(i, j)$ for $i > 1$ if we apply a tail-recursive evaluation. We only need to evaluate $p(1, j)$ for $1 \leq j \leq m$ and additional tuples for some newly introduced relations. Thus, the total number of evaluated tuples is smaller than that of the standard approach. $\triangleleft$

## 1.2 Our Contributions

In this paper, we propose a method called QSQN-TRE for evaluating queries to Horn knowledge bases. The method integrates the QSQN method from [10] with a form of tail-recursion elimination.

Our method has many advantages: it reduces the number of evaluated intermediate tuples/subqueries during processing, increases flexibility and eliminates redundant computation. Furthermore, since unnecessary intermediate results are not stored, it usually performs better than the QSQN method for the case as in Example 1. To deal with function symbols, we use a term-depth bound for atoms and substitutions occurring in the computation and propose to use iterative deepening search which iteratively increases the term-depth bound. Similarly to the QSQN method, our new method allows various control strategies such as Depth-First Search (DFS), Improved Depth-First Search (IDFS) and Disk Access Reduction (DAR), which have been proposed in [10,3,5].

## 2 Preliminaries

We assume that the reader is familiar with the notions of *term, atom, predicate, substitution, unification, mgu (most general unifier)* and related ones. We refer the reader to [1,11] for further reading.

We classify each predicate either as *intensional* or as *extensional.* A *generalized tuple* is a tuple of terms, which may contain function symbols and variables. A *generalized relation* is a set of generalized tuples of the same arity.

A *program clause* is a formula of the form $(A \vee \neg B_1 \vee \ldots \vee \neg B_n)$ with $n \geq 0$, written as $A \leftarrow B_1, \ldots, B_n$, where $A, B_1, \ldots, B_n$ are atoms. $A$ is called the *head*, and $B_1, \ldots, B_n$ the *body* of the program clause. If $p$ is the predicate of $A$ then the program clause is called a program clause defining $p$.

A *positive* (or *definite*) *logic program* is a finite set of program clauses. From now on, by a "program" we will mean a positive logic program.

A *goal* is a formula of the form $(\neg B_1 \vee \ldots \vee \neg B_n)$, written as $\leftarrow B_1, \ldots, B_n$, where $B_1, \ldots, B_n$ are atoms and $n \geq 0$. If $n = 1$ then the goal is called a *unary goal.* If $n = 0$ then the goal is referred to as the *empty goal.*

Given substitutions $\theta$ and $\delta$, the composition of $\theta$ and $\delta$ is denoted by $\theta\delta$, the domain of $\theta$ is denoted by $dom(\theta)$, the range of $\theta$ is denoted by $range(\theta)$, and the restriction of $\theta$ to a set $X$ of variables is denoted by $\theta_{|X}$. The *term-depth* of an expression (resp. a substitution) is the maximal nesting depth of function symbols occurring in that expression (resp. substitution). Given a list/tuple $\alpha$ of terms or atoms, by $Vars(\alpha)$ we denote the set of variables occurring in $\alpha$.

A *Horn knowledge base* is defined as a pair that consists of a positive logic program $P$ (which may contain function symbols and not be "range-restricted") for defining *intensional* predicates and a *generalized extensional instance* $I$, which is a function mapping each extensional $n$-ary predicate to an $n$-ary generalized relation.

A *query* to a Horn knowledge base $(P, I)$ is a formula $q(\overline{x})$, where $q$ is an $n$-ary *intensional* predicate and $\overline{x}$ is a tuple of $n$ pairwise different variables. An *answer* to the query is an $n$-ary tuple $\overline{t}$ of terms such that $P \cup I \models q(\overline{t})$, treating $I$ as the corresponding set of atoms of the extensional predicates.

**Definition 1 (Tail-recursion).** A program clause $\varphi_i = (A_i \leftarrow B_{i,1}, \ldots, B_{i,n_i})$, for $n_i > 0$, is said to be *recursive* whenever some $B_{i,j}$ $(1 \leq j \leq n_i)$ has the same predicate as $A_i$. If $B_{i,n_i}$ has the same predicate as $A_i$ then the clause is *tail-recursive.* $\lhd$

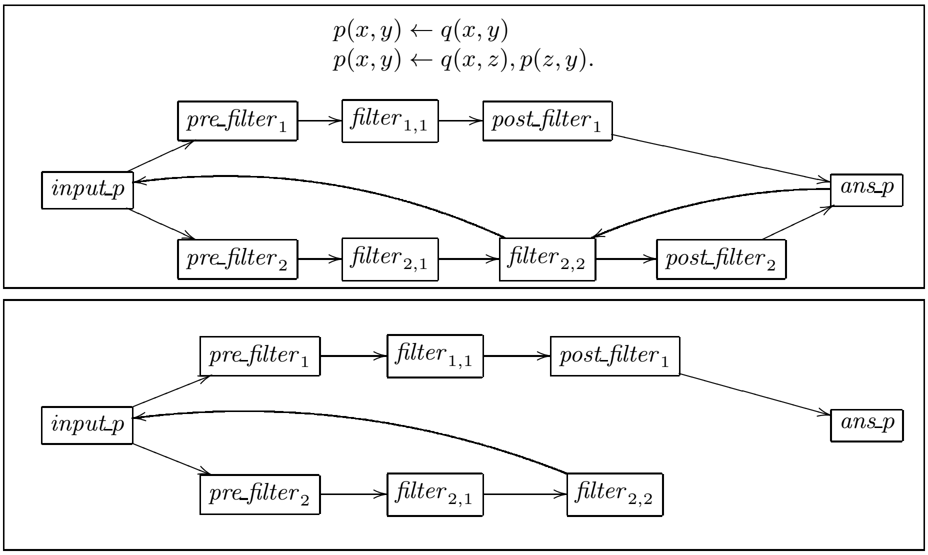


**Fig. 1.** The QSQ topological structure and the QSQN-TRE topological structure of the program given in Example 2

## 3 QSQ-Nets with Tail-Recursion Elimination

In this section we specify the notion QSQN-TRE (QSQ-net with tail-recursion elimination) and describe our QSQN-TRE evaluation method for Horn knowledge bases. QSQN-TRE is an extension of QSQN introduced in [11,10] and can be viewed as a flow control network for determining which set of tuples/subqueries should be processed next.

*Example 2.* This example is taken from [10]. The upper part of Figure 1 illustrates a logic program and its QSQ topological structure, where $p$ is an *intensional* predicate, $q$ is an *extensional* predicate and $x$, $y$, $z$ are variables. $\lhd$

In what follows, $P$ is a positive logic program and all $\varphi_1, \ldots, \varphi_m$ are the program clauses of $P$, with $\varphi_i = (A_i \leftarrow B_{i,1}, \ldots, B_{i,n_i})$, for $1 \leq i \leq m$ and $n_i \geq 0$. The following definition shows how to make a QSQN-TRE structure from the given positive logic program $P$.

**Definition 2 (QSQN-TRE structure).** A *query-subquery net structure with tail-recursion elimination* (QSQN-TRE structure for short) of $P$ is a tuple $(V, E, T)$ such that:

- $T$ is a pair $(T_{edb}, T_{idb})$, called the *type* of the net structure.
- $T_{idb}$ is a function that maps each *intensional* predicate to *true* or *false*. (If $T_{idb}(p) =$ *true* then the *intensional* relation $p$ will be computed using tail-recursion elimination).
- $V$ is a set of nodes that includes:
  - *input_p* and *ans_p*, for each *intensional* predicate $p$ of $P$,

  - $\mathit{pre\_filter}_i$, $\mathit{filter}_{i,1}$, $\ldots$, $\mathit{filter}_{i,n_i}$, for each $1 \leq i \leq m$,
  - $\mathit{post\_filter}_i$ if either $\varphi_i$ is not tail-recursive or $T_{idb}(p) = \mathit{false}$, for each $1 \leq i \leq m$, where $p$ is the predicate of $A_i$.
- $E$ is a set of edges that includes:
  - $(\mathit{input\_p}, \mathit{pre\_filter}_i)$, for each $1 \leq i \leq m$, where $p$ is the predicate of $A_i$,
  - $(\mathit{pre\_filter}_i, \mathit{filter}_{i,1})$, for each $1 \leq i \leq m$ such that $n_i \geq 1$,
  - $(\mathit{filter}_{i,1}, \mathit{filter}_{i,2})$, $\ldots$, $(\mathit{filter}_{i,n_i-1}, \mathit{filter}_{i,n_i})$, for each $1 \leq i \leq m$,
  - $(\mathit{filter}_{i,n_i}, \mathit{post\_filter}_i)$, for each $1 \leq i \leq m$ such that $n_i \geq 1$ and either $\varphi_i$ is not tail-recursive or $T_{idb}(p) = \mathit{false}$, where $p$ is the predicate of $A_i$,
  - $(\mathit{pre\_filter}_i, \mathit{post\_filter}_i)$, for each $1 \leq i \leq m$ such that $n_i = 0$,
  - $(\mathit{post\_filter}_i, \mathit{ans\_p})$, for each $1 \leq i \leq m$ such that either $\varphi_i$ is not tail-recursive or $T_{idb}(p) = \mathit{false}$, where $p$ is the predicate of $A_i$,
  - $(\mathit{filter}_{i,j}, \mathit{input\_p})$, for all $1 \leq i \leq m$ and $1 \leq j \leq n_i$ such that the predicate $p$ of $B_{i,j}$ is an *intensional* predicate,
  - $(\mathit{ans\_p}, \mathit{filter}_{i,j})$, for each *intensional* predicate $p$ and for all $1 \leq i \leq m$ and $1 \leq j \leq n_i$ such that $B_{i,j}$ is an atom of $p$ and either $\varphi_i$ is not tail-recursive or $T_{idb}(p) = \mathit{false}$.
- $T_{edb}$ is a function that maps each $\mathit{filter}_{i,j} \in V$ such that the predicate of $B_{i,j}$ is *extensional* to *true* or *false*. (If $T_{edb}(\mathit{filter}_{i,j}) = \mathit{false}$ then subqueries for $\mathit{filter}_{i,j}$ are always processed immediately without being accumulated at $\mathit{filter}_{i,j}$). ◁

From now on, $T(v)$ denotes $T_{edb}(v)$ if $v$ is a node $\mathit{filter}_{i,j}$ such that $B_{i,j}$ is an *extensional* predicate, and $T(p)$ denotes $T_{idb}(p)$ for an *intensional* predicate $p$. Thus, $T$ can be called a *memorizing type* for *extensional* nodes (as in QSQ-net structures), and a *tail-recursion-elimination type* for *intensional* predicates.

We call the pair $(V, E)$ the QSQN-TRE topological structure of $P$ w.r.t. $T_{idb}$. The lower part of Figure 1 illustrates the QSQN-TRE topological structure of the positive logic program given in Example 2 w.r.t. the $T_{idb}$ with $T_{idb}(p) = \mathit{true}$.

We now specify the notion QSQN-TRE and the related ones. We also show how the data is transferred through the edges in QSQN-TRE.

**Definition 3 (QSQN-TRE).** A *query-subquery net with tail-recursion elimination* (QSQN-TRE for short) of $P$ is a tuple $N = (V, E, T, C)$ such that $(V, E, T)$ is a QSQN-TRE structure of $P$, $C$ is a mapping that associates each node $v \in V$ with a structure called the *contents* of $v$, and the following conditions are satisfied:

- If either ($v = \mathit{input\_p}$ and $T(p) = \mathit{false}$) or $v = \mathit{ans\_p}$ then $C(v)$ consists of:
  - $\mathit{tuples}(v)$ : a set of generalized tuples of the same arity as $p$,
  - $\mathit{unprocessed}(v, w)$ for $(v, w) \in E$: a subset of $\mathit{tuples}(v)$.
- If $v = \mathit{input\_p}$ and $T(p) = \mathit{true}$ then $C(v)$ consists of:
  - $\mathit{tuple\_pairs}(v)$: a set of pairs of generalized tuples of the same arity as $p$,
  - $\mathit{unprocessed}(v, w)$ for $(v, w) \in E$: a subset of $\mathit{tuple\_pairs}(v)$.
- If $v = \mathit{pre\_filter}_i$ then $C(v)$ consists of:
  - $\mathit{atom}(v) = A_i$ and $\mathit{post\_vars}(v) = \mathit{Vars}((B_{i,1}, \ldots, B_{i,n_i}))$.
- If $v = \mathit{post\_filter}_i$ then $C(v)$ is empty, but we assume $\mathit{pre\_vars}(v) = \emptyset$.

– If $v = \mathit{filter}_{i,j}$ and $p$ is the predicate of $B_{i,j}$ then $C(v)$ consists of:
  - $\mathit{kind}(v) = \mathit{extensional}$ if $p$ is extensional, and
    $\mathit{kind}(v) = \mathit{intensional}$ otherwise,
  - $\mathit{pred}(v) = p$ (called the predicate of $v$) and $\mathit{atom}(v) = B_{i,j}$,
  - $\mathit{pre\_vars}(v) = \mathit{Vars}((B_{i,j}, \ldots, B_{i,n_i}))$ and
    $\mathit{post\_vars}(v) = \mathit{Vars}((B_{i,j+1}, \ldots, B_{i,n_i}))$,
  - $\mathit{subqueries}(v)$: a set of pairs of the form $(\bar{t}, \delta)$, where $\bar{t}$ is a generalized tuple of the same arity as the predicate of $A_i$ and $\delta$ is an idempotent substitution such that $\mathit{dom}(\delta) \subseteq \mathit{pre\_vars}(v)$ and $\mathit{dom}(\delta) \cap \mathit{Vars}(\bar{t}) = \emptyset$,
  - $\mathit{unprocessed\_subqueries}(v) \subseteq \mathit{subqueries}(v)$,
  - in the case $p$ is *intensional*:
    $\mathit{unprocessed\_subqueries_2}(v) \subseteq \mathit{subqueries}(v)$,
    $\mathit{unprocessed\_tuples}(v)$: a set of generalized tuples of the same arity as $p$;
  - if $v = \mathit{filter}_{i,n_i}$, $\mathit{kind}(v) = \mathit{intensional}$, $\mathit{pred}(v) = p$ and $T(p) = \mathit{true}$ then $\mathit{unprocessed\_subqueries}(v)$ and $\mathit{unprocessed\_tuples}(v)$ are empty (and can thus be ignored).

– If $v = \mathit{filter}_{i,j}$, $\mathit{kind}(v) = \mathit{extensional}$ and $T(v) = \mathit{false}$ then $\mathit{subqueries}(v)$ and $\mathit{unprocessed\_subqueries}(v)$ are empty (and can thus be ignored).

A QSQN-TRE of $P$ is *empty* if all the sets of the form $\mathit{tuple\_pairs}(v)$, $\mathit{tuples}(v)$, $\mathit{unprocessed}(v, w)$, $\mathit{subqueries}(v)$, $\mathit{unprocessed\_subqueries}(v)$, $\mathit{unprocessed\_subqueries_2}(v)$ or $\mathit{unprocessed\_tuples}(v)$ are empty. $\triangleleft$

If $(v, w) \in E$ then $w$ is referred to as a *successor* of $v$. Observe that:

– if $v \in \{\mathit{pre\_filter}_i, \mathit{post\_filter}_i\}$ or $v = \mathit{filter}_{i,j}$ and $\mathit{kind}(v) = \mathit{extensional}$ then $v$ has exactly one successor, which we denote by $\mathit{succ}(v)$;
– if $v$ is $\mathit{filter}_{i,n_i}$ with $\mathit{kind}(v) = \mathit{intensional}$, $\mathit{pred}(v) = p$ and $T(p) = \mathit{true}$ then $v$ has exactly one successor, which we denote by $\mathit{succ}_2(v) = \mathit{input\_p}$;
– if $v$ is $\mathit{filter}_{i,j}$ with $\mathit{kind}(v) = \mathit{intensional}$, $\mathit{pred}(v) = p$ and either $j < n_i$ or $T(p) = \mathit{false}$ then $v$ has exactly two successors: $\mathit{succ}(v) = \mathit{filter}_{i,j+1}$ if $n_i > j$; $\mathit{succ}(v) = \mathit{post\_filter}_i$ otherwise; and $\mathit{succ}_2(v) = \mathit{input\_p}$.

By a *subquery* we mean a pair of the form $(\bar{t}, \delta)$, where $\bar{t}$ is a generalized tuple and $\delta$ is an idempotent substitution such that $\mathit{dom}(\delta) \cap \mathit{Vars}(\bar{t}) = \emptyset$. The set $\mathit{unprocessed\_subqueries_2}(v)$ (resp. $\mathit{unprocessed\_subqueries}(v)$) contains the subqueries that were not transferred through the edge $(v, \mathit{succ}_2(v))$ (resp. $(v, \mathit{succ}(v))$ – when it exists).

For an *intensional* predicate $p$ with $T(p) = \mathit{true}$, the intuition behind a pair $(\bar{t}, \bar{t}') \in \mathit{tuple\_pairs}(\mathit{input\_p})$ is that:

– $\bar{t}$ is a usual input tuple for $p$, but the intended goal at a higher level is $\leftarrow p(\bar{t}')$,
– any correct answer for $P \cup I \cup \{\leftarrow p(\bar{t})\}$ is also a correct answer for $P \cup I \cup \{\leftarrow p(\bar{t}')\}$,
– if a substitution $\theta$ is a computed answer of $P \cup I \cup \{\leftarrow p(\bar{t})\}$ then we will store in $\mathit{ans\_p}$ the tuple $\bar{t}'\theta$ instead of $\bar{t}\theta$.

We say that a tuple pair $(\bar{t}, \bar{t}')$ is *more general* than $(\bar{t}_2, \bar{t}_2')$, and $(\bar{t}_2, \bar{t}_2')$ is an *instance* of $(\bar{t}, \bar{t}')$, if there exists a substitution $\theta$ such that $(\bar{t}, \bar{t}')\theta = (\bar{t}_2, \bar{t}_2')$.

For $v = \mathit{filter}_{i,j}$ and $p$ being the predicate of $A_i$, the meaning of a subquery $(\bar{t}, \delta) \in \mathit{subqueries}(v)$ is as follows: if $T(p) = \mathit{false}$ (resp. $T(p) = \mathit{true}$) then there exists $\bar{s} \in \mathit{tuples}(\mathit{input\_p})$ (resp. $(\bar{s}, \bar{s}') \in \mathit{tuple\_pairs}(\mathit{input\_p})$) such that for processing the goal $\leftarrow p(\bar{s})$ using the program clause $\varphi_i = (A_i \leftarrow B_{i,1}, \ldots, B_{i,n_i})$, unification of $p(\bar{s})$ and $A_i$ as well as processing of the subgoals $B_{i,1}, \ldots, B_{i,j-1}$ were done, amongst others, by using a sequence of mgu's $\gamma_0, \ldots, \gamma_{j-1}$ with the property that $\bar{t} = \bar{s}\gamma_0 \ldots \gamma_{j-1}$ (resp. $\bar{t} = \bar{s}'\gamma_0 \ldots \gamma_{j-1}$) and $\delta = (\gamma_0 \ldots \gamma_{j-1})_{|Vars((B_{i,j}, \ldots, B_{i,n_i}))}$. Informally, a subquery $(\bar{t}, \delta)$ transferred through an edge to $v$ is processed as follows:

– if $v = \mathit{filter}_{i,j}$, $\mathit{kind}(v) = \mathit{extensional}$ and $\mathit{pred}(v) = p$ then, for each $\bar{t}' \in I(p)$, if $\mathit{atom}(v)\delta = B_{i,j}\delta$ is unifiable with a fresh variant of $p(\bar{t}')$ by an mgu $\gamma$ then transfer the subquery $(\bar{t}\gamma, (\delta\gamma)_{|\mathit{post\_vars}(v)})$ through $(v, \mathit{succ}(v))$.
– if $v = \mathit{filter}_{i,j}$, $\mathit{kind}(v) = \mathit{intensional}$, $\mathit{pred}(v) = p$ and either $T(p) = \mathit{false}$ or ($T(p) = \mathit{true}$ and (either $j < n_i$ or $p$ is not the predicate of $A_i$)) then
  • if $T(p) = \mathit{false}$ then transfer the input tuple $\bar{t}'$ such that $p(\bar{t}') = \mathit{atom}(v)\delta = B_{i,j}\delta$ through $(v, \mathit{input\_p})$ to add a fresh variant of it to $\mathit{tuples}(\mathit{input\_p})$,
  • if $T(p) = \mathit{true}$ and either $j < n_i$ or $p$ is not the predicate of $A_i$ then transfer the input tuple pair $(\bar{t}', \bar{t}')$ such that $p(\bar{t}') = \mathit{atom}(v)\delta = B_{i,j}\delta$ through $(v, \mathit{input\_p})$ to add a fresh variant of it to $\mathit{tuple\_pairs}(\mathit{input\_p})$,
  • for each currently existing $\bar{t}' \in \mathit{tuples}(\mathit{ans\_p})$, if $\mathit{atom}(v)\delta = B_{i,j}\delta$ is unifiable with a fresh variant of $p(\bar{t}')$ by an mgu $\gamma$ then transfer the subquery $(\bar{t}\gamma, (\delta\gamma)_{|\mathit{post\_vars}(v)})$ through $(v, \mathit{succ}(v))$,
  • store the subquery $(\bar{t}, \delta)$ in $\mathit{subqueries}(v)$, and later, for each new $\bar{t}'$ added to $\mathit{tuples}(\mathit{ans\_p})$, if $\mathit{atom}(v)\delta = B_{i,j}\delta$ is unifiable with a fresh variant of $p(\bar{t}')$ by an mgu $\gamma$ then transfer the subquery $(\bar{t}\gamma, (\delta\gamma)_{|\mathit{post\_vars}(v)})$ through $(v, \mathit{succ}(v))$.
– if $v = \mathit{filter}_{i,n_i}$, $\mathit{kind}(v) = \mathit{intensional}$, $\mathit{pred}(v) = p$, $T(p) = \mathit{true}$ and $p$ is the predicate of $A_i$ then transfer the input tuple pair $(\bar{t}', \bar{t})$ such that $p(\bar{t}') = \mathit{atom}(v)\delta = B_{i,n_i}\delta$ through $(v, \mathit{input\_p})$ to add a fresh variant of it to $\mathit{tuple\_pairs}(\mathit{input\_p})$.
– if $v = \mathit{post\_filter}_i$ and $p$ is the predicate of $A_i$ then transfer the answer tuple $\bar{t}$ through $(\mathit{post\_filter}_i, \mathit{ans\_p})$ to add it to $\mathit{tuples}(\mathit{ans\_p})$.

Formally, the processing of a subquery, an input/answer tuple or an input tuple pair in a QSQN-TRE is designed so that:

– every subquery or input/answer tuple or input tuple pair that is subsumed by another one or has a term-depth greater than a fixed bound $l$ is ignored;
– the processing is divided into smaller steps which can be delayed at each node to maximize flexibility and allow various control strategies;
– the processing is done set-at-a-time (e.g., for all the unprocessed subqueries accumulated in a given node).

The procedure `transfer2`$(D, u, v)$ given in [11] specifies the effects of transferring data $D$ through an edge $(u, v)$ of a QSQN-TRE. If $v$ is of the form $\mathit{pre\_filter}_i$ or $\mathit{post\_filter}_i$ or ($v = \mathit{filter}_{i,j}$ and $\mathit{kind}(v) = \mathit{extensional}$ and $T(v) = \mathit{false}$) then the input $D$ for $v$ is processed immediately and an appropriate data $\Gamma$ is produced and transferred through $(v, \mathit{succ}(v))$. Otherwise, the input $D$ for $v$ is not processed immediately, but accumulated into the structure of $v$ in an appropriate way. The function `active-edge`$(u, v)$ given in [11] returns *true* for an edge $(u, v)$ if the data accumulated in $u$ can be processed to produce some data to transfer through $(u, v)$, and returns *false* otherwise. If `active-edge`$(u, v)$ is *true*, the procedure `fire2`$(u, v)$ given in [11][1] processes the data accumulated in $u$ that has not been processed before to transfer appropriate data through the edge $(u, v)$. Both the procedures `fire2`$(u, v)$ and `transfer2`$(D, u, v)$ use a parameter $l$ as a term-depth bound for tuples and substitutions.

Algorithm 2 of [11] presents our QSQN-TRE evaluation method for Horn knowledge bases. It repeatedly selects an active edge and fires the operation for the edge. Such a selection is decided by the adopted control strategy, which can be arbitrary. If there is no tail-recursion to eliminate or $T(p) = \mathit{false}$ for every *intensional* predicate $p$, the QSQN-TRE method reduces to the QSQN evaluation method. See [11] for properties on soundness, completeness and data complexity of the QSQN-TRE method.

## 4 Preliminary Experiments

This section presents our experimental results and a discussion about the performance of the QSQN-TRE evaluation method in comparison with the QSQN method using the IDFS control strategy [5]. All the experiments have been performed using our Java codes [4] and *extensional* relations stored in a MySQL database. The package [4] also contains all the experimental results reported below. In the following tests, we use typical examples that appear in many well-known articles related to deductive databases, including tail/non-tail recursive logic programs as well as logic programs with or without function symbols. Our implementation allows queries of the form $q(\overline{t})$, where $\overline{t}$ is a tuple of terms.

### 4.1 The Settings

The experiments are divided into two stages. All the experimental results reported below are for the first stage.

1. In the first stage, we assume that the computer memory is large enough to load all the involved *extensional* relations and keep all the intermediate relations. During execution of the program, for each operation of reading from a relation (resp. writing a set of tuples to a relation), we increase the counter of read (resp. write) operations on this relation by one. For counting the maximum number of kept tuples/subqueries in the memory, we increase (resp. decrease)

[1] The step 3 in the macro `compute-gamma` for the procedure `fire2` in [11] should be replaced by "**else if** $j < n_i$ or $p$ is not the predicate of $A_i$ **then**".

**Table 1.** A comparison between the QSQN and QSQN-TRE methods w.r.t. the number of read/write operations. The "Reading *inp_/ans_/sup_/edb*" column means the number of read operations from *input/answer/supplement/extensional* relations, respectively. Similarly, the "Writing *inp_/ans_/sup_*" column means the number of write operations to *input/answer/supplement* relations, respectively. The last column shows the maximum number of kept tuples in the memory for each test.

| **Tests** | **Methods** | **Reading (times)** | **Writing (times)** | **Max No. of kept tuples** |
|---|---|---|---|---|
| | | *inp_/ans_/sup_/edb* | *inp_/ans_/sup_* | |
| Test 1 (a) | QSQN | 156 (40+38+57+21) | 58 (20+19+19) | 248 |
| | QSQN-TRE | 100 (40+1+38+21) | 40 (20+1+19) | 97 |
| Test 1 (b) | QSQN | 64 (3+38+21+2) | 21 (1+19+1) | 229 |
| | QSQN-TRE | 100 (40+1+38+21) | 40 (20+1+19) | 781 |
| Test 2 (a) | QSQN | 190 (41+59+69+21) | 69 (20+29+20) | 992 |
| | QSQN-TRE | 101 (40+1+39+21) | 40 (20+1+19) | 151 |
| Test 2 (b) | QSQN | 95 (3+59+31+2) | 31 (1+29+1) | 963 |
| | QSQN-TRE | 151 (60+1+59+31) | 60 (30+1+29) | 3573 |
| Test 3 | QSQN | 43 (5+21+16+1) | 13 (1+9+3) | 101 |
| | QSQN-TRE | 58 (19+15+19+5) | 19 (5+5+9) | 237 |
| Test 4 | QSQN | 56 (15+14+20+7) | 20 (7+6+7) | 136 |
| | QSQN-TRE | 39 (14+4+14+7) | 15 (7+2+6) | 93 |
| Test 5 | QSQN | 403 (101+101+150+51) | 150 (50+50+50) | 10,350 |
| | QSQN-TRE | 253 (101+1+100+51) | 101 (50+1+50) | 600 |
| Test 6 | QSQN | 184 (48+46+66+24) | 67 (23+22+22) | 930 |
| | QSQN-TRE | 126 (48+8+46+24) | 49 (23+4+22) | 566 |
| Test 7 | QSQN | 91 (7+39+25+20) | 25 (3+19+3) | 195 |
| | QSQN-TRE | | | |

the counter of kept tuples by two if a tuple pair is added to (resp. removed from) *tuple_pairs*(*input_p*), otherwise we increase (resp. decrease) it by one. The returned value is the maximum value of this counter.

2. The second stage follows the first one. We will limit the space available in computer memory for storing the tuples/subqueries on each test. This will require load and unload operations on disk when the computer memory is not enough to hold all the relations. The aim of this stage is to estimate the number of disk accesses. This stage is still in progress.

### 4.2 Experimental Results

We compare the QSQN-TRE and QSQN methods with respect to the number of accesses to the intermediate relations and *extensional* relations as well as the number of kept tuples/subqueries in the memory for the following tests.

**Test 1.** Reconsider the logic program from Example 2, where $p$ is an *intensional* predicate, $q$ is an *extensional* predicate, and $x$, $y$, $z$ are variables. Let the *extensional* instance $I$ for $q$ be as follows: $I(q) = \{(a_i, a_{i+1}) \mid 1 \leq i < n\}$, where $a_i$ are constant symbols and $n$ is a natural number.

$$p(x,y) \leftarrow q(x,y)$$
$$p(x,y) \leftarrow q(x,z), p(z,y).$$

We perform this test using the following queries: $(a)\ \ p(a_1, x)$, $\quad (b)\ \ p(x, y)$.

Similar to the discussion in Example 1, in order to answer a query as in the part ($a$) or ($b$), QSQN has to evaluate all tuples of the form $(a_i, a_j)$, where $1 \leq i < j \leq n$. However, for the query $p(a_1, x)$ as in the part ($a$), by applying tail-recursion elimination, QSQN-TRE only needs to evaluate a set of tuples of the form $(a_1, a_j)$ with $1 < j \leq n$. Thus, in this case, the number of evaluated tuples for QSQN-TRE is much smaller than QSQN. In contrast, for queries without any bound parameter such as $p(x, y)$ as in the part ($b$), QSQN-TRE has to evaluate also all the related tuples and may be worse than QSQN. The reason is that, after the processing at node $v = \mathit{filter}_{i,n_i}$ with $p = \mathit{pred}(v)$, if $T(p) = \mathit{true}$, QSQN-TRE produces a set of tuple pairs and accumulates them in $\mathit{tuple\_pairs}(\mathit{input\_p})$, which are not instances of each other. Meanwhile, QSQN adds the answers to $\mathit{tuples}(\mathit{ans\_p})$ for later processing, and also transfers data through $(v, \mathit{input\_p})$ without adding any new tuple to $\mathit{tuples}(\mathit{input\_p})$ because it already contains a fresh variant of $(x, y)$ that is more general than all the other tuples. As the result, in this case, QSQN-TRE may keep more tuples than QSQN. We use $n = 20$ for this test.

**Test 2.** This test uses the logic program $P$ and the queries as in Test 1, but the *extensional* instance $I$ for $q$ is extended to contain cycles as follows, where $a_i$ and $b_i$ are constant symbols:

$$\begin{aligned} I(q) = \{(a_i, a_{i+1}) \mid (1 \leq i < 20)\} \;\cup\; \{(a_{20}, a_1)\} \;\cup \\ \{(a_1, b_1)\} \;\cup\; \{(b_i, b_{i+1}) \mid 1 \leq i < 10\} \;\cup\; \{(b_{10}, a_1)\}. \end{aligned}$$

**Test 3.** This test involves the transitive closure of a binary relation [2,3]. Consider the following logic program $P$, where *path* is an *intensional* predicate, *arc* is an *extensional* predicate, and $x$, $y$, $z$ are variables. The query is $\mathit{path}(x, y)$ and the *extensional* instance $I$ is specified by $I(\mathit{arc}) = \{(1,2), (2,3), \ldots, (9,10)\}$.

$$\begin{aligned} &\mathit{path}(x, y) \leftarrow \mathit{arc}(x, y) \\ &\mathit{path}(x, y) \leftarrow \mathit{path}(x, z), \mathit{path}(z, y). \end{aligned}$$

**Test 4.** This test is taken from [9]. Consider the following program $P$, where $p$, $s$ are *intensional* predicates, $q$ is an *extensional* predicate, and $x$, $y$, $z$ are variables. The query is $s(x)$ and the *extensional* instance $I$ for $q$ consists of the following pairs, where $a - o$ are constant symbols: $I(q) = \{(a, b)$, $(b, c)$, $(c, d)$, $(d, e)$, $(e, f)$, $(f, g)$, $(a, h)$, $(h, i)$, $(i, j)$,$(i, d)$, $(j, k)$, $(k, f)$, $(a, l)$, $(l, m)$, $(l, i)$, $(m, n)$, $(n, o)$, $(n, k)$, $(o, g)\}$.

$$\begin{aligned} p(x, y) &\leftarrow q(x, y) \\ p(x, y) &\leftarrow q(x, z), p(z, y) \\ s(x) &\leftarrow p(a, x). \end{aligned}$$

**Test 5.** This test is taken from Example 1 using $m = 200$, $n = 50$. As shown in Table 1, the maximum number of kept tuples for the QSQN evaluation method is much larger than for the QSQN-TRE evaluation method.

**Test 6.** This test is taken from [3]. Consider the following program $P$ and the following *extensional* instance $I$, where $p$, $q_1$, $q_2$ are *intensional* predicates, $r_1$, $r_2$ are *extensional* predicates, $x$, $y$, $z$ are variables, and $a_i$, $b_{i,j}$ are constant symbols.

– the positive logic program $P$:

$$\begin{aligned} p(x,y) &\leftarrow q_1(x,y) \\ p(x,y) &\leftarrow q_2(x,y) \\ q_1(x,y) &\leftarrow r_1(x,y) \\ q_1(x,y) &\leftarrow r_1(x,z), q_1(z,y) \\ q_2(x,y) &\leftarrow r_2(x,y) \\ q_2(x,y) &\leftarrow r_2(x,z), q_2(z,y). \end{aligned}$$

– the *extensional* instance $I$:

$$\begin{aligned} I(r_1) &= \{(a_i, a_{i+1}) \mid 0 \leq i < 10\} \\ I(r_2) &= \{(a_0, b_{1,j}) \mid 1 \leq j \leq 9\} \cup \\ &\quad \{(b_{i,j}, b_{i+1,j}) \mid 1 \leq i < 9 \\ &\qquad \text{and } 1 \leq j \leq 9\} \cup \\ &\quad \{(b_{9,j}, a_{10}) \mid 1 \leq j \leq 9\}. \end{aligned}$$

– the query: $p(a_0, x)$.

**Test 7.** This test is taken from Test 2 of [5] to show a case with function symbols. It is a non-tail-recursive program. In this case, the QSQN-TRE evaluation method reduces to the QSQN method, and they have the same results. See [5] for details of the logic program and its *extensional* instance.

### 4.3 Discussion

Table 1 shows the comparison between the QSQN and QSQN-TRE evaluation methods. As can be seen in this table, if we use a tail-recursive program with at least a bound parameter either in the query as in Tests $[1(a), 2(a), 5, 6]$ or in the body of a rule that is related to a tail-recursive predicate as in Test 4, by not representing intermediate results during the computation, the QSQN-TRE method usually outperforms the QSQN method. In these cases, as shown in Table 1, the QSQN-TRE method reduces not only the number of accesses to the mentioned relations but also the number of kept tuples/subqueries in the memory in comparison with the QSQN method.

In contrast, for queries without any bound parameter as in Tests $[1(b), 2(b)]$ and for cases with a tail-recursive clause with more than one *intensional* predicate $p$ in the body such that $T(p) = true$ as in Test 3, QSQN-TRE may be worse than QSQN. The explanation is similar to that of Test 1.

## 5 Conclusions

We have proposed the QSQN-TRE method for evaluating queries to Horn knowledge bases. It extends the QSQN method with tail-recursion elimination that allows to avoid materializing intermediate results during the processing. Similarly to QSQN, our new method also allows various control strategies such as DFS, IDFS and DAR [10,3,5].

The experimental results in Table 1 show that QSQN-TRE is better than QSQN for tail-recursive cases with at least a bound parameter in the query, especially for the positive logic program and the query given in Example 1 (as shown for Test 5 in Table 1). The preliminary comparison between QSQN, QSQR and Magic-Set reported in [3] justifies the usefulness of QSQN and hence also the usefulness of QSQN-TRE. As a future work, we will compare the methods in more detail, especially w.r.t. the number of accesses to the secondary storage when the computer memory is limited, as well as apply our method to Datalog-like rule languages for the Semantic Web [6,7].

**Acknowledgments.** This work was supported by Polish National Science Centre (NCN) under Grants No. 2011/02/A/HS1/00395 (for the first author) and 2011/01/B/ST6/02759 (for the second author). The first author would like to thank the Warsaw Center of Mathematics and Computer Science for support. We would also like to express our special thanks to Dr. Joanna Golińska-Pilarek from the University of Warsaw for very helpful comments and suggestions.

## References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1995)
2. Beeri, C., Ramakrishnan, R.: On the power of magic. J. Log. Program. 10, 255–299 (1991)
3. Cao, S.T.: On the efficiency of Query-Subquery Nets: an experimental point of view. In: Proceedings of SoICT 2013, pp. 148–157. ACM (2013)
4. Cao, S.T.: An implementation of the QSQN-TRE evaluation methods (2014), `http://mimuw.edu.pl/~sonct/QSQNTRE14.zip`
5. Cao, S.T., Nguyen, L.A.: An Improved Depth-First Control Strategy for Query-Subquery Nets in evaluating queries to Horn knowledge bases. In: van Do, T., Thi, H.A.L., Nguyen, N.T. (eds.) Advanced Computational Methods for Knowledge Engineering. AISC, vol. 282, pp. 281–296. Springer, Heidelberg (2014)
6. Cao, S.T., Nguyen, L.A., Szalas, A.: The Web ontology rule language OWL 2 RL+ and Its extensions. T. Computational Collective Intelligence 13, 152–175 (2014)
7. Cao, S.T., Nguyen, L.A., Szalas, A.: WORL: a nonmonotonic rule language for the Semantic Web. Vietnam J. Computer Science 1(1), 57–69 (2014)
8. Freire, J., Swift, T., Warren, D.S.: Taking I/O seriously: Resolution reconsidered for disk. In: Naish, L. (ed.) Proc. of ICLP 1997, pp. 198–212. MIT Press (1997)
9. Madalińska-Bugaj, E., Nguyen, L.A.: A generalized QSQR evaluation method for Horn knowledge bases. ACM Trans. on Computational Logic 13(4), 32 (2012)
10. Nguyen, L.A., Cao, S.T.: Query-Subquery Nets. In: Nguyen, N.-T., Hoang, K., Jędrzejowicz, P. (eds.) ICCCI 2012, Part I. LNCS, vol. 7653, pp. 239–248. Springer, Heidelberg (2012)
11. Nguyen, L.A., Cao, S.T.: Query-Subquery Nets (2012), `http://arxiv.org/abs/1201.2564`
12. Ramakrishnan, R., Srivastava, D., Sudarshan, S.: Efficient bottom-up evaluation of logic programs. In: Vandewalle, J. (ed.) The State of the Art in Computer Systems and Software Engineering. Kluwer Academic Publishers (1992)
13. Ross, K.A.: Tail recursion elimination in deductive databases. ACM Trans. Database Syst. 21(2), 208–237 (1996)
14. Tamaki, H., Sato, T.: OLD resolution with tabulation. In: Shapiro, E. (ed.) ICLP 1986. LNCS, vol. 225, pp. 84–98. Springer, Heidelberg (1986)
15. Vieille, L.: Recursive axioms in deductive databases: The query/subquery approach. In: Proceedings of Expert Database Conf., pp. 253–267 (1986)
16. Vieille, L.: Recursive query processing: The power of logic. Theor. Comput. Sci. 69(1), 1–53 (1989)
17. Zhou, N.-F., Sato, T.: Efficient fixpoint computation in linear tabling. In: Proceedings of PPDP 2003, pp. 275–283. ACM (2003)

# Part V
# Spatial and Temporal Data

# Reasoning over Spatial Orientation Relations Using Rules

Sotiris Batsakis, Grigoris Antoniou, and Ilias Tachmazidis

Department of Informatics
University of Huddersfield
Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK
{S.Batsakis,G.Antoniou,Ilias.Tachmazidis}@hud.ac.uk

**Abstract.** Representation of spatial information for the Semantic Web often involves qualitative defined information (i.e., information described using natural language terms such as "Left"), since precise arithmetic descriptions using coordinates and angles are not always available. A basic aspect of spatial information is directional relations, thus embedding directional spatial relations into ontologies along with their semantics and reasoning rules is an important practical issue. This work proposes a new representation for directional spatial information in ontologies by means of OWL properties and reasoning rules in SWRL embedded into the ontology. The proposed representation is based on the combination of object orientations (e.g., same direction or opposite) and cone shaped directional relations of positions using an egocentric reference (e.g., left or right of an object). The proposed representation is to the best of our knowledge a novel one, and in this work, the proposed representation is analysed, implemented and evaluated.

## 1 Introduction

Understanding the meaning of Web information requires formal definitions of concepts and their properties, using the Semantic Web Ontology definition language OWL. OWL provides the means for defining concepts, their properties and their relations, and allows for reasoning over the definitions and the assertions of specific individuals using reasoners such as HermiT. Furthermore, reasoning rules can be embedded into the ontology using the SWRL rule language.

Spatial information is an important aspect of represented objects in many application areas. Spatial information in turn can be defined using quantitative (e.g., using coordinates) and qualitative terms (i.e., using natural language expressions such as "Behind"). Qualitative spatial terms have specific semantics which can be embedded into the ontology using reasoning rules. In previous work [1,6], such a representation is proposed for allocentric (i.e., using an external reference frame, such as North-South) directional relations in OWL.

Current work deals with the case of egocentric directional spatial information, and proposes a new representation for such information. Egocentric directional relations are applied over local reference frames e.g., using terms such as "front" or "left", that are defined with respect to specific objects and the placement of objects relative to these

N. Bassiliades et al. (eds.), *New Trends in Database and Information Systems II*,
Advances in Intelligent Systems and Computing 312, DOI: 10.1007/978-3-319-10518-5_10

points of reference. Egocentric orientation relations are analysed into two sets of relations; The first set represents the directional orientation relation between two objects (e.g., “same” or “opposite” direction). This set of relations is a modified form of the relations proposed in [7]. The second set represents the positional orientation relations in terms of the egocentric reference frame of each object (e.g., “front” or “behind”). This set is a modified form of *OPRA* calculi proposed in [8]. Thus, for example if object B is in front of object A and is directed towards it, the following relations hold: *B opposite A* and *B front-of A*. Both relations correspond to cone shaped regions in the plane, and their definitions and semantics are introduced in the current work. Reasoning is applied on directional orientation relations separately, since orientation of directed objects does not depend on the position of one wrt the other (e.g., an object can be directed to an opposite direction wrt another, and simultaneously can be left, right, front or back of it). On the other hand, reasoning over positional orientation relations combines directional orientation relations as well. Current work proposes a reasoning mechanism for the proposed representation. Properties of the reasoning mechanism are analysed and the mechanism is implemented and evaluated. Furthermore, the implementation is based on OWL axioms and SWRL rules embedded into an ontology, thus it is suitable for Semantic Web applications, since reasoning can be achieved using only standard reasoners that support SWRL such as HermiT [9].

Current work is organized as follows: related work in the field of spatial knowledge representation is discussed in Section 2. The proposed representation is presented in Section 3 and the corresponding reasoning mechanism in Section 4 followed by evaluation in Section 5 and conclusions and issues for future work in Section 6.

## 2 Background and Related Work

Definition of ontologies for the Semantic Web is achieved using the Web Ontology Language OWL[1]. The current W3C standard is the OWL 2[2] language, offering increased expressiveness while retaining decidability of basic reasoning tasks. Reasoning tasks are applied both on the concept and property definitions into the ontology (TBox) and the assertions of individual objects and their relations (ABox). Reasoners include among others Pellet[3], and HermiT[4]. Reasoning rules can be embedded into the ontology using SWRL[5]. To guarantee decidability, the rules are restricted to *DL-safe rules* that apply only on named individuals in the ontology ABox. *Horn Clauses* (i.e., a disjunction of atoms with at most one positive literal), can be expressed using SWRL, since Horn clauses can be written as implications (i.e., $\neg A \lor \neg B ... \lor C$ can be written as $A \land B \land ... \Rightarrow C$).

Qualitative spatial reasoning (i.e., inferring implied relations and detecting inconsistencies in a set of asserted relations) typically corresponds to Constraint Satisfaction problems which are *NP-hard*, but tractable sets (i.e., solvable by polynomial algorithms)

[1] `http://www.w3.org/TR/owl-ref/`
[2] `http://www.w3.org/TR/owl2-overview/`
[3] `http://clarkparsia.com/pellet/`
[4] `http://hermit-reasoner.com/`
[5] `http://www.w3.org/Submission/SWRL/`

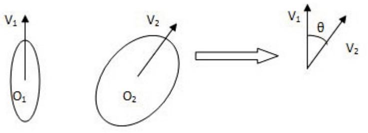
are known to exist [2]. Formal spatial representations have been studied extensively within the Semantic Web community. Relations between spatial entities in ontologies can be topological, directional, or orientation relations. Furthermore, spatial relations are distinguished into qualitative (i.e., relations described using lexical terms such as "Behind") and quantitative (i.e., relations described using numerical values such as "45 degrees Right").

Embedding spatial reasoning into the ontology by means of SWRL rules applied on spatial object properties forms the basis of the SOWL model proposed in [1,6]. Based on the representation proposed in [1] the dedicated Pellet-Spatial reasoner [3] has been extended for directional relations in the CHOROS system [4] (Pellet-Spatial supports only topological relations). None of the above supports orientation relations which typically appear in natural language scene descriptions and in robotics among others. In this work, a representation of orientation relations based on a combination of modified versions of the relations proposed in [7,8] is proposed. The proposed representation is combined with a tractable reasoning mechanism over specific sets of relations, containing basic relations of both sets that are parts of the mechanism. Furthermore, the reasoning mechanism is implemented by means of OWL axioms and SWRL rules that are fully compliant with existing Semantic Web standards and tools.

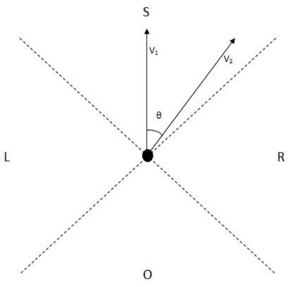
## 3 Spatial Representation

Orientation relations in this work are represented as object properties between OWL objects representing spatial entities. For example if $Object1$ is $Left\ Of\ Object2$, user asserts the binary relation $Object1\ Left\ Object2$, or equivalently *Left(Object1, Object2)*. This approach is similar to the approach used in [1] for cardinal directional relations, as part of the SOWL model. In [7] and [8] orientation relations are defined between objects based on cone-shaped regions around objects. In both cases lines separating the cone-shaped regions are also different relations, similar to the *star calculus* proposed in [5]. Reasoning over *star calculus* have been proven to be *NP-complete*, even if reasoning is restricted over basic relations. On the other hand, lines separating cone-shaped regions can belong to one of these regions instead of being separate basic relations. This calculi is called the *revised star calculus* and it is also presented in [5]. Furthermore reasoning over basic relations of the modified calculi is decided by path consistency and is tractable [5]. This approach is also used for representing cardinal directional relations in [1,6]. In this work, orientation relations correspond to cone-shaped regions, and lines separating the regions belong to only one of these regions. This is the basic difference to relations proposed in [7] and [8].

Note that representations based on projections on orthogonal 2D axis and reasoning over the pairs of relations on these one-dimensional spaces, instead of cone shaped regions in bi-dimensional space have been proposed as well in [2]. Projection based representations have different semantics than the proposed cone-shaped representation, thus it can not be consider as an alternative to it. For example, using the projection based approach, if a point is located far left relatively to another point and slightly behind it, following the projection based approach relations $Left$ and $Behind$ will hold at the horizontal and the vertical axis respectively. Following the cone-shaped approach



(a) Directional Orientation of Objects



(b) Directional Orientation Relations

**Fig. 1.** (a) Egocentric Directional Orientation of Objects (b) Egocentric Directional Orientation Relations

only the relation $Left$ holds, which is conceptually right according to the way humans usually refer to orientation relations.

The basic directional orientation relations are: *Same, Opposite, Left and Right* as presented in Figure 1(b). These relations are abbreviated as *S,O,L,R* respectively. The relations are defined as follows, if 2D vectors $v_1$ and $v_2$ represent the orientation of objects $o_1$ and $o_2$ on the 2D plane, then the angle $\theta_o$ between vector $v_1$ and $v_2$ specify the egocentric orientation relation between $o_1$ and $o_2$ as illustrated in Figure 1(a). Note that lines separating the cone shaped regions belong only to one of these regions according to definitions of relations. Specifically:

$$-\frac{\pi}{4} \leq \theta_o < \frac{\pi}{4} \equiv S(o_1, o_2)$$

$$\frac{\pi}{4} \leq \theta_o < \frac{3\pi}{4} \equiv R(o_1, o_2)$$

$$\frac{3\pi}{4} \leq \theta_o < \frac{5\pi}{4} \equiv O(o_1, o_2)$$

$$\frac{5\pi}{4} \leq \theta_o < \frac{7\pi}{4} \equiv L(o_1, o_2)$$

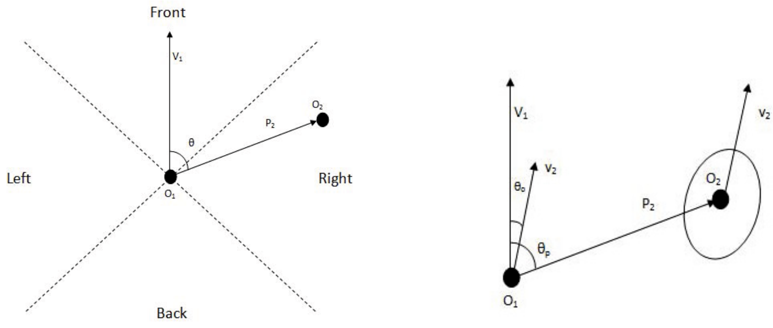
Positional orientation relations are *Front, Back, Left and Right*, presented in Figure 2(a) (note that terms *Back* and *Behind* can be used interchangeably). Lines separating the cone-shaped regions belong to only one of the adjacent regions, as in the case of directional orientation relations. By convention, they also belong to the region to the right of the line, moving clockwise (but other conventions are valid as long as each line belongs to exactly one adjacent cone shaped region). Although positional orientation relations seem similar to directional orientation relations, their definition and semantics are different. Specifically, positional orientation relations are defined as follows, if 2D vector $p_2$ represents the position (and not orientation, as for directional orientation relations) of object $o_2$ on the 2D plane, that has $o_1$ position as reference frame, y-axis defined using $v_1$ (orientation of $o_1$) and x-axis perpendicular to y-axis, then the angle $\theta_p$ between vector $v_1$ and $p_2$ specify the egocentric positional orientation relation between $o_1$ and $o_2$ as illustrated in Figure 2(a). Relations are defined as follows:

$$-\frac{\pi}{4} \leq \theta_p < \frac{\pi}{4} \equiv Front(o_1, o_2)$$

$$\frac{\pi}{4} \leq \theta_p < \frac{3\pi}{4} \equiv Right(o_1, o_2)$$

$$\frac{3\pi}{4} \leq \theta_p < \frac{5\pi}{4} \equiv Back(o_1, o_2)$$

$$\frac{5\pi}{4} \leq \theta_p < \frac{7\pi}{4} \equiv Left(o_1, o_2)$$



(a) Positional Orientation Relations (b) Orientation Example

**Fig. 2.** (a) Positional Orientation Relations (b) Orientation Example

An example presenting both relations is illustrated in Figure 2(b). Directional orientation relation is defined by angle $\theta_o$ between vector $v_1$ representing the orientation of object $o_1$ and vector $v_2$ representing the orientation of object $o_2$. Positional orientation relation is defined by angle $\theta_p$ between vector $v_1$ representing orientation of object $o_1$ and vector $p_2$ representing position of object $o_2$. In this example, object $o_2$ has the same orientation as object $o_1$ and it is located at the right of object $o_1$.

Additional OWL axioms required for the proposed representation; basic relations of each set are pairwise disjoint e.g., $Left$ is disjoint with $Front$. Also $Left$ is $inverse$ of $Right$ (in both sets) and $Front$ is $inverse$ of $Back$. On the other hand, directional orientation relations $same$ and $opposite$ are symmetric. Note also that if two objects are identical then the equality relation holds between them. Instead of using a separate equality relation the OWL $sameAs$ keyword can be used instead for this case as in [1].

## 4 Spatial Reasoning

Reasoning is realized by introducing a set of SWRL rules operating on spatial relations. Reasoners that support DL-safe rules such as HermiT can be used for inference and consistency checking over orientation relations. Defining compositions of relations is a basic part of the spatial reasoning mechanism. Table 1 represents the result of the composition of two directional orientation relations of Figure 1(b) (relations $Same$, $Right$, *Opposite* and *Left*, are denoted by "*S*","*R*","*O*", "*L*" respectively).

**Table 1.** Composition Table for Directional Orientation Relations

| Relations | $S(Same)$ | $R(Right)$ | $O(Opposite)$ | $L(Left)$ |
|---|---|---|---|---|
| $S$ | $S, R, L$ | $S, R, O$ | $R, O, L$ | $S, O, L$ |
| $R$ | $S, R, O$ | $R, O, L$ | $S, O, L$ | $S, R, L$ |
| $O$ | $R, O, L$ | $S, O, L$ | $S, R, L$ | $S, R, O$ |
| $L$ | $S, O, L$ | $S, R, L$ | $S, R, O$ | $R, O, L$ |

Composition Table can be interpreted as follows: if relation $R_1$ holds between object $o_2$ and object $o_1$ and relation $R_2$ holds between object $o_3$ and object $o_2$, then the entry of the Table 1 corresponding to line $R_1$ and column $R_2$ denotes the possible relation(s) holding between object $o_3$ and object $o_1$. For example, if object $o_2$ is at $Same$ direction to object $o_1$ and object $o_3$ is *Right* (in terms of directional orientation) to object $o_2$ then object $o_3$ is *right, same direction or opposite* to object $o_1$. Entries in the above composition table are determined using the following observation: composition of two relations corresponds to the addition of two angles representing the relative directional orientation of $point2$ to $point1$ and $point3$ to $point2$, forming angles $\theta_1$ and $\theta_2$ respectively with the reference (vertical) axis. Combining this observation with the definition of relations in Section 3 the above compositions of Table 1 are defined. So for example composition of $same$ and $opposite$ is interpreted as adding $\frac{7\pi}{4} \leq \theta_1 < \frac{\pi}{4}$ and $\frac{3\pi}{4} \leq \theta_2 < \frac{5\pi}{4}$ which yields $\frac{2\pi}{4} \leq \theta_{12} < \frac{6\pi}{4}$ which corresponds to a cone shaped region into the region defined by the disjunction of *Right, Opposite, Left.*

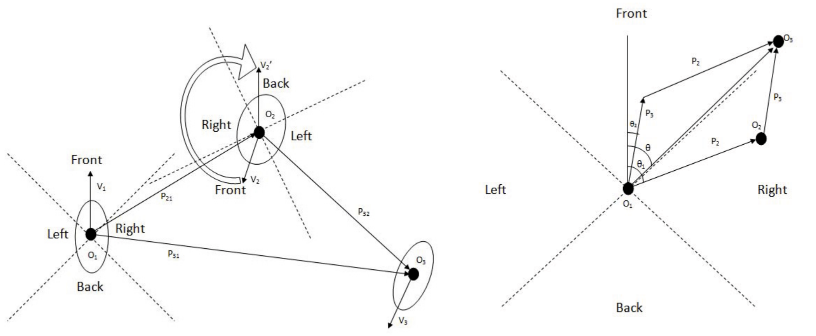
Composing positional orientation relations (Figure 2(a)) requires combining also directional orientation relations (Figure 1(b)). Specifically, composing orientation relations can be defined as follows: if object *o1* is related with object *o2* with directional orientation relation $R_{o_{21}}$ and positional orientation relation $R_{p_{21}}$ and object *o2* is related with object *o3* with directional orientation relation $R_{o_{32}}$ and positional orientation relation $R_{p_{32}}$ then between object *o1* and object *o3* the directional orientation relation $R_{o_{31}}$ is defined using the compositions of Table 1 as: $R_{o_{31}} \equiv R_{o_{21}} \circ R_{o_{32}}$ ($\circ$ denotes composition).

Composition of positional relations is more complex: when composing relations $R_{p_{21}}$ and $R_{p_{32}}$, the fact that object *o2* may have a different directional orientation wrt object *o1* must be also taken into account. For example, if object $o2$ is $Right$ of object $o1$, object $o3$ is $Left$ of object $o2$, but because object $o2$ is $opposite$ of object $o1$ (see Figure 3(a)), directional orientation of objects $o1$ and $o2$ must be aligned, before composing positional relations. Specifically, after rotating object $o2$ wrt object $o1$ (i.e., aligning their directional orientation, by changing direction of object $o2$ from $v_2$ to $v_2\prime$), then object $o3$ is not considered to be located $Left$ of object $o2$, but $Right$, $Front$ or $Back$ of object $o2$ (see Table 2). Then we can compose positional orientation relations and infer possible relations holding between objects $o1$ and $o3$.

Intuitively, for the composition of $R_{p_{21}}$ and $R_{p_{32}}$ object *o1* is now the reference point for both object *o2* and object *o3*, thus before composing the two relations, $R_{p_{32}}$ must be adjusted to the reference frame of object *o1* and this is achieved by performing the rotation specified by relation $R_{o_{21}}$. The result of this rotation, which will be described in detail is denoted by $R_{o_{21}} \Diamond R_{p_{32}}$. The resulting relation which is a positional orientation relation of Figure 2(a) can be composed with relation $R_{p_{21}}$, thus $R_{p_{31}} \equiv R_{p_{21}} \circ (R_{o_{21}} \Diamond R_{p_{32}})$. Since rotation is defined as an addition of two angles, the result of the rotation is similar to compositions of Table 1. Specifically, rotations are defined in Table 2. Given a directional orientation relation $R_o$ and a positional orientation relation $R_p$, each entry in Table 2 corresponding to row $R_o$ and column $R_p$ represent the result of the rotation of relation $R_p$ with respect to relation $R_o$, yielding a set of positional orientation relations.

**Table 2.** Rotation Table for Positional Orientation with respect to Directional Orientations

| Relations | $F(Front)$ | $R(Right)$ | $B(Back)$ | $L(Left)$ |
|---|---|---|---|---|
| $S(Same)$ | $F, R, L$ | $F, R, B$ | $R, B, L$ | $F, B, L$ |
| $R(Right)$ | $F, R, B$ | $R, B, L$ | $F, B, L$ | $F, R, L$ |
| $O(Opposite)$ | $R, B, L$ | $F, B, L$ | $F, R, L$ | $F, R, B$ |
| $L(Left)$ | $F, B, L$ | $F, R, L$ | $F, R, B$ | $R, B, L$ |



(a) Composition Example (b) Positional Composition Example

**Fig. 3.** (a) Composition Example (b) Positional Composition Example

After performing the rotation of the second positional orientation relation, the two positional orientation relations can be composed. Table 3 represents the result of the composition of two positional orientation relations of Figure 2(a) (relations $Front$, $Right$, *Back* and *Left*, are denoted by "$Fr$","$Ri$","$Ba$", "$Le$" respectively, and $All$ denotes the disjunction of all relations).

**Table 3.** Composition Table for Positional Orientation Relations

| Relations | $Fr$ | $Ri$ | $Ba$ | $Le$ |
|---|---|---|---|---|
| $Fr$ | $Fr$ | $Fr, Ri$ | $All$ | $Fr, Le$ |
| $Ri$ | $Fr, Ri$ | $Ri$ | $Ri, Ba$ | $All$ |
| $Ba$ | $All$ | $Ri, Ba$ | $Ba$ | $Ba, Le$ |
| $Le$ | $Fr, Le$ | $All$ | $Ba, Le$ | $Le$ |

Composition Table can be interpreted as follows: if relation $R_1$ holds between $point2$ and $point1$ and relation $R_2$ holds between $point3$ and $point2$, then the entry of the Table 3 corresponding to line $R_1$ and column $R_2$ denotes the possible relation(s) holding between $point3$ and $point1$ (points represent the centroid of corresponding objects). For example, if $point2$ is $Front$ of $point1$ and $point3$ is $Left$ to $point2$ (after rotating object $o2$ to so as to point to the same direction as object $o1$) then $point3$ is $Front$ or $Left$ to $point1$. Entries in the above composition tables are determined using the following observation: composition of two relations corresponds to the addition of two vectors representing the relative placement of $point2$ to $point1$ and $point3$ to $point2$ (after performing the aforementioned rotation) forming angles $\theta_1$ and $\theta_2$ respectively with the reference axis. The resulting vector represents the relative placement of $point3$ to $point1$, i.e., the composition of two vectors, as illustrated in Figure 3(b). When adding the two vectors the resulting vector forms an angle $\theta$ with the reference axis such that $\theta_2 \leq \theta \leq \theta_1$. Angle $\theta$ defines the relation between $point1$ and $point3$. Using this observation it can be concluded for example that composing relations $Right$ and $Front$ yields the disjunction of these two relations as a result.

A series of compositions of relations may yield relations which are inconsistent with existing ones (e.g., the above example will yield a contradiction if *point3 back of of point1* has been also asserted into the ontology). Consistency checking is achieved by ensuring path consistency by applying formula:

$$\forall x, y, k\ R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_k(k, y))$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and $R_i$, $R_j$, $R_k$, $R_s$ denote directional relations). The formula is applied until a fixed point is reached (i.e., the application of the rules above does not yield new inferences) or until the empty set is reached, implying that the ontology is inconsistent. Implementing path consistency formula requires rules for both compositions and intersections of pairs of relations.

Compositions of relations $R_1$, $R_2$ yielding a unique relation $R_3$ as a result are expressed in SWRL using rules of the form:

$$R_1(x, y) \wedge R_2(y, z) \rightarrow R_3(x, z)$$

Note that for directional orientation relations of Figure 1(b), rules of the above form apply, but for positional orientation relations of Figure 2(a), rules are of the form:

$$R_1(x, y) \wedge R_{o_1}(x, y) \wedge R_2(y, z) \rightarrow R_3(x, z)$$

where $R_{o_1}$ is a relation of the set presented in Figure 1(b).

The following is an example of such a composition rule:

$$Front(x, y) \wedge Front(y, z) \rightarrow Front(x, z)$$

Rules yielding a set of possible relations cannot be represented directly in SWRL, since disjunctions of atomic formulas are not permitted as a rule head. Instead, disjunctions of relations are represented using new relations, whose compositions must also be defined and asserted into the knowledge base. For example, the composition of relations $Front$ and *Right* yields the disjunction of two possible relations (*Front* and *Right*) as a result:

$$Front(x, y) \wedge Right(y, z) \rightarrow (Front \vee Right)(x, z)$$

If the relation $Front\_Right$ represents the disjunction of relations *Front* and *Right*, then the composition of $Front$ and $Right$ can be represented using SWRL as follows:

$$Front(x, y) \wedge Right(y, z) \rightarrow Front\_Right(x, z)$$

A set of rules defining the result of intersecting relations holding between two points must also be defined in order to implement path consistency. These rules are of the form:

$$R_1(x, y) \wedge R_2(x, y) \rightarrow R_3(x, y)$$

where $R_3$ can be the empty relation. For example, the intersection of relations $Left$ and $Right$ yields the empty relation, and an inconsistency is detected:

$$Left(x, y) \wedge Right(x, y) \rightarrow \bot$$

Intersection of relations $Right$ and $Right\_Back$ (representing the disjunction of $Right$ and *Back* yields relation $Right$ as a result:

$$Right(x, y) \wedge Right\_Back(x, y) \rightarrow Right(x, y)$$

Thus, path consistency is implemented by defining compositions and intersections of relations using SWRL rules and OWL axioms for inverse relations as presented in Section 3. Another important issue for implementing path consistency is the identification of the additional relations, such as the above mentioned $Right\_Back$ relation, that represent disjunctions. Specifically *minimal* sets of relations required for defining compositions and intersections of all relations that can be yielded when applying path consistency on the basic relations of Figures 1(b) and 2(a) are identified. The identification of the additional relations is required for the construction of the corresponding SWRL rules.

In this work, the *closure method* [2] of Table 4 is applied for computing the minimal relation sets containing the set of basic relations: starting with a set of relations, intersections and compositions of relations are applied iteratively until no new relations are yielded forming a set closed under composition, intersection and inverse. Since compositions and intersections are constant-time operations (i.e., a bounded number of table lookup operations at the corresponding composition tables is required) the running time of closure method is linear to the total number of relations of the identified set. This method is applied over both sets of relations.

**Table 4.** Closure method

```
Input: Set S of tractable relations
Table C of compositions
WHILE S size changes
    BEGIN
        Compute C:Set of compositions of relations in S
        S=S ∪ C
        Compute I:set of intersections of relations in S
        S= S ∪ I
    END
RETURN S
```

A reduction to required relations and rules can be achieved by observing that the disjunction of all basic relations when composed with other relations yields the same relation, while intersections yield the other relation. Specifically, given that $All$ represents the disjunction of all basic relations and $R_x$ is a relation in the supported set, then the following holds for every $R_x$:

$$All(x,y) \wedge R_x(x,y) \rightarrow R_x(x,y)$$

$$All(x,y) \wedge R_x(y,z) \rightarrow All(x,z)$$

$$R_x(x,y) \wedge All(y,z) \rightarrow All(x,z)$$

Since relation $All$ always holds between two points, because it is the disjunction of all possible relations, all rules involving this relation, both compositions and intersections, do not add new relations into the ontology and they can be safely removed. Also, all rules yielding the relation $All$ as a result of the composition of two supported relations $R_{x1}, R_{x2}$:

$$R_{x1}(x,y) \wedge R_{x2}(y,z) \rightarrow All(x,z)$$

can be removed as well. Thus, since intersections yield existing relations and the fact that the disjunction over all basic relations must hold between two objects, all rules involving the disjunction of all basic relations, and consequently all rules yielding this relation, can be safely removed from the knowledge base. After applying the closure method and optimizations the required number of relations for representation and reasoning (basic and disjunctive) is 23 (14 directional and 9 positional).

## 5 Evaluation

In the following the proposed representation and reasoning mechanism is evaluated both theoretically and experimentally.

### 5.1 Theoretical Evaluation

The required expressiveness of the proposed representation is within the limits of OWL 2 expressiveness. Reasoning is achieved by employing DL-safe rules expressed in SWRL that apply on named individuals in the ontology ABox, thus retaining decidability.

Specifically, any object can be related with every other object with two basic directional relations (one of each set presented in Figures 1(b) and 2(a)). Since relations of each set are mutually exclusive, between $n$ objects, at most $2n(n-1)$ relations can be asserted. Furthermore, path consistency has $O(n^5)$ time worst case complexity (with $n$ being the number of points). In the most general case where disjunctive relations are supported, in addition to the basic ones, any object can be related with every other object by at most $k$ relations, where $k$ is the size of the set of supported relations. Therefore, for $n$ objects, using $O(k^2)$ rules, at most $O(kn^2)$ relations can be asserted into the knowledge base.

The $O(n^5)$ upper limit for path consistency running time referred to above is obtained as follows: At most $O(n^2)$ relations can be added in the knowledge base. At each such addition step, the reasoner selects 3 variables among $n$ objects which corresponds to $O(n^3)$ possible different choices. Clearly, this upper bound is pessimistic, since the overall number of steps may be lower than $O(n^2)$, because an inconsistency detection may terminate the reasoning process early, or the asserted relations may yield a small number of inferences. Also, forward chaining rule execution engines employ several optimizations, thus the selection of appropriate variables usually involves fewer than $O(n^3)$ trials. Nevertheless, since the end user may use any reasoner supporting SWRL, a worst case selection of variables can be assumed in order to obtain an upper bound for complexity. Also, retaining control over the order of variable selection and application of rules yields an $O(n^3)$ upper bound for path consistency [3].

## 5.2 Experimental Evaluation

Measuring the efficiency of the proposed representation requires the spatial ontology of Section 3, containing instances, thus a data-set of 10K to 100K objects generated randomly was used for the experimental evaluation. Reasoning response times of the spatial orientation reasoning rules are measured as the average over 10 runs. HermiT 1.3.8 running as a library of a Java application was the reasoner used in the experiments. All experiments where run on a PC, with Intel Core CPU at 2.4 GHz, 6 GB RAM, and Windows 7.

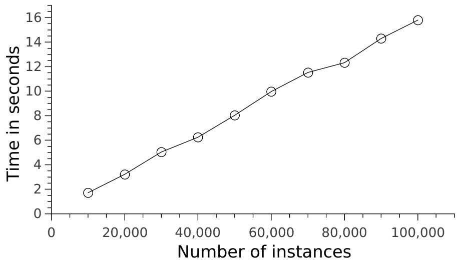


**Fig. 4.** Average reasoning time for orientation relations as a function of the number of objects

Measurements illustrate that the proposed approach can efficiently represent thousands of objects and reason over them in a few seconds, without using specialized software besides a standard OWL reasoner such as HermiT.

## 6 Conclusions and Future Work

In this work, a representation framework for handling orientation spatial information in ontologies is introduced. The proposed framework handles both, egocentric directional and positional information using an inference procedure based on path consistency.

The proposed representation is fully compliant with existing Semantic Web standards and specifications, which increases its applicability. Being compatible with W3C specifications, the proposed framework can be used in conjunction with existing editors, reasoners and querying tools such as Protégé and HermiT, without requiring any additional specialized software. Therefore, information can be easily distributed, shared and modified. Directions of future work include the development of applications based on the proposed mechanism. Such applications could combine temporal and topological spatial representations with the proposed orientation representation and reasoning mechanism.

## References

1. Batsakis, S., Petrakis, E.G.M.: SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 242–249. Springer, Heidelberg (2011)
2. Renz, J., Nebel, B.: Qualitative Spatial Reasoning using Constraint Calculi. In: Handbook of Spatial Logics, pp. 161–215. Springer, Netherlands (2007)
3. Stocker, M., Sirin, E.: PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine. In: OWLED 2009. CEUR Workshop Proceedings, vol. 529, pp. 2–31 (2009)
4. Christodoulou, G., Petrakis, E.G.M., Batsakis, S.: Qualitative Spatial Reasoning using Topological and Directional Information in OWL. In: Proc. of 24th International Conference on Tools with Artificial Intelligence (ICTAI 2012), November 7-9 (2012)
5. Renz, J., Mitra, D.: Qualitative Direction Calculi with Arbitrary Granularity. In: Zhang, C., Guesgen, H.W., Yeap, W.-K. (eds.) PRICAI 2004. LNCS (LNAI), vol. 3157, pp. 65–74. Springer, Heidelberg (2004)
6. Batsakis, S.: Reasoning over 2D and 3D directional relations in OWL: a rule-based approach. In: Morgenstern, L., Stefaneas, P., Lévy, F., Wyner, A., Paschke, A. (eds.) RuleML 2013. LNCS, vol. 8035, pp. 37–51. Springer, Heidelberg (2013)
7. Baruah, R., Hazarika, S.M.: Qualitative directions in egocentric spatial reference frame. In: 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA). IEEE (2012)
8. Mossakowski, T., Moratz, R.: Qualitative reasoning about relative direction of oriented points. Artificial Intelligence 180, 34–45 (2012)
9. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-Efficient OWL Reasoner. In: OWLED, vol. 432 (2008)

# An Efficient Approach for Detecting and Repairing Data Inconsistencies Resulting from Retroactive Updates in Multi-temporal and Multi-version XML Databases

Hind Hamrouni, Zouhaier Brahmia, and Rafik Bouaziz

University of Sfax, Tunisia
hindhamrouni@gmail.com,
{zouhaier.brahmia,raf.bouaziz}@fsegs.rnu.tn

**Abstract.** In multi-temporal XML databases supporting schema versioning, updating a past element with retroactive effect is not always a graceful task, since it could give rise to inconsistencies in the database. In fact, modifying a past element due to a detected error means that the database has included erroneous information during some period and, therefore, its consistency should be restored by correcting all errors and inconsistencies that have occurred in the past. In this paper, we propose an efficient approach which preserves data consistency in multi-temporal and multi-version XML databases. More precisely, after any retroactive update, the proposed approach allows (i) determining the period of database inconsistency, which results from that update, and (ii) repairing of all data inconsistencies and their consequent side effects.

**Keywords:** Temporal XML Databases, Schema Versioning, Retroactive Update, Data Inconsistency, Inconsistency Period, Repairing Inconsistencies.

## 1 Introduction

Nowadays, supporting the temporal aspect is a requirement for most computer applications, including processing of scientific and census data, banking and financial transactions, and record-keeping applications. In fact, these applications need to store and manipulate data while taking into account the time dimension. This has led to the appearance of temporal databases [1, 2] which retain data evolution over transaction-time dimension and/or valid-time dimension [3]: (i) the valid-time of a datum is the time when this datum is true in the real world; each datum is timestamped with a validity start time (VST) and a validity end time (VET); (ii) the transaction-time of a datum is the time when this datum is current in the database; each datum is timestamped with a transaction start time (TST) and a transaction end time (TET).

Thus, according to the temporal dimensions they support, temporal databases are classified into five categories [3]: transaction-time (including only transaction-time data), valid-time (containing only valid-time data), bitemporal (storing only bitemporal data), snapshot (containing only nontemporal/conventional data) or multi-temporal (supporting data of different temporal formats).

Besides, evolution of database schema (e.g., dropping or adding entities, dropping or adding attributes of entities) over time is unavoidable in the context of information systems and may occur due to several reasons: meeting new user requirements, taking into account new regulations, etc. In temporal databases, changing schema could lead to some problems like data loss when dropping some entities and/or attributes. Thus, although temporal databases allow keeping track of data history when the schema is static, they do not provide a complete data history when they do not keep track also of database schema evolution over time. Therefore, to avoid this drawback and to provide a complete data history which allows performing operations on data defined under any schema version, researchers in the database community have proposed to adopt the schema versioning technique in temporal databases [4, 5].

On the other hand, XML databases [6] are widely used, especially on the Web. The introduction of temporal and schema versioning aspects in such databases gave rise to multi-temporal and multi-version XML databases [7] in which any XML document can store elements of different temporal formats (snapshot, transaction-time, valid-time and bitemporal) and any XML schema evolves over time through multiple versions. Moreover, these databases are very useful for several domains (e.g., managing evolution of customer profiles and requirements in e-commerce systems, managing evolution of legal texts in e-government systems...).

In multi-temporal and multi-version XML databases, there are three types of updates concerned with the time when updates are made: retroactive, proactive [8], and real-time (or on-time) updates. Indeed, a retroactive update is done after the change occurred in reality (i.e., the TST of the datum is superior to its VST). A proactive update is done before the change occurs in reality (i.e., the TST of the datum is inferior to its VST). A real-time update is done when the change occurs in reality (i.e., the TST of the datum is equal to its VST).

Retroactive and proactive updates occur naturally in many applications. For example, a postdated check is a proactive update, and a salary increase may be retroactive to some past date. However, retroactive updates are not always performed safely since they could have a harmful effect on the consistency of the database. Let's take the sample of correcting a past XML element representing a past banking interest rate in a bank, which was applied during the period going from 2013-01-01 to 2013-12-31. All existing data (e.g., interest bank accounts, balances of bank accounts, and scheduled payment amounts) that have been obtained using the erroneous past banking interest rate are consequently erroneous and should be corrected. The database was inconsistent during the period where this interest rate was effective.

In this paper, we focus on the impacts of retroactive updates on the consistency of the database and we propose an efficient approach that preserves such a consistency: we show how to repair automatically and safely data inconsistencies which result from a transaction that includes some operations acting on past data.

The rest of this paper is organized as follows: the next section motivates the need for a new approach for preserving the consistency of a multi-temporal and multi-version XML database; Section 3 describes data inconsistencies resulting from retroactive updates; Section 4 presents our approach for an automatic and safe repairing of data inconsistencies that occur due to a retroactive update; Section 5 discusses related work; Section 6 concludes the paper.

## 2 Motivation

In this section, we first present an example that illustrates how maintaining consistency in temporal XML databases after a retroactive update is a complicated task that could not be achieved using supports provided by existing database management systems (DBMSs). Then we show the need for systems providing supports for preserving consistency of multi-temporal and multi-version XML databases.

### 2.1 Motivating Example

Suppose that on 2014-06-10, the personnel officer detects an error that has occurred on 2013-01-03: he/she saved an erroneous value for the salary of the employee Fateh: 1120 TND (the erroneous value) instead of 1210 TND (the correct value); thus, an amount of 90 TND has not been considered in the salary, and during a period of twenty-nine months. Obviously, this error was propagated to all results of operations that have been performed using this salary (i.e., 1120), especially those which calculated taxes and social security contributions, as well as to all successor salaries.

Currently, the semantics of the temporal data update operation [9], which should be used to correct the erroneous salary that was inserted on 2013-01-03, does not support the correction of all effects of this error (i.e., it does not correct all taxes and social security contributions, that were calculated after 2013-01-03 based on the erroneous salary, as well as all other successor salaries). Such an operation corrects only the value of the corresponding salary. The XML element that represents the erroneous salary is stored as a past erroneous element, and the correct salary is stored in a new XML element which represents a past correct element <salary/> (see Fig. 1); the modification is performed in a non-destructive manner, since we are in a temporal setting.

To repair all data inconsistencies, the database administrator should proceed in an ad hoc manner: first, he/she should determine the list of all operations that were done using the erroneous salary, in order to know all data that were calculated based on the erroneous salary or on other data obtained from the erroneous salary, going from 2013-01-03 to 2014-06-10. Then, he/she should correct all erroneous data by writing an appropriate XML update [10, 11]. Fig. 1 shows that all salaries introduced after the corrected salary have been also corrected by inserting new correct past <salary/> elements, except the last salary which is replaced by a new correct current <salary/> element (i.e., the <salary/> element with TET attribute equal to "UC"); notice that "UC" (Until Change) [3] means that the salary is current until new change. The figure 1 shows also that both the social security contributions and taxes corresponding to the corrected salaries have also been corrected accordingly (a social security contribution is equal to 10% of the salary paid during the same period, whereas a tax is equal to 5% of the corresponding salary).

```
<employees>
  <employee>
    <SSN>12345678</SSN>
    <name>Fateh</name>
    <salaries>
      <salary VST=″2013-01-01″ VET=″2013-12-31″ TST=″2013-01-03″ TET=″2014-01-03″>1120</salary>
      <salary VST=″2013-01-01″ VET=″2013-12-31″ TST=″2014-06-10″ TET=″2014-06-10″>1210</salary>
      <salary VST=″2014-01-01″ VET=″2014-12-31″ TST=″2014-01-03″ TET=″2014-06-10″>1250</salary>
      <salary VST=″2014-01-01″ VET=″2014-12-31″ TST=″2014-06-10″ TET=″UC″>1300</salary>
    </salaries>
    <socialSecurityContributions>
      <socialContribution VST=″2013-01-01″  VET=″2013-12-31″
                          TST=″2013-01-03″  TET=″2014-01-03″>112</socialContribution>
      <socialContribution VST=″2013-01-01″  VET=″2013-12-31″
                          TST=″2014-06-10″  TET=″2014-06-10″>121</socialContribution>
      <socialContribution VST=″2014-01-01″  VET=″2014-12-31″
                          TST=″2014-01-03″  TET=″2014-06-10″>125</socialContribution>
      <socialContribution VST=″2014-01-01″  VET=″2014-12-31″
                          TST=″2014-06-10″  TET=″UC″>130</socialContribution>
    </socialSecurityContributions>
    <taxes>
      <tax VST=″2013-01-01″ VET=″2013-12-31″ TST=″2013-01-03″ TET=″2014-01-03″>56</tax>
      <tax VST=″2013-01-01″ VET=″2013-12-31″ TST=″2014-06-10″ TET=″2014-06-10″>60.5</tax>
      <tax VST=″2014-01-01″ VET=″2014-12-31″ TST=″2014-01-03″ TET=″2014-06-10″>62.5</tax>
      <tax VST=″2014-01-01″ VET=″2014-12-31″ TST=″2014-06-10″ TET=″UC″>65</tax>
    </taxes>
  </employee>
  …
<employees>
```

**Fig. 1.** Salaries of the employee Fateh, and corresponding social security contributions and taxes, corrected after a retroactive update

## 2.2 Need for New DBMS Supports

The consistency of a multi-temporal and multi-version XML database could not be ensured easily, since (i) all temporal dimensions are supported (i.e., data can evolve over transaction time and/or valid time), (ii) there are several schema versions and consequently several database instances, and (iii) a data management operation that is originally devoted to insert, delete, or update an XML element could involve several other XML elements (e.g., when the temporal interval of a new element that modifies an existing one overlaps, completely or partially, temporal intervals of other existing XML elements or when the value of an element is deducted from the value of erroneous element), defined under the same XML schema version or under several XML schema versions. Thus, the challenges described above show that end users/applications, interacting with multi-temporal and multi-version XML databases, need DBMSs with built-in support for repairing inconsistencies that result from retroactive updates.

## 3 Data Inconsistencies Resulting from Retroactive Updates

In a multi-temporal and multi-version XML database, inserting, updating or deleting past data give rise to data inconsistencies during some periods. In [12], the authors defined an inconsistency period resulting from a retroactive update of data as the temporal interval which delimits the scope of side effects that are expected to be generated by this update. Such an inconsistency period could be of one of the following three types: "Wrong Absence of Data", "Wrong Presence of Data", or "Errors in Data".

- Wrong Absence of Data: the inconsistency is due to the absence of a datum that had to be present in the database during this period;
- Wrong Presence of Data: the inconsistency comes from the presence of a datum that had to be absent in the database during this period;
- Errors in Data: the inconsistency results from the existence of some data with erroneous values during this period.

An inconsistency period, resulting from a retroactive update can be divided into several sub-periods; each one of these sub-periods should be interpreted according to the nature (i.e., insertion, deletion, or correction) of the retroactive update. In the following, we study periods of inconsistency resulting from retroactive updates.

## 3.1 Data Inconsistencies Resulting from a Retroactive Insertion of Data

The insertion of an XML element with retroactive effect generates an inconsistency period of "Wrong Absence of Data" type: the inserted element, which was absent before its TST, should be normally present in the temporal database since its VST. Fig. 2 illustrates such a period of inconsistency. In the following, we use CT to denote the "current time".

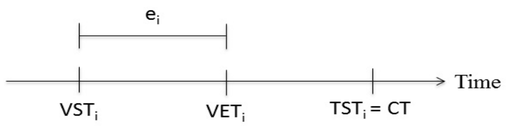


**Fig. 2.** The inconsistency period resulting from a retroactive insertion of data

As shown by Fig. 2, the period of inconsistency is delimited by the VST (period beginning) and the TST (period ending) of $e_i$; it can be divided into two sub-periods:

- $[VST_i – VET_i]$: the interval during which the consequent side effects concern all processings that had to use the element $e_i$ while it had to be a current element;
- $]VET_i – TST_i]$: the interval during which the generated side effects concern all processings that had to use the element $e_i$ while it had to be a past element.

## 3.2 Data Inconsistencies Resulting from a Retroactive Deletion of Data

The removal of an XML element with retroactive effect generates an inconsistency period of "Wrong Presence of Data" type: the deleted element, which was present before the instant of its deletion (i.e., before the TST of the deletion element [9, 11], that is the XML element which is used to delete the corresponding element), should not normally exist in the temporal database since its VST. Fig. 3 illustrates such a period.

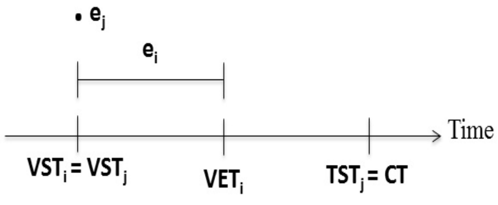


**Fig. 3.** The inconsistency period resulting from a retroactive deletion of data

As shown by Fig. 3, the period of inconsistency, which starts at the VST of $e_i$ and ends at the TST of $e_j$, can be divided into two sub-periods:

- [$VST_j$ - $VET_i$]: the interval during which the resulting side effects concern all processings that had used the element $e_i$ while it was a current element;
- ]$VET_i$ - $TST_j$]: the interval during which the consequent side effects concern all processings that had used the element $e_i$ while it was a past element.

## 3.3 Data Inconsistencies Resulting from a Retroactive Correction of Data

Retroactive correction operations could be done only on valid-time and bitemporal data. In this paper, we deal only with retroactive correction of bitemporal data, since we think that it is the most complicated case and, thus, it requires much attention.

The correction of a bitemporal element is performed by inserting a new element containing the correct values, called the element of correction [9, 11]. A correction operation can affect (i) the contents of the corrected element, (ii) values of non-temporal attributes (i.e., attributes different of VST, VET, TST, and TET attributes) of the corrected element, and/or (iii) the valid-time interval of the corrected element (i.e., values of VST and VET attributes); obviously, the transaction-time interval (i.e., values of TST and TET attributes) of any element cannot be modified owing to the definition of transaction time. In the first and/or the second case (i.e., points (i) and (ii)), the correction operation generates an inconsistency period of type "Errors in Data". However, in the third case (i.e., point (iii)), it generates an inconsistency period which can be divided into several sub-periods each one of them has a different type. A more detailed analysis of inconsistency periods can be found in [13].

In the following, we deal with inconsistencies resulting from a retroactive correction operation that updates the contents and/or the values of non-temporal attributes of a bitemporal element; it modifies neither the VST attribute, nor the VET attribute. This correction generates an inconsistency period of type "Errors in Data": it means that the corrected element had an erroneous value. Fig. 4 illustrates such a period of inconsistency.

As shown by Fig. 4, the period of inconsistency, which is delimited by the VST of $e_i$ (period beginning) and the TST of $e_j$ (period ending), can be divided into two sub-periods:

- [$VST_i$ - $VET_i$]: the interval during which the generated side effects concern all processings that had used the element $e_i$ while it was a current element;
- ]$VET_i$ - $TST_j$]: the interval during which the consequent side effects concern all processings that had used the element $e_i$ while it was a past element.

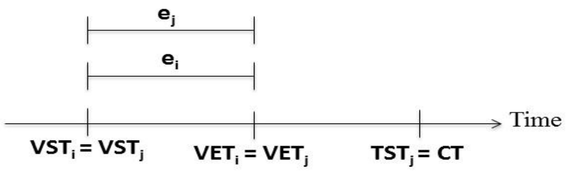


**Fig. 4.** The inconsistency period resulting from a retroactive correction operation which does not modify the valid-time interval of a bitemporal element

## 4 The Proposed Approach for Repairing Data Inconsistencies Resulting from Retroactive Updates of Temporal XML Data

In this section, we propose an approach that allows repairing automatically and safely data inconsistencies resulting from retroactive updates in multi-temporal XML databases supporting schema versioning. First, we describe the process of repairing data inconsistencies. Then, we present the architecture of a native temporal XML DBMS which supports repairing such data inconsistencies.

### 4.1 Process of Repairing Automatically and Safely Data Inconsistencies

When an end user or an application submits to the temporal XML DBMS a retroactive update of temporal XML data, it performs the following sequence of tasks:

**Task 1:** it updates the database as required by the end user or the application (obviously after checking the update syntactically).

**Task 2:** it determines the period of inconsistency resulting from the retroactive update of data, and its sub-periods.

**Task 3:** it determines the list of transactions that were executed during each sub-period of inconsistency and had used erroneous past data (in case that the corresponding sub-period of inconsistency is of type "Wrong Presence of Data" or "Errors in Data") or had to use new data (in case that the corresponding sub-period of inconsistency is of type "Wrong Presence of Data" or "Errors in Data"); for each concerned transaction, it should provide its commit time, all its elementary operations (for the sake of simplicity, we suppose that a transaction is composed of a single operation, i.e., a single insert, delete, or update operation), and all data that were written and read by this transaction.

**Task 4:** it re-executes in a provisory workspace the list of corresponding transactions during each sub-period of inconsistency either (a) without using the corresponding past data, if this sub-period is of type "Wrong Presence of Data", or (b) while using (b.1) the correct values of past data, if this sub-period is of type "Errors in Data", or (b.2) the specified past data, if this sub-period is of type "Wrong Absence of Data".

**Task 5:** it compares the old results of determined transactions (i.e., results already stored in the database, as written data by these transactions) with their new results (i.e., the new data that are written by these transactions in the provisory workspace).

**Task 6:** it replaces every old result with the corresponding new result when there is a difference between them.

In the following, we provide main requirements of some tasks presented above:

- Task 1 requires that the DBMS supports management of temporal XML data under schema versioning; we have studied this aspect in our previous works [9, 11].
- Task 3 requires beforehand keeping track of all transactions which are executed: for each transaction, the DBMS should store all operations which compose this transaction, all written and read data, and its commit time;
- Task 4 requires that the provisory workspace should be a copy of the database (schema and instances) during the inconsistency period and before the execution of the retroactive update operation.
- Task 6 requires that replacing old data with new data should be performed logically and not physically. Each existing erroneous data is logically corrected by a new correct data (i.e., in a non-destructive manner). After restoring the database consistency, only correct data must be used by the DBMS to answer user/application queries; erroneous data could be vacuumed later by the database administrator.

## 4.2 Architecture of a DBMS Supporting Repair of Data Inconsistencies

Owing to the architecture of a DBMS [14], the transaction manager is the component which is devoted to managing transactions resulting from user/applications queries and updates submitted to the database. Therefore, if we would like to have an automatic and graceful repairing of data inconsistencies which result from any retroactive update of temporal XML data in a multi-schema context, we think that (i) the transaction manager of a temporal XML DBMS should be extended by three new components: "Retroactive Update Checker", "Inconsistency Period Manager" and "Side Effect Recovery Manager", and (ii) the temporal XML DBMS itself should include a "Transaction Catalog Manager", a "Transaction Catalog", and a "Provisory Workspace". The new general architecture of such a DBMS is depicted in Fig. 5.

The "Retroactive Update Checker" checks that the Temporal XML Update submitted by the end user/application is a retroactive update operation.

The "Inconsistency Period Manager" determines the period of inconsistency which results from a retroactive update of data, and its sub-periods with their types. It is invoked by the "Retroactive Update Checker" in case this latter detects a retroactive update.

After determining all sub-periods of inconsistency, the "Inconsistency Period Manager" (i) invokes the "Transaction Catalog Manager" in order to retrieve from the

"Transaction Catalog" the list of transactions that were executed during each one of these sub-periods, and (ii) sends all retrieved transactions to the "Side Effect Recovery Manager".

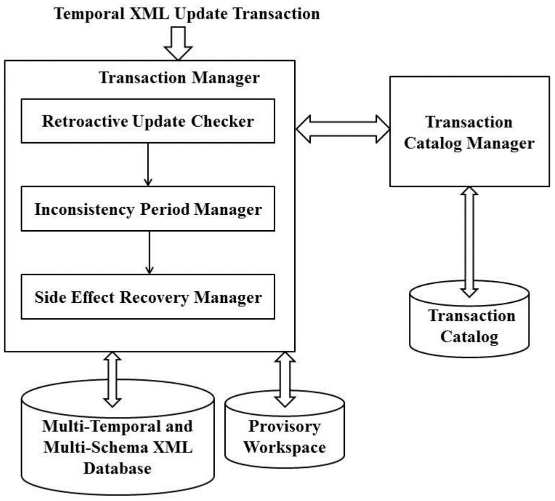

Temporal XML Update Transaction
Transaction Manager
Retroactive Update Checker
Inconsistency Period Manager
Side Effect Recovery Manager
Transaction Catalog Manager
Transaction Catalog
Multi-Temporal and Multi-Schema XML Database
Provisory Workspace

**Fig. 5.** General architecture of a temporal XML DBMS supporting automatic and graceful repairing of data inconsistencies resulting from retroactive updates

The "Side Effect Recovery Manager" controls the re-execution of transactions which have used erroneous data (i.e., transactions executed during a period of inconsistency of type "Wrong Presence of Data" or "Errors in Data") and transactions that had to use newly added data (i.e., transactions executed during a period of inconsistency of type "Wrong Absence of Data"). The concerned transactions are re-executed in a "Provisory Workspace" which is a copy of the database during the corresponding period of inconsistency. At the end of the re-execution of transactions, the "Side Effect Recovery Manager" (i) compares the results of determined transactions already stored (as written data) in the database with their results in the "Provisory Workspace", and (ii) replaces each old result with the corresponding new one when a difference between them is detected.

The "Transaction Catalog Manager" is added in order to have a history of transactions, which is complete (all details of transactions) and useful (i.e., easy-to-use by Side Effect Recovery Manager). For each transaction, it saves its commit time, the specified insert, delete, or update operation which composes it, data read from the database, data written to the database.

## 5 Related Work

Despite the importance of preserving consistency of databases after retroactive updates, this issue has been considered only to a limited extent in current literature.

However, it has not yet been studied in a multi-temporal and multi-version XML environment.

In [12], the authors proposed a solution for redressing side effects generated by a retroactive update, named "correction propagation". This solution is defined to repair only inconsistencies which affect cumulative attributes (i.e., attributes which can undergo only operations of additions or subtractions of values, like the balance of a bank account or the turnover of a company). So, this solution does not preserve the database consistency under retroactive updates of values of attributes which are not cumulative. Furthermore, this solution was defined for a temporal relational environment which does not support schema versioning.

In [15], the author proposed the use of temporal active rules and retroactive rules in order to redress side effects resulting from a retroactive update in a temporal relational active database. An active rule is said to be temporal if (i) the event is temporal, or (ii) the condition is temporal. A retroactive rule is a rule whose action includes a retroactive update.

On the other hand, retroactive and proactive updates were studied in temporal active databases [16] and in conventional (non-temporal) databases [8]. Furthermore, Brahmia et al. [9] and Hamrouni [11] have studied data updates in multi-temporal XML databases supporting schema versioning. However, none of these works has taken into account data inconsistencies resulting from retroactive updates.

Other works have dealt with the issue of preserving data consistency in temporal databases but with regard to (i) the concurrency control of transactions, by proposing new pessimistic [17] and optimistic algorithms [18], (ii) the forensic analysis of database tampering [19], or (iii) the respect of integrity constraints [20].

Pardede et al. [21] propose a generic methodology for the management of XML data update in XML-enabled databases. Such a methodology preserves the conceptual semantic constraints and avoids inconsistencies in XML data during update operations. But, the authors did not deal with retroactive updates, and define a data inconsistency as a data invalidity resulting from an XML data update.

In [22], the author proposed an approach for managing inconsistency in databases, within the framework of database repairs; a repair of an inconsistent database is a database over the same schema that satisfies the integrity constraints at hand and differs from the given inconsistent database in some minimal way.

Consistency of data, which takes into account the violation of semantic rules defined over a set of data items, has been also studied within the issue of data cleansing [23] and considered as a data quality dimension.

Zellag et al. [24] propose an approach for detecting consistency anomalies and automatically reducing their occurrence, in multi-tier architectures. Since the authors introduce a completely DBMS-independent approach, they propose that the system implementing their approach should be embedded into the middle-tier.

## 6 Conclusion

In this paper, we propose an efficient approach for an automatic and graceful repairing of data inconsistencies resulting from retroactive updates in multi-temporal databases supporting schema versioning. It allows the DBMS (i) to detect any database inconsistency that happens after a retroactive update operation, and (ii) to perform necessary processings, in a transparent way, in order to repair inconsistencies automatically.

We think that our approach (i) maintains effectively the consistency of the database, and (ii) provides a low-impact solution since it requires neither modifications of existing temporal database, nor extensions to existing temporal XML models (e.g., τXSchema [25]) and query languages (e.g., τXQuery [26]).

In order to show the feasibility of our approach, currently we are developing a prototype system as a temporal stratum on top of the existing XML DBMS xDB [27].

As a part of our future work, we envisage to extend our work by (i) dealing with retroactive updates which concern several temporal XML elements (in our present work, we have supposed that a retroactive update consider always one temporal XML element), and (ii) studying transactions that include several temporal XML updates with retroactive effect (in the current work, we supposed that a transaction include always a single temporal XML retroactive update operation). Furthermore, we also plan to study how to repair inconsistencies resulting from on-time and proactive updates of temporal XML databases, since the update of a current or a future temporal XML element could also give rise to some inconsistencies.

## References

[1] Etzion, O., Jajodia, S., Sripada, S. (eds.): Dagstuhl Seminar 1997. LNCS, vol. 1399. Springer, Heidelberg (1998)

[2] Grandi, F.: Temporal Databases. In: Koshrow-Pour, M. (ed.) Encyclopedia of Information Science and Technology, 3rd edn. IGI Global, Hershey (in press)

[3] Jensen, C.S., et al.: The Consensus Glossary of Temporal Database Concepts – February 1998 Version. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Dagstuhl Seminar 1997. LNCS, vol. 1399, pp. 367–405. Springer, Heidelberg (1998)

[4] De Castro, C., Grandi, F., Scalas, M.R.: Schema versioning for multitemporal relational databases. Information Systems 22(5), 249–290 (1997)

[5] Brahmia, Z., Mkaouar, M., Chakhar, S., Bouaziz, R.: Efficient management of schema versioning in multi-temporal databases. International Arab Journal of Information Technology 9(6), 544–552 (2012)

[6] Bourret, R.: XML and Databases, `http://www.rpbourret.com/xml/XMLAndDatabases.htm` (last updated in September 2005)

[7] Brahmia, Z., Grandi, F., Oliboni, B., Bouaziz, R.: Schema Change Operations for Full Support of Schema Versioning in the τXSchema Framework. International Journal of Information Technology and Web Engineering (in press)

[8] Etzion, O., Gal, A., Segev, A.: Retroactive and Proactive Database Processing. In: Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems (RIDE-ADS 1994), Houston, Texas, February 14-15, pp. 126–131 (1994)

[9] Brahmia, Z., Bouaziz, R.: Data Manipulation in Multi-Temporal XML Databases Supporting Schema Versioning. In: Proceedings of the 4th International EDBT Workshop on Database Technologies for Handling XML Information on the Web (DaTaX 2009), Saint-Petersburg, Russia (March 22, 2009)

[10] W3C, XQuery Update Facility 1.0, W3C Candidate Recommendation (March 17, 2011), http://www.w3.org/TR/2011/REC-xquery-update-10-20110317/

[11] Hamrouni, H.: Extending XQuery Update Facility to Temporal and Versioning Aspects, Master thesis, Faculty of Economics and Management of Sfax, Tunisia (December 2012)

[12] Bouaziz, R., Moalla, M.: Historisation of Data and Recovery of Side Effects. In: Proceedings of the 14th Journées de Bases de Données Avancées (BDA 1998), Hammamet, Tunisia, October 23-26, pp. 487–507 (1998) (in french)

[13] Hamrouni, H., Brahmia, Z., Bouaziz, R.: An Efficient Approach for Detecting and Repairing Data Inconsistencies Resulting from Retroactive Updates in Multi-Temporal and Multi-version XML Databases. TimeCenter Technical Report TR-97, 22 pages (June 17, 2014), http://timecenter.cs.aau.dk/TimeCenterPublications/TR-97.pdf

[14] Hellerstein, J.M., Stonebraker, M., Hamilton, J.: Architecture of a Database System. Foundations and Trends® in Databases 1(2), 141–259 (2007)

[15] Samet, A.: Automatic Recovery of Side Effects in a Multi-Version Environment, Master thesis, Faculty of Science of Tunis, Tunisia (March 1997)

[16] Deng, M., Sistla, A.P., Wolfson, O.: Temporal Conditions with Retroactive and Proactive Updates. In: Proceedings of the 1st International Workshop on Active and Real-Time Database Systems (ARTDB 1995), Skövde, Sweden, June 9-11, pp. 122–141 (1995)

[17] De Castro, C.: On Concurrency Management in Temporal Relational Databases. In: Proceedings of the 6th Italian Symposium on Advanced Database Systems (SEBD 1998), Ancona, Italy, pp. 189–202 (June 1998)

[18] Shirvani, M.H., Mohsenzadeh, M., Shirvani, S.M.H.: A New Concurrency Control Algorithm in Temporal Databases. Journal of Advances in Computer Research 4(2), 63–73 (2013)

[19] Pavlou, K.E., Snodgrass, R.T.: Generalizing database forensics. ACM Transactions on Database Systems 38(2) (2013), paper 12

[20] Švirec, M., Mlýnková, I.: Efficient Detection of XML Integrity Constraints Violation. In: Benlamri, R. (ed.) NDT 2012, Part I. CCIS, vol. 293, pp. 259–273. Springer, Heidelberg (2012)

[21] Pardede, E., Rahayu, J.W., Taniar, D.: XML data update management in XML-enabled database. Journal of Computer and System Sciences 74(2), 170–195 (2008)

[22] Afrati, F.N., Kolaitis, P.G.: Repair Checking in Inconsistent Databases: Algorithms and Complexity. In: Proceedings of the 12th International Conference on Database Theory (ICDT 2009), March 23-25, pp. 31–41. St. Petersburg, Russia (2009)

[23] Mezzanzanica, M., Boselli, R., Cesarini, M., Mercorio, F.: Automatic Synthesis of Data Cleansing Activities. In: Proceedings of the 2nd International Conference on Data Management Technologies and Applications (DATA 2013), Reykjavík, Iceland, July 29-31, pp. 138–149 (2013)

[24] Zellag, K., Kemme, B.: Consistency anomalies in multi-tier architectures: automatic detection and prevention. The VLDB Journal 23(1), 147–172 (2014)

[25] Snodgrass, R.T., Dyreson, C.E., Currim, F., Currim, S., Joshi, S.: Validating Quicksand: Schema Versioning in *τ*XSchema. Data Knowledge and Engineering 65(2), 223–242 (2008)

[26] Gao, D., Snodgrass, R.T.: Temporal slicing in the evaluation of XML documents. In: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003), Berlin, Germany, September 9-12, pp. 632–643 (2003)

[27] EMC, Documentum xDB (2014), http://www.emc.com/products/detail/software2/documentum-xdb.htm

# Integrated Representation of Temporal Intervals and Durations for the Semantic Web

Sotiris Batsakis, Grigoris Antoniou, and Ilias Tachmazidis

Department of Informatics
University of Huddersfield
Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK
{S.Batsakis,G.Antoniou,Ilias.Tachmazidis}@hud.ac.uk

**Abstract.** Representation of temporal information for the Semantic Web often involves qualitative defined information (i.e., information described using natural language terms such as "before" or "lasts longer than"), since precise dates, times and durations are not always available. A basic aspect of temporal information is duration of intervals, thus embedding duration relations into ontologies along with their semantics and reasoning rules is an important practical issue. This work proposes a new representation for intervals and their durations in ontologies by means of OWL properties and reasoning rules in SWRL embedded into the ontology. The proposed representation is based on the decomposition of Interval Duration calculus relations (INDU) offering a compact representation and a tractable reasoning mechanism. Furthermore, by embedding reasoning rules using SWRL into the ontology, reasoning semantics are an integrated part of the representation which can be easily shared and modified without requiring additional specialized reasoning software.

## 1 Introduction

Understanding the meaning of Web information requires formal definitions of concepts and their properties, using the Semantic Web Ontology definition language OWL. This language provides the means for defining concepts, their properties and their relations, allowing for reasoning over the definitions and the assertions of specific individuals using reasoners such as Pellet and HermiT. Furthermore, reasoning rules can be embedded into the ontology using the SWRL rule language.

Temporal information is an important aspect of represented objects in many application areas involving change. Temporal information in turn can be defined using quantitative (e.g., using dates) and qualitative terms (i.e., using natural language expressions such as "During"). Qualitative temporal terms have specific semantics which can be embedded into the ontology using reasoning rules. In previous work [1,2] such a representation is proposed for interval based temporal information in OWL, but duration information, which is necessary for representing assertions such as "process P2 starts after and lasts longer than process P1", was not represented.

Current work deals exactly with the case of combined interval and duration information which is more expressive than the representation proposed in [1]. Interval information is represented using the combined Interval Duration (INDU) relations introduced in [4]. The proposed representation, which is to the best of our knowledge the first such representation for the Semantic Web, is based on the decomposition of the INDU calculus relations. Specifically, between each pair of intervals two relations are asserted (e.g., “Before” and “Longer”). The first relation represents the relative placement of intervals along the axis of time (Allen relation [3]) and the second the relation of their durations (i.e., “Longer”, “Shorter” or “Equal”). Representation and reasoning for both sets of relations are presented in detail. Reasoning is applied on each set of relations separately, achieving a decomposition of INDU temporal relations. After decomposing the relations, the two reasoning mechanisms must be combined by taking into account the interactions of these two sets of relations. For example, when the duration of an interval *i1* is shorter than that of a second interval *i2*, then the fist interval cannot *contain* the second (i.e., Allen relation *contains* is incompatible with duration relation shorter). Implementation of reasoning is based on *4-consistency*, applied on a specific tractable set of INDU relations that is identified in this work. Furthermore, reasoning is implemented using SWRL rules that are embedded into OWL ontology containing the definitions of temporal properties. A total set of 180 Allen relations and 4 duration relations are part of the representation and reasoning mechanism.

Embedding reasoning rules into the ontology makes sharing of data easier since all SWRL compliant reasoners (such as Pellet and HermiT) can be used for temporal reasoning. To the best of our knowledge, this work is the first to represent INDU relations for the Semantic Web using a representation based on the decomposition of INDU relations, and also the first to deal with INDU reasoning based on a rule based approach embedded into OWL ontologies.

Current work is organized as follows: related work in the field of temporal knowledge representation is discussed in Section 2. The proposed representation is presented at Section 3 and the corresponding reasoning mechanism in Section 4. The extension to a combined interval-duration reasoning mechanism is presented in Section 5, followed by evaluation in Section 6 and conclusions and issues for future work in Section 7.

## 2 Background and Related Work

Definition of ontologies for the Semantic Web is achieved using the Web Ontology Language OWL. Specifically, the current W3C standard is the OWL 2[1] language, offering increased expressiveness while retaining decidability of basic reasoning tasks. Reasoning tasks are applied both on concept and property definitions into the ontology (TBox), and on assertions of individual objects and their relations (ABox). Reasoners include among others Pellet[2] and HermiT[3]. Reasoning rules can be embedded into the ontology using SWRL[4]. To guarantee decidability, the rules are restricted to *DL-safe rules* that apply only on named individuals in the ontology ABox. *Horn Clauses* (i.e., a disjunction of literals with at most one positive literal), can be expressed using SWRL,

[1] `http://www.w3.org/TR/owl2-overview/`
[2] `http://clarkparsia.com/pellet/`
[3] `http://hermit-reasoner.com/`
[4] `http://www.w3.org/Submission/SWRL/`

since Horn clauses can be written as implications (i.e., $\neg A \vee \neg B ... \vee C$ can be written as $A \wedge B \wedge ... \Rightarrow C$).

Qualitative temporal reasoning (i.e., inferring implied relations and detecting inconsistencies in a set of asserted relations) typically corresponds to Constraint Satisfaction problems which are $NP$, but tractable sets (i.e., solvable by polynomial algorithms) are known to exist [6]. Formal temporal representations have been studied extensively within the Semantic Web and Database communities (e.g., the temporal capabilities in SQL 2011 standard or TSQL2 support for qualitative Allen's intervals). For almost thirty years, researchers have been using temporal logic of Allen for representing point, interval and relative events in databases [12]. Relations between dynamic (i.e. evolving in time and having time dependent properties) entities in ontologies are typically represented using Allen temporal relations. In this work, duration information about intervals is combined with Allen relations. Furthermore, temporal relations are distinguished into qualitative (i.e., relations described using lexical terms such as “After”) and quantitative (i.e., relations described using specific dates for representing the starting and ending points of intervals).

Embedding spatial reasoning into the ontology, by means of SWRL rules applied on temporal intervals, forms the basis of the SOWL model proposed in [1] and the CHRONOS system [9]. CHRONOS and the underlying SOWL model were both not addressing the issue of combined interval duration relations. Such information is required for representing expressions such as “process P2 and P3 follow process P1, P2 lasts longer than P3, while process P4 starts directly after P2 finishes”, that commonly appear in scheduling problems. Current work addresses this issue and to the best of our knowledge is the first such representation for the semantic Web.

## 3 Temporal Representation

The proposed representation deals with duration relations between intervals in addition to their Allen relations. Qualitative duration relation of two intervals is represented using an object property specifying their relative duration. Specifically between two intervals three possible size relations can hold, these relations are “$<$”, “$>$”, “$=$” also referred to as *shorter, longer* and *equals* respectively.

An interval temporal relation can be one of the 13 pairwise disjoint Allen relations [3] of Figure 1. In cases where the exact durations of temporal intervals are unknown (i.e., their starting or ending points are not specified), their temporal relations to other intervals can still be asserted qualitatively by means of temporal relations (e.g., “event A happens before event B” even in cases where the exact durations of either A, B, or both are unknown).

Objects with dynamic properties (or fluent properties) can be represented into ontologies using the *N-ary relations* approach [11], which suggests representing an n-ary relation as two properties each related with a new object. Note that this approach requires one additional object for every temporal relation. An example of a representation of a relation holding during a specific temporal interval is illustrated in Figure 2(a). An alternative approach called the *4D-fluents* (*perdurantist*) approach [10] can be applied for representing temporal information and the evolution of temporal concepts in OWL.

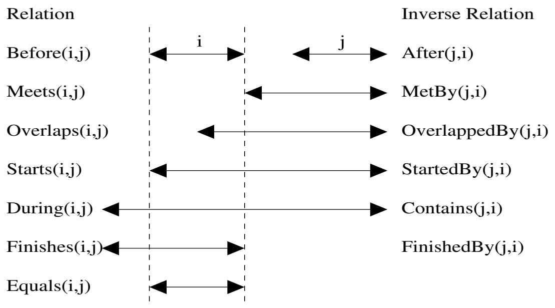
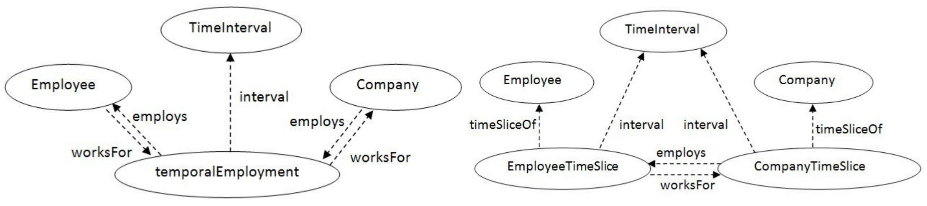


**Fig. 1.** Allen Temporal Relations



(a) N-ary Relations (b) 4D fluents

**Fig. 2.** Example of (a) N-ary Relations and (b) 4D-fluents

An example of a representation of a relation holding during a specific temporal interval using 4D-fluents is illustrated in Figure 2(b). Since the qualitative temporal relations that are used in this work (duration and Allen relations) hold between interval objects, the proposed representation can be combined with both N-ary relations and 4D-fluents by just adding the new relations and rules into the corresponding ontology without any additional modification.

Besides temporal property definitions, additional OWL axioms are required for the proposed representation; basic relations on each set are pairwise disjoint i.e., "$<$","$>$" and "$=$" size relations are pairwise disjoint and all Allen relations of Figure 1 are pairwise disjoint as well. In addition to that "$<$" is inverse of "$>$", while "$=$" is symmetric and transitive. Also $Before$ is $inverse$ of $After$, $Meets$ is $inverse$ of $MetBy$, $During$ is $inverse$ of $Contains$, $Finishes$ is $inverse$ of $FinishedBy$, $Starts$ is $inverse$ of $startedBy$ and $Overlaps$ is $inverse$ of $OverlappedBy$. $Equals$ is symmetric and transitive. Note that in this work the representation of Interval-Duration relations is achieved by decomposing the relations into two different sets instead of representing directly the 25 basic INDU relation as proposed in [4]. By applying this decomposition, the required number of relations is reduced to 13 basic Allen relations and 3 duration (size) relations, thus 16 basic relations in total. Between each pair of intervals, two basic relations (one from each set) can hold.

## 4 Temporal Reasoning

Inferring implied relations and detecting inconsistencies are handled by a reasoning mechanism. In the case of qualitative relations, assertions of relations holding between temporal entities (i.e., intervals) restrict the possible assertions holding between other temporal entities in the knowledge base. Then, reasoning on qualitative temporal relations can be transformed into a *constraint satisfaction problem*, which is known to be an *NP-hard* problem in the general case [6]. Inferring implied relations is achieved by specifying the result of $compositions$ of existing relations. Specifically, when a relation (or a set of possible relations) $R_1$ holds between intervals $i1$ and $i2$ and a relation (or a set of relations) $R_2$ holds between intervals $i2$ and $i3$ then, the $composition$ of relations $R_1, R_2$ (denoted as $R_1 \circ R_2$) is the set (which may contain only one relation) $R_3$ of relations holding between $i1$ and $i3$.

Qualitative relations under the intended semantics may not apply simultaneously between a pair of individuals. For example, given the time intervals $i_1$ and $i_2$, $i_1$ can not be simultaneously $before$ and $after$ $i_2$. Typically, in temporal representations (e.g., using Allen relations), all basic relations (i.e., simple relations and not disjunctions of relations) are pairwise disjoint. When disjunctions of basic relations hold true simultaneously, then their set intersection holds true as well. For example, if $i_1$ is *before or equals* $i_2$ and simultaneously $i_1$ is *after or equals* $i_2$ then $i_1$ *equals* $i_2$. In case the intersection of two relations is empty, these relations are disjoint. Checking for consistency means, whenever asserted and implied relations are disjoint, an inconsistency is detected.

### 4.1 Reasoning over Interval Allen Relations

Reasoning is realized by introducing a set of SWRL rules operating on temporal relations. Reasoners that support DL-safe rules such as HermiT can be used for inference and consistency checking over INDU (Allen-duration) relations. The temporal reasoning rules for Allen relations are based on the composition of pairs of the basic Allen relations of Figure 1 as defined in [3]. Specifically, if relation $R_1$ holds between intervals *i1* and *i2*, and relation $R_2$ holds between intervals *i2* and *i3*, then the composition table defined in [3] denotes the possible relation(s) holding between intervals *i1* and *i3*. Not all compositions yield a unique relation as a result. For example, the composition of relations $During$ and $Meets$ yields the relation $Before$ as a result, while the composition of relations $Overlaps$ and $During$ yields three possible relations namely *Starts, Overlaps and During*.

A series of compositions of relations may yield relations which are inconsistent with existing ones (e.g., if *i1 before i2* is inferred using compositions, a contradiction arises if *i1 after i2* has been also asserted into the ontology). Reasoning over temporal relations is known to be an NP-hard problem and identifying tractable cases of this problem has been in the center of many research efforts over the last few years [6]. The notion of *k-consistency* is very important in this research. Given a set of $n$ intervals with relations asserted between them imposing certain restrictions, *k-consistency* means that every subset of the $n$ intervals containing at most $k$ intervals does not contain an inconsistency.

Notice that, checking for all subsets of $n$ entities for consistency is exponential on the $n$.

There are cases where, although *k-consistency* does not imply *n-consistency* in general, there are specific sets of relations $R_t$ (which are subsets of the set of all possible disjunctions of basic relations $R$), with the following property: if asserted relations are restricted to this set, then *k-consistency* implies *n-consistency* and $R_t$ is a *tractable set* of relations or a *tractable subset of* $R$ [6]. Tractable sets of Allen interval algebra have been identified in [8], tractable subsets for Point Algebra which can represent duration relations have been identified in [7] and tractable subsets for INDU relations have been identified in [4]. Additional tractable sets for Allen relations have been identified in [1]. Consistency checking in [1] is achieved by ensuring path consistency by applying formula:

$$\forall x, y, k\ R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_k(k, y))$$

For the case of INDU relations, consistency over basic relations is not decided by path-consistency (or 3-consistency), but by 4-consistency, which corresponds to applying formula:

$$\forall x, y, k, l\ R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_k(k, z) \circ R_l(z, y))$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and $R_i$, $R_j$, $R_k$,$R_l$, $R_s$ denote temporal relations). The formula is applied until a fixed point is reached (i.e., the application of the rules above does not yield new inferences) or until the empty set is reached, implying that the ontology is inconsistent. Implementing 4-consistency formula requires rules for both compositions and intersections of pairs of relations. If a set of relations $R$ is closed under composition, then any arbitrary sequence of compositions can be implemented using compositions of pairs of relations and then composing the results.

The above holds for composing pairs of relations by definition. Also compositions of triples of relations can be computed from compositions of pairs of relations: $R_1 \circ R_2 \circ R_3 \equiv ((R_1 \circ R_2) \circ R_3)$. This can be extended using induction for any composition sequence for sets of relations closed under composition. This proves that 4-consistency can be enforced by composing and intersecting pairs of relations for all pairs of intervals. Compositions of relations $R_1$ *and* $R_2$ yielding a unique relation $R_3$ as a result are expressed in SWRL using rules of the form:

$$R_1(x, y) \wedge R_2(y, z) \rightarrow R_3(x, z)$$

The following is an example of such a composition rule:

$$Before(x, y) \wedge Contains(y, z) \rightarrow Before(x, z)$$

Rules yielding a set of possible relations cannot be represented directly in SWRL since, disjunctions of atomic formulas are not permitted as a rule head. Instead, disjunctions of relations are represented using new relations whose compositions must also be defined and asserted into the knowledge base. For example, the composition of relations

$Overlaps$ and $During$ yields the disjunction of three possible relations (*During, Overlaps* and *Starts*) as a result:

$$Overlaps(x, y) \wedge During(y, z) \rightarrow$$

$$During(x, z) \vee Starts(x, z) \vee Overlaps(x, z)$$

If the relation $DOS$ represents the disjunction of relations *During, Overlaps* and *Starts*, then the composition of $Overlaps$ and $During$ can be represented using SWRL as follows:

$$Overlaps(x, y) \wedge During(y, z) \rightarrow DOS(x, z)$$

The set of possible disjunctions over all basic Allen's relations contains $2^{13}$ relations, and complete reasoning over all temporal Allen relations has exponential time complexity. However, tractable subsets of this set that are closed under composition (i.e., compositions of relation pairs from this subset yield also a relation in this subset) are also known to exist [8,7]. In this work, we use the subset identified in Section 5, which is identified by taking into account interactions between interval and duration relations.

An additional set of rules defining the result of intersection of relations holding between two intervals is also required. These rules are of the form:

$$R_1(x, y) \wedge R_2(x, y) \rightarrow R_3(x, y),$$

where $R_3$ can be the empty relation. For example, the intersection of relation $DOS$ (represents the disjunction of $During$, $Overlaps$ and $Starts$) with relation $During$, yields relation $During$ as a result:

$$DOS(x, y) \wedge During(x, y) \rightarrow During(x, y).$$

The intersection of relations $During$ and $Starts$ yields the empty relation, and an inconsistency is detected:

$$Starts(x, y) \wedge During(x, y) \rightarrow \perp.$$

Thus, path consistency is implemented by defining compositions and intersections of relations using SWRL rules and OWL axioms for inverse relations as presented in Section 3.

## 4.2 Reasoning over Duration relations

Possible relations between interval durations are $shorter$, $longer$ and $equals$, denoted as "$<$","$>$","$=$" respectively. Table 1 illustrates the set of reasoning rules defined on the composition of existing relation pairs. The three temporal relations are declared as pairwise disjoint, since they cannot simultaneously hold between two intervals. Not all compositions yield a unique relation as a result. For example, the composition of relations $shorter$ and $longer$ yields all possible relations as a result. Because such compositions do not yield new information these rules are discarded. Rules corresponding to compositions of relations $R_1$ *and* $R_2$ yielding a unique relation $R_3$ as a result are

**Table 1.** Composition Table for point-based temporal relations

| Relations | $<$ | $=$ | $>$ |
|---|---|---|---|
| $<$ | $<$ | $<$ | $<,=,>$ |
| $=$ | $<$ | $=$ | $>$ |
| $>$ | $<,=,>$ | $>$ | $>$ |

retained (7 out of the 9 entries of Table 1 are retained), and are directly expressed in SWRL. The following is an example of such a temporal inference rule:

$$shorter(x, y) \wedge equals(y, z) \rightarrow shorter(x, z)$$

Therefore, 7 out of 9 entries in Table 1 can be expressed using SWRL rules, while the two remaining entries do not convey new information. A series of compositions of relations may imply relations which are inconsistent with existing ones. In addition to rules implementing compositions of temporal relations, a set of rules defining the result of intersecting relations holding between two instances must also be defined in order to check consistency. For example, the intersection of the relation representing the disjunction of *shorter, longer* and *equals* (abbreviated as $All$), and the relation $shorter$ yields the relation $shorter$ as result:

$$All(x, y) \wedge shorter(x, y) \rightarrow shorter(x, y)$$

The intersection of relations $shorter$ and $longer$ yields the empty relation, and an inconsistency is detected:

$$shorter(x, y) \wedge longer(x, y) \rightarrow \perp$$

As shown in Table 1, compositions of relations may yield one of the following four relations: *shorter, longer, equals* and the disjunction of these three relations. Intersecting the disjunction of all three relations with any of these leaves existing relations unchanged. Intersecting any one of the tree basic (non disjunctive) relations with itself also leaves existing relations unaffected. Only intersections of pairs of different basic relations affect the ontology by yielding the empty relation as a result, thus detecting an inconsistency. By declaring the three basic relations *shorter, longer, equals* as pairwise disjoint, all intersections that can affect the ontology are defined. Thus, checking consistency of duration relations is implemented by defining compositions of relations using SWRL rules and by declaring the three basic duration relations as disjoint.

## 5 Combining Interval and Duration Representation and Reasoning

The implementation of the reasoning mechanism over INDU relations requires the definition of all interactions between duration and Allen relations. Furthermore, since both sets of relations require the identification of a tractable set that is closed under compositions and intersections, interaction rules must be taken into account prior to tractable relations identification.

The first constraint is that when the duration relation between two intervals is the relation $shorter$ then the Allen relation can be one of *Before, After, Meets, MetBy, Overlaps, OverlappedBy, During, Starts* and *Finishes*:

$shorter(x,y) \rightarrow Before(x,y) \vee After(x,y) \vee Meets(x,y) \vee MetBy(x,y)$
$\vee Overlaps(x,y) \vee OverlappedBy(x,y) \vee During(x,y) \vee Starts(x,y) \vee Finishes(x,y)$

The second constraint is that when the duration relation between two intervals is the relation $longer$ then the Allen relation can be one of *Before, After, Meets, MetBy, Overlaps, OverlappedBy, Contains, StartedBy* and *FinishedBy*:

$longer(x,y) \rightarrow Before(x,y) \vee After(x,y) \vee Meets(x,y) \vee MetBy(x,y)$
$\vee Overlaps(x,y) \vee OverlappedBy(x,y) \vee Contains(x,y)$
$\vee StartedBy(x,y) \vee FinishedBy(x,y)$

The third constraint is that when the duration relation between two intervals is the relation $equals$ then the Allen relation can be one of *Before, After, Meets, MetBy, Overlaps, OverlappedBy*, and *Equals*:

$equals(x,y) \rightarrow Before(x,y) \vee After(x,y) \vee Meets(x,y) \vee MetBy(x,y)$
$\vee Overlaps(x,y) \vee OverlappedBy(x,y) \vee Equals(x,y)$
All the above relations that appear on the right hand side of a rule introduce disjunctions that must be supported by the reasoning mechanism. Similar restrictions on the duration relations are imposed by the Allen relation holding between two intervals. Specifically:

$$During(x,y) \rightarrow shorter(x,y)$$
$$Finishes(x,y) \rightarrow shorter(x,y)$$
$$Starts(x,y) \rightarrow shorter(x,y)$$
$$Contains(x,y) \rightarrow longer(x,y)$$
$$FinishedBy(x,y) \rightarrow longer(x,y)$$
$$StartedBy(x,y) \rightarrow longer(x,y)$$
$$Equals(x,y) \rightarrow equals(x,y)$$

Notice that Allen relations do not impose restrictions on duration relations requiring additional disjunctive relations that are not part of the representation mechanism of section 4.2. Thus, the representation and reasoning mechanism of section 4.2 combined with above rules is sufficient for representing desired relations. On the other hand, the set of relations used in the reasoning mechanism of section 4.1 must contain the imposed relations, thus propagating constraints from duration relations to Allen relations.

In order to implement the reasoning mechanism, while enforcing consistency over Allen relations, additional relations and rules must be minimized. Existing work (e.g., [8]) emphasizes on determining maximal tractable subsets of relations, while practical implementations calls for minimization of such relation sets (i.e., finding the minimal tractable set that contain the required relations).

**Table 2.** Closure method

```
Input:Set S of tractable relations
Table C of compositions
WHILE S size changes
    BEGIN
        Compute C:Set of compositions of relations in S
        S=S ∪ C
        Compute I:set of intersections of relations in S
        S= S ∪ I
    END
RETURN S
```

In this work, we apply the closure method of Table 2 for computing the minimal relation sets containing a tractable set of basic relations: starting with a set of relations, intersections and compositions of relations are applied iteratively until no new relations are produced. Since compositions and intersections are constant-time operations (i.e., a bounded number of table lookup operations at the corresponding composition table) the running time of closure method is linear to the total number of relations of the identified tractable set.

Applying the closure method over the set of basic Allen relations yields a tractable set containing 29 relations [1], but after adding the additional relations that result from propagating duration constraints, the minimal set of Allen relations containing the imposed restrictions consists of 180 relations. The new set is also tractable since the 13 basic relations and the additional three disjunctive Allen relations imposed by propagating constrains from duration relations belong to the maximal tractable set of Allen relation as identified in [8]. The full set will not be presented due to space limitations. Besides defining compositions and intersections between relations of the closed set, propagation of restrictions of the set to duration relations must be implemented as well. After checking all 180 relations, two additional restrictions must be imposed:

$$(Finishes(x, y) \lor FinishedBy(x, y)) \land equals(x, y) \rightarrow \perp$$

$$(Starts(x, y) \lor StartedBy(x, y)) \land equals(x, y) \rightarrow \perp$$

## 6 Evaluation

The required expressiveness of the proposed representation is within the limits of OWL 2 expressiveness. Furthermore, since the proposed representation is equivalent to the direct INDU representation, reasoning using the polynomial time 4-consistency algorithm is sound and complete, as in the case of non-decomposed INDU relations [4].

Specifically, any interval can be related with every other interval with two basic relations (one Allen and one duration relation) . Since relations of each set are mutually exclusive, between $n$ intervals, at most $2n(n-1)$ relations can be asserted. Furthermore, 4-consistency has $O(n^5)$ time worst case complexity (with $n$ being the number of intervals). In the most general case where disjunctive relations are supported in addition to the basic ones, any interval can be related with every other interval by at most $k$

relations, where $k$ is the size of the set of supported relations (containing 180 relations in case of Allen algebra subset identified in this work). Therefore, for $n$ intervals, using $O(k^2)$ rules, at most $O(kn^2)$ relations can be asserted into the knowledge base.

The $O(n^5)$ upper limit for consistency running time referred to above is obtained as follows: At most $O(n^2)$ relations can be added in the knowledge base. At each such addition step, the reasoner selects 3 variables among $n$ intervals (composition relations) which corresponds to $O(n^3)$ possible different choices. Clearly, this upper bound is pessimistic, since the overall number of steps may be lower than $O(n^2)$ because an inconsistency detection may terminate the reasoning process early, or the asserted relations may yield a small number of inferences. Also, forward chaining rule execution engines employ several optimizations (e.g., the Rete algorithm employed at the SWRL implementation of Pellet), thus the selection of appropriate variables usually involves fewer than $O(n^3)$ trials. Nevertheless, since the end user may use any reasoner supporting SWRL, a worst case selection of variables can be assumed in order to obtain an upper bound for complexity.

## 6.1 Experimental Evaluation

Measuring the efficiency of the proposed representation requires the temporal ontology of section 3 containing instances, thus a data-set of 100 to 1000 intervals generated randomly was used for the experimental evaluation. Reasoning response times of the temporal reasoning rules are measured as the average over 10 runs. HermiT 1.3.8 running as a library of a Java application was the reasoner used in the experiments. All experiments where run on a PC, with Intel Core CPU at 2.4 GHz, 6 GB RAM, and Windows 7.

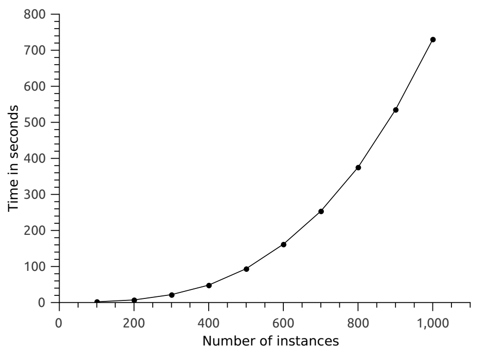


**Fig. 3.** Average reasoning time for INDU relations as a function of the number of intervals

Measurements illustrate that the proposed approach can efficiently represent hundreds of intervals and reason over them in a few minutes, or even seconds without using any specialized software besides a standard OWL reasoner such as HermiT.

## 7 Conclusions and Future Work

In this work, a representation framework for handling duration and interval relations in ontologies is introduced. The proposed framework handles both duration and Allen relations using an inference procedure based on 4-consistency, while the proposed representation is based on decomposition of INDU relations.

The proposed representation is fully compliant with existing Semantic Web standards and specifications, which increases its applicability. Being compatible with W3C specifications the proposed framework can be used in conjunction with existing editors, reasoners and querying tools such as Protégé and HermiT without requiring specialized software. Therefore, information can be easily distributed, shared and modified. Directions of future work include the development of real world applications based on the proposed mechanism. Such applications could combine duration and interval information in order to facilitate scheduling tasks. Furthermore, parallelizing our rule based reasoning mechanism is another promising direction of future research.

## References

1. Batsakis, S., Petrakis, E.G.M.: SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 242–249. Springer, Heidelberg (2011)
2. Batsakis, S., Stravoskoufos, K., Petrakis, E.G.M.: Temporal Reasoning for Supporting Temporal Queries in OWL 2.0. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011, Part I. LNCS, vol. 6881, pp. 558–567. Springer, Heidelberg (2011)
3. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26(11), 832–843 (1983)
4. Pujari, A., Kumari, V., Sattar, A.: INDU: An interval & duration network. In: Advanced Topics in Artificial Intelligence, pp. 291–303 (1999)
5. Welty, C., Fikes, R.: A Reusable Ontology for Fluents in OWL. Frontiers in Artificial Intelligence and Applications 150, 226–236 (2006)
6. Renz, J., Nebel, B.: Qualitative Spatial Reasoning using Constraint Calculi. In: Handbook of Spatial Logics, pp. 161–215. Springer, Netherlands (2007)
7. van Beek, P., Cohen, R.: Exact and approximate reasoning about temporal relations. Computational Intelligence 6(3), 132–147 (1990)
8. Nebel, B., Burckert, H.J.: Reasoning About Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. Journal of the ACM (JACM) 42(1), 43–66 (1995)
9. Preventis, A., Makri, X., Petrakis, E., Batsakis, S.: CHRONOS: A Tool for Handling Temporal Ontologies in Protege. In: Proceedings of 24th International Conference on Tools with Artificial Intelligence (ICTAI 2012), Athens, Greece, November 7-9 (2012)
10. Welty, C., Fikes, R.: A Reusable Ontology for Fluents in OWL. In: Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006), pp. 226–336 (2006)
11. Noy, N., Rector, A., Hayes, P., Welty, C.: Defining N-ary Relations on the Semantic Web. W3C Working Group Note 12 (2006)
12. Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M., Telos, M.: Representing knowledge about information systems. ACM Transactions on Information Systems (TOIS) 8(4), 325–362 (1990)

# Temporal State Management for Supporting the Real-Time Analysis of Clinical Data

Andreas Behrend[1], Philip Schmiegelt[2], Jingquan Xie[3], Ronny Fehling[4], Adel Ghoneimy[4], Zhen Hua Liu[4], Eric Chan[4], and Dieter Gawlick[4]

[1] University of Bonn, Germany
behrend@cs.uni-bonn.de
[2] Fraunhofer FKIE, Germany
{philip.schmiegelt}@fkie.fraunhofer.de
[3] Fraunhofer IAIS, Germany
jingquan.xie@iais.fraunhofer.de
[4] Oracle, USA
{first.last}@oracle.com

**Abstract.** Database systems are more and more employed to analyze an ever increasing amount of temporal data by applying a continuously evolving knowledge and are expected to do this in a timely fashion. Examples are financial services, computer systems monitoring, air traffic monitoring, and patient care. In each of these cases data are processed in order to understand current situations and to determine optimal responses. In this paper, we exemplarily investigate system requirements for a patient care scenario in which patient data are continuously collected and processed by a database system. We show that the concepts provided by today's systems are still not enough for supporting the complex reasoning process needed. In particular, we identify situation-based reasoning as a missing database component and propose a temporal state concept for leveraging simple event processing. States provide a high level (and qualitative) description of past and current situations defined over streams of medical data, complemented by projections into the future. Our proposed database extension allows for a compact and intuitive representation of medical data; much like physicians use abstraction from details and dramatically simplifies the analysis of medical data.

**Keywords:** Monitoring, Medical Databases, Data Streams, CEP, Provenance, Knowledge Management, Declarative Programming.

## 1 Introduction

Databases are able to store, manage, and retrieve large amounts and a broad variety of data. However, the task of understanding and reacting to the data is often left to tools or user applications outside the database. As a consequence, monitoring applications are often relying on problem-specific imperative code for data analysis, scattering the application logic. This leads to isolated applications which are hard to maintain, give raise to security and performance problems due to the separation of data storage and analysis. In particular, the process of problem solving, which requires understanding and tracking the current status and evolution of data, knowledge, and events, is still handled mostly by humans and not by the recording databases. The lack of a database support for the problem of process solving and data monitoring led to various approaches (e.g., CEP. MapReduce or DSMS) or specialized systems such as moving object databases. Especially CEP appeared to be well-suited for handling the dynamics of an application and we considered this paradigm for various projects (e.g., Air traffic surveillance [15] or patient care [4,5]) in which a database serves as a backend in order to deal with the strict auditing requirements. Pure event-based calculus turned out to be useful for general temporal reasoning over continuous changes but it is not well-suited for modelling the discrete changes between domain-specific situations. In our context, a situation is defined as a period of time during which a certain set of properties holds [10]. Observed events may change properties such that a situation is replaced by another (subsequent) one. This way, state transitions are performed which allow for various kinds of temporal analysis such as the detection of valid transition sequences or hypothetical reasoning over future state transitions.

Let us consider an air traffic scenario in which various events (e.g. changes of altitude, speed, and heading of an aircraft) are frequently processed for detecting anomalies (cf. Figure 1). As long as an aircraft is at its predefined course, however, all detected events (e.g. radar measurements) do not really have to be processed as they just confirm the situation "cruising". This changes as soon as the airplane enters the landing phase which is usually much more dangerous and should be monitored more closely. Such an additional abstraction would also allow for querying flight histories in a more natural way, e.g., if we want to count all flights with a starting and landing phase but no cruising phase in between. In this paper, we present an ongoing project from database industry and academia for adding a combination of event and situation calculus to the Oracle database. We use the notion *states* instead of ”situations“ or ”phases“ for describing a set of properties with a certain duration. We refine the state concept from [16] by means of the following contributions:

– Introducing abstraction levels for states.
– Determining the basic functionality needed for managing probabilistic states.
– Investigating the possibilities for hypothetical reasoning using predicted states.
– Identifying implementation issues and discussing a possible solution by integrating CEP into a DBMS.

The paper is organized as follows: After motivating the state concept more deeply in Section 2, we formally define the state concept and its properties. To this end, the state concept is embedded into a high-level data model called *KIDS* [8] for designing complex reactive systems in a systematic way. Afterwards, Section 3 shows how states could be used for a comprehensive in-database analytics including retrospective as well as predictive analysis of temporal data. Finally, first implementation issues are discussed in Section 5 and possible solutions are identified.

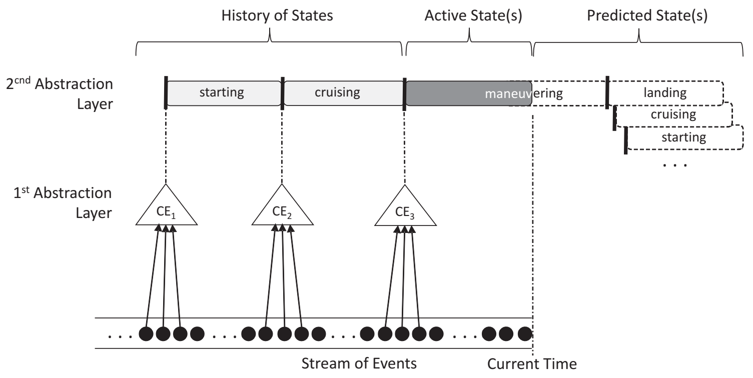


**Fig. 1.** States Abstracting From Data Changes or Events

## 2 Motivation

In order to support the declarative analysis of data in highly dynamic environments, CEP has been proven to be a well-suited paradigm for various applications. CEP is based on an event calculus - a well-established formalism for temporal reasoning with explicit references to object properties at certain timepoints. These explicit references to timepoints, however, may sometimes complicate the reasoning process over object properties which are not instantaneous but rather last over a certain period of time.

### 2.1 Simplifying Queries on Periods

In the following we show how difficult situation-based reasoning can become using a CEP approach. Let us assume we want to know the time interval during which a patient has had fever but not tachycardia (called 'FnT' for brevity). In case that only one tachycardia interval lies completely inside a fever period, two resulting FnT periods have to be determined by the system. In general, however, a fever period may coincide with several tachycardia periods leading to various sub-intervals of FnT periods to be computed.

In a CEP system, we may assume that the starting and ending times of fever as well as of tachycardia periods is provided by corresponding input streams. It is, however, hard to test whether a condition like "fever" is satisfied over a certain time interval by means of a CEP query. To this end, we would need to store the first and last fever respectively tachycardia event and query all possible sequences. Let us assume we store the respective event information in two tables called "feverStatus" and "tachycardiaStatus". For initializing the reasoning process, both tables are filled with a default event indicating the end of the respective diagnosis. To cover all possible sequences, four queries are necessary which determine the begin and end of an FnT state within the incoming stream. Using the Esper system[1], the query pattern looks like this:

[1] `http://esper.codehaus.org`

```
SELECT fever.timestamp AS FnT.start FROM fever
 WHERE fever.type=start AND
       last(tachycardiaStatus).type=end ...

SELECT tachycardia.timestamp AS FnT.end FROM tachycardia
 WHERE tachycardia.type=start AND
       last(feverStatus).type=start ....
```

Let us assume that the results of the above queries are materialized using the output stream *FnT_Stream*. For determining the duration of FnT periods, one more query would be necessary:

```
SELECT * FROM FnT_Stream.win:length(2) match_recognize
 measures B.timestamp-A.timestamp AS duration,
 pattern (A B) define A AS FnT.type=start,B AS FnT.type=end
```

The inherent complexity of the reasoning process is already visible and becomes even more apparent as soon as more than two conditions are compared this way. This is due to the fact that all possible sequences of time points in the underlying event calculus have to be considered. In contrast, a situation calculus could directly provide operations on time intervals which free the programmer from those explicit time point comparisons. In a database system supporting this kind of a (temporal) state concept, the following query could be used to detect FnT occurrences:

```
SELECT Name, Period.Start, Period.End FROM  SELECT ID FROM patient
WHERE P.Name="John",
      TABLE (TDIFF(vwFever(ID),vwTachycardia(ID)) Period
```

The complexity of the query is drastically reduced as all time point comparisons are hidden in the table function TDIFF (see Section 4.2) which is laterally joined with the ID generating subquery. The latter provides one or more patient IDs for persons named John. Apparently, the resulting expression is more easy to read and to maintain, making the query less error-prone.

## 2.2 Advantages of a State Concept

Another advantage of a state concept is the possibility to associate a state with appropriate directives (actions or treatments) which may be employed in order to initiate a desired state transition within a certain time interval. In addition, an abstraction hierarchy of states could be employed in order to prescind from certain minor important situations. For example, the fever period of a patient could be the result of a liver disease while the tachycardia phases may be related to a heart problem. A state hierarchy would allow physicians to get a brief overview of the general status of a patient before initiating a detailed investigation. A state concept could also be used to automatically detect the violation of transition constraints. States may be exclusive or non-exclusive. For example, a patient may have fever and tachycardia at the same time while fever and hypothermia cannot coincide. Quite similar, the transition of two consecutive states may violate domain specific condition constraints. For example, a certain therapy may not be replaced by another one as their combination may cause serious adverse effects. In order to automatically detect these constraint violations, a transition graph containing all valid state transitions [16] can be used. Corresponding continuous queries could be automatically derived from this graph for detecting and signaling invalid state sequences. Note that the resulting continuous queries cannot be regarded as classical integrity constraints because of the assumed stream context.

Another positive aspect of a state concept is the possibility to perform predictive reasoning complemented by a strict provenance support. The reasoning process can also be easily customized which allows for various state-based reasoning lines at the same time. In this way, the individual knowledge of more than one physician could be simultaneously applied for interpreting the underlying observations. Query optimization could be guided by currently active states because they provide valuable context information in which the processing of certain observed events may be avoided. In sum, a state concept would simplify the development of various monitoring applications in which the analysis of situations plays a prominent role. In particular, we believe that states would allow to drastically simplify the development, maintenance as well as debugging of database software for monitoring scenarios.

## 3 Use Case

We consider a health care scenario because it very naturally illustrates the requirements for designing a state concept. In health care states are already implicity used by many physicians in order to describe the temporal status of a patient. For example, [11] presents a Bayesian network which represents a probabilistic causal model for the diagnosis of liver disorders (LD). Every node is one symptom for LD which represents a currently active/inactive (e.g. 'fatigue') or historic temporal state (e.g. 'history of alcoholic abuse') of the respective patient. The given BN could be employed as decision support system but it also shows how physicians really abstract from symptoms provided in a stream of patient-related observations. This abstraction can be even increased by introducing an hierarchical version of the BN where groups of nodes are clustered in submodels.

In Figure 2 a subgraph of the respective state hierarchy is depicted which is subsequently employed for illustrating the reasoning process using a state concept. To this end, we have organized the states in form of a superstate and substate relationship. For example, the state LD is a superstate of the state 'Sweating'. In addition, we have introduced a new derived state 'Heart Problems' (HP) at the same abstraction level as LD and the new state 'Organ Problems' (OP) for abstracting from these two. Furthermore, we use the new states 'Measurements' (ME) and 'Visible Symptoms' (VS) for abstracting from respective LD or HP indicating symptoms by means of abnormal platelet counts (PC) or

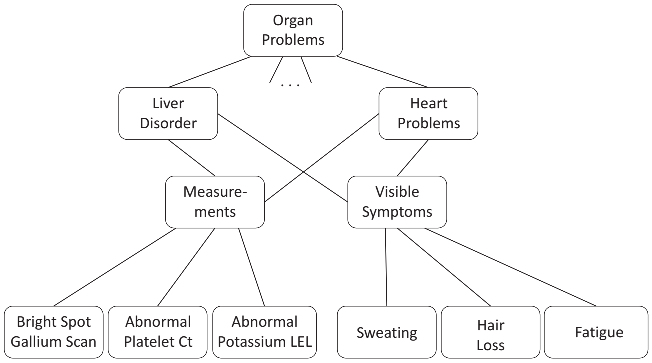
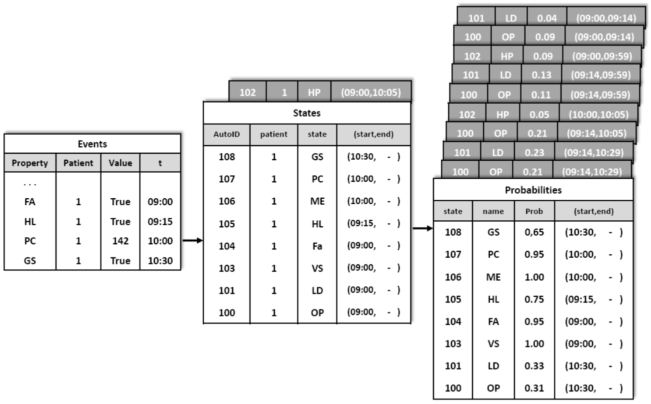


**Fig. 2.** Hierarchy of States for Detecting Liver Disorder

potassium levels (PL) respectively sweating or hair loss. We have chosen a representation in which not every substate is really contributing to the activation of every superstate in order to have a very concise state representation. For example, an abnormal PL may indicate heart problems but it is not an indicator for liver disorder. On the other hand, every connection line from a substate to a (direct) superstate really represents a measurable influence. These design decisions have to be guided by a corresponding developer tool which also provides the user with information about possible performance issues. In fact, the abstraction hierarchy is meant to provide a focus on the most important issues first but allows for drilling down to more specific states for explaining details later.

## 3.1 States for Detecting Liver Disorder

Let us consider a patient who suffers from fatigue syndrome (FA) indicating heart problems as the most likely cause. After a short time, hair loss (HL) is observed as additional symptom such that the diagnosis liver disorder becomes more likely. As a consequence, a platelet count is ordered. The resulting value increases the likelihood of a liver disorder situation. After a certain time, the probability of the still active diagnosis HP falls below a given threshold such that the respective state is reset to be inactive. Finally, a gallium scan is ordered leading to bright spots in the respective area which further increases the probability of LD. We assume that the corresponding symptoms are continuously recorded by a database system leading to the activation of various states. Note that the patient is in the respective state with a certain probability which could be determined, e.g., by using a Bayesian Network, decision trees or any other techniques for performing probabilistic reasoning. A state is considered active as soon as certain threshold for its probability has been reached. The state probability changes due to new observed symptoms. In case of HP, however, the probability value was decreased automatically as new confirming observations were not recorded by the system. This form of information decay has to be modelled individually for each state. A state may refer to a certain set of associated



Events

| Property | Patient | Value | t |
|---|---|---|---|
| . . . | | | |
| FA | 1 | True | 09:00 |
| HL | 1 | True | 09:15 |
| PC | 1 | 142 | 10:00 |
| GS | 1 | True | 10:30 |

| | | | |
|---|---|---|---|
| 102 | 1 | HP | (09:00,10:05) |

States

| AutoID | patient | state | (start,end) |
|---|---|---|---|
| 108 | 1 | GS | (10:30, - ) |
| 107 | 1 | PC | (10:00, - ) |
| 106 | 1 | ME | (10:00, - ) |
| 105 | 1 | HL | (09:15, - ) |
| 104 | 1 | Fa | (09:00, - ) |
| 103 | 1 | VS | (09:00, - ) |
| 101 | 1 | LD | (09:00, - ) |
| 100 | 1 | OP | (09:00, - ) |

| | | | |
|---|---|---|---|
| 101 | LD | 0.04 | (09:00,09:14) |
| 100 | OP | 0.09 | (09:00,09:14) |
| 102 | HP | 0.09 | (09:00,09:59) |
| 101 | LD | 0.13 | (09:14,09:59) |
| 100 | OP | 0.11 | (09:14,09:59) |
| 102 | HP | 0.05 | (10:00,10:05) |
| 100 | OP | 0.21 | (09:14,10:05) |
| 101 | LD | 0.23 | (09:14,10:29) |
| 100 | OP | 0.21 | (09:14,10:29) |

Probabilities

| state | name | Prob | (start,end) |
|---|---|---|---|
| 108 | GS | 0,65 | (10:30, - ) |
| 107 | PC | 0.95 | (10:00, - ) |
| 106 | ME | 1.00 | (10:00, - ) |
| 105 | HL | 0.75 | (09:15, - ) |
| 104 | FA | 0.95 | (09:00, - ) |
| 103 | VS | 1.00 | (09:00, - ) |
| 101 | LD | 0.33 | (10:30, - ) |
| 100 | OP | 0.31 | (10:30, - ) |

**Fig. 3.** Evolving State Hierarchy for Detecting Organ Problems

directives which allows for strengthening/weakening the certainty of being in the respective state or initiate a state transition. In the example, the directive "perform a PC" is used in order to increase the probability of the state LD. Once a directive is issued, the system could be programmed to expect the observation of corresponding results within a provided time interval. This way the reaction to the non-occurrence of events could be implemented in an elegant way.

## 3.2 Evolving States in the Use Case

In Fig. 3 the temporal development of the affected states are presented using temporal tables. The table `Events` stores the observed symptoms. Its first entry FA at 9:00 o'clock leads to the initialization of an FA state for patient 1 which is stored in the table `States` with a respective valid time interval. The probability attribute of the states are maintained using the temporal table `Probabilities` and a corresponding entry is made for FA setting the probability value to 95%. This is the default value for FA observations which can be overridden by another probability value directly provided by the observed event. The Fatigue state (FA) is a substate of Visible Symptoms (VS) which becomes activated with 100% at the same time indicating for patient 1 that visible disease symptoms have been observed. Furthermore, the superstates LD and HP are activated with probability 4% resp. 9%, leading to the activation of the state Organ Problems (OP). At 9:15, the additional symptom HL is observed leading to an increase of the probabilities for LD and OP (making LD the most likely cause for OP). For supporting full provenance, the old probability values of LD and OP are shifted into the history of the table `Probabilities` by closing the corresponding valid time interval at 9:14. Every new observation recorded in table `Events` leads to a recomputation of probability values. However, the state HP is deactivated at 10:05 as its probability value fell below 4%. This drop was initiated by an internal timer associated with the HP state which models the information decay over time if no state confirming events are detected.

## 4 A State Concept for Databases

A calculus for situations was first introduced by McCarthy in 1963, refined by Ray Reiter [12] in 1992. It represents a classical approach in AI and is basically the counterpart to an event calculus [6]. While situation calculus is as expressive as the event calculus, it has been shown that the combination of both could be beneficial [2]. In fact, a state concept would really complement a CEP system and simplify certain parts of the underlying reasoning process. Our initial attention to this concept was drawn during the development of the KIDS [8] model for realizing data as well as knowledge intensive applications.

### 4.1 The Abstract State Concept

In general, a named state is employed to describe the status of a tracked object with respect to certain properties (or features) managed by the database. We use an extensible user-defined datatype known form ORDBMS for defining abstract state types. The employed standard attributes (e.g., Start,End, object_ID, decay_fun, etc..) are depicted in Fig. 4. The start and end variables capture the time points during which the state was considered active. A state may also contain other attributes such as a probability value. All attributes are managed by the bi-temporal support of current database systems and have at least one associated valid time interval. Apart from state-related attributes, a state contains references to the super- and sub-states in the provided state hierarchy. In addition, a list of applicable directives as well as covered properties is maintained by a state. The latter provides all recorded properties (or features) in the given monitoring scenario. In the investigated clinical context the properties basically cover all data from electronic medical records. Finally, the function `decayfun` specifies the decay of information over time. More formal details on the definition of states are given in [14,16]. In Figure 4 a sample instance of the LD state for the patient with ID=1 is depicted. This state became active at 09:00 and its end is still undetermined. The list of applicable directives refers to an MRI scan, a biopsy and a gallium scan. The property list refers to LD symptoms like abnormal platelet counts (PC) or fatigue syndromes (FA).

### 4.2 State Related Operators

A complete list of the different temporal relationships between intervals (situations) is already given by means of the Allen's operators, e.g., equals/2, includes/2, overlaps/2, begins/2 or ends/2. They led to corresponding proposals for the development of TSQL [17] as well as system-specific solutions such as the valid time support of Oracle. If you want to query a history of situations, then it is usually necessary to compare sets of intervals and the above-mentioned operators are not suited anymore. As an example, consider a relation $\{$(FE,[10:00,11:00]),(TA,[10:10,10:20]),...$\}$ for storing fever (FE) and tachycardia states (TA) of a patient. For determining the FnT intervals, every fever state

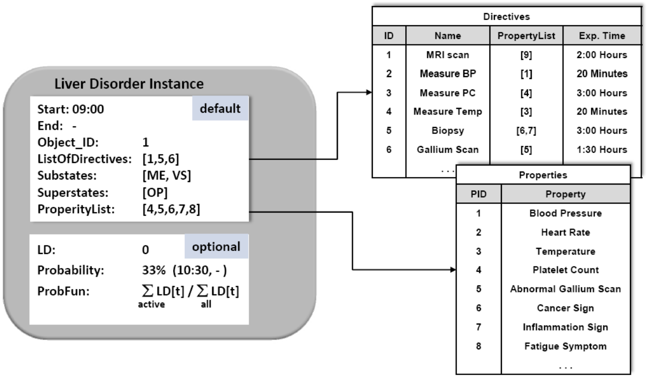

Liver Disorder Instance
Start: 09:00
default
End: -
Object_ID: 1
ListOfDirectives: [1,5,6]
Substates: [ME, VS]
Superstates: [OP]
ProperityList: [4,5,6,7,8]
LD: 0
optional
Probability: 33% (10:30, - )
ProbFun: ∑ LD[t] / ∑ LD[t] active all
Directives
ID Name PropertyList Exp. Time
1 MRI scan [9] 2:00 Hours
2 Measure BP [1] 20 Minutes
3 Measure PC [4] 3:00 Hours
4 Measure Temp [3] 20 Minutes
5 Biopsy [6,7] 3:00 Hours
6 Gallium Scan [5] 1:30 Hours
Properties
PID Property
1 Blood Pressure
2 Heart Rate
3 Temperature
4 Platelet Count
5 Abnormal Gallium Scan
6 Cancer Sign
7 Inflammation Sign
8 Fatigue Symptom

**Fig. 4.** Sample Instance of Liver Disorder State

has to be compared with every tachycardia state. In particular, several FnT intervals may result from these comparisons even if only one fever and one tachycardia period are considered as input: { (FnT,[10:00,10:09]), (FnT,[10:21,11:00]) }. Therefore, a table generating function is necessary in order to model the required one-to-many mapping. In Section 2 we have already employed the function `TDIFF` for determining the difference periods of two input sets of time intervals. In a similar way, various other interval operators from above have to be translated into table functions for simplifying the reasoning process over states including `TINTERSECTION` and `TUNION`. The handling with states is further homogenized by using access functions for their attributes such as getObject/1, getName/1, getBegin/1, getEnd/1, or getProbability/1. In addition, some auxiliary functions like sameObject/2 or duration/2 are used to facilitate the reasoning with states.

## 4.3 Reasoning with States

The knowledge for performing diagnostic as well as predictive reasoning over states can be represented in various ways. This includes functionalities provided by the database system (such as materialized views or CQN techniques) but also external rule representations such as neural or (dynamic) Bayesian networks. The latter are often used in the medical domain for handling the intrinsic uncertainty of medical data and diagnostic reasoning. Decision trees or logic programs would be another choice for performing a definite state derivation. No matter how we represent the knowledge and reasoning process, every time a new event has been observed by the system the corresponding reasoning process has to be triggered and the affected state related attribute values updated. The simplest way to access the states maintained this way is the usage of a snapshot query, e.g.:

```
SELECT getObject(S) AS Patient
FROM  tblStates AS OF PERIOD FOR valid_time TO_TIMESTAMP('04-Jun-13') AS S
WHERE getName(S)='HP'
```

This query based on Oracle's SQL would return the list of patients with heart problems on June 4th. We have already used functions for simplifying the needed state attribute access. The UDF `getName` is employed to determine the state name 'HP' while `getObject` provides the corresponding patient ID using the state attribute ID from tuple S. If we are interested in the states over a period of time, the above query could be modified by using version statement within the FROM part. For analyzing historical states, very often sequences of states have to be found. To this end, the two tables `tblSequence` and `tblStrictSequence` are automatically maintained by our system for a fast identification of (strict) consecutive states. This facilitates the determination of complex patterns of state occurrences, e.g.:

```
SELECT count(*) FROM sequence P1, strictSequence P2
WHERE sameObject(P1.S1,P1.S2,P2.S1,P2.S2) AND WM_LESSTHAN(P1.S2,P2.S1) AND
  getName(P1.S1)='AA' AND getName(P1.S2)= 'HL' AND getName(P2.S1)='FE' AND
  getName(P2.S2)='PC' AND getObject(P1.S1)='John' AND getMax(P2.S1)>41 AND
  duration(P1.S1)> INTERVAL '01' YEAR
```

This query counts the number of disease phases for patient John in which he first suffered from alcohol addiction ('AA') for more than one year and afterwards hair loss followed by a period of fever with over 41 degrees which was directly followed by an abnormal platelet count phase. Note that the sequence table just stores two states S1 and S2 with S1.END $<$ S2.END which means that the 'AA' and 'HL' periods in this query may even overlap. The complexity of the reasoning process becomes obvious and can even be increased by utilizing operators on time intervals as already indicated by the table function `TDIFF` from Section 2. This way, complex interval queries such as the determination of time periods over more than one day in which a patient suffered from fever and liver disorder but not hair loss can be expressed in a rather condensed form. Additionally, the state concept quite naturally allows for expressing the non-occurrence of events which is rather difficult for an event-based calculus.

States represent derived data which is generated by applying domain knowledge over stored observations and already determined states. Consequently, our query language should allow for deriving new states based on other ones. The following query serves as an example how a new state instance for organ problems 'OP' could be determined for patient John with probability 40% after observing the state sequence fever and hypothermia 'HT':

```
SELECT op(seq.nval,'OP',getObject(S1),NOW,null,0.4)
FROM strictSequence WHERE sameObject(S1,S2) AND getName(S1)='FE' AND
                          getName(S2)='HT' AND getObject(S1)='John'
```

The keyword op within the select part represents a constructor for the respective user-defined state type. The function `seq.nval` automatically generates a unique internal identifier for this new instance. The start attribute of this state is set to NOW while its end is still undefined. In order to really activate this state, the corresponding data must be materialized, e.g., by using an INSERT statement. This algebraic approach to generating and activating derived states is especially useful for performing predictive reasoning.

## 5 Implementation Issues

The presented sample queries already showed that most of the needed functionality can be realized by adding new user-defined data types and functions to a CEP system. For maintaining the entire workflow of a state-driven monitoring application, a new middleware component is necessary placed on top of the underlying and already extended CEP engine. The proposed query language for states is supposed to be plain SQL which is seamlessly extended by various new UDFs and data types. One of the key features needed for this extension, however, is that the underlying CEP or database system fully supports the management of bi-temporal data. The possibility to perform probabilistic reasoning represents another functionality necessary for implementing the proposed state concept. In our example we have used rules from a LD detecting BN for determining the continuously changing state probabilities. Every time a new observation is made, the BN rules are called and the state values updated accordingly. In order to incorporate information aging, we have extended the BN rules by a monotonically decreasing e-function. As an example, consider the rule for determining the probability of HL after the observation of an HL event E at E.time:

$$s.prob[t] := \begin{cases} 0.75e^{-0.1(t-E.t)}, & \exists \text{ last}(E) : E.o = s.o \wedge E.\text{t} \leq t \wedge E.n = \text{'HL'} \wedge \\ & E.v = \text{TRUE} \\ 0, & \text{else} \end{cases}$$

$E.o$ refers to the affected patient while *last* returns the newest event $E$ in the event history. The term $s.LD[t] := \frac{s.prob[t]}{7.5}$ is used in the HL state in order to describe the impact of the HL state's probability to the state LD. In a similar way, the potassium level (PL) is modelled using

$$s.prob[t] := \begin{cases} 0.95e^{-0.5(t-E.t)}, & \exists \text{ last}(E) : E.o = s.o \wedge E.\text{t} \leq t \wedge E.n = \text{'PL'} \wedge \\ & (E.v < 3.7 \vee E.v > 5.2) \\ 0, & \text{else} \end{cases}$$

while the LD impact value is set to zero $s.LD[t] := 0$ as a PL state indicates HP but not LD. Finally, liver disorder is derived using the accumulated probability values of its substates normalized by the sum of all possible contribution:

$$s.prob[t] := \sum_{l \in \text{actState}(s.o,t)} l.LD[t] \;\div \sum_{l \in \text{allStates}(s.o,t)} l.LD[t].$$

## 6 Discussion

We have presented a refined version of the state concept from [16] which combines the best of two worlds: The scalable real-time processing capabilities of an event calculus and the complex analysis and planing features of a situation calculus [2]. In particular one can think of event processing as the technology to identify changes and states as the abstraction that captures the current situation in a compact way and that also allows us to focus our attention on what is currently important and how facts have to interpreted in that context. Our work is related to various other research fields: Stream processing has first been mentioned in [9]. After that, numerous event processing engines and languages have been proposed most of them supporting discrete timestamps, only. There are only few proposals for events with a duration [1,3,13]. Some CEP engines (like [7]) allow at least for the specification of time intervals. Such interval specifications, however, differ from the state concept, as a state is allowed to only have a starting time with an unknown end. Probabilistic databases represents another active research field related to the proposed reasoning process in this paper.

## References

1. Adaikkalavan, R., Chakravarthy, S.: Formalization and Detection of Events using Interval-Based Semantics. In: COMAD, pp. 58–69 (2005)
2. Belleghem, K.V., Denecker, M., Schreye, D.D.: Combining Situation Calculus and Event Calculus. In: ICLP, pp. 83–97 (1995)
3. Galton, A., Augusto, J.C.: Two Approaches to Event Definition. In: Hameurlain, A., Cicchetti, R., Traunmüller, R. (eds.) DEXA 2002. LNCS, vol. 2453, pp. 547–556. Springer, Heidelberg (2002)
4. Gawlick, D., Ghoneimy, A., Liu, Z.H.: How to build a modern patient care application. In: HEALTHINF, pp. 427–432 (2011)
5. Guerra, D., Gawlick, U., Bizarro, P., Gawlick, D.: An Integrated Data Management Approach to Manage Health Care Data. In: BTW, pp. 596–605 (2011)
6. Kowalski, R.A., Sergot, M.J.: A Logic-Based Calculus of Events. New Generation Computing 4(1), 67–95 (1986)
7. Krämer, J., Seeger, B.: PIPES - A Public Infrastructure for Processing and Exploring Streams. In: SIGMOD, pp. 925–926 (2004)
8. Liu, Z.H., Behrend, A., Chan, E., Gawlick, D., Ghoneimy, A.: Kids - a model for developing evolutionary database applications. In: DATA, pp. 129–134 (2012)
9. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems, 1st edn. Addison-Wesley (2002)
10. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502 (1969)
11. Onisko, A., Druzdzel, M.J., Wasyluk, H.: A Bayesian Network Model for Diagnosis of Liver Disorders. Biocybernetics and Biomedical Engineering, 842–846 (1999)
12. Reiter, R.: Formalizing Database Evolution in the Situation Calculus. In: FGCS, pp. 600–609 (1992)
13. Roncancio, C.L.: Toward Duration-Based, Constrained and Dynamic Event Types. In: Andler, S.F., Hansson, J. (eds.) ARTDB 1997. LNCS, vol. 1553, pp. 176–193. Springer, Heidelberg (1999)
14. Schmiegelt, P., Xie, J., Schüller, G., Behrend, A.: Database functionalities for evolving monitoring applications. In: DATA (2013)
15. Schüller, G., Behrend, A., Manthey, R.: AIMS: An SQL-based System for Airspace Monitoring. In: IWGS, pp. 31–38 (2010)
16. Schüller, G., Schmiegelt, P., Behrend, A.: Supporting Phase Management in Stream Applications. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS 2012. LNCS, vol. 7503, pp. 332–345. Springer, Heidelberg (2012)
17. Snodgrass, R.T. (ed.): The TSQL2 Temporal Query Language. Kluwer (1995)

# Part VI
# Streams

# A Concept of Time Windows Length Selection in Stream Databases in the Context of Sensor Networks Monitoring

Monika Chuchro, Michał Lupa, Anna Pięta,
Adam Piórkowski, and Andrzej Leśniak

Department of Geoinformatics and Applied Computer Science
AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Cracow, Poland
{chuchro,apieta}@geol.agh.edu.pl,
{mlupa,pioro,lesniak}@agh.edu.pl
http://www.geoinf.agh.edu.pl

**Abstract.** Monitoring systems are a source of large amounts of data. These streams of data flow down as information which, in the case of sensor networks is often associated with the measurement of the selected physical signals. Processing of these data is a non-trivial issue, because accurate calculations often require dedicated solutions and large computing power.

In the case of flood embankment monitoring systems the essence of the calculation is the analysis of time series in terms of similarities to herald danger scenarios. This analysis also includes data series from neighboring sensors, which increases the difficulty of the calculation. This paper proposes the concept of a data analysis system, allowing for dynamic evaluation of the embankments.

**Keywords:** time series, time windows, stream database, embankments, flood.

## 1 Introduction

Integrated embankment monitoring systems have been created in a few countries, such as in the Netherlands [2,12]. In Poland, in cooperation with the AGH University of Science and Technology in Cracow and Cracow companies SWECO Hydroprojekt Ltd and Neosentio, project ISMOP (Computer Monitoring System for Flood Embankments) has been developed. This was founded under the NCBiR project (National Centre for Research and Development, Poland), which involves the creation of a monitoring system of static and dynamic behavior of the embankments that works in real time [17,19].

The basic problem of flood embankment is to control their condition, both during the flood season and during exposure to the interruption of the embankment. Visual observation of the shaft during the flood does not answer whether the section of the shaft has lost stability, and if so, how long it can effectively resist the flood. The aim of the research is to provide answers to the question of whether the measurement of technical parameters inside the shaft, such as temperature, pore pressure and humidity, can allow us to estimate the probability of damage to the embankment. An additional difficulty in giving a clear answer to this question is heterogeneous building embankments - the most common material that is near the bund.

## 1.1 Sensor Network for Embankments Monitoring System

This article concerns the difficult issue of assessment of flood embankment stability. This is a complex problem due to the amount of data processed from sensors located in the experimental embankment and the calculations required to assess flood embankment stability.

In order to facilitate the evaluation, an experimental embankment was divided into sections with a length of 1km. In each section of the embankment, there are 1,000 sensors arranged in 5 layers. A cross section of the experimental embankment is shown in Fig. 1. Each sensor measures parameters which can influence the condition of the embankment. These include: temperature, pressure, pore water, humidity, stress, strain and electrical conductivity. A sensor network is thus created [17,13]. Due to the advances in the electronic design, sensor network use IPv6 [3,18] for communication purposes. It enables possibility to expose its functionality as services [20] and make them coherent with the server side part of the system. Depending on the depth of the sensor in the embankment, the average measured values may vary slightly. These differences in measured values are caused by weather conditions and hydrogeological sensors located in the subsurface layers. All measurements are carried out with a time step of 15 minutes,

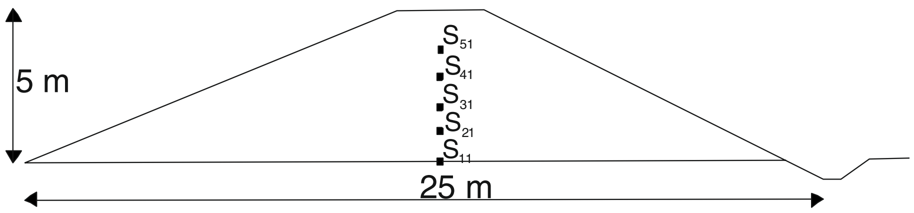


**Fig. 1.** The schema of experimental flood embankments with location of sensors ($S_{nm}$)

giving 540 000 observations every hour (15×1000×9×4), and almost 13 million per day on a 1km length of the experimental embankment. Taking into account the length of the flood embankment in the municipalities, the huge amount of incoming data is a big challenge. The first problem that must be solved is the collection and storage of data. The second problem is assessment of the state of the experimental embankment. This state assessment includes a comparison of data flowing from the sensors with the previously created numerical models.

Also the state assessment contains the visualization of models in a short period of time, shorter than getting the new batch of data from sensors.

Real data flowing from the sensors can be written as a multivariate time series of moments of time step of 15 min (1) [21,14].

$$\begin{gathered} Y(t) = \{y_a, y_b, ..., y_i\} \\ t = 1, 2, ..., \tau. \end{gathered} \tag{1}$$

where:

- $t$ - time step,
- $y_a$ - observation of one parameter on the time step $t$,
- $i$ - measured parameters $(a, ..., i)$.

The time series of a single sensor includes, in addition to the seven parameters, a timestamp and the coordinates of the point in space where the sensor is located. In addition to the data from the sensors, weather data from a weather station located in close proximity to the experimental flood embankment are processed and collected.

Assessment of the embankment stability is based on a comparison of data flowing from the sensor with dynamic numerical models of the embankment stability performed in the Flac [9]. Flac is a numerical modeling code for advanced geotechnical analysis, used in project for dynamical modeling. The concept used in Flac which is based on Lagrangian analysis is applied in many field (e.g. in obtaining a fluid-flow-based mechanical model for prediction of probability distribution [1]). Simulations in this environment are conducted on a numerical model of the embankment with the same parameters as the experimental embankment. In addition to the initial conditions, external conditions are used to simulate real phenomena occurring on and within the experimental embankment and in its proximity. Due to the possibly infinite number of opportunities to model the initial and external parameters, team modeling in Flac have to uniformly sampled space of possible initial and external parameters. The final result is a dynamic simulation of the embankment stability under the influence of different values of various external factors.

The simulated evaluation of the embankment stability should be consistent with the actual experimental embankment state. To facilitate searching the Flac models database, a treelike structure was created in which the first divisions concerned the initial conditions, and further divisions covered external conditions.

## 2 A Concept of Flood Embankment Condition Assessment

Due to the nature of the data and analysis requirements, no solution available is able to properly evaluate the condition of the experimental flood embankment. The number of observations in 1 km of the embankment, and the requirement that the assessment be realized in not more than 15 minutes, exclude a relational database. Daily, seasonal and annual cyclicality hinder the creation of generalized models, which is why correct space sampled of possible solutions will be a big challenge, especially for air temperature and humidity. On the other hand a strong dependence of autocorrelation means that the best solution seems to be to analyze the data into time windows using stream databases.

The general process of flood embankment condition assessment is shown in Fig. 2. Data flowing from the 1,000 sensors are evaluated for correctness. In the case of abnormalities or anomalies an error message is sent to the system administrator. An important step is the preprocessing of data, which includes examining the similarity of the values measured by the sensors in the moment $\tau$ and $\tau - 1$. If the module assessment of the shaft is not running, and the difference between the measured values of the parameters for the individual sensor does not exceed 5%, the system goes into idle mode until time $\tau + 1$. If the difference between the values of the parameters of the time $\tau$ and $\tau - 1$ (even of a single parameter or the sensor) is higher than 5%, that module of embankment stability assessment is enabled. Shutdown of assessment modules occur only when all of the scenarios created in Flac suggest "very good condition of experimental flood embankment" and the subsequent n measured values for all parameters did not show differences between them more than 5%. In the first step after starting the module assessment of the embankment stability, the last ten values for each sensor and each parameter are read from the database.

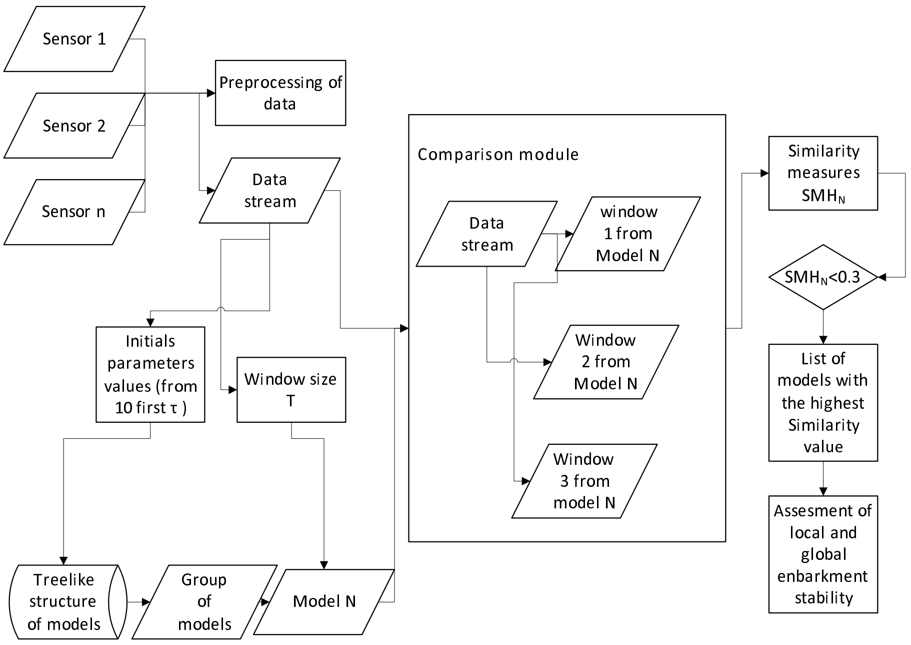

Sensor 1
Sensor 2
Sensor n
Preprocessing of data
Data stream
Initials parameters values (from 10 first τ )
Window size T
Treelike structure of models
Group of models
Model N
Comparison module
Data stream
window 1 from Model N
Window 2 from Model N
Window 3 from model N
Similarity measures SMH_N
SMH_N<0.3
List of models with the highest Similarity value
Assesment of local and global enbarkment stability

**Fig. 2.** The concept of flood embankment condition assessment

With these values, average values are calculated, which we consider as initial values. Calculated initial values are compared with the initial values for the Flac scenarios. The group of scenarios with the highest degree of similarity are selected for further evaluation of the embankment stability.

As a measure of similarity between two multidimensional windows ranks time series considered one of several measures of mathematical and statistical. Initially, a typical measure of similarity like as Pearson's correlation coefficient is excluded because of the lack of fit to the nature of multivariate and nonlinear time series. Eight similarity and dissimilarity measures has been selected from many existing measures to test. In many fields it has been shown that the different similarity measures can lead to substantial differences in final results and the similarity measure performance can be influenced by dataset characteristic [11]. For testing were chosen: Jacard coefficient, Sorensen coefficient, Czekanowski coefficient, determination coefficient, average measure of the error, mean absolute percentage error (MAPE), average absolute error known as $L_1$ Norm (2), root mean square error (square $L_2$ Norm) [4,8,5].

$$L_1 = \frac{1}{n} \sum_{i=1}^{n} |Y_\tau - Y_p| \tag{2}$$

The selection of the optimal similarity measure to use was made on the basis of experiment. 100 pairs of series of 1000 observations were generated in terms discraibed in [16], corresponding to series of a single sensor (real data) and to Flac models. The "real" time series were composed of a slowly varying sinusoidal function with strong irregular components, which were Gaussian noise. Flac models deviated slightly from the real data time series, but the Flac model time series do not have Gaussian noise. Due to the nature of the modeled data the similarity measure should:

- properly evaluate similarity ranks in the case of one of the time series having a stochastic component (noise),
- evaluate the similarity in the case of linear and nonlinear dependencies,
- lower the value of similarity in the models which do not adapt to the extreme values, which may indicate deterioration of embankment stability,
- be easy to interpret.

The measure of average absolute error (2) fulfills the abovementioned requirements, and this measure will be used in the project as a measure of similarity. Experimental embankment stability assessment runs iteratively. As the first group of embankments dynamic models was selected, this group, which was calculated for the highest similarity measure for the initial parameters values. Generally speaking, the same number of observations are selected for the first sensor and the first dynamic model, counting the real time series from a sensor 1. Then, for both multidimensional time series a measure of similarity is calculated. The calculated value of similarity and the number of initial observations of the time window are written to a temporary table. In the next iteration, the time window is shifted to the right by one observation in the dynamic model and then the measure of similarity is calculated again. If the new measured value is higher than the similarity stored in a temporary table, these values are overwritten. If the iterations come to the last observation in the first dynamic model, the highest value of the similarity measure, together with the number of initial observations of the time window, the number of dynamic model and the result of the script is saved to the table with the best match scenarios, provided that the measure of similarity does not exceed the limit value. The predetermined threshold based on the similarity measure experiments is 0.3. After calculating the measure of similarity for the first dynamic model and real times series from the first sensor, similarity measures are calculated for all dynamic model parameters preselected by initial values of parameters.

The final of the dynamic model is stored in binary form, in which 0 represents break or destruction of the flood embankment and 1 is maintained stability of the embankment. The local assessment of embankment stability, called "state", for a single sensor positioned in the experimental embankment is calculated based on Bayesian probability from all chosen by similarity measue models finals and is stated as $P(C_k)$, where $P$ is a Bayesian probablility and $C_k$ is a sensor number [15,24]. For a single sensor at time $t$ can take one of five states chosen arbitrary:

- Alarm status - threat of break or damage to the embankment. The system sends a message indicating the status of the emergency to the designated administrative units along with the location of the threat (3).

$$P(C_k) < 0.2 \tag{3}$$

- Warning status - threat of break or destruction of the experimental embankment, but also with the possible improvement of the embankment condition. The system sends a message indicating the status of the emergency to the designated administrative units along with the location of the threat (4).

$$0.2 \leq P(C_k) < 0.4 \tag{4}$$

- Neutral status - depending on external conditions the experimental embankment may stabilize, or a real threat of break or destruction of the embankment may occur (5).

$$0.4 \leq P(C_k) < 0.6 \tag{5}$$

- Good status - the likelihood of disturbances of the state of the experimental embankment is low, however, some dynamic models indicate the possibility of destruction of the embankment in the future (6).

$$0.6 \leq P(C_k) < 0.8 \tag{6}$$

- Very good status - the likelihood of disturbances to the state of the shaft is very low (7).

$$P(C_k) \geq 0.8 \tag{7}$$

The final threshold values of $P(C_k)$ will be modified during the experiments on real flood embankment.

Global assessment of the stability state of the entire experimental embankment is calculated as the average value of local states calculated in parallel to each other. A special case is the occurrence of a warning or alarm condition even for a single sensor (9 parameters). The status of the warning or alarm for a single sensor automatically grants the same status to the global assessment. In the case of an output state indicating possible damage or break of the experimental embankment, it is possible to run predictive modules allowing the assessment of the future state of the embankment on the basis of current data. For each evaluation time reports are generated with graphical presentation of the results, which are sent to the control and administration unit.

## 3 The Issue of Time Window Length Selection

Data in the stream database are processed and modeled in the form of streams of data. A single stream S can be called a multicollection of elements in the form $< s, \tau >$ in which $s$ is a tuple, and $\tau$ is the time of appearance of the item, or timestamp. A characteristic feature of streams is their possible unlimited length and repeatable values $\tau$. One of the most important features of the stream database is the time of arrival of the data to the database (timestamp). The older the data is, the less that data is associated with the current data streaming to the database. For this reason and because of the computational complexity, for processing streaming data generally only the last n observations are used with sliding time windows [6,7,22]. The time window converts the data stream into a table. In the stream each tuple has a timestamp $\tau$. The time window is a function that defines the life of the component in a table based on timestamp tuple. The arrival of a new tuple will update the current time of the operator in accordance with the time stamp of the tuple. Depending on the determining method, we distinguish agglomeration, sliding and sequencing time windows. The initial time window is in the agglomerative windows parameter, and the window size changes with each new input tuple. After updating, the current time window is increased by the unit of time. A sliding (sliding window) - parameter window is the size of $T$ in units of time. Each new input tuple is inserted into the table. The next step is to update operator current time. Then the table is cleared of tuples whose timestamp is beyond the scope of the window. The sliding window allows the grouping of tuples from within $T$ units of time. This can be expressed as (8):

$$R(\tau) = \{s|s, \tau' \in S \wedge (\tau' \leq \tau) \wedge (\tau' \geq max(\tau - T, 0)\}. \tag{8}$$

where:

- $S$ - data stream,
- $\tau$ - current time operator,
- $\tau'$ - the newest tuple,
- $T$ - window size.

Time-based sliding window contains all the elements at the time $\tau$ of the stream $S$, which appeared from time $\tau - T$ to time $\tau$. Another type of window is the tuple-base sliding window, which is associated with a fixed number of tuples, and contains the $N$ most recent tuples that have emerged to date $\tau$. The last type of sliding window is the partitioned sliding window. The stream $S$ is divided into subattributes with identical features. The next step is selection of $N$ most recent tuples to time $\tau$. Shifting / fixed window - window parameter is the size of $T$ in units of time. The interval is defined as:

– Beginning marker: $i * T$
– End marker: $(i + 1) * T$

## 3.1 A Concept of Experimental Flood Embankments Condition Assessment in Time Windows

Assessment of the experimental embankment is carried out iteratively, in the time windows. Already, on the basis of preliminary findings, two types of time windows were excluded (partitioned sliding and shifting windows) as not fulfilling the requirements of the project. Two types of windows were chosen as the most promising, fitted simultaneously to the data coming from the sensors and the planned evaluation of the stability of experimental embankment. The first type of time window selected is the sliding window with a fixed number of observations of 100 tuples from a single sensor, with a time-resolution of 15 minutes (Fig. 3). The advantage of such a solution, based on streaming databases, is that it does not require intermediary relational database calculations and so performance problems that could result from this kind of data storage are avoided. In addition, fixed window size reduces the assessment time, by reducing the number of function calls which count the length of the time series of measurements from the sensor, and function calls choosing an adequate number of observations for the dynamic models. The disadvantage of fixed size time windows is the possibility of loss of part of the dynamic models, due to the loss of initial readings from the sensors. To prevent the loss of significant parts of the models it would be necessary, for each time window $\tau - T$, to calculate the new initial value and to

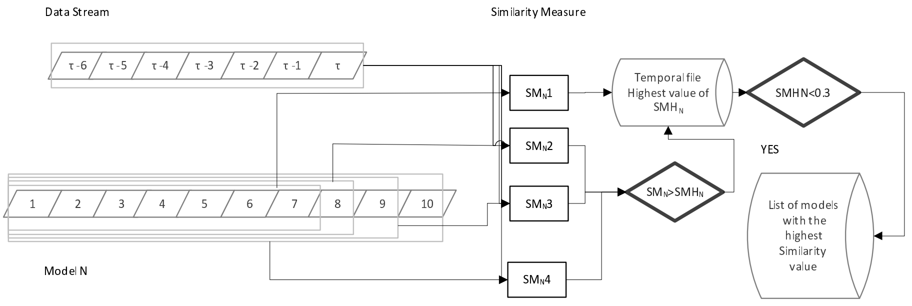


**Fig. 3.** Processing in agglomerative window

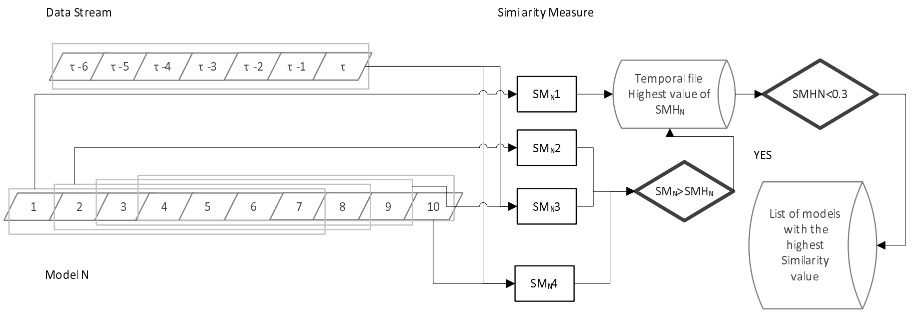


**Fig. 4.** Processing in sliding window

research the treelike structure of dynamic models. This research, unfortunately, will increase the number of time-consuming calculations and increase hardware resource requirements. The second solution is to use the agglomerative time windows (Fig. 4). The disadvantage of this solution is an increase in the number of calculation and in the time of calculation. With each new moment, a time window increases by 9 parameter values for a single sensor, which gives 9 000 new observations for a 1 km stretch of the experimental shaft for a moment, and during the day up to 864 000 additional observations. Additionally, the computing time also increases because of selecting and counting invoke functions. In the classical approach to stream databases, the computationally less expensive solution is to introduce time windows with a fixed number of observations, despite the possibility of losing a valuable part of dynamic models of the stability flood embankment development. However, this solution is not optimal in terms of obtaining full information about the status of the flood embankment. To be able to take advantage of the benefits of the agglomerative window, a way to shorten the time of analysis is needed, which will be iteratively longer and longer with each new $\tau$. Reducing the time may be accomplished by searching and reading tuples in the stream database, or reducing the number of parameters in the analysis. However the second solution reduces the possibility of a correct assessment of the embankment stability, the same as sliding windows. For this reason, we decided to use indexation of data dynamic model and in time series from sensors.

## 4 Conclusions and Further Work

This project on embankments monitoring is important due to the possibility of providing an early warning about embankment breach. This results in greater security in flood risk areas. It is necessary to establish and implement a dedicated system due to the lack of existing software that meets the requirements associated with the collection, analysis, and prediction of embankment condition. For this reason, the authors have proposed their own solutions, based on stream databases, where the information is evaluated by time series analysis tools.

The essence of time series analysis of data from the sensor network is the use of time windows, and the selection of the type and length of the windows, which will be adequate to the cyclicality periodic data (e.g. daily, monthly, yearly). This work also considers the selection of optimal time windows in the context of determining the similarity measure of time series and model data. The advantages and disadvantages of the proposed solution in relation to computational complexity were also presented. In future, the management of stream databases for discussed flood embankment will be considered [23,10].

**Acknowledgments.** This work is financed by the National Centre for Research and Development (NCBiR), Poland, project PBS1/B9/18/2013 - (no 180535).

This work was partly support by the AGH - University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection, as a part of statutory project number No.11.11.140.032.

## References

1. Augustyn, D.R.: Using the model of continuous dynamical system with viscous resistance forces for improving distribution prediction based on evolution of quantiles. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B. z. (eds.) BDAS 2014. CCIS, vol. 424, pp. 1–9. Springer, Heidelberg (2014)
2. Balis, B., Kasztelnik, M., Bubak, M., Bartynski, T., Gubała, T., Nowakowski, P., Broekhuijsen, J.: The urbanflood common information space for early warning systems. Procedia Computer Science 4, 96–105 (2011)
3. Brzoza-Woch, R., et al.: Implementation, Deployment and Governance of SOA Adaptive Systems. In: Ambroszkiewicz, S., Brzezinski, J., Cellary, W., Grzech, A., Zielinski, K. (eds.) Advanced SOA Tools and Applications. SCI, vol. 499, pp. 261–323. Springer, Heidelberg (2014)
4. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. International Journal of Mathematical Models and Methods in Applied Sciences 1(4), 300–307 (2007)
5. Clements, M., Hendry, D.: Forecasting economic time series. Cambridge University Press (1998)
6. Golab, L., Özsu, M.T.: Issues in data stream management. ACM Sigmod Record 32(2), 5–14 (2003)
7. Golab, L., Özsu, M.T.: Processing sliding window multi-joins in continuous queries over data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 500–511. VLDB Endowment (2003)
8. Hamilton, J.D.: Time series analysis, vol. 2. Princeton University Press, Princeton (1994)
9. Itasca Consulting Group, Inc.: FLAC Fast Lagrangian Analysis of Continua and FLAC/Slope – User's Manual (2008)
10. Koudas, N., Ooi, B.C., Tan, K.L., Zhang, R.: Approximate nn queries on streams with guaranteed error/performance bounds. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 804–815. VLDB Endowment (2004)

11. Gruca, A., Kozielski, M.: Correlation of genes similarity measures based on GO terms similarity and gene expression values. In: Czachórski, T., Kozielski, S., Stańczyk, U. (eds.) Man-Machine Interactions 2. AISC, vol. 103, pp. 137–144. Springer, Heidelberg (2011)
12. Krzhizhanovskaya, V.V., Shirshov, G., Melnikova, N., Belleman, R.G., Rusadi, F., Broekhuijsen, B., Gouldby, B., Lhomme, J., Balis, B., Bubak, M., et al.: Flood early warning system: design, implementation and computational modules. Procedia Computer Science 4, 106–115 (2011)
13. Li, J., Cai, Z., Li, J.: Data management in sensor networks. In: Wireless Sensor Networks and Applications, pp. 287–330. Springer (2008)
14. McGovern, A., Rosendahl, D.H., Brown, R.A., Droegemeier, K.K.: Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. Data Mining and Knowledge Discovery 22(1-2), 232–258 (2011)
15. McKenzie, C.R.: The accuracy of intuitive judgment strategies: Covariation assessment and bayesian inference. Cognitive Psychology 26(3), 209–239 (1994)
16. Pieta, A., Bala, J., Dwornik, M., Krawiec, K.: Stability of the levees in case of high level of the water. In: 14th SGEM Geoconference On Informatics, Geoinformatics And Remote Sensing – Conference Proceedings, vol. 1, pp. 809–815 (2014)
17. Piórkowski, A., Leśniak, A.: Using data stream management systems in the design of monitoring system for flood embankments. Studia Informatica 35(2), 297–310 (2014)
18. Szydlo, T., Gut, S., Puto, B.: Smart applications: Discovering and interacting with constrained resources ipv6 enabled devices. Przeglad Elektrotechniczny, 221–226 (June 2013)
19. Szydlo, T., Nawrocki, P., Brzoza-Woch, R., Zielinski, K.: Power aware mom for telemetry-oriented applications using gprs-enabled embedded devices – levee monitoring use case. In: Federated Conference on Computer Science and Information Systems (FedCSIS), September 7-10 (in print, 2014)
20. Szydło, T., Suder, P., Bibro, J.: Message oriented communication for ipv6 enabled pervasive devices. Computer Science 14(4) (2013)
21. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.: Indexing multidimensional time-series. The VLDB Journal—The International Journal on Very Large Data Bases 15(1), 1–20 (2006)
22. Wang, W., Li, J., Zhang, D., Guo, L.: Processing sliding window join aggregate in continuous queries over data streams. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 348–363. Springer, Heidelberg (2004)
23. Zhang, R., Koudas, N., Ooi, B.C., Srivastava, D.: Multiple aggregations over data streams. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 299–310. ACM (2005)
24. Zyphur, M.J.: Bayesian probability and statistics in management research: A new horizon. Journal of Management 39, 5–13 (2013)

# Partitioning for Scalable Complex Event Processing on Data Streams

Omran Saleh, Heiko Betz, and Kai-Uwe Sattler

Department of Computer Science and Automation
Technische Universität Ilmenau, Germany
{first.last}@tu-ilmenau.de

**Abstract.** Many applications processing dynamic data require to filter, aggregate, join as well as to recognize event patterns in streams of data in an online fashion. However, data analysis and complex event processing (CEP) on high volume and/or high rate streams are challenging tasks. Typically, partitioning techniques are leveraged for achieving low latency and scalable processing. Unfortunately, sequence-based operations such as CEP operations as well as long-running continuous queries make partitioning much more difficult than for batch-oriented approaches.

In this paper, we address this challenge by presenting partitioning strategies for CEP queries. We discuss two strategies for stream and pattern partitioning and we present a cost-based optimization approach for determining the number of partitions as well as the split points in the queries to achieve better load balancing and avoid congestions of processing nodes in a cluster environment.

## 1 Introduction

Over the last few years, several approaches [3,2,6,5] have been developed to address the big data challenge also for dynamic data. This is motivated by applications on environmental and systems monitoring, stock trading, realtime business analytics, or social media analysis. One of the basic strategy for scalable processing both for batch processing and online processing is to exploit data and/or task parallelism. For complex event processing on data streams, this means either *(1)* to partition the input data stream and send the sub-streams to different cluster nodes processing the same sub-graph of the dataflow (*stream partitioning*) or *(2)* to split complex operators such as CEP pattern matchers and assign them to different nodes (*pattern partitioning*). However, such partitioning strategies are not as easily applicable like in batch processing for the following reasons. First, whereas stateless operators do not cause problems, partitioning for stateful operators is more challenging. This is particularly the case for window- and sequence-based operators which includes CEP. Here, stream partitioning is applicable only if sub-streams can be processed independently, e.g., if partition criteria are induced by pattern specification or if punctuations allow to separate sub-streams. Furthermore, partitioning across multiple compute nodes may increase latencies which have to be considered, too. Second, queries on data streams are typically long-running queries prohibiting to simply restart and reallocate resources if data loss is not acceptable. Thus, planning and deploying such queries (which includes to find an optimal partitioning) is a critical issue.

In this paper, we address these challenges by presenting and evaluating techniques for stream and pattern partitioning in CEP queries. We discuss strategies which work both locally for multicore processing as well as for distributed cluster environments. In order to make the partitioning transparently to the user and to support a resource-aware query deployment, we present a cost-based approach for partitioning dataflow graphs that forms the rewriting step of query planning. Our cost model is based on existing rate-based models but takes also resource requirements into account.

## 2 The PipeFlow Language and Engine

The techniques proposed in the following are developed in the context of the PipeFlow system – a scalable distributed stream processing engine (SPE) with complex event processing features. In our system, we assume an event stream model (or tuples model) based on existing works on semantics of data streams [4] and CEP operators [8]. It is based on a library of C++ modules implementing various data stream operators as well as auxillary functionalities (e.g., a ZooKeeper interface). The frontend to our system is formed by the PipeFlow compiler which compiles declarative dataflow specifications into executable PipeFlow programs. PipeFlow is inspired by dataflow languages like Pig or Jaql: a script describes a directed acyclic graph of dataflow operators which are connected by named pipes. A single statement is given by:

```
$out := op($in1, $in2, ...) using (parameters);
```

where `$out` denotes a pipe variable referring to the typed output stream of operator `op` and `$in`*i* refers to input streams. By using the output pipe of one operator as input pipe of another operator, a dataflow graph is formed. PipeFlow provides a large set of predefined operators such as sources (file and database readers, network socket readers, ...), filters and projections, streaming and relation joins, grouping and aggregations as well as sliding and tumbling windows. Furthermore, it supports (externally implemented) user-defined operators as well as composite operators. The different operators are parametrized by a `using` clause and can be augmented with the schema (e.g., for source operators), and additional operator-specific clauses. For network communication, we use the 0MQ middleware[1] by providing dedicated operators such as `zmq_publisher` and `zmq_subscriber`. The following PipeFlow script shows an example for receiving data via a 0MQ socket, filtering tuples, and calculating the median over a sliding window of 5 minutes. The result is then published via a 0MQ sink:

```
$in := zmq_subscriber() with (x int, y int) using (endpoint = "...");
$f := filter($in) by x >100;
$w := window($f) using (slide_len = 300);
$a := aggregate($w) produce median(y) using (slide_len = 0);
$res := zmq_publisher($a) using (endpoint = "tcp://*:1235");
```

[1] http://zeromq.org

Matching of complex event patterns is supported by a dedicated `matcher` operator where the pattern is specified by the following clauses:

- `event` allows to specify the pattern over primitive events like `SEQ`, `AND`, `OR` etc. [8], e.g., `SEQ(A, B)` as the sequence of events `A` followed by `B`.
- `where` defines the primitive events and predicates, i.e., each primitive event A, B, ... is defined in terms of a predicate on a tuple. Predicates can be intra-event predicates (on a single tuple) like `A = x < 10`, or intra-event predicates, e.g., `B = (id == A.id && x >= 10)`.
- `using` is used for additional parameters such as time window of validity (e.g., "`within 60 seconds`") and the matching strategy (`first`, `every`, etc.).

The following example shows the basic usage of the `matcher` operator for detecting a complex event where the value of `x` of the same item (`id`) is increased within 10 seconds:

```
$res := matcher($in) event SEQ(A, B)
     where B = (id == A.id && x >=A.x * 1.1) using (within = 10);
```

The matcher operator is implemented by Non-deterministic Finite Automata (NFA) to evaluate event patterns over tuple streams in a way similar to regular expressions. A NFA represents the event pattern as a collection of connected states that must be traversed. The structure of a NFA is created by the PipeFlow compiler by mapping consecutive event types to NFA states.

A PipeFlow script is compiled into a physical execution plan represented as C++ code. Particularly, this involves graph rewriting steps (e.g., for partitioning as described in Sect. 3) as well as generating native code for implementing predicates, automata etc. Note that a PipeFlow script can be compiled to multiple executables which can be deployed to different machines for distributed query processing. This is supported by additional higher-level operators which we discuss in the following section.

## 3 Partitioning Techniques

The main motivation for leveraging partition techniques in data stream and complex event processing is to increase performance/throughput and/or scalability by load distribution. For CEP on data streams, two basic strategies exist: stream partitioning and pattern partitioning.

**Stream Partitioning.** The stream partitioning pattern exploits data parallelism by partitioning the input stream and sending the tuples of each partition (sub-stream) to a replicated subgraph of the dataflow program. This approach requires to split the stream in some way (e.g., round-robin or value-based such as range- or hash-partitioned), process the sub-streams independently, and finally merge the results into a single stream. Depending on the processing tasks applied to the sub-streams and the split strategy, the merge step could be either *(1)* a simple union of the sub-streams or *(2)* could require post-processing (e.g., post-aggregation or even a reordering of the tuples). In PipeFlow, we support this by introducing a `map` operator as a simplified version of the corresponding MapReduce operator. Our `map` version takes a dataflow graph as parameter, partitions the input stream according to a given key, and applies the parameter graph to each partition. There is no explicit reduce step because the resulting sub-streams produced by the mappers are simply merged into a single stream allowing to apply global post-processing (including aggregation or reordering).

The following example illustrates the use of the `map` operator together with the `matcher` introduced above. A user-defined composite operator is introduced which is then used as mapper task. The stream is partitioned on the field `C` because there exist no dependencies between the individual mappers.

```
define find_raise($in) returns $res {
   $res := matcher($in) event SEQ(A, B) where ...
};
$out := map($stream) on C do find_raise;
```

The map operator supports both a local and a distributed mode: with the local mode, the mappers run in separate threads on the same machine, whereas in the distributed mode, the mappers run on remote machines by inserting 0MQ send/receive primitives into the dataflow graph. Furthermore, the number of initial partitions/mappers can be specified explicitly by the parameter `num_partitions`, additional partitions can be added on-the-fly to a running query (but not yet automatically).

However, the application of stream partitioning is restricted to cases where the mappers can process their sub-streams independently. For the CEP operator this can be checked as follows. Given a pattern specification like

```
event SEQ(A,B) where A = x > 100, B = (x < 100 && id == A.id)
```

It is obvious that matching tuples always share the same value for the field `id`. Thus, tuples with different `id` values will never match and can be independently processed. In summary, this means a pattern matcher is *stream partitionable* on an attribute $x$ iff all pairs of primitive events $A, B$ of the pattern specification are pairwise connected by an equality predicate on $x$, i.e., `B = (x == A.x)`.

**CEP Pattern Partitioning.** If a pattern matcher is not stream partitionable but the operator is still too expensive for a single compute node, the CEP pattern itself could be split up and distributed over different nodes to meet the throughput requirements. There has been a few recent works focusing on CEP pattern partitioning, e.g., [1]. The main idea behind these approaches is to segment each NFA state to be run on a separate node in parallel as a *pipeline*. To each state in the NFA, a predicate is assigned which is evaluated for the whole incoming event stream. After matching an event by the corresponding state, a match from this state has to be appended as a partial result and forwarded to the next state in the chain. This continues until reaching the final state.

Obviously, this approach has some potential issues: *(i)* it produces a communication overhead between states running on separate machines. For example, if the pattern consists of a sequence of length 9, the system has to forward the partial matches among the consecutive nodes 9 times *(ii)* the input stream must be replicated and forwarded to every node representing a state in the NFA which leads to add up more overhead on the system *(iii)* different nodes in the system receive and process events at different rates depending on the match result from their predecessor nodes. This occurs as a result of different processing times for evaluating the predicates associated with the different states. To overcome these issues, CEP patterns should be decomposed into a more efficient way by segmenting the NFA into automata for sub-patterns where multiple states are combined in a single sub-pattern. In other words, one matcher operator can be rewritten into multiple independent matcher operators.

As an example, we consider the following pattern specification which contains a hypothetical predicate that is computational expensive:

```
$res := matcher($in) event SEQ(A, B, C) where A = expensive_pred(x),
    B = (x < 100 && id == A.id), C = (x < 200 && id == B.id)
    using (within = 60);
```

This pattern can be rewritten into the following query that separates the processing of the `A` event from the pattern on `SEQ(B, C)`. For this purpose, the original input stream (`$in`) is joined using a variant of a hash join (called `combine_hash_join`) with the result stream of the matcher on `A` (denoted by `$res1`) and fed into the matcher for `SEQ (B, C)`. This join implementation maintains a queue for all incoming tuples of the first stream and tries to combine the incoming tuple from the second stream with the first element of this queue. As soon as a match was found, the tuple is removed from the queue. Note, that the join condition (`$in1.id == $res1.id`) can be derived from the original pattern specification:

```
$res1 := matcher($in) event SEQ(A) where A = expensive_pred(x);
$res2 := combine_hash_join($in1, $res1) on $in1.id == $res1.id;
$res := matcher($res2) event SEQ(B, C)
        where B = x < 100,  C = x < 200  using (within = 60);
```

Therefore, by using rewriting rules for each pattern supported by the `matcher` operator, a modified dataflow graph can be derived where the pattern matcher is decomposed into a sub-graph of operators.

## 4 Partitioning Dataflow Graphs

Given the partitioning techniques for dataflow graphs with CEP operators as described in the previous section, our goal is to decide about the number of partitions and the split points for partitioning. The main motivation for a partitioning is to avoid congestions in the processing nodes. Redundancy issues for increasing reliability is an orthogonal aspect and not covered here.

However, whether a dataflow graph should be partitioned at all and how many partitions are introduced depends heavily on the work done on the individual partitions. Therefore, a cost-based decision model and a partition planning phase are required which we describe in the following.

**Cost Model.** The foundation for our decision model is a basic rate-based cost model as presented in [7]. A query is represented as dataflow graph $G$ of operators. We assume that tuples arrive with a known average input rate *irate* measured in tuples per second. Furthermore, each operator $op_i$ is characterized by the selectivity $sel_i$ as the ratio of output tuples vs. input tuples per time unit and the tuple processing time $ptime_i$ as the average time for processing a tuple. We assume that these parameters are obtained from a preceding calibration process where runtime statistics of the individual operators on the specific underlying hardware were collected. PipeFlow provides appropriate facilities for this monitoring. Using these parameters, we can estimate for each operator the tuple output rate $orate_i = \min\left\{\frac{1}{ptime_i}, irate_i\right\} \cdot sel_i$. Furthermore, the CPU utilization $u_i$ of an operator $op_i$ is for unary operators $u_i = irate_i \cdot ptime_i$ and for binary operators with two input streams $l$ and $r$: $u_i = (irate_l + irate_r) \cdot ptime_i$. Obviously, $irate_i = orate_j$ if operator $op_j$ provides the input for $op_i$. Similarly, we can estimate the memory consumption $mem_i$ of operator $op_i$ which is for stateless operators $mem_i = c$ where $c$ denotes a constant value and for stateful operators an operator-specific estimation, e.g., for a sliding range-based window of size $wlen$: $mem_i = wlen \cdot irate_i \cdot c$. Based on the estimations for all operators of a dataflow graph $G$, the total utilization and memory consumption are $U_G = \sum_{i \in G} u_i$ and $Mem_G = \sum_{i \in G} mem_i$. A graph $G$ is congestion-free as long as the following conditions are satisfied: $U_G \leq 1$ and $Mem_G \leq AvailMem$. In this case, the complete graph can be deployed to a single machine without congestion in the tuple processing.

However, this does not necessarily mean that this topology represents a graph with minimal processing latency. We define latency as the average time needed for a tuple from arriving at the system until producing a result at a sink operator. Given a path $op_1 op_2 \dots op_k$ in a dataflow graph from a source operator $op_1$ to any other operator $op_k$, the latency for $op_k$ is just

$$latency(op_k) = \begin{cases} ptime_i + latency(op_{i-1}) & \text{if } op_k \text{ is unary} \\ ptime_i + \max\{latency(op_l), latency(op_r)\} & \text{if } op_k \text{ is binary} \end{cases}$$

where for a binary operator $op_l$ and $op_r$ denote the input operators.

Thus, our goal is to find a partitioning $\mathcal{P}(G)$ of graph $G$ with $\mathcal{P}(G) = \{P_1, \dots, P_n\}$ where *(1)* $P_i \subseteq G$, *(2)* each partition is congestion-free, i.e., $\forall P_i \in \mathcal{P} : U_{P_i} \leq 1$ and $Mem_{G_i} \leq AvailMem$, and *(3)* the latency for sink operators is minimal. Note, that partitioning means not only to partition the graph but also to split operators whose utilization exceeds the threshold of 1. Furthermore, if partitions are placed on different servers additional send/receive operators have to be inserted which increase the latency on this edge. We consider only the case of a single sink operator, but this approach can be easily extended to cases with multiple sinks by just aggregate latencies (e.g., average of latencies at all sinks or maximum).

**Partition Planning.** Being able to estimate costs in terms of latency and memory consumption of a given dataflow graph, the second task is to determine an optimal partitioning. Graph partitioning is a well-known problem in graph theory and, in principle, two problems are related here. First, the max-flow min-cut problem theorem states that the maximum amount of flow passing from a source to the sink is equal to the minimum capacity of all cuts. Applied to our problem, an algorithm like Ford-Fulkerson would split the dataflow graph into subgraphs where the required network bandwidth between the subgraphs is minimized. However, in our case cutting a graph requires to change the capacity on this link (due to inserting a pair of send/receive operators) and we want to be able to change the number of partitions, i.e., replicating subgraphs. The second related problem is bin packing as a combinatorial optimization problem where $n$ packages have to be packed into $m$ larger bins which correspond to computing nodes in our case and have a maximum capacity. Bin packing is a NP-hard problem and does not take into account the latency resulting from putting connected operators of the dataflow graph into separate bins.

Thus, we present in the following a greedy algorithm for this problem. Given a dataflow graph $G = \{op_1, op_2, \dots\}$ and a set of computing nodes $M = \{m_1, m_2, \dots\}$, we try to find a partitioning $\mathcal{P}(G)$ where each partition $P_i$ can be assigned to one node of $M$. Starting at sink operators, we try to place operators along the path of $G$ at the same node. As soon as the underlying node of partition $P_i$ is overloaded, a split point is introduced and a new partition is created. A split point is represented by a pair of send/receive operators which contribute to the total costs. In this way, we take communication costs into account. Particularly, we have to handle three cases: *(i)* the resources of a single operator $op_i$ exceed the maximum utilization of a partition, i.e., $u_i > U_{max}$, *(ii)* assigning an operator $op_i$ to an existing partition $P_j$ would exceed its utilization, i.e., $U_{P_j} + u_i > U_{max}$, or *(iii)* otherwise. $U_{max} = 1$ if only this partition is assigned to the node or $U_{max} = 1 - U_G$ if another partition is already running there. For case *(i)* either stream or pattern partitioning have to be applied, for case *(ii)* the dataflow graph has to be split by introducing send/receive operators. However, we can handle both cases with our partition planning approach. Note, that for illustration purposes we use only the CPU utilization $U_{P_i}$ here, but in fact we have to consider a vector of resource utilization, i.e., $\langle u_i, mem_i \rangle$.

Our planning algorithm implements a variant of bin packing and works in three phases. In the preparation phase, all operators are assigned to equivalence classes $\mathcal{C}$ according their maximum path length to the source operators (in number of hops). As the result, each class $C_k \in \mathcal{C}$ contains the set of operators with the same maximum path length. In the second phase shown in Algorithm 1, the operators are processed class-wise starting at the equivalence class with the highest index. The first step (line 6) is to check whether a single operator would overload a node. In this case, the operator has to be split by applying either stream or pattern partitioning. Which of this strategy is used depends on the operator semantics. If multiple strategies are applicable, all of them should be considered as alternatives in the following steps (not shown here). Otherwise, connected operators are assigned to the same partition as long as the maximum utilization is not exceeded. In line 11, the partition with the lowest utilization is determined which has a connection to the operator $op_i$ and will not be overutilized if the operator is added. Then, the operator is assigned to this partition (line 18). If no partition satisfying these condition can be found, a new partition is created (line 14). The predicate `is_connected(`$P_j, op_i$`)` is satisfied iff $op_i$ has either a direct link to any operator in $P_j$ or is already a member of $P_j$, i.e., $op_i \in P_j$. Finally, in the third phase the number of required computing nodes is minimized by merging partitions. Remember that operator $op_i$ is already assigned to partition $P_{\mathsf{current}}$. Now, we try recursively to merge a second partition to which $op_i$ is also connected with $P_{\mathsf{current}}$. As before, this is done only if the partition is not overloaded and with the least utilized partition (Algorithm 2).

**Algorithm 1.** Greedy-based graph partitioning

```
   input : equivalence classes $\mathcal{C} = \{C_0, \dots, C_n\}$
   input : partitioning of operators $\mathcal{P} = \{P_1, \dots\}$
1  maxp := 1;
2  for $k := n$ to 0 do
3      /* process equivalence classes in descending order */
4      foreach $op_i \in C_k$ do
5          /* check whether we have to split the operator */
6          if $u_i > U_{max}$ then
7              split operator $op_i$ and assign the resulting sub-graph to the
               corresponding classes in $\mathcal{C}$;
8              continue;
9          end
10         /* determine the least utilized partition connected to $op_i$ */
11         $P_{\mathsf{current}} := \arg\min_{U_{P_j}} P_j$ with $U_{P_j} + u_i \leq U_{max} \wedge$ is_connected $(P_j, op_i)$;
12         if no $P_{\mathsf{current}}$ found then
13             /* create new partition $P_{maxp}$ and assign $op_i$ to it */
14             $P_{\mathsf{maxp}}$.assign $(op_i)$;
15             maxp := maxp + 1;
16         else
17             /* assign $op_i$ to $P_{\mathsf{current}}$ */
18             $P_{\mathsf{current}}$.assign $(op_i)$;
19             merge_partition $(P_{\mathsf{current}}, op_i)$;
20         end
21     end
22 end
```

Sorting the operators according their distance to the source operators in the query and assigning them to equivalence classes ensures that operators are always processed before their input operators. Second, operators of the same class are processed equally and disjoint paths are assigned to different partitions if necessary. Furthermore, if such branches share a common input operator, the algorithm tries to merge both partitions again.

## 5 Evaluation

For our experiments, we have implemented a synthetic stock ticker events generator which creates a stream of events and allows to control various parameters, e.g., the data distribution. We have produced multiple sets of different sizes: 1M, 3M, 5M, 12M, 20M, and 30M. The schema of the datasets comprises five attributes of type integer: `timestamp`, `seqno`, `symbol`, `price`, and `volume`. We have mainly focused on the `price` attribute which is generated randomly with uniform distribution and various maximum values (10, 50, 100, 300, 500, 700).

**Algorithm 2.** Merging partitions

```
   input: partition $P_m$, operator $op_i$
          set of all partitions $\mathcal{P}$
 1 finished:= false;
 2 $P_{\mathsf{old}} := \{\}$;
 3 while $\neg$ finished do
 4     $P_{\mathsf{cand}} := \arg\min_{U_{P_j}} P_j$ with $U_{P_j} + U_{P_m} \leq U_{max} \land op_i \notin P_j \land$ is_connected
       $(P_j, op_i)$;
 5     if $P_{\mathsf{cand}}$ found then
 6         $P_m := P_m \cup P_{\mathsf{cand}}$;
 7         $P_{\mathsf{cand}} := \{\}$;
 8         if $P_{\mathsf{old}} = P_m$ then finished:= true $P_{\mathsf{old}} := P_m$;
 9     else
10         finished:= true;
11     end
12 end
```

For simplicity, we refer to each set with *price-size-max_value*, e.g., a dataset with the maximum price 100 and 3M tuples is referenced by *price-3M-100*.

In the experiments, we have used different machines as evaluation environments. For centralized and local (i.e., multi-core) experiments, we used a single machine which comes with an Intel Xeon E5-2630 CPU with 24 cores at clock-speed 2.30GHz. The machine runs Ubuntu Linux and has a 128GB main memory. For the distributed setup, we used several machines from our local cluster. Each of this machine has an Intel Xeon E5645 CPU with 6 cores at 2.4GHz and a 8GB main memory. Based on the datasets described above, we have defined a number of queries with various properties for our experiments. These queries differ in their structures, i.e., in pattern syntax. Table 1 shows these queries. Each query was executed ten times and the average throughput of all runs was calculated.

**Baseline and Comparison with Other Systems.** In order to get a baseline for evaluation of the partitioning techniques, we measured the throughput of the queries for different datasets with a pure centralized setup. In addition, we compared the results of implementation with two existing CEP engines SASE[2] and Siddhi[3] by adapting the queries accordingly. We have chosen these engines because they are open source engines and are used as reference in several research articles. We used the same prior two set queries which SASE has already used for the evaluation [8]. In the first query set, the sequence length of the pattern is varied from 2 to 6 with one equality attribute (i.e., `price`) in a way similar to the following query pattern:

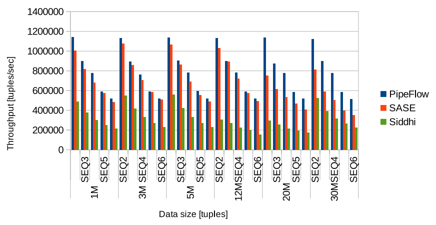
```
event SEQ(A, B, C, ...) where A = true, B = price == A.price,
   C = price == B.price ...
```

For the second query set, multiple equality attributes (e.g., `price` and `volume`) are used indicating that these values should be stable for three consecutive

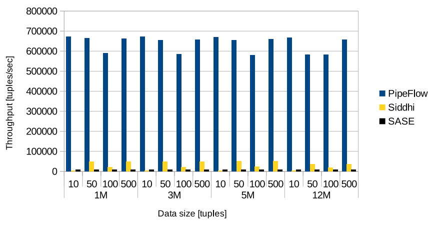
[2] V. 1.0 from `http://code.google.com/p/sase-umass/`

[3] V. 2.0 from `http://svn.wso2.org/repos/wso2/branches/commons/siddhi/`

states (i.e., `B = (price == A.price && volume == A.volume)`). The *price-X-100* and *price-X-[10,50,100,500]* datasets with varying sizes denoted by *X* were used for the first and second query sets, respectively. Fig. 1a and Fig. 1b show the throughput results of the three engines for the first and second query sets, respectively. In all cases, the PipeFlow engine achieves significantly better throughput; particularly with increasing data volume (i.e., greater then 12M tuples) and sequence length and when multiple equivalence attributes are used.



(a) Results for query set #1



(b) Results for query set #2

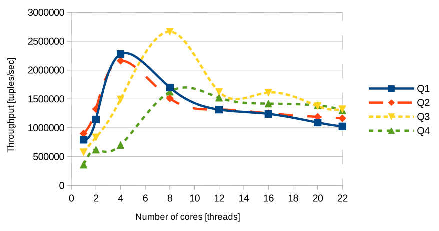
**Fig. 1.** Baseline comparisons

**Stream Partitioning.** The goal of these experiments was to evaluate the performance of the stream partitioning technique described in Sect. 3 both for local partitioning (i.e., on a multi-core machine) and distributed processing in a cluster. The evaluation was done using the synthetic *price-30M-100* data set.

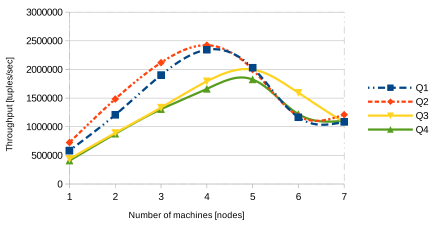
**Table 1.** CEP queries for the experiments

| **ID** | **Pattern** | **Predicate** |
|---|---|---|
| Q1 | `SEQ(A, B, C, D, F)` | `A.price < 20, B.price > 30 AND B.price < 40, C.price > 60, D.price == 90 OR D.price == 95, F.price > 95` |
| Q2 | `SEQ(A, B, C, D, F)` | `A == true, B.price > A.price, C.price > B.price .... F.price > D.price` |
| Q3 | `SEQ(A, B*, C+, D?, F)` | `A.price < 20, B.price > 30 AND B.price < 40, C.price == 50, D.price > 70, F.price == 80` |
| Q4 | `SEQ(A, B+, C)` | `A == true, price > AVG(B.price), C.price > 2*B.price` |

**Local Stream Partitioning.** In this setup, the `map` operator was used to partition the input stream on `symbol` attribute into sub-streams which are processed in parallel by the actual `matcher` operator in separate threads (cores). In Fig. 2a, the throughput results of runs on the 24-core machine are shown. The lowest throughput of all queries was obtained when non-partitioned setup (=1 core) was used. The results show that local stream partitioning helps to improve the performance (throughput) of the `matcher` operator. However, the figure highlights an important issue: the optimal degree of parallelism, i.e., the number of partitions for which the maximum throughput can be achieved. For example, the highest throughput was 2,700,000 events/sec. for Q3 on 8 cores and 2,270,000 events/sec. for Q1 on 4 cores. Depending on the query, the best degree parallelism could be achieved with 4 and 8 cores while further partitioning does not yield better results (according to Amdahl's law). Obviously, this is caused by synchronization overhead and depends on the characteristics of the `map` tasks.



(a) Local partitioning



(b) Distributed partitioning

**Fig. 2.** Baseline comparisons

**Distributed Stream Partitioning.** In this experiment, stream partitioning was used for distributed processing on a shared-nothing cluster of several 6-core machines. The setup used in this experiment have several nodes. The `map` node is used to perform the partition using a hash function into sub-streams. It also forwards the tuples to the actual CEP operators running on other nodes via 0MQ pub-sub operator. The latter nodes (`processing nodes`) process a replicated sub-graphs with a 0MQ subscriber as an input and another 0MQ operator to publish the results. Finally, the `merger` node collects the results from all processing nodes. Fig. 2b shows the results of this experiment. For three of the queries, a higher throughput than with local stream partitioning was achieved – only Q3 is the exception. Moreover, in the range of 2-7 machines (compared to 2-7 threads) the results are better than in Fig. 2a. So, we were able to process approx. 2,420,000 events/secs on four machines. However, also in this setup a further increase of partitions does not yield better results.

**Pattern Partitioning.** In the next experiment, we have evaluated the CEP pattern partitioning technique. To clarify the idea of pattern partitioning, we employed the same CEP query as in Sect. 3 (i.e., `x` replaced with `price` and `id` with `symbol`) with the *price-30M-100* dataset as an input stream. For simplification, *eexpensive_pred* was simulated by a sleep call with varying time and a `price < 20` predicate evaluation. This query is rewritten as described in Sect. 3. In Fig. 3 the throughput results are shown in both cases: in the centralized processing and when CEP pattern partitioning strategy was applied. For the query with different sleep period, the speed-up achieved was in the range between 2.5-2.9. In general, we can observe that the first node determines the maximum throughput and, therefore, is critical for partition planning.

**Partition Planning.** The goal of the experiment described below was to get an impression of the quality of the partitioning scheme obtained using the planning approach from Sect. 4. In order to obtain operator statistics, we ran the queries for a short time with activated profiling statistics provided by PipeFlow. Based on the operator-specific processing time $ptime_i$ and selectivity $sel_i$ derived from these profiling information as well as from the global input rate of the stream, the planner calculated the values form operator utilization $u_i$ and determined the partitioning as described in Algorithm 1.

The results in terms of the number of partitions (#partitions) for the four queries are shown in Table 2. Here, "measured" denotes the optimal values obtained empirically (Fig. 2a) and "planner" the partitioning scheme derived by the planner. We do not give the estimated costs here, because they are very close to the actual costs simply due to the usage of a calibrated cost model. Instead, the results show that the partition planner produces the optimal partitioning scheme for all examples queries. Though, the planner recommends 13 partitions instead of the optimal value of 8, the results in Fig. 2a show that the difference between the throughput on 8 up to 20 partitions is very small. Note, that the results are given only for the local stream partitioning case, but results for the distributed case are similar.

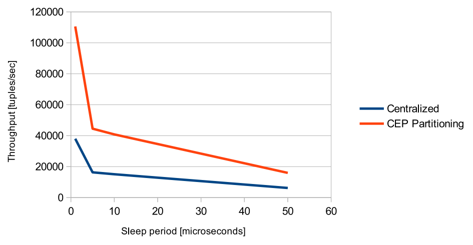


**Fig. 3.** Pattern partitioning result

**Table 2.** Results of partition planning

| Query | #partitions (measured) | #partitions (planner) |
|---|---|---|
| 1 | 4 | 5 |
| 2 | 4 | 4 |
| 3 | 8 | 8 |
| 4 | 8 | 13 |

## 6 Conclusion

In this paper, we have discussed several techniques taking the specifics of CEP operators into account which we have implemented and evaluated in the context of our PipeFlow system. Furthermore, we have introduced a simple rate-based cost model allowing to determine a nearly optimal partitioning of a query. In our ongoing work we plan to extend this static planning approach towards a dynamic autoscaling solution allowing to add new partitions on-the-fly.

## References

1. Brenna, L., Gehrke, J., Hong, M., Johansen, D.: Distributed event stream processing with non-deterministic finite automata. In: DEBS 2009, pp. 3:1–3:12 (2009)
2. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: MapReduce online. Technical Report UCB/EECS-2009-136, EECS Department, UC, Berkeley (October 2009)

3. Hirzel, M.: Partition and compose: Parallel complex event processing. In: DEBS Conference, pp. 191–200. ACM (2012)
4. Krämer, J., Seeger, B.: Semantics and implementation of continuous sliding window queries over data streams. ACM Trans. Database Syst. 34(1) (2009)
5. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: Distributed stream computing platform. In: ICDMW 2010, pp. 170–177 (2010)
6. Schultz-Møller, N.P., Migliavacca, M., Pietzuch, P.: Distributed complex event processing with query rewriting. In: DEBS Conference, pp. 4:1–4:12. ACM (2009)
7. Viglas, S., Naughton, J.F.: Rate-based query optimization for streaming information sources. In: SIGMOD Conference, pp. 37–48 (2002)
8. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: SIGMOD Conference, pp. 407–418 (2006)

# Part VII
# GID 2014 Workshop

# Improving High-Performance GPU Graph Traversal with Compression*

Krzysztof Kaczmarski[1], Piotr Przymus[2], and Paweł Rzążewski[1]

[1] Warsaw University of Technology, Poland
{p.rzazewski,k.kaczmarski}@mini.pw.edu.pl
[2] Nicolaus Copernicus University, Poland
eror@mat.umk.pl

**Abstract.** Traversing huge graphs is a crucial part of many real-world problems, including graph databases. We show how to apply Fixed Length lightweight compression method for traversing graphs stored in the GPU global memory. This approach allows for a significant saving of memory space, improves data alignment, cache utilization and, in many cases, also processing speed. We tested our solution against the state-of-the-art implementation of BFS for GPU and obtained very promising results.

**Keywords:** graph searching, BFS, graph compression, data-intensive computation, GPU, CUDA, graph databases.

## 1 Introduction

Graph algorithms are a foundation of many fields of computer science, including graph databases. Since the graphs appearing in applications tend to be bigger and bigger, both science and industry conduct research to find some more efficient and powerful methods allowing to process them.

Recently, implementations of graph algorithms for the Graphic Processing Units (GPUs) have received considerable attention. A prominent speed-up has been expected due to a massive parallelism offered by the GPU technology. Although the parallel threads programming is now much simplified in this programming model, most of the algorithms (except the ones for embarrassingly parallel problems) need to be redesigned and rewritten. For this reason, new GPU implementations of already known graph algorithms are extensively studied. An example of such an algorithm is the *Breadth-first search (BFS)*, being a building block of many more complicated algorithms and data mining techniques. There have been many studies addressing the implementation of this algorithm on a GPU, followed by a novel work of Merrill, Garland and Grimshaw [11], which outperformed all previous achievements. As the GPU cards often have severe memory limitations, Merril *et al.* also cover the usage of multiple GPU cards, which allows to scale the problem, when the size of the data increases.

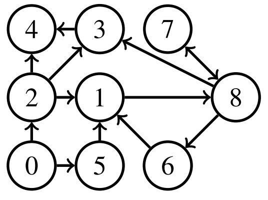
In this work we propose an extension and improvement to the work by Merrill *et al.* [11], by combining BFS with a lightweight compression algorithm. As a result, it

* The project was funded by National Science Centre, decision DEC-2012/07/D/ST6/02483.

is possible to decrease an overall communication cost between the CPU and the GPU and fit significantly larger graphs in a single GPU. Surprisingly, for large graphs it is also possible to improve the processing time of the algorithm on a single GPU device.

### 1.1 Preliminaries

In this paper, we consider directed graphs $G = (V, E)$, being a pair of a vertex set $V$ and a directed edge (arc) set $E \subseteq V \times V$. Neither multiple edges of loops are allowed. We use a well-known compressed sparse row (CSR) format to represent an adjacency matrix of our graph. The vertices are indexed with successive non-negative integers. We store a graph $G = (V, E)$ using two arrays $C$ and $R$. The array $C$ is a concatenation of adjacency list of successive vertices. Therefore its length is exactly $|E|$. The array $R$ has $|V| + 1$ elements. The value of $R[i]$ (for $i \in \{0, 1, .., |V| - 1\}$) points to the index in $C$ of the first element of adjacency list of $i$. In $R[|V|]$ we keep the total number of edges $|E|$. See Figure 1 for an example. Let $C_v$ be a subarray (segment) of $C$ containing nodes pointed by edges of vertex $v$, i.e. an adjacency list of $v$. Clearly $C_v = C[R[v]], \dots, C[R[v+1] - 1]$.



C:

| 2 | 5 | 8 | 1 | 3 | 4 | 4 | 1 | 1 | 8 | 3 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

R:

| 0 | 2 | 3 | 6 | 7 | 7 | 8 | 9 | 10 | 13 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

| Iter. | Vertex frontier | Edge frontier |
|---|---|---|
| 1 | {0} | {2,5} |
| 2 | {2,5} | {1,1,3,4} |
| 3 | {1,3,4} | {4,8} |
| 4 | {8} | {3,6,7} |
| 5 | {6,7} | {1,8} |
| 6 | ∅ | |

**Fig. 1.** A graph, its CSR representation and vertex- and edge-frontiers in iterations of BFS started with node 0

BFS (*Breadth-first search*), one of the best known graph algorithms, starts from a given vertex $v$ and traverses the set of all vertices in a breadth-first manner. Every vertex is labelled with its distance from $v$ (measured in the number of edges). Sometimes we also store the immediate predecessor of each vertex on the shortest path from $v$. This allows us to re-create all such paths. The complexity of a sequential BFS is $O(|V| + |E|)$.

Although BFS is simple and not very interesting on itself, it is a crucial part of many more complicated and useful graph algorithms, e.g. detecting (strongly) connected components, detecting cycles, checking for bipartiteness or finding a maximum flow. We refer the reader to a classical book by Cormen *et al.* [3] for more information about possible uses of BFS.

Parallel versions of BFS have also been investigated. In this approach, we also start with some initial vertex, forming a one-element set $V_1$, called a *vertex-frontier*. Then, in every iteration $i$, the current vertex frontier $V_i$ is expanded, forming a multiset $V'_{i+1}$ of neighbours of vertices from vertex frontier. This multiset is called an *edge-frontier*. To obtain the next vertex frontier $V_{i+1}$, we need to remove from $V'_{i+1}$ all duplicates and all vertices that have already been visited. If so obtained vertex frontier is empty, the algorithms stops. For an example, consult Figure 1.

## 1.2 Short History of BFS Implementations for the GPU

In 2006 NVIDIA published the first version of the CUDA platform which enabled programmers to write arbitrary programs executed by vectorised parallel threads with simplified random memory access. Simplified programming paradigm and spectacular benefits in many applications have led CUDA to become a de-facto standard in the general purpose graphic processor unit programming (GPGPU). In this paper, we assume that a reader is familiar with general purpose GPU computing problems and NVIDIA CUDA architecture. Due to the strong page limit we shall not describe notions like SIMD computing, threads, warps, blocks and various memory hierarchy levels. We kindly suggest reading the documentations provided by NVIDIA [12,13], if necessary.

Breadth-first search, being a building block of many graph algorithms and data mining techniques, appeared to be an important task for GPU devices. In 2007 Harish and Narayanan [7] presented the first CUDA implementation. Their approach processes a graph in levels, starting from a given source vertex. In each iteration all vertices which have to be visited in the next step are marked by parallel threads. No global queue of vertices is created due to possible conflicts in memory access by parallel threads and necessity of duplicates removal. This becomes a problem for graphs of high average degree, where the same vertex may be pointed to by many edges. Parallel post-processing and removal of duplicates in the vertex frontier may become a complex and expensive task itself. An obvious strategy is then not to create the vertex queue at all and visit all vertices in each iteration, checking if each vertex has to be visited or not. Unfortunately this solution leads to quadratic processing time (compared to $O(|V|+|E|)$ for the sequential algorithm).

Deng *et al.* [5] achieved the same quadratic complexity for sparse graphs represented by adjacency matrices. In each iteration a frontier propagation is computed by multiplication of a matrix and a vertex vector. Since the entire graph traversal will require $O(|V|)$ multiplications in the worst case and each multiplication has a complexity of $O(|E|)$, we again get at least quadratic processing time.

The only way to achieve an efficiency comparable to a sequential algorithm is to organize a set of vertices to be visited in the next step optimally, without duplicates. Luo, Wong and Hwu [9] were the first who presented such a solution for the GPU. Their hierarchical queue structure produces a vertex frontier array processing incoming vertices. It is first initiated in the shared memory, using the warp-level threads cooperation. Then, on the block-level, the next step of a queue processing is performed. The final shape of the array is formed in the global memory by a proper copying of block level frontiers. It is worth to mention that an efficient implementation is guaranteed by memory coalesced writes and reads.

In 2011 Hong *et al.* [8] noticed that performance may be significantly improved, if the edge frontier array can be processed by warps instead of single threads. They described a *warp-based* task allocation model and extended it further to virtual warps (smaller than normal 32-thread warps), which allowed for better utilization of threads in multiprocessors. In this approach each thread in a virtual warp processes the same vertex and then it processes a single edge coming out of this vertex. Although authors were successful in general description of a better task allocation method, their BFS implementation is based on the Harish and Narayanan solution and achieves only quadratic performance.

In 2011 Merrill, Garland and Grimshaw [11] presented a new implementation of the BFS, which significantly outperformed all previous works. Since this solution is a starting point for our research, a more detailed description follows in the next section.

## 1.3 Highly Optimized BFS Implementation

The most important part of various BFS implementations is the generation of a vertex frontier for iteration $i$ from a vertex-frontier in iteration $i-1$ (and possibly some other data). Merrill *et al.* [11] described and evaluated a few possible strategies listed below.

*Expand-contract.* In this approach a single kernel takes care of the current vertex-frontier, expands it into an edge-frontier and then contracts it into the next vertex-frontier. First, threads try to detect duplicates within the warp using some heuristic methods. Then the majority of duplicates and already visited vertices is discarded on the block level. Finally, the whole block is assigned to gather the neighbours from unexpanded adjacency lists of the vertices from the vertex-frontier. This assignment is fine-grained and uses a prefix-sum operation.

*Contract-expand.* This approach is very similar to the previous one. A single kernel first contracts a given edge-frontier by deleting already visited vertices and identifying most of the duplicates. Then it expands this vertex frontier to an edge frontier, again using prefix-sum operation.

*Two-phase.* This approach provides separate kernels for the expansion and contraction steps. The expansion kernel uses a clever synchronization strategy, using block-level and warp-level synchronization. Then the duplicates and previously visited vertices are deleted by the contraction kernel, thus producing the next vertex frontier.

The authors finally decide to use the *hybrid strategy*. They perform the *two-phase* approach for large iterations and the *contract-expand* approach in small iterations (i.e. when the edge frontier is small).

It is important to mention that using refined strategies for transforming a vertex-frontier to an edge-frontier and then again to a next vertex-frontier, requires a random access to arrays in which we store the graph. This was not the case in previous, less complicated implementations.

Merrill *et al.* [11] test their solution against the previous solutions for the GPU by Hong *et al.* [8] and Luo *et al.* [9] and also some CPU implementations (both serial and parallel). They were able to obtain a significant improvement in all test cases. The performance they achieve reaches $3.3 \cdot 10^9$ traversed edges per second.

## 1.4 Motivation

Graphs that need to be processed in real-life applications tend to be really huge. For example, Facebook has about $1.23 \cdot 10^9$ monthly active users ($1.26 \cdot 10^9$ total). The number of friendships (i.e. edges in the social graph) is $125 \cdot 10^9$. The average degrees may vary depending on the region. For example there are $128 \cdot 10^6$ Facebook users in the US and an average number of Facebook friends for them is 350. For a detailed analysis of the Facebook social graph, we refer the reader to Ugander *et al.* [19].

Above mentioned examples shows that the time needed to process an input graph is not the only constraint – we also need to be able to store it in the memory (in our case, memory of the GPU device) somehow. A common solution here is to distribute the algorithm to multiple GPUs, each of which keeps a part of the graph.

In this paper we show how to compress the graph, so that we can store more data in a memory of a single GPU device. This, combined with the distribution of work to multiple GPUs, would allow us to process some extremely large graphs. Our compression proposal has the following properties:

– a smooth integration with existing algorithms,
– a minimal instruction overhead,
– a decreased overall memory footprint (so that bigger graphs can be stored on single GPU device),
– a localized warp-centric and thread-centric decompression, minimizing the number of instructions necessary for the decompression.

Our experience from other applications of a compression shows that in many data-intensive algorithms, the decompression procedure may increase instruction throughput, since threads often have to wait for memory reads to complete. We expect that our approach will improve graph processing algorithms in graph databases.

### 1.5 Lightweight Compression Methods

The GPU compression topic was raised in several studies. A considerable has been paid to the so-called *lightweight compression* algorithms, which are primarily intended for real-time applications and favor compression/decompression speed over the compression ratio. Their main purpose is to increase a data throughput by a reduction of a data volume. A detailed description of presented compression algorithms may be found in [6,22,4,15].

Interesting results on the GPU compression were presented by Andrzejewski *et al.* [1], where Word Aligned Hybrid compression algorithm for the GPU was presented. Wu *et al.* [20] discussed an implementation of Lempel-Ziv 77 (LZ77) algorithm on CUDA framework and showed that the performance of this algorithm was poor on the GPU processor when compared to the classical CPU implementation, due to many branches and threads divergence problem. Interesting results in the area of lossless compression on the GPU were presented by Fang *et al.* [6]. Using a compression planner it was possible to achieve a significant improvement in overall query processing on the GPU by reducing a data transfer time from RAM to the global device's memory space.

In our previous work we studied possibilities of composing several lightweight compression methods to improve the compression ratio. We have shown that finding the optimal compression plan by a dynamic data analysis may significantly improve results without sacrificing the decompression speed [16].

The **Fixed Length (FL)** compression method works by removing leading zeros at the most significant bits and thus truncating each value to a fixed length, which is the same for all input elements. The main advantage of the FL algorithm (and its variants) is the fact that both compression and decompression are highly effective on the GPU, because these routines contain minimal number of branching-conditions, which decrease parallelism of SIMD operations. For the best efficiency, dedicated compression and decompression routines are prepared for every bit encoding length with unrolled loops and using only shift and mask operations. In our case this method may be used for both arrays $R$ and $C$. Additionally for $C$, the number of bits may be different for $C_v$ for each vertex $v$. This may lead to a better compression ratio, but also a more complicated decompression.

Another method, which may be used, is the **Frame Of Reference (FOR)**. It works in a similar way to FL, but before a compression it transforms each value into an offset from the reference value (for example the smallest value in the set) in a compression block. The reference value is then stored in the compression header. In this situation, we need exactly $\lfloor \log_2(\max - \min) \rfloor + 1$ bits to encode each value in the frame and $\lfloor \log_2 \min \rfloor + 1$ to store the reference value. The best efficiency can be reached if the ordering of vertices reflects the structure of a graph in such a way that the vertices that are close to each other (in the graph) have relatively close indexes. This can be achieved for example by clustering the graph and ordering the vertices of each cluster separately.

Let us now analyse the applicability of FOR compression of $C$ and $R$ arrays (having in mind that we require a random access to its elements). As explained above, this method divides data into segments named frames. All values in one frame are compressed together but it is possible to decompress any of them by reading a header (the reference value) and a compressed value itself. Reading two values (instead of one) in each read operation is a waste of bandwidth and processing time.

The **Differential Representation** approach stores only the differences between successive data points. Since we need a random access to all elements, a decompression of a value would require to scan the whole array, which takes a linear time. Also the **Dictionary** method (in all variants including the Tunstall encoding) is not suitable for CSR graph representation, since there are too many different values to encode. Creating a dictionary of them would make no sense. The **Run length encoding** stores an array of repeating values as an array of pairs: value and run length (the number of successive appearances of an element). This method is not suitable for us, since we do not expect values to repeat often on consecutive positions in the arrays $R$ (they would correspond to subsequent vertices with no out-edges) or $C$ (they would correspond to subsequent vertices of out-degree 1, having a common neighbour).

All integer encoding methods of variable length based on prefix-codes (e.g. Elias, Shannon-Fano, Huffman, Golomb – consult a comprehensive book by Salomon [18] for more information) do not support a random access and thus are not applicable in case of graph algorithms.

The main drawback of many lightweight compression schemes is that they are prone to outliers in the data frame. For example, consider the following data frame $\{1,2,3,5,7,128\}$ and the FL compression scheme. One could use a 3-bit fixed-length compression to encode almost all values in the frame, but due to the outlier (the value

128) we have to use 7-bit fixed-length compression. A solution of this problem is to use the so-called **Patched Lightweight Compression**. An example of this approach has been proposed by Zukowski *et al.* [22] as a modification of three lightweight compression algorithms. Their main idea is to store outliers as exceptions in an additional array. However, variable number of exceptions lead to many branches in code and decrease efficiency of parallel threads. Various solutions have been proposed to cope with this problem, such as reducing the frame size [22], avoiding too many exceptions [4,21] or separating decoding and patching processes [14,17].

## 2 Compression of the CSR Graph Representation

To choose a compression method which is suitable for graph processing using a GPU device, we need to analyse the behavior of the BFS algorithm in the aspect of memory access and the graph representation in memory.

To our best knowledge, in parallel implementations of BFS there are two possibilities of parallel threads behavior, when reading the edges to be visited in the next algorithm iteration:

- A single thread reads a vertex to be visited and then performs a series of sequential read operations in the array $C_v$, looking for the corresponding edges.
- A group of threads (a warp or a block) reads the same vertex to be visited and then, in parallel, reads all its edges from the array $C_v$. If the number of edges is bigger than the number of threads, then this process iterates until all edges are visited. Similarly, if number of edges is smaller, then some threads may be idle.

In both options, both arrays $C$ and $R$ are accessed randomly, but in the second solution bigger fragments of array $C$ can be read together.

In the case of our example in Figure 1, the first approach would lead to one thread reading vertex number 0 and then its neighbours $\{2,5\}$ followed by two threads processing two vertices 2 and 5 in parallel and producing two edge frontiers $\{1,3,4\}$ and $\{1\}$, respectively. The number of read operations is three in the case of the the first thread and only one for the second thread.

The second approach would use a group of threads (most effectively a warp) for reading vertex 0 from the vertex frontier. Next two threads would concurrently read two edges from the $C_0$ array and then produce one edge frontier containing $\{2,5\}$. In the second iteration two groups (warps) would access two distant places in array $C$ reading corresponding edge frontiers. Each thread in a group would read its corresponding value: 1,3 and 4 in the first group and 1 in the second group. Two resulting frontiers will be created simultaneously in one step. If a coalesced memory access is possible, then the process would end in two memory read operations. This fact of a better thread utilization in the case of a warp-level edge-frontier access was already noted by other authors [9,11] and is related to the parallel processing model of the GPU device.

Considering the memory space needed to store the graph, we observe that the array $R$ is sorted and stores indices of the much bigger array of edges $C$. Each segment $C_v$ can also be sorted. Both arrays contain only non-negative integers. We also need to note that $R$ contains much bigger values (its last element is the sum of degrees of all vertices, which is equal to the total number of edges).

The threads behaviour and memory representation leads to important conclusions:
1. the array $R$ after compression must allow for a random access to any of its elements;
2. the same holds for the array $C$, but this array may be divided into blocks $C_v$;
3. a group of threads may cooperate in decompression of edge frontiers read from $C$.

Note that BFS is a data intensive algorithm. It performs very few computations and can significantly slow down if a decompression method is too expensive or creates some unwanted threads divergence by branching.

The above statements focus our attention on lightweight compression methods, which are local, do not use patching mechanisms and allow a value to be decompressed solely upon information from the data read in a constant time. According to the analysis from Section 1.5, the FL compression method seems to be the most flexible and promising. Therefore we chose to use it in our approach.

## 2.1 Fixed-Length Compression of Large In-Memory Arrays

In this section we discuss in detail the consequences of choosing the FL compression scheme for a large array, which require a random access.

**Memory Organization Consequences of the FL Compression Method.** Consider an array compressed with the FL algorithm, with each value written on $\ell$ bits. We store them in an array of $k$-element memory cells (in most cases we shall use $k = 32$, as it is best supported by current GPU devices). Observe that some values will be stored in two consecutive memory cells. In those cases, to retrieve the value, we need to read two cells, which significantly increases the cost of the read operation. Therefore we want to keep the number of such values as small as possible. In a perfect situation, when $\ell$ divides $k$, there are no values spanning over multiple cells.

Consider a block $A$ of our array, whose length is equal to $\mathrm{lcm}(k,\ell)$, being the least common multiple of $k$ and $\ell$ (as the whole array consists of such blocks and some remainder, which has constant length and therefore can be omitted in our analysis). Let $x$ and $y$ be integers such that $\mathrm{lcm}(k,\ell) = x \cdot \ell = y \cdot k$. The number of values spanning over two consecutive cells is exactly $y - 1 = \frac{\ell}{\gcd(k,\ell)} - 1$ $(\star)$.

From this we can see that there are two ways to minimize the number of read operations – by making $\ell$ small or by making $\gcd(k,\ell)$ large.

**Expected Cost of Random Array Access.** The above statements lead to an important conclusion that the additional cost of memory operations (when compared to an array without any compression) depends on the values of $\ell$ and $k$. Suppose we have an array of $X$ values, consisting of blocks of size $\frac{k}{\gcd(k,\ell)}$ (as mentioned before, we do not care about some remainder, as its length is constant). Therefore, from $(\star)$, the total number of values which occupy two consecutive cells is $\frac{X}{x} \cdot \left(\frac{\ell}{\gcd(k,\ell)} - 1\right) = X \cdot \frac{\ell - \gcd(k,\ell)}{k}$. Let $\alpha := \frac{\ell - \gcd(k,\ell)}{k}$. Now $\alpha X$ is the number of values spanning over two consecutive cells. If we choose a random value (with a uniform probability), we get a value in two cells with probability $\frac{\alpha X}{X} = \alpha$ and a value in just one cell with probability $\frac{1-\alpha X}{X} = 1 - \alpha$. Suppose we want to read $m$ random values, chosen with uniform probability. The expected value of read operations is:

$$\mathbb{EX}(\text{number of read operations when reading } m \text{ values}) =$$
$$m \cdot \mathbb{EX}(\text{number of read operations when reading one value}) =$$
$$= m \cdot (1 \cdot (1-\alpha) + 2 \cdot \alpha) = m \cdot (1+\alpha) = m \cdot (1 + \frac{\ell - \gcd(k,\ell)}{k}).$$

This is compared with $m$ read operations needed to retrieve $m$ values from a non-compressed array. Obviously if $\ell$ divides $k$ then additional cost is 0.

Moreover, observe that so far we only considered the simplest case when each value was immediately followed by the next one and we had no unused bits. However, this may not be an optimal approach. Consider for example cells of size $k = 32$ bits and $\ell = 5$. We may consider storing 6 values in a single cell and leaving two remaining bits unused (so the next value starts in the next cell). With this approach we increase the size of the data (and thus reduce the compression ratio), but we never have to read more than one cell to retrieve a single value, which improves the efficiency of processing (we need $m$ read operations to read $m$ values).

Actually, in our experiment we use such a modification. Instead of storing each value on $\ell = \max\{\lfloor \log_2 z \rfloor + 1 \colon z \text{ is a value to be stored}\}$ bits, we chose some $\ell' \geq \ell$, which allows us to reduce the number of values spanning over two cells. Table 1 shows the optimal values $\ell$ and chosen values $\ell'$ for benchmark graph. If $\ell' = 21$, then we just stored three values in two 32-bit memory cells and left one last bit unused. For the graphs with $\ell = \ell' = 16$, we just stored two values in a single 32-bit memory cell. Observe that a small loss in the compression ratio is justified by fewer read operations.

## 3 Benchmark Graphs and Results of Experiments

In order to confirm the effectiveness of our approach we test it against the fastest known BFS implementation, which was already discussed in Section 1.3. Unfortunately most of the data sets mentioned by Merrill *et al.* were not available when we performed the tests. We only managed to download several Citeseer and DBLP graphs. However, we were able to use the same graph generator: R-MAT (see Chakrabarti *et al.* [2] for details). Such graphs reflect specific properties of large graphs appearing in real-world applications.

We run the experiments on graphs having from $65.5 \cdot 10^3$ to $2 \cdot 10^6$ vertices and up to $300 \cdot 10^6$ edges. Table 1 lists the parameters of benchmark graphs.

The code of the solution by Merrill *et al.* [11] is available for public as a part of the *back40computing (b40c)* project [10]. Therefore we were able to apply our improvements directly into their fine-tuned implementation. Although the changes were not straight-forward, eventually we modified the original code in two aspects (altogether highly touching many places in the code):

– creating graph representation in the memory (by adding the FL compression),
– the function call controlling an access to the elements of $C$ (decoding vertices).

The graph compression depends on the selected method's parameters $\ell$, $k$ and was explained in Section 2.1.

The internal architecture of the *b40c* implementation is using almost all available on-chip shared memory. We were not able to utilize it in the decompression process. Therefore the vertex decoding had to be done in a very simple way just using threads' private registers and without any additional intercommunication between threads. This solution required more memory operations and processing when compared to the ideal one. The differences between the thread behavior in the original approach and our approach is presented in Figure 2.

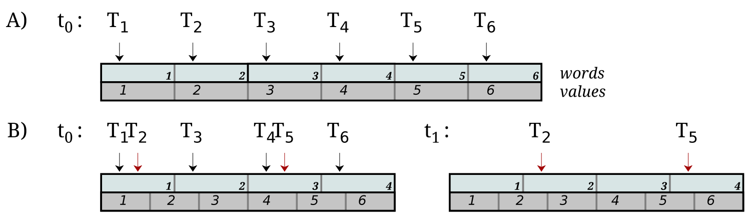


**Fig. 2.** A simple strategy of parallel values decoding. A) no compression. A single memory cell stores a single value, which is read by a single thread. B) The FL compression. One memory cell stores 1.5 values. Some threads need to read two cells with 100% cache hits. Overall cache usage is decreased. No threads intercommunication.

An example of a decompression function is shown in Figure 3. It was prepared for $\ell' = 21$ bits. Choosing another length would require slight changes in the function. The key point in this code are modulo operations which are done by bit operations only. They are necessary to find beginnings of compressed values. In some cases, if the value is split across two memory cells, a thread needs to perform another read operation (see line 13). This is necessary if we assume no communication between threads.

We believe that this approach will be successful also in other applications since it allows for a random array access and imposes no additional restrictions.

```
 1  efine NBITSTOMASK(n) ((1<<(n)) - 1)
 2  efine GETNBITS(a,n)  ((a) & NBITSTOMASK(n)) // returns a0a1..an-1
 3  efine GETNPBITS(a, n, p) GETNBITS((a>>p), (n)) // returns apap+1..ap+n-1
 4
 5  device__ __forceinline__ static
 6  id Ld_21_64(T &val, T *ptr, long n) {
 7  unsigned int a = (unsigned) n;
 8  a = (0x55555555*a+(a>>1)-(a>>3))>>30;
 9  unsigned int pos = ((unsigned)n-a)*0xAAAAAAAB;
10  pos = pos * 2 + (a>>1);
11  val = GETNPBITS(ptr[pos],21-10*(a&1),((a*21)&31));
12  if(a&1)
13    val = val|GETNPBITS(ptr[pos+(a&1)],10*(a&1),0)<<11;
```

**Fig. 3.** An example of a data retrieving function. **Ld_21_32** function with FL decompression for $\ell' = 21$ and $k = 32$ (three values are encoded in two subsequent integers) $n$ – an index of an element to be decoded.

**Table 1.** Experimental data sets. The first group of columns shows the number of vertices, edges and an average degree of each graph. The second group shows an optimal ($\ell$) and a chosen by us ($\ell'$) length of an encoding of a single value and $k$, being the size of a single memory cell. Third group shows the size of $C$ before compression, after compression with each value encoded on $\ell$ or $\ell'$ bits and corresponding compression ratios.

| **Graph** | vert. $\cdot 10^3$ | edges $\cdot 10^6$ | avg. degree | $\ell$ bits | $\ell'$ bits | $k$ bits | $C$ [MB] | $\text{FL}_\ell(C)$ [MB] | compr. ratio ($\ell$) | $\text{FL}_{\ell'}(C)$ [MB] | compr. ratio ($\ell'$) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| citationCiteseer | 268 | 1.15 | 4.3 | 20 | 21 | 32 | 4.39 | 2.74 | 0.63 | 2.92 | 0.67 |
| coAuthorsCiteseer | 227 | 0.81 | 3.58 | 20 | 21 | 32 | 3.09 | 1.93 | 0.63 | 2.06 | 0.67 |
| coAuthorsDBLP | 299 | 0.98 | 3.26 | 20 | 21 | 32 | 3.74 | 2.34 | 0.63 | 2.49 | 0.67 |
| coPapersCiteseer | 434 | 16.03 | 37.55 | 20 | 21 | 32 | 61.15 | 38.22 | 0.63 | 40.77 | 0.67 |
| coPapersDBLP | 540 | 15.24 | 28 | 20 | 21 | 32 | 58.14 | 36.33 | 0.63 | 38.76 | 0.67 |
| RM131Kv19Me | 131 | 19.65 | 150 | 17 | 21 | 32 | 74.96 | 39.82 | 0.53 | 49.97 | 0.67 |
| RM131Kv39Me | 131 | 39.30 | 300 | 17 | 21 | 32 | 149.92 | 79.64 | 0.53 | 99.95 | 0.67 |
| RM131Kv78Me | 131 | 78.60 | 600 | 17 | 21 | 32 | 299.84 | 159.29 | 0.53 | 199.89 | 0.67 |
| RM2Mv150Me | 2000 | 150 | 150 | 21 | 21 | 32 | 572.20 | 375.51 | 0.66 | 381.47 | 0.67 |
| RM2Mv301Me | 2000 | 301 | 301 | 21 | 21 | 32 | 1148.22 | 753.52 | 0.66 | 765.48 | 0.67 |
| RM2Mv350Me | 2000 | 350 | 350 | 21 | 21 | 32 | 1335.14 | 876.19 | 0.66 | 890.10 | 0.67 |
| RM2Mv400Me | 2000 | 400 | 400 | 21 | 21 | 32 | 1525.88 | 1001.36 | 0.66 | 1017.25 | 0.67 |
| RM65.5Kv10Me | 65.5 | 10 | 152.67 | 16 | 16 | 32 | 38.15 | 19.07 | 0.50 | 19.07 | 0.50 |
| RM65.5Kv67Me | 65.5 | 67 | 1022.90 | 16 | 16 | 32 | 255.58 | 127.79 | 0.50 | 127.79 | 0.50 |
| RM65.5Kv104Me | 65.5 | 104 | 1587.78 | 16 | 16 | 32 | 396.73 | 198.36 | 0.50 | 198.36 | 0.50 |
| RM65.5Kv268Me | 65.5 | 268 | 4091.60 | 16 | 16 | 32 | 1022.34 | 511.17 | 0.50 | 511.17 | 0.50 |

All the experiments were executed on the same model of GPU processor as the experiments by Merrill *et al.* [11]. Detailed hardware configuration: two six-core processors *Intel® Xeon® E5649 2.53GHz*, 8GB RAM and *Nvidia® Tesla M2070* card.

## 3.1 Discussion on Results

The results of our experiments are shown in Table 2 (average value of 10 executions).

*Compression.* Due to the limitation of the BFS implementation we worked with, we could only use a very simple FL compression scheme with a random access to the array of edges ($C$). Obviously in such a case the compression ratio depends on the number of bits which are used to store a vertex identifier. In most of the reference sample data sets only 21 bits were used, which was enough to pack three nodes into two integers (i.e. a 64 bit segment). In such cases, achieved compression ratio varied from 0.53 to 0.62 (of the original size).

Let us now analyse how big graphs may be stored in a GPU device with 6 GB of memory (this is the theoretical storage space of *Nvidia® Tesla M2070*), assuming that the average degree of a node is 40 and the values are stored as 32-bit integers. Using the CSR representation, a single vertex $v$ requires 4 bytes for a corresponding cell in $R$ array and $40 \cdot 4$ bytes on average for the $C_v$ array. Therefore, in theory, a professional GPU device with a memory of 6 442 450 944 bytes lets us to store a graph of up to

$n := 39\,283\,237$ vertices (of course this would require to use all the memory just for the graph representation, leaving no space for e.g. some additional structures used by the algorithm, so it is just a theoretical upper bound). To store the indices of these nodes we need 26 bits. By compressing the array $C$ with FL method and using $\ell = 26$ and $k = 32$, we could pack 6 values in 5 integers (by wasting 4 bits). A single vertex with its out-edges needs now $4(1 + \lceil\frac{5 \cdot 40}{6}\rceil) = 140$ bytes on average. Therefore our graph with $n$ vertices would occupy only 5 499 653 180 bytes together gives 5.12GB. Memory we saved in such a way would let us to store 6 734 629 additional vertices and their compressed edges (note that the vertex indices still can be stored on 26 bits). In this configuration we managed to increase the practical device capacity by over 17%. Notice that this can be significantly improved for graphs with larger average degree (as the number of bits needed to represent a vertex remains low and the size of $C$ grows).

This may be crucial is some cases – e.g. a graph with $2 \cdot 10^9$ vertices and $400 \cdot 10^9$ was too big to fit into the memory of the GPU device without a compression (Table 2).

We also observe that it would need a device with memory storage of 328 GB to use all 32 bits in an integer encoding the vertex indices. Therefore, in the case of current GPU devices, savings using the FL compression are always possible.

Another benefit of this compression method is that it can significantly decrease the memory bandwidth when executing multi-GPU algorithms and whenever memory transfer of a graph or its parts is used.

*BFS Algorithm Time.* At the beginning we have to observe that the time of processing of compressed graphs depends on two factors: a ratio between $\ell$ (or $\ell'$) and $k$, i.e. the efficiency of a compression (being the number of values we can pack into a single memory cell) and an average degree of a vertex.

Moreover, we observe that the additional compression/decompression cost is compensated for medium-sized graphs. For large graphs we are even able to speed up the computation.

**Table 2.** The time of BFS processing for benchmark graphs [ms] (smaller is better). The last column shows the improvement over the original solution (greater is better). The graph RM2Mv400Me with $400 \cdot 10^9$ edges could not be processed without the compression.

| | **Processing time [ms]** | | **Speed up** |
|---|---|---|---|
| **Graph** | **b40c** | **b40c+FL** | **[%]** |
| citationCiteseer | 2.9948 | 3.4951 | -14.31 |
| coAuthorsCiteseer | 2.0487 | 2.6138 | -21.62 |
| coAuthorsDBLP | 2.4007 | 2.9336 | -18.17 |
| coPapersCiteseer | 12.7294 | 14.2871 | -10.90 |
| coPapersDBLP | 11.5055 | 12.5845 | -8.57 |
| RM131Kv19.6Me | 7.0177 | 7.0178 | 0.00 |
| RM131Kv39.3Me | 12.6992 | 12.5758 | 0.98 |
| RM131Kv78.6Me | 23.897 | 23.5884 | 1.31 |

| | **Processing time [ms]** | | **Speed up** |
|---|---|---|---|
| **Graph** | **b40c** | **b40c+FL** | **[%]** |
| RM2Mv150Me | 61.3072 | 59.4512 | 3.12 |
| RM2Mv301Me | 115.656 | 111.2375 | 3.97 |
| RM2Mv350Me | 132.7916 | 127.9881 | 3.75 |
| RM2Mv400Me | X | 144.6979 | X |
| RM65.5Kv10Me | 4.6509 | 4.3214 | 7.62 |
| RM65.5Kv67Me | 22.9852 | 20.9861 | 9.53 |
| RM65.5Kv104Me | 35.6296 | 32.466 | 9.74 |
| RM65.5Kv268Me | 97.943 | 86.212 | 13.61 |

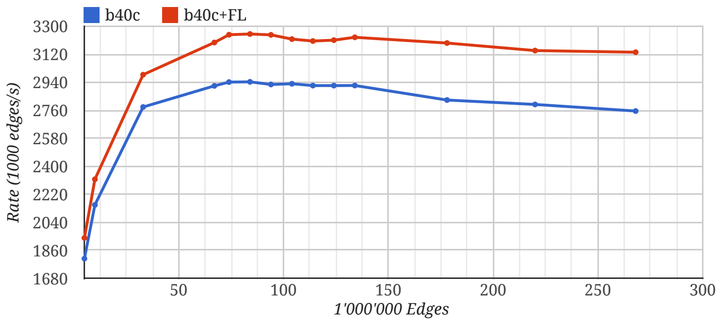


**Fig. 4.** Rate (edges per millisecond, greater is better) of the BFS algorihtm for graphs with $65.5 \cdot 10^3$ vertices and different number of edges

## 4 Conclusions and Future Work

We have presented a method of compressing graphs stored in the CSR format and processed in GPU devices. Our solution is characterized by an ultra-fast decompression time, a simplicity of integration with already existing algorithms and an optimization of parallel threads computation.

We evaluated our solution against the state-of-the-art in graph algorithms – the highly-optimized BFS implementation for GPU devices by Merrill *et al.* [11]. Our results show that for big graphs the compression not only allows to fit more vertices and edges into a single GPU, but also speeds up the processing by a better utilization of memory caches.

We believe that our improvement can also be used in a case of a distributed computation performed on multiple GPU nodes or in clusters. Using a compression should significantly speed up the most critical operation, which is a data transfer.

## References

1. Andrzejewski, W., Wrembel, R.: GPU-WAH: Applying gPUs to compressing bitmap indexes with word aligned hybrid. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part II. LNCS, vol. 6262, pp. 315–329. Springer, Heidelberg (2010)
2. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: A recursive model for graph mining. In: SDM, pp. 442–446 (2004)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
4. Delbru, R., Campinas, S., Samp, K., Tummarello, G.: Adaptive frame of reference for compressing inverted lists. Technical report, DERI – Digital Enterprise Research Institute (December 2010)
5. Deng, Y.S., Wang, B.D., Mu, S.: Taming irregular EDA applications on GPUs. In: Proceedings of the 2009 International Conference on Computer-Aided Design, ICCAD 2009, pp. 539–546. ACM, New York (2009)
6. Fang, W., He, B., Luo, Q.: Database compression on graphics processors. Proceedings of the VLDB Endowment 3(1-2), 670–680 (2010)

7. Harish, P., Narayanan, P.J.: Accelerating large graph algorithms on the GPU using CUDA. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 197–208. Springer, Heidelberg (2007)
8. Hong, S., Kim, S.K., Oguntebi, T., Olukotun, K.: Accelerating CUDA graph algorithms at maximum warp. In: Cascaval, C., Yew, P.-C. (eds.) PPOPP, pp. 267–276. ACM (2011)
9. Luo, L., Wong, M.D.F., Mei, W., Hwu, W.: An effective GPU implementation of breadthfirst search. In: Sapatnekar, S.S. (ed.) DAC, pp. 52–55. ACM (2010)
10. Merrill, D.: Back40computing (2013), `https://code.google.com/p/back40computing/`
11. Merrill, D., Garland, M., Grimshaw, A.S.: Scalable gpu graph traversal. In: Ramanujam, J., Sadayappan, P. (eds.) PPOPP, pp. 117–128. ACM (2012)
12. NVIDIA Corporation. NVIDIA CUDA C programming guide 5.5 (2013)
13. NVIDIA Corporation. CUDA C Toolkit v.5.5 (2014)
14. Przymus, P., Kaczmarski, K.: Improving efficiency of data intensive applications on GPU using lightweight compression. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM 2012 Workshops. LNCS, vol. 7567, pp. 3–12. Springer, Heidelberg (2012)
15. Przymus, P., Kaczmarski, K.: Dynamic compression strategy for time series database using GPU. In: Catania, B., et al. (eds.) New Trends in Databases and Information Systems. AISC, vol. 241, pp. 235–244. Springer, Heidelberg (2014)
16. Przymus, P., Kaczmarski, K.: Dynamic compression strategy for time series database using GPU. In: Catania, B., et al. (eds.) New Trends in Databases and Information Systems. AISC, vol. 241, pp. 235–244. Springer, Heidelberg (2014)
17. Przymus, P., Kaczmarski, K.: Time series queries processing with GPU support. In: Catania, B., et al. (eds.) New Trends in Databases and Information Systems. AISC, vol. 241, pp. 53–60. Springer, Heidelberg (2014)
18. Salomon, D.: Data Compression: The Complete Reference. Springer (1998)
19. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the Facebook social graph. CoRR, abs/1111.4503 (2011)
20. Wu, L., Storus, M., Cross, D.: CS315A: Final project CUDA WUDA SHUDA: CUDA compression project (2009)
21. Yan, H., Ding, S., Suel, T.: Inverted index compression and query processing with optimized document ordering. In: Proc. of the 18th Intern. Conf. on World Wide Web, pp. 401–410. ACM (2009)
22. Zukowski, M., Heman, S., Nes, N., Boncz, P.: Super-scalar RAM-CPU cache compression. In: Proc. of the 22nd Intern. Conf. on Data Engineering, ICDE 2006, pp. 59–59. IEEE (2006)

# GPU-Accelerated Method of Query Selectivity Estimation for Non Equi-Join Conditions Based on Discrete Fourier Transform

Dariusz Rafal Augustyn and Lukasz Warchal

Silesian University of Technology, Institute of Informatics,
16 Akademicka St., 44-100 Gliwice, Poland
{draugustyn,lukasz.warchal}@polsl.pl

**Abstract.** Selectivity factor is obtained by database query optimizer for estimating the size of data that satisfy a query condition. This allows to choose the optimal query execution plan. In this paper we consider the problem of selectivity estimation for inequality predicates based on two attributes, therefore the proposed solution allows to estimate the size of data that satisfy theta-join conditions. The proposed method is based on Discrete Fourier Transform and convolution theorem. DFT spectrums are used as representations of distribution of attribute values. We compute selectivity either performing Inverse DFT (for an inequality condition based on two attributes) or avoiding it (for a single-attribute range one). Selectivity calculation is a time-critical operation performed during an on-line query preparing phase. We show that by applying parallel processing capabilities of Graphical Processing Unit, the implementation of the method satisfies the assumed time constraint.

**Keywords:** Query Selectivity Estimation, Theta-Join Condition, Discrete Fourier Transform, CUDA.

## 1 Introduction

Query processing includes an optimization phase when the most efficient plan of query execution should be obtained. This activity is performed by the cost query optimizer (CQO). To choose the optimal method of query execution CQO needs an early estimation of the size of data that satisfy a selection condition. To do this CQO calculates so-called query selectivity factors for particular predicates that are included in a query condition. Selectivity for a query condition is the number of table rows satisfying it, divided by the number of all rows in this table. For a single-table range query with a condition based on a single attribute with continuous domain the selectivity is defined:

$$sel(Q(a < x_j < b) = \int_a^b f_{x_j}(x_j)dx_j. \qquad (1)$$

where: $x_j$ – a table attribute, $a$, $b$ – range query boundaries, $f_{x_j}(x_j)$ – a probability density function (PDF) of $x_j$ distribution. Thus, to calculate the selectivity estimation we need some representation of PDF.

There are many approaches to the problem of selectivity estimation for a complex query condition based on many table attributes. Then obtaining a selectivity value requires to have a multidimensional representation of multivariate PDF of attributes values. For high dimensions we should overcome the curse of dimensionality problem, because multidimensional representations become too much space-consuming when dimensionality increases. There are well-known methods of creating a space-efficient representation of multidimensional distribution of attributes values. Some of them are based on Discrete Cosine Transform [13], Cosine Series [17] or Discrete Wavelet Transform [7]. While Discrete Fourier Transform has been used in selectivity estimation for equi-join conditions [16] where attributes comes from different tables.

Sometimes a distribution of values in database enables to apply the attribute value independency rule (AVI) which allows to obtain a selectivity without having multidimensional representation of joint distribution but using only a space-efficient 1-dimensional representations of marginal distribution. This is the well-know technique commonly used in CQO. AVI rule will be also used in the method proposed in this paper.

In this paper we consider the problem of estimating selectivity values for a query condition which determines non equi-join between two database tables. Such condition is equivalent to a range condition based on a difference between joined attributes. Calculation of a selectivity value for such query condition is based on a distribution of attribute difference values and it uses Discrete Fourier Transform (DFT) and convolution theorem.

The query optimization phase is performed during an on-line query processing and it is a time-critical operation. Commonly, we assume that an execution time of task of query preparing should be less than about 10ms, so a subtask of selectivity estimation should take significantly less, e.g. let us assume no more than 1ms.

The method of selectivity estimation for query selection conditions based on sum of attributes has been already considered [3]. The problem of DFT-based selectivity estimation for such condition (a range condition for a sum of attributes) is similar to the one considered in this paper. However, the experimental results from [3] shows that the simple Java-based implementation of that approach satisfies the assumed time constraint ($<$ 1ms) only for vectors with a length less or equals 256. Thus, we propose the enough time-efficient hardware-accelerated implementation. It enables our method of selectivity estimation for non equi-join conditions and high resolution representations of PDFs of attribute values.

Hardware-accelerated modules supporting database operations becomes popular including those that use parallelization capabilities of modern Graphical Processing Unit (GPU) (e.g. [6,10,5] and many others). At the field of query selectivity estimation, there are many methods which involve GPU processing (e.g. [4,11,18]), like the one proposed in this paper.

The contributions of this paper are:

- the algorithm of query selectivity estimation (based on DFT spectrums) for query conditions like $a < x_j - x_l < b$, where independent attributes $x_j$ and $x_l$ may come either from the same table (selection condition) or from different tables (theta-join condition),
- the algorithm of query selectivity estimation for range conditions like $a < x_j < b$; the algorithm operates directly on a DFT spectrum (with no need to perform inverse DFT),
- implementations of these algorithms that use capabilities of CUDA [14] and SSE technologies [12], i.e. the GPU-based implementation which uses cuFFT library [15] and the CPU-based one which uses FFTW library [8].

## 2 Selectivity Estimation for Non Equi-Join Condition

A theta-join is any Cartesian product of two tables $t_1$, $t_2$ that is filtered by a condition $t_1.x_j \Theta t_2.x_l$ which compares attribute values of $x_j$ and $x_l$ from both tables (where $x_j$ is an attribute from $t_1$ and $x_l$ from $t_2$, respectively).

An execution plan of a query which contains a theta-join condition depends on an estimated size of result set which includes joined tuples. We propose the query selectivity estimation method for non equi-join conditions i.e. $t_1.x_j < t_2.x_l$.

Let us assume that $f_{x_j}(x_j)$ and $f_{x_l}(x_l)$ are PDFs of $x_j$, and $x_l$. We assume that values of $x_j$ and $x_l$ are independent because they come from different tables. Instead of considering $x_j < x_l$ we may consider the following condition:

$$z = x_j - x_l < 0. \tag{2}$$

We may find continuous Fourier transforms of PDF of $x_j$ and $x_l$:

$$\begin{aligned} \mathcal{F}(f_{x_j}) = \mathcal{F}_{x_j}(t) = \int_{-\infty}^{+\infty} e^{-itx_j} f_{x_j}(x_j)\mathrm{d}x_j, \\ \mathcal{F}(f_{x_l}) = \mathcal{F}_{x_l}(t) = \int_{-\infty}^{+\infty} e^{-itx_l} f_{x_l}(x_l)\mathrm{d}x_l, \end{aligned} \tag{3}$$

We may find PDF of z denoted by $f_z(z)$ using:

$$\mathcal{F}(f_z) = \mathcal{F}_{x_j}(t)\mathcal{F}_{x_l}(-t) = \mathcal{F}_{x_j}(t)\overline{\mathcal{F}_{x_l}}(t), \tag{4}$$

where $\overline{\mathcal{F}_{x_l}}(t)$ is a complex conjugate of $\mathcal{F}_{x_l}(t)$.

This allows to obtain the required selectivity as follows:

$$\mathrm{sel}(x_j < x_l) = \mathrm{sel}(z < 0) = \int_{-\infty}^{0} f_z(z)\mathrm{d}z. \tag{5}$$

Using (4) and (5) we may obtain the selectivity value from spectrums:

$$\mathrm{sel}(z < 0) = \int_{-\infty}^{0} \mathcal{F}^{-1}\left(\mathcal{F}_{x_j}(t)\overline{\mathcal{F}_{x_l}}(t)\right)\mathrm{d}z. \tag{6}$$

### 2.1 Using a DFT Spectrum as a Representation of Attribute Values Distribution

Let us find $min_{all}$ and $max_{all}$ values:

$$min_{all} = \min(\min(x_j), \min(x_l)) \wedge max_{all} = \max(\max(x_j), \max(x_l)). \tag{7}$$

We use equi-width histograms as representations of PDFs. $N$ is the number of buckets of the equi-width histograms. $(max_{all} - min_{all})/N$ is the length of a bucket of the equi-width histograms. Both histograms' domains start at $min_{all}$. For the $x_j$ attribute we define:

$$\boldsymbol{F}_j = \left(f_{jn}\right)_{n=0}^{N-1} \tag{8}$$

– a vector whose elements represent frequencies of occurrences of attribute values in a histogram bucket. $f_{j_n}$ is the probability that $x_j$ belongs to the $(n+1)$-th bucket of the equi-width histogram.

To utilize the formula (6) we use Discrete Fourier Transform (DFT).
Let us introduce a DFT spectrum – the vector of complex values:

$$\boldsymbol{S_j} = \left(s_{jk}\right)_{k=0}^{N-1}. \tag{9}$$

Because of $\boldsymbol{S}_j = \mathrm{DFT}(\boldsymbol{F}_j)$ we may obtain spectrum coefficients as follows:

$$s_{jk} = \sum_{n=0}^{N-1} f_{jn} e^{-i\frac{2\pi kn}{N}}. \tag{10}$$

By applying inverse DFT ($\mathrm{DFT}^{-1}$) we may find $f_{jn}$ using $s_{jk}$ as follows:

$$f_{jn} = \frac{1}{N} \sum_{k=0}^{N-1} s_{jk} e^{i\frac{2\pi kn}{N}}. \tag{11}$$

The procedure of selectivity estimation requires to create a statistics data. This preliminary step is preformed during an update statistics.
In this phase we create temporary equi-width histograms that describe $x_j$ and $x_l$ distributions (i.e. we create $\boldsymbol{F}_j$ and $\boldsymbol{F}_l$ vectors).
To obtain the distribution of difference between $x_j$ and $x_l$ we have to use zero-padded frequency vectors $\boldsymbol{F}'_j$ and $\boldsymbol{F}'_l$, with lengths equal $2N$. For example $\boldsymbol{F}'_j$ is defined as follows:

$$\boldsymbol{F}'_j = \left(f'_{jn}\right)_{n=0}^{2N-1} \wedge f'_{jn} = \begin{cases} f_{jn} & \text{for } n = 0, \ldots, N-1 \\ 0 & \text{for } n = N, \ldots, 2N-1 \end{cases} \tag{12}$$

Then we obtain relevant spectrum vectors: $\boldsymbol{S}'_j = \mathrm{DFT}(\boldsymbol{F}'_j)$ and $\boldsymbol{S}'_l = \mathrm{DFT}(\boldsymbol{F}'_l)$, both also with lengths equal $2N$.

During the update statistics we prepare $\boldsymbol{S}''_j$, $\boldsymbol{S}''_l$ so-called shifted spectrum vectors. $\boldsymbol{S}''_j$ is obtained by swapping left and half halves of $\boldsymbol{S}'_j$ and moving $s'_{j0}$

(so-called DC component of the spectrum) to the center of $\boldsymbol{S}''_j$. $\boldsymbol{S}''_l$ is obtained from $\boldsymbol{S}'_l$ analogously.

Values of $\boldsymbol{S}''_j$ and $\boldsymbol{S}''_l$ are stored in a database metadata dictionary and they will be directly used by a selectivity estimation procedure described below. There is no need to persist $\boldsymbol{F}'_j$ and $\boldsymbol{F}'_l$.

## 2.2 Selectivity Estimation of a Range Condition Based on Difference of Attributes

The proposed method will allow to calculate selectivity value for such range query condition:

$$\mathrm{sel}(a < z = x_j - x_l < b). \tag{13}$$

For $a = min_{all} - max_{all}$ and $b = 0$ the formula (13) is equivalent to $\mathrm{sel}(-\infty < z < 0)$.

Let us denote

$$\boldsymbol{F}_z = \Big(f_{zn}\Big)_{n=0}^{2N-1} \tag{14}$$

as a frequency vector for $z$ variable, i.e. a sequence of values of an equi-width histogram which describes $z$ distribution. $\boldsymbol{F}_z$ will be obtained by applying Inverse DFT (see eq. (6)). The equi-width histogram will have $2N$ buckets and its domain is $[min_{all} - max_{all}, max_{all} - min_{all}]$.

To estimate selectivity value we use a sum of selected elements from $\boldsymbol{F}_z$ (instead of using the definite integral in eq. (5)):

$$\mathrm{sel}(a < z < b) = \sum_{n=0}^{2N-1} f_{zn}\mathrm{I}_n(a,b), \tag{15}$$

where $\mathrm{I}_n(a,b) \in [0,1]$ is an including function which determines a degree of overlapping the $n$-th bucket $[r_n, l_n)$ of the equi-width histogram by the query range interval $[a,b]$:

$$\mathrm{I}_n(a,b) = \begin{cases} 0 & \text{for } r_n \le a \vee l_n \ge b \\ 1 & \text{for } l_n \ge a \wedge r_n \le b \\ \dfrac{b-a}{r_n - l_n} & \text{for } l_n < a \wedge r_n > b \\ \dfrac{r_n - a}{r_n - l_n} & \text{for } l_n < a \wedge a < r_n < b \\ \dfrac{b - l_n}{r_n - l_n} & \text{for } a < l_n < b \wedge r_n > b \end{cases} \tag{16}$$

where $l_n = (min_{all} - max_{all}) + n\dfrac{max_{all} - min_{all}}{N}$ and $r_n = l_n + \dfrac{max_{all} - min_{all}}{N}$ for $n = 0 \ldots 2N-1$.

In this section we introduce the algorithm of selectivity estimation for range condition for a difference between $x_j$ and $x_l$. The proposed Algorithm 1 uses known values of $\boldsymbol{S}''_j$ , $\boldsymbol{S}''_l$, $min_{all}$, $max_{all}$, and $N$ obtained earlier during the update statistics.

**Algorithm 1.** The algorithm for calculating selectivity of a range condition based on a difference between attributes

```
1: $\boldsymbol{S}_z \leftarrow \boldsymbol{S}''_l$
2: $\boldsymbol{S}_z \leftarrow \overline{\boldsymbol{S}_z}$                          ▷ complex conjugate (eq. (4))
3: $\boldsymbol{S}_z \leftarrow \boldsymbol{S}_z .* \boldsymbol{S}''_j$          ▷ complex element wise multiplication (eq. (4))
4: $\boldsymbol{S}_z \leftarrow \text{ifttshift}(\boldsymbol{S}_z)$     ▷ operation of swapping right and left halves of array
5: $\boldsymbol{F}_z \leftarrow \text{IDFT}(\boldsymbol{S}_z)$          ▷ applying fast algorithm of inverse DFT
6: $sel \leftarrow 0$
7: for all $f_{zn}$in $\boldsymbol{F}_z$ do
8:     $sel \leftarrow sel + f_{zn} * \mathrm{I}_n(min_{all} - max_{all}, 0)$
9: end for
10: return $sel$
```

As mentioned earlier, calculated selectivity value may be used for estimating the size of a query result, i.e. the number of tuples that satisfy a query condition as follows:

- for a non equi-join ($t_1.x_j < t_2.x_l$): $M_1 \cdot M_2 \cdot sel$, where $M_1$ and $M_2$ are numbers of rows in tables $t_1$ and $t_2$, respectively,
- for a single-table selection condition ($t_1.x_j < t_1.x_l$): $M1 \cdot sel$.

Furthermore, because the proposed method calculates selectivity for condition (13), it can also compute the following selectivity $\mathrm{sel}(x_j \approx x_l)$, where $x_j$ is similar to $x_l$, if we define the similarity operator as follows:

$$x_j \approx x_l \overset{def}{\equiv} \left| x_j - x_l \right| < \varepsilon \equiv -\varepsilon < x_j - x_l < \varepsilon, \tag{17}$$

where $\varepsilon$ is a permissible absolute error value.

## 3 Calculating Selectivity Values Directly from Spectrum (Without Inverse DFT) for Simple Range Predicates

In the previously described solution we assume that someone creates and stores statistics metadata that describe a set of selected attributes $\left(x_j\right)_{j=1}^{D}$, i.e.:

$$\left(\boldsymbol{S}''_j\right)_{j=1}^{D}, min_{all} = \min_{j=1\ldots D}(\min(x_j)), max_{all} = \max_{j=1\ldots D}(\max(x_j)). \tag{18}$$

In calculation of selectivity we may use any two attributes $x_j$ and $x_l$ ($j, l = 1, \ldots, D \wedge j \neq l$). They may come from two different tables or from the same table (in the second case, we assume truth of attribute values independence rule).

Having already prepared set of spectrums $\left(\boldsymbol{S}''_j\right)_{j=1}^{D}$ we may ask if they may be useful for fast selectivity estimation of the simplest range condition based only on a single attribute like $a < x_j < b$. This may eliminate the necessity of storing a set of additional histograms $\left(\boldsymbol{F}_j\right)_{j=1}^{D}$.

Selectivity estimation based on elements of $\boldsymbol{F}_j$ for the simple range condition may be calculated as follows:

$$\mathrm{sel}(a < x_j < b) \approx \mathrm{sel}(j, n_a, n_b) = \sum_{n=n_a}^{n_b} f_{jn}, \tag{19}$$

where:

- $n_a$ – the number of the first bucket of the equi-width histogram (with domain $[min_{all}, max_{all}]$ and $N$ buckets) which is overlapped (in at least 50%) by range interval $[a, b]$,
- $n_b$ – the number of the last bucket which is overlapped (in at least 50%) by range interval $[a, b]$,
- $n_a, n_b = 0, \ldots, N-1$, and $n_a \leq n_b$.

We do not have $\boldsymbol{F}_j$ and we want to obtain it during selectivity estimation from spectrum $\boldsymbol{S}_j$.

The problem is that we have vector $\boldsymbol{S}''_j$ instead of $\boldsymbol{S}_j$. We will show that we may obtain $\boldsymbol{S}_j$ from some elements of $\boldsymbol{S}''_j$.

Of course, we may obtain $\boldsymbol{S}'_j$ by shifting elements of $\boldsymbol{S}''_j$.

Using DFT definition (10) we may find elements of $\boldsymbol{S}'_j$:

$$s'_{jk} = \sum_{n=0}^{2N-1} f'_{jn} e^{-i\frac{2\pi kn}{2N}} \quad \text{for } k = 0, \ldots, 2N-1, \tag{20}$$

and

$$s'_{jk} = \sum_{n=0}^{N-1} f'_{jn} e^{-i\frac{2\pi kn}{2N}} + \sum_{n=N}^{2N-1} f'_{jn} e^{-i\frac{2\pi kn}{2N}}. \tag{21}$$

Using (11) we obtain:

$$s'_{jk} = \sum_{n=0}^{N-1} f_{jn} e^{-i\frac{\pi kn}{N}} + \sum_{n=N}^{2N-1} 0 e^{-i\frac{\pi kn}{N}}. \tag{22}$$

Using only $k = 2r$ for $r = 0, \ldots, N-1$ we obtain:

$$s'_{j2r} = \sum_{n=0}^{N-1} f_{jn} e^{-i\frac{\pi 2rn}{N}} \quad \wedge \quad \sum_{n=0}^{N-1} f_{jn} e^{-i\frac{2\pi rn}{N}} = s_{jr} \tag{23}$$

Thus using (23) for obtaining $\boldsymbol{S}_j$ we use every second element from $\boldsymbol{S}'_j$.

It is obvious that the simplest way to obtain $\boldsymbol{F}_j$ is to apply IDFT for already obtained $\boldsymbol{S}_j$. But we propose calculating the selectivity without using IDFT (i.e. without explicit calculating elements $f_{jn}$).

Using (11) and (19) we obtain:

$$\mathrm{sel}(j, n_a, n_b) = \sum_{n=n_a}^{n_b} f_{jn} = \frac{1}{N} \sum_{n=n_a}^{n_b} \sum_{k=0}^{N-1} s_{jk} e^{i\frac{2\pi kn}{N}} = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{n=n_a}^{n_b} s_{jk} e^{i\frac{2\pi kn}{N}}, \tag{24}$$

$$\begin{aligned}\mathrm{sel}(j,n_a,n_b) &= \frac{1}{N}\sum_{k=0}^{N-1}\left(s_{jk}\sum_{n=n_a}^{n_b} e^{i\frac{2\pi kn}{N}}\right) = \\ &= \frac{1}{N}\left((n_b-n_a+1)s_{j0} + \sum_{k=1}^{N-1}\left(s_{jk}\sum_{n=n_a}^{n_b} e^{i\frac{2\pi kn}{N}}\right)\right).\end{aligned} \tag{25}$$

Using $z_k \overset{def}{=} e^{i\frac{2\pi k}{N}}$ we may obtain:

$$\mathrm{sel}(j,n_a,n_b) = \frac{1}{N}\left((n_b-n_a+1)s_{j0} + \sum_{k=1}^{N-1}\left(s_{jk}\sum_{n=n_a}^{n_b} e^{z_k n}\right)\right). \tag{26}$$

We can define and calculate the difference between two geometric series with ratio $z_k$:

$$\mathrm{w}(k,n_a,n_b) = \sum_{n=n_a}^{n_b} e^{z_k n} = \sum_{n=1}^{n_b} e^{z_k n} - \sum_{n=1}^{n_a} e^{z_k n} = \frac{e^{z_k n_a} - e^{z_k(n_b+1)}}{1-e^{z_k}} \tag{27}$$

for $k = 1,\ldots,N-1$.
Let us define $\mathrm{w}(k,n_a,n_b)$ also for $k=0$ as follows:

$$\mathrm{w}(0,n_a,n_b) = n_b - n_a + 1. \tag{28}$$

Finally, using (26), (27) and (28) we obtain:

$$\mathrm{sel}(j,n_a,n_b) = \frac{1}{N}\sum_{k=0}^{N-1} s_{jk}\mathrm{w}(k,n_a,n_b). \tag{29}$$

All above allows to formulate the following algorithm (Algorithm 2):

**Algorithm 2.** The algorithm for calculating selectivity of a range condition based on a single attribute without applying IDFT

```
1: $n_a \leftarrow \ldots; n_b \leftarrow \ldots$        ▷ values obtained using: $a$, $b$, $min_{all}$, $max_{all}$, $N$
2: $\boldsymbol{S}'_j \leftarrow \mathrm{fftshift}(\boldsymbol{S}''_l)$        ▷ shifting halves of an array
3: $\boldsymbol{S}_j \leftarrow \boldsymbol{S}'_j \downarrow 2$        ▷ downsampling - keeping the even indexed elements
4: $sel \leftarrow 0$
5: for $k = 0 \to N-1$ do        ▷ for every spectrum coefficient
6:     $sel \leftarrow sel + s_{j_k} * \mathrm{w}(k,n_a,n_b)$
7: end for
8: $sel \leftarrow sel/N$
9: return $sel$
```

In a real implementation of the Algorithm 2 there is no need to copy vector $\boldsymbol{S}''_l$ to $\boldsymbol{S}'_j$ (line 2) and then to create vector $\boldsymbol{S}_j$ form $\boldsymbol{S}'_j$ (line 3), because elements $s_{jk}$ (later used in line 6) may be obtained directly from $\boldsymbol{S}''_j$:

$$s_{jk} = s'_{j2k} = \begin{cases} s''_{j(2k+N)} & \text{for } 2k < N \\ s''_{j(2k-N)} & \text{for } 2k \geq N \end{cases} \quad \text{for } k = 0,\ldots,N-1. \tag{30}$$

## 4 GPU-Accelerated Selectivity Estimation

Algorithms described in sections 2 and 3 were implemented in C language and designed in a way that leverage the parallel computing model, which Compute Unified Device Architecture (CUDA) technology delivers.

DFT and IDFT were performed with cuFFT library [15], which is provided off-the-shelf with CUDA toolkit. It offers simple API to compute discrete Fourier transforms with FFT algorithm on NVIDIA GPU devices. Its design is similar to the FFTW library [8], which is widely used in CPU-only applications. FFTW was used in single-threaded CPU benchmarking application, which was written to compare the speed-up achieved with GPU. In both cases we used 1D vectors of single precision floating point numbers to obtain better performance and reasonably good accuracy.

Each component of the sum i.e. $f_{zn} * \mathrm{I}_n(min_{all} - max_{all}, 0)$ (in line 8 of the Algorithm 1) or $s_{jk} * \mathrm{w}(k, n_a, n_b)$ (in line 6 of the Algorithm 2) is computed by a separate GPU-thread.

In the Algorithm 1 (line 4) we use FFT-Shift operation to center the DC component of the spectrum. However, neither of mentioned earlier libraries provide such function out of the box. In our implementations we use the solution for 1D vectors from library described in [2].

In proposed algorithms the resulting selectivity value is calculated as a sum of some components (lines 7-9 in the Algorithm 1 and 5-7 in the Algorithm 2). The sum is computed partially on the GPU and CPU. To achieve better performance, on the GPU side we used reduction mechanism [9]. This imposes that the kernel function which performs these calculations is invoked with thread block of size $2 \times WARP\_SIZE$ (64 in case of used GPU cards).

The module which implements the Algorithm 1 (for calculation a selectivity of a range condition based on difference between the attributes $a < x_j - x_l < b$) invokes:

- the first GPU-kernel function for preparing $S_z$ (lines 1-4),
- *cufftExec* to obtain $F_z$ from $S_z$ (usage of cuFFT library) (line 5),
- the second GPU-kernel function for calculating a share of each bucket $f_{zn} * \mathrm{I}_n(a, b)$ (line 8),
- the third GPU-kernel function for summing shares within a block of threads (here we use warp-synchronous mode),
- the CPU-based part of the implementation designated for gathering partial results from GPU and for summing them by CPU.

In the most efficient implemenation of Algorithm 2 (for calculation a selectivity of a range condition based on one attribute $a < x_j < b$) we do not use cuFFT library at all. The implementation is based on calls of two GPU-kernel functions. The detailed sequence of calls is following:

- the CPU-based part of the algorithm to obtain $n_a, n_b$ values, that determinate the number of histogram buckets overlapped by interval $[a, b]$ (line 1),

- the first GPU-kernel function for calculating w(...) value and multiply it be the proper spectrum coefficient $s_{jk}$ (eq. (30) ) (line 6),
- the second GPU-kernel function for summing the previously obtained shares i.e. $s_{jk} * \mathrm{w}(k, n_a, n_b)$ within a block of threads (run in warp-synchronous mode),
- the CPU-based part of the implementation designated for gathering partial results from GPU and for summing them by CPU, and finally dividing the result by $N$.

## 5 Experimental Results

Experiments were conducted on two NVIDIA GPU devices: Quadro FX 580 (compute capability 1.0) and GeForce GTX 560 Ti (compute capability 2.1) and CPU Intel Xeon W3550 @ 3.07 GHz. In our implementations we used CUDA toolkit version 5.5 and FFTW library version 3.3.4. Additionally, FFTW was compiled with SSE instruction set [12] enabled to leverage Intel Xeon compute capabilities.

In reported results we do not take into account the time needed to compute spectrum vectors $\left(\boldsymbol{S}''_j\right)_{j=1}^{D}$. This can be done once during the update statistics phase, and obtained values can be transferred to the GPU memory and then directly used during selectivity calculations.

The experimental results for a non equi-join condition i.e. computation times of selectivity estimation for $N = 512, 1024, \ldots, 16384$ (0.5K, 1K,...,16K) using GPU kernels (for FX 580 and GTX 560Ti) and cuFFT library and CPU modules invoking FTTW library (with SSE enabled or disabled) are shown in Fig. 1a. Only for $N$ about 512 CPU-based implementations are faster than GPU-ones. For greater values of $N$ they become slower and for $N$ greater than 4K they become unacceptable with subject to the assumed absolute estimation time constraint ($T_{estim} < T_{max} = 1ms$). The GPU-based implementation satisfies the constraint for all $N$ = 0.5K,...,16K and for high $N$ it is significantly faster (see the ratios in Fig. 1b – GPUs-based implementation speedup relative to CPU-SSE-based one). Among others, justification for this is that cuFFT library outperforms FFTW when $N$ grows [1].

We also conducted experiments for selectivity estimation (based on existing DFT spectrums) of a simple range single-attribute-based condition. Fig. 1c shows calculation times of the selectivity estimation for:

- GPU-based implementation of the Algorithm 2 (using spectrum coefficients directly, without inverse DFT) launched on GPU FX 580,
- GPU-based implementation utilizing cuFFT (performing inverse DFT),
- CPU implementation of the Algorithm 2 utilizing FFTW (with SSE enabled).

Only GPU-based implementations are acceptable for all $N$=0.5K,...,16K with subject to $T_{estim} < T_{max}$. Furthermore, the GPU-based implementation of the

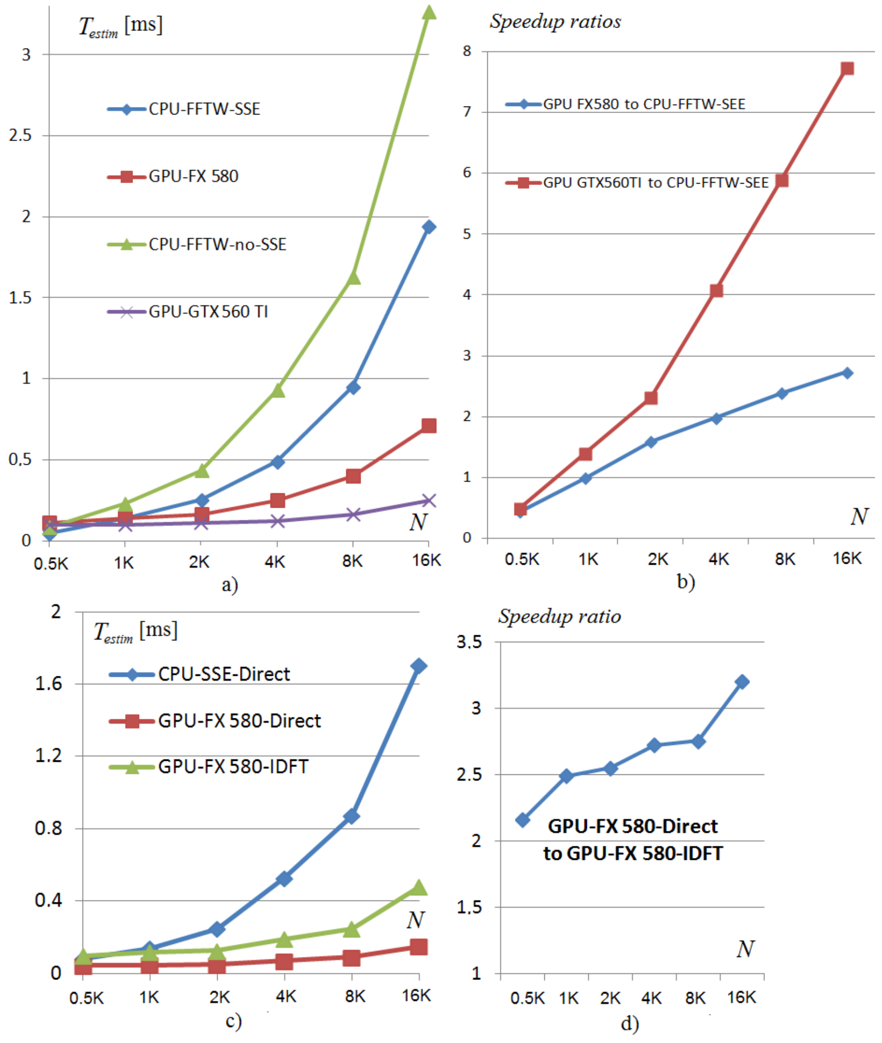


**Fig. 1.** Experimental results for selectivity estimations for $a < x_j - x_l < b$ condition (a – times of estimation, b – speedup ratios) and for $a < x_j < b$ condition (c – times of estimation, d – speedup ratios)

Algorithm 2 which avoids IDFT is always a few times faster than those one which uses it (see the speedup ratio *GPU-FX-580-Direct to GPU-FX-580-IDFT* in Fig. 1d).

## 6 Conclusions

In this paper we propose the method of selectivity estimation for non equi-join conditions, which is based on Discrete Fourier Transform and convolution theorem.

We describe its high performance implementation leveraging parallel processing capabilities of CUDA-enabled GPUs. We show that this approach outperforms the CPU-based solution (speed-up up to 8X for $N$=16K) and assures meeting the assumed time constraint i.e. less then 1ms for selectivity calculation.

This paper describes also the method of query selectivity estimation for range conditions based on a single attribute. In the proposed algorithm selectivity is calculated from prepared DFT spectrums, but without the DFT inverse operation.

We present experimental results for GPU-based implementation of this algorithm compared to pure CPU-based solution and another GPU-based version, which performs DFT inversion. Both GPU-based implementations reveal the benefits of parallel processing with respect to the computation time, however only the implementation of the proposed algorithm complies with the established time constraint for all considered vector sizes.

## References

1. CUFFT vs FFTW comparison (2008), `http://www.sharcnet.ca/~merz/CUDA_benchFFT`
2. Abdellah, M., Saleh, S., Eldeib, A., Shaarawi, A.: High performance multi-dimensional (2d/3d) fft-shift implementation on graphics processing units (gpus)
3. Augustyn, D.R.: The method of query selectivity estimation for selection conditions based on sum of sub-independent attributes. In: Gruca, A., Czachórski, T., Kozielski, S. (eds.) Man-Machine Interactions 3. AISC, vol. 242, pp. 601–609. Springer, Heidelberg (2014)
4. Augustyn, D.R., Zederowski, S.: Applying cuda technology in dct-based method of query selectivity estimation. In: Pechenizkiy, M., Wojciechowski, M. (eds.) New Trends in Databases & Inform. AISC, vol. 185, pp. 3–12. Springer, Heidelberg (2012)
5. Breß, S., Beier, F., Rauhe, H., Sattler, K.U., Schallehn, E., Saake, G.: Efficient co-processor utilization in database query processing. Inf. Syst. 38(8), 1084–1096 (2013), `http://dx.doi.org/10.1016/j.is.2013.05.004`, doi:10.1016/j.is.2013.05.004
6. Breß, S., Heimel, M., Siegmund, N., Bellatreche, L., Saake, G.: Exploring the design space of a GPU-aware database architecture. In: Catania, B., Cerquitelli, T., Chiusano, S., Guerrini, G., Kämpf, M., Kemper, A., Novikov, B., Palpanas, T., Pokorny, J., Vakali, A. (eds.) New Trends in Databases and Information Systems. AISC, vol. 241, pp. 225–234. Springer, Heidelberg (2014)

7. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. The VLDB Journal 10(2-3), 199–223 (2001), `http://dl.acm.org/citation.cfm?id=767141.767147`
8. Frigo, M., Johnson, S.G.: FFTW Library (2014), `http://www.fftw.org`
9. Harris, M.: Optimizing Parallel Reduction in CUDA (2011), `http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf`
10. He, B., Lu, M., Yang, K., Fang, R., Govindaraju, N.K., Luo, Q., Sander, P.V.: Relational query coprocessing on graphics processors. ACM Transactions on Database Systems (TODS) 34(4), 21 (2009)
11. Heimel, M., Markl, V.: A first step towards gpu-assisted query optimization. In: The Third International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures, Istanbul, Turkey, pp. 1–12. Citeseer (2012)
12. Intel Corporation: Intel®65 and IA-32 Architectures Software Developers Manual (2001)
13. Lee, J.H., Kim, D.H., Chung, C.W.: Multi-dimensional selectivity estimation using compressed histogram information. SIGMOD Rec. 28(2), 205–214 (1999), `http://doi.acm.org/10.1145/304181.304200`, doi:10.1145/304181.304200
14. NVidia Corporation: NVIDIA CUDA$^{TM}$C Programming Guide, version 6.0 (2014), `http://docs.nvidia.com/cuda/pdf/CUDA/C/ProgrammingGuide.pdf`
15. NVidia Corporation: NVIDIA cuFFT Library User's Guide (2014), `http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf`
16. Saraç, K., Egecioglu, Ö., Abbadi, A.E.: Dft techniques for size estimation of database join operations. Int. J. Found. Comput. Sci. 10(1), 81–102 (1999), `http://dblp.uni-trier.de/db/journals/ijfcs/ijfcs10.html#SaracEA99`
17. Yan, F., Hou, W.C., Jiang, Z., Luo, C., Zhu, Q.: Selectivity estimation of range queries based on data density approximation via cosine series. Data Knowl. Eng. 63(3), 855–878 (2007), `http://dx.doi.org/10.1016/j.datak.2007.05.003`, doi:10.1016/j.datak.2007.05.003
18. Zhang, J., You, S., Gruenwald, L.: Parallel selectivity estimation for optimizing multidimensional spatial join processing on gpus

# GPU-Accelerated Quantification Filters for Analytical Queries in Multidimensional Databases[*]

Peter Tim Strohm, Steffen Wittmer, Alexander Haberstroh, and Tobias Lauer

Jedox AG
Bismarckallee 7a
D-79106 Freiburg, Germany
{peter.strohm,steffen.wittmer,alexander.haberstroh,
tobias.lauer}@jedox.com

**Abstract.** In online analytical processing (OLAP), filtering elements of a given dimensional attribute according to the value of a measure attribute is an essential operation, for example in top-*k* evaluation. Such filters can involve extremely large amounts of data to be processed, in particular when the filter condition includes "quantification" such as ANY or ALL, where large slices of an OLAP cube have to be computed and inspected. Due to the sparsity of OLAP cubes, the slices serving as input to the filter are usually sparse as well, presenting a challenge for GPU approaches which need to work with a limited amount of memory for holding intermediate results. Our CUDA solution involves a hashing scheme specifically designed for frequent and parallel updates, including several optimizations exploiting architectural features of Nvidia's Fermi and Kepler GPUs.

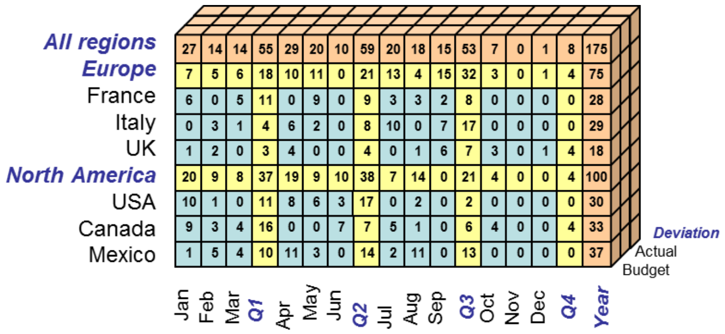
## 1 Introduction

Multidimensional databases allow users to analyze data from different perspectives by applying OLAP operations such as roll up, drill down, or slice and dice. Handling very large dimensions with hundreds of thousands or even millions of elements requires filter methods in order to show users only those data that are relevant to their analytical needs. Apart from simple filters based on characteristics of the elements themselves (e.g. element names, hierarchy levels, etc.), more advanced methods can filter elements based on conditions regarding the values of the (numeric) measures associated with the elements and stored in the OLAP cube.

An example from the sales analytics domain would be to show only those products of which at least *k* units were sold in *any* store (or, alternatively: *all* stores) during a certain time. Filters involving such an ANY or ALL requirement are called *quantification filters* and are the central topic of this paper. Other filters might require that an aggregated value such as SUM, MIN, MAX, or AVG fulfills a condition; those are called aggregation filters.

[*] Parts of the research described in this paper were presented by the authors at Nvidia's GPU Technology Conference in San Jose, CA (USA) in March 2014.

The calculation of all such filters involves the processing of very large amounts of data. Essentially, for each of the input elements to be filtered, an ($n$-1)-dimensional slice of an $n$-dimensional cube must be scanned with respect to the given condition(s). The slices may consist of aggregated values, which have to be computed first (cf. Fig. 1). While research on traditional relational approaches (ROLAP) has often concentrated on the partial pre-computation of aggregate values and the question which subset of aggregates should be pre-calculated to be stored in a given amount of available memory, in recent years a multitude of in-memory database systems have been emerging, many of them non-relational but using specialized multidimensional data structures [11, 12, 13]. Those systems store only base data (Fig. 2) and compute all calculated values on demand, i.e. at query time. This obviates the need for pre-computation but requires algorithms and computational resources for extremely fast aggregation in order to deliver results to users in (near) real time to support interactive operations such as slicing, dicing, roll-up and drill-down.



| | Jan | Feb | Mar | Q1 | Apr | May | Jun | Q2 | Jul | Aug | Sep | Q3 | Oct | Nov | Dec | Q4 | Year |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All regions | 27 | 14 | 14 | 55 | 29 | 20 | 10 | 59 | 20 | 18 | 15 | 53 | 7 | 0 | 1 | 8 | 175 |
| Europe | 7 | 5 | 6 | 18 | 10 | 11 | 0 | 21 | 13 | 4 | 15 | 32 | 3 | 0 | 1 | 4 | 75 |
| France | 6 | 0 | 5 | 11 | 0 | 9 | 0 | 9 | 3 | 3 | 2 | 8 | 0 | 0 | 0 | 0 | 28 |
| Italy | 0 | 3 | 1 | 4 | 6 | 2 | 0 | 8 | 10 | 0 | 7 | 17 | 0 | 0 | 0 | 0 | 29 |
| UK | 1 | 2 | 0 | 3 | 4 | 0 | 0 | 4 | 0 | 1 | 6 | 7 | 3 | 0 | 1 | 4 | 18 |
| North America | 20 | 9 | 8 | 37 | 19 | 9 | 10 | 38 | 7 | 14 | 0 | 21 | 4 | 0 | 0 | 4 | 100 |
| USA | 10 | 1 | 0 | 11 | 8 | 6 | 3 | 17 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 30 |
| Canada | 9 | 3 | 4 | 16 | 0 | 0 | 7 | 7 | 5 | 1 | 0 | 6 | 4 | 0 | 0 | 4 | 33 |
| Mexico | 1 | 5 | 4 | 10 | 11 | 3 | 0 | 14 | 2 | 11 | 0 | 13 | 0 | 0 | 0 | 0 | 37 |

**Fig. 1.** OLAP cube with base cells (light blue) and aggregated cells (yellow and orange)

Hence, it is essential that aggregates along the relevant dimensions of a data cube can be calculated as efficiently as possible.

General-purpose computing on graphics processing units (GPGPU) is a trend used in many computing domains with the potential for tremendous speedups through the massively data-parallel computation available on such devices. During the past decade, a growing number of database operations and applications have been examined in this respect, with relatively mixed results [4, 5, 8]. One reason for this diversity lies in the nature of database operations, which are more often bandwidth-bound rather than computation-bound. In addition, if GPUs are used as pure co-processors, the bottleneck is typically the transfer of the data to and from GPU memory, rather than the computation itself [4]. As the vast majority of research contributions try to accelerate the computation of individual relational operators [2, 4, 5, 8], this problem remains inherent. Only in few approaches – such as [7], [10] and our own system – the graphics processors form an integral part of the database system design and GPU memory is also explicitly used for data storage.

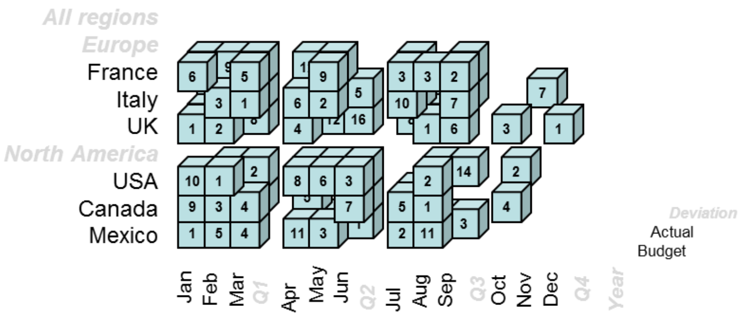


**Fig. 2.** Only base cells are stored with in-memory databases. All aggregates are calculated on demand.

The main contribution of this paper is a massively parallel approach to the computation of quantification filters, with an implementation for graphics processing units using Nvidia's CUDA. Since the actual filter is applied on a pre-aggregated area (or, sub-cube), the first part of our contribution is an efficient parallel aggregation method for large (and possible sparse) areas of aggregated cells, also implemented in CUDA. All of the described algorithms have been integrated in the commercially available Jedox OLAP Server [13].

## 2 Motivation: Computation of Quantification Filters

A quantification filter as described above can be seen as a two-stage process. In a first step the values that need to be inspected are computed and collected such that each element of the filter dimension is associated with a corresponding slice of the OLAP cube. For instance, assume the filter dimension consists of the products sold by a company. In this case, each product might be associated with the cube slice containing sales figures of this product (over a time span) for each individual store, but totaled over all other dimensions. Obtaining all the slices involves a lookup and pre-aggregation procedure resulting in an *area* or "grid" of values, whose parallel computation is described in section 3.

In the second step, explained in section 4, the filter condition is checked for each element of the filter dimension, i.e. for the values in each slice of the area computed in step 1. For instance, the filter condition might require that the total sales for a product in ALL stores (i.e. in every single store) are greater than US$ 100,000. In this case, all values in the slice have to be inspected to verify the condition. Note that each slices can be an up to ($n$-1)-dimensional sub-cube of the original $n$-dimensional OLAP cube. Hence, both of the above steps can become computationally expensive, the first due to its potentially large output, the second due to the size of its input (the output of the filter is limited by the cardinality of the filter dimension).

## 3 Parallel Computation of Aggregate Areas

It has been shown in previous work that individual OLAP aggregations as well as reasonably sized bulks of aggregations can be accelerated significantly by using GPUs [4]. However, in quantification filters a large structured area of aggregated values, potentially consisting of billions of cells, must be computed. In OLAP scenarios, such areas are typically very sparse, meaning that a majority of the values are zero because no base data exist to enter the aggregate.

If such values are calculated by a standard target-driven approach such as the one described in [4] and summarized in section 3.1, many unnecessary queries will be launched. Added up, these queries can significantly increase overall response time.

We therefore propose an alternative, source-driven aggregation approach designed for large areas of aggregated values, in section 3.2.

### 3.1 Target-Driven Aggregation

In target-driven approaches, the computation takes the targets as a starting point. A variable is assigned (i.e. memory is allocated) for each target cell to hold its output value (and any intermediate values during the computation). Then, for any source cell to be examined, the aggregation algorithm would typically check whether that source contributes to the target, and add the source value to the current aggregate value, taking into account possible weight factors defined on dimensional hierarchies. For checking source cells, a parent-to-child map of dimensional hierarchies is required, which provides, for each coordinate in a target path, the coordinate(s) of all matching source cells and corresponding weights.

Such an approach works very well for individual targets and can also nicely be parallelized with GPUs: checking source cells can be distributed among as many threads as are available, and results can be aggregated by parallel reduction, a well-known parallel building block [3]. The CUDA implementation minimizes thread divergence and maximizes coalesced memory accesses. If the data is kept in GPU memory, tremendous speedups over CPU solutions can be achieved, especially for high-level aggregations. More details can be found in [6].

The main shortcoming of this approach is that this parallelization only works with a limited amount of target cells. This is because of memory limitations, regarding both the number of registers available per thread and of the on-device shared memory required for efficient thread communication and cooperation. If a large number of aggregated cells are to be computed, they will have to be split into smaller bulks which must be computed by separate kernel calls. Hence, roughly speaking, the approach can be described as a sequence of parallel computations, and its runtime depends heavily on the number of targets to be computed (apart from the size of the input, i.e. the number of source values involved).

There are two more disadvantages when dealing with large and sparse areas of targets. First, a lot of memory has to be allocated, much of which will be unused when many of the results are zero. Given the limited memory of GPUs, this soon creates a problem. Second, even just checking for a value that will be zero is computationally expensive if it happens many times. In such a case, GPU-based calculation can actually become a disadvantage rather than an advantage, as the invocation of GPU kernels typically takes longer than invocation of a pure host method.

## 3.2 Source-Driven Aggregation

An alternative approach to computing aggregates is to drive the calculation not by targets, but by the source cells. Instead of a parent-to-child map in dimensions, we use a child-to-parent mapping, which allows a thread to look up, for a given source cell, all the target paths of the query to which it contributes (see Fig. 3), together with the corresponding weights. The thread can then "construct" the target paths and add the value of this source to all appropriate target values. Hence, unlike the target-driven method, which could be labeled a "sequence of parallel aggregations" (in CUDA, it would consist of multiple calls of one simple kernel), the source-driven one is a "parallel execution of serial aggregations" (one complex CUDA kernel).

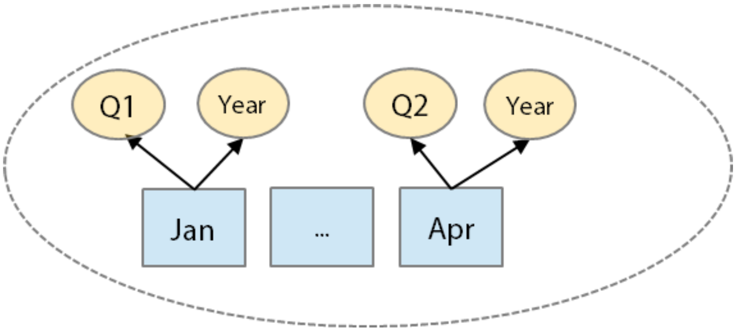


**Fig. 3.** Child-to-parent map required for source-driven aggregation

The most important advantage of such an approach is that targets with zero value are never created, i.e. no memory must be allocated for them and no computational resources are wasted for them. To achieve this, existing targets will be placed (and looked up) in a common hash table [1] located in GPU global memory. Using the terminology of [9], our approach would classify as "shared" for each GPU, but "independent" between multiple GPUs. Figure 4 gives an overview of the overall method on one GPU. The dotted line of cells labeled "target cell area" is not fully materialized – only existing target cells are created and placed in the hash table.

This approach results in a time complexity that is independent of the target area size, but depends on the actual number of non-empty targets. Moreover, since only one kernel call is require even for a large number of targets, each input will be inspected only once, which reduces actual running time.

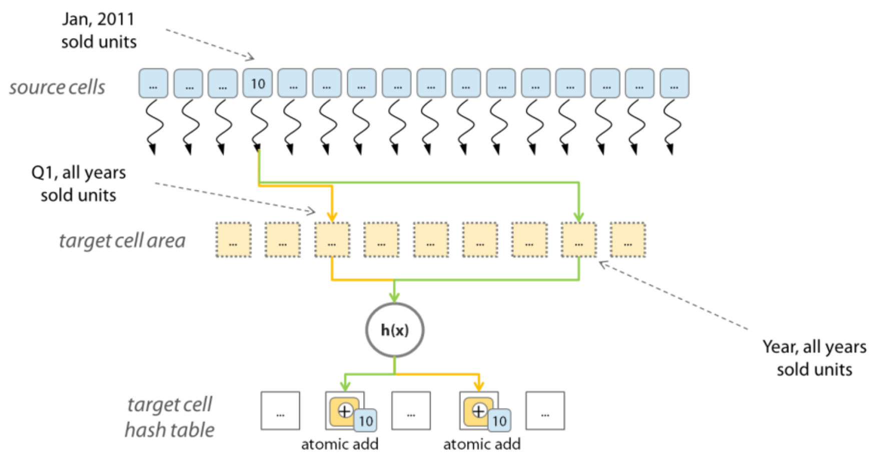


**Fig. 4.** Algorithm for source-driven aggregation. Source values are added to targets stored in a hash table in GPU RAM.

However, there are several challenges, regarding the SIMD-like parallelization that GPUs are built for: First, the number of targets for different source cells may vary dramatically; hence some threads will do significantly more work than others, and considerable thread divergence is not only likely but almost inevitable. Our solution benefits from the locality effect caused by a global sorting of source cells: adjacent source cells, which are handled by threads in the same *warp* (a group of 32 threads scheduled together on the same multiprocessor), are likely to have a similar number of targets to which they contribute.

Second, coalesced memory accesses (especially writes) are almost impossible to achieve, with target cells being scattered across a hash table. Even worse, many threads may compete to update (i.e. add their value to) the same target at a time, which makes it necessary to use atomic operations on the hash table, leading to congestion (see Fig. 5). On Nvidia GPUs with compute capability 1.3 or lower, the latter problem made the source-driven approach completely infeasible, leading to crashes and instabilities.

However, atomic operations have been improved significantly with both the Fermi and the Kepler architectures, and new features of the CUDA programming model can improve this even further. Since version 4.0, CUDA supports the *__ballot* operation, which allows threads in the same warp to compare their values of the same thread-local variable. This allows the pre-aggregation of values from all those threads which try to write to the same target location at the same time, and then let only one thread do the writing to the hash table. This warp-internal pre-aggregation (see. Fig. 6) can drastically reduce congestion, but warps from different multiprocessors may still compete for writing to the same memory address.

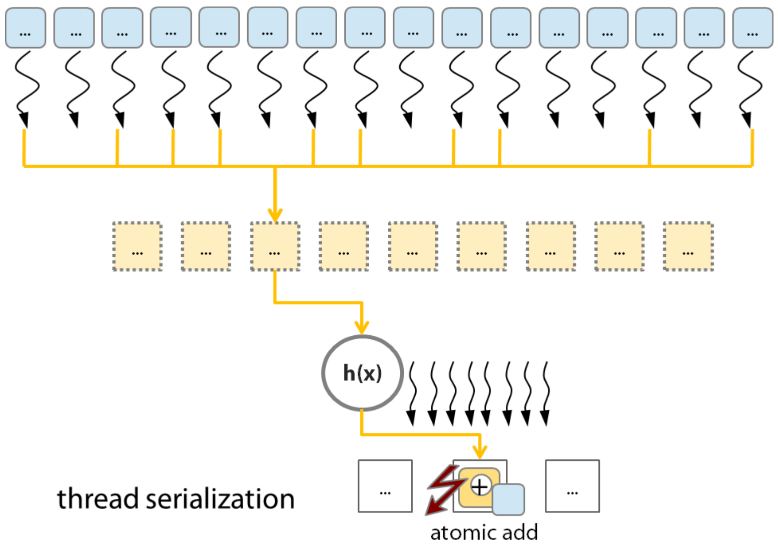


**Fig. 5.** Contention caused by thread serialization through atomic memory operations

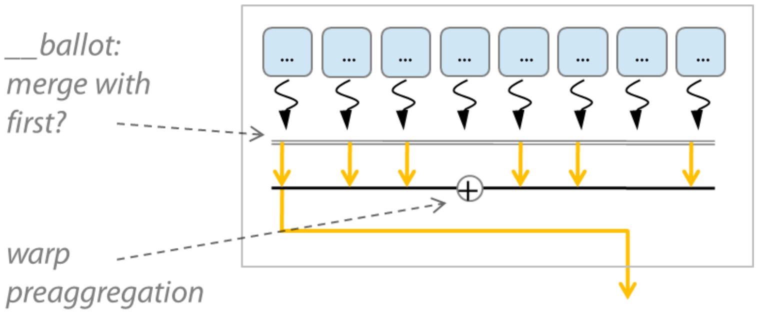


**Fig. 6.** Warp-wise pre-aggregation on Fermi or newer architectures

To further avoid contention, we can trade off some memory for better performance by allowing multiple hash functions. This is similar (though not identical) to the concept of "cloning", as described in [9]. If different warps use different hash functions, they will write their results to different locations in the hash table. Of course this will increase the size of the hash table *k*-fold, if *k* hash functions are used. Also, an extra step is required to summarize the (up to) k results for the same target before returning them. This can be combined with a sort and compact step that removes the unused slots of the hash table and sorts the results according to their target path. The best value for the number *k* of hash functions cannot be determined a priori, since the number of existing targets in unknown; our implementation chooses $k$ heuristically, taking into account the sizes of source and target areas, the number of warps used in the kernel, and the maximum allowed size of the hash table.

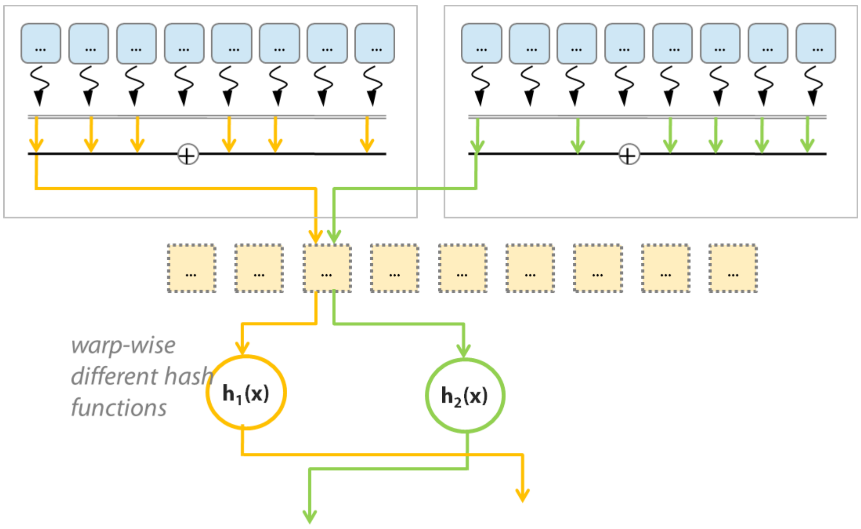


**Fig. 7.** Using different hash functions for different warps

### 3.3 Performance of GPU Aggregation Algorithms

While the target-driven approach is faster for small numbers of target cells, the source-driven method matches and outperforms it already at around 70 targets aggregated simultaneously. Fig. 8 shows a comparison of aggregation runtimes between the two approaches, on a cube containing approximately 128 million base cells. As can be seen, the time required by the source-driven algorithm is much less dependent on the target area size.

## 4 Parallel Filtering

As a result from step 1, we have a sparse aggregated area of cells, which can be imagined as a table, with each row corresponding to one element of the filter dimension and containing the slice with the respective values. Note, however, that the actual representation is simply a list of key/value pairs. Moreover, only cells with non-zero values are represented to account for sparsity. Since empty cells are still counted as 0 (zero) values for the filter condition, we have to distinguish 4 cases, depending on

(a) whether or not a 0 (zero) value satisfies the specified condition, and
(b) whether ANY or ALL cells of a slice are required to satisfy the condition.

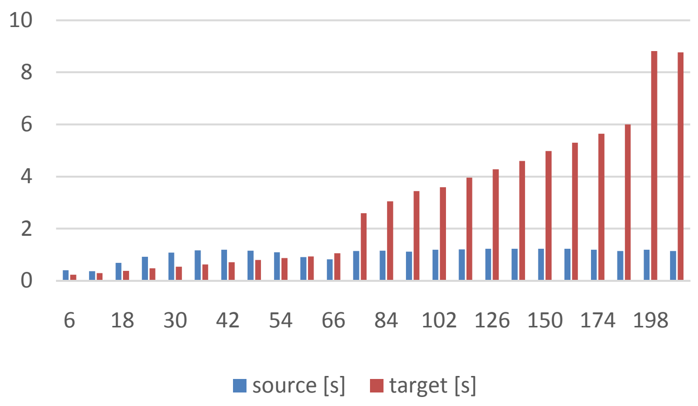
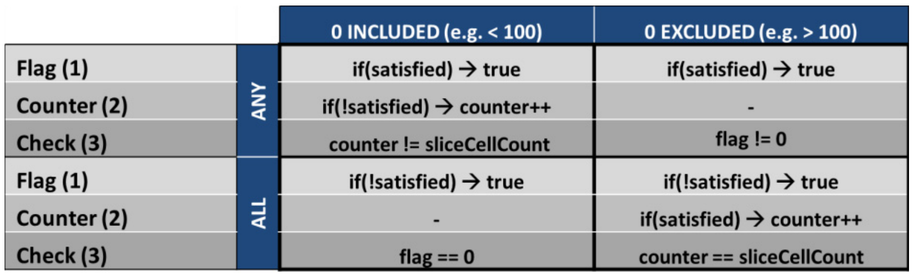


**Fig. 8.** Time (in seconds) for target-driven vs. source-driven aggregation by target area size

The four cases are depicted in Fig. 9. For example, consider the condition “ALL < 100”, which means that all cells in the slice corresponding to an element of the filter dimension must have a value less than 100 for that element to be included in the output. In this case, any empty cells (i.e. cells with value 0) fulfill the condition. On the other hand, if the condition is “ALL > 100”, the mere existence of an empty cell in the slice means that the element does not meet the condition.



| | | 0 INCLUDED (e.g. < 100) | 0 EXCLUDED (e.g. > 100) |
|---|---|---|---|
| Flag (1) | ANY | if(satisfied) → true | if(satisfied) → true |
| Counter (2) | ANY | if(!satisfied) → counter++ | - |
| Check (3) | ANY | counter != sliceCellCount | flag != 0 |
| Flag (1) | ALL | if(!satisfied) → true | if(!satisfied) → true |
| Counter (2) | ALL | - | if(satisfied) → counter++ |
| Check (3) | ALL | flag == 0 | counter == sliceCellCount |

**Fig. 9.** Four cases of the Quantification Filter: ALL | ANY vs. zero included | not included.
(1) Flag: current cell value satisfies given condition (e.g. val > 1000)
(2) counter is incremented in special cases to keep track of all flag results
(3) Condition which has to be satisfied to add an element to result set

Our CUDA approach implements the computation of the second step filters by a parallel scan of all cells in all slices, using tens of thousands of concurrent threads. We use a variant of the parallel hashing scheme described in step 1 to hold the results. In addition, we employ a mechanism for minimizing unnecessary checks as much as possible, and a space-efficient method for checking multiple conditions (condition tree). The following paragraphs describe the procedure in more detail.

First, we need to initialize a GPU hash table with enough slots capable to hold all of the elements in the filter dimension (in the extreme case, all elements meet the filter criteria). Each slot in the hash table has space for an element ID, a value, a (binary) flag, a counter, and a lock. The lock is needed because all threads scanning cells of the same slice will try to write to the same hash table position; hence a synchronization mechanism is required. In order to save memory, our implementation uses the same variable for the flag and the lock.

The scan of the cells is implemented as a CUDA kernel, where each thread examines one cell at a time as follows:

(a) Compute the hash table position for the corresponding element, check existing flag and – if necessary – acquire the lock for that position.
(b) In the ANY (ALL) case, set the flag to TRUE (FALSE) if the value satisfies the condition and to FALSE (TRUE) if it does not (see Fig. 9).
(c) In the ANY (ALL) case, increment the counter if the condition is not satisfied (is satisfied) AND a zero value would satisfy (would not satisfy) the condition (see Fig. 9).
(d) Release the lock of the hash table position.

After the kernel has finished, the flags (1) and counters (2) contain all the information required for deciding whether or not the corresponding element has to be included in the result. The bottom line (3) in each quadrant of Fig. 9 lists the condition to be verified for the decision.

However, the result might still be incomplete. This is because there might be elements for which no hash entry was created because no cells exist in the corresponding slice. If zero values also satisfy the filter condition (left quadrants of Fig. 9), these elements must be added to the result.

Fig. 10 summarizes the overall procedure. Starting with the preprocessed cells from step 1, the kernel checks each cell and fills the hash table. If necessary, elements for zero-values are added, yielding the list of result cells after some postprocessing.

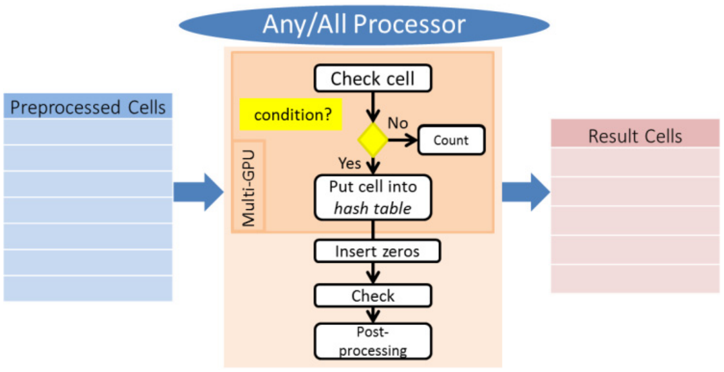


**Fig. 10.** Workflow of quantification process for ALL | ANY. Preprocessed cells are the result from step 1 (aggregation).

In order to minimize unnecessary checks, some optimization can be done. In particular, if the result is already clear, the check of a cell can be terminated without accessing the hash table, thus avoiding expensive locks and memory accesses. By checking the flag/lock at the hash table position of the processed cell we can discard those cells contributing to an element of the filter dimension which is already marked as "satisfied | not satisfied". The flow diagram of the optimized algorithm is depicted in Fig. 11.

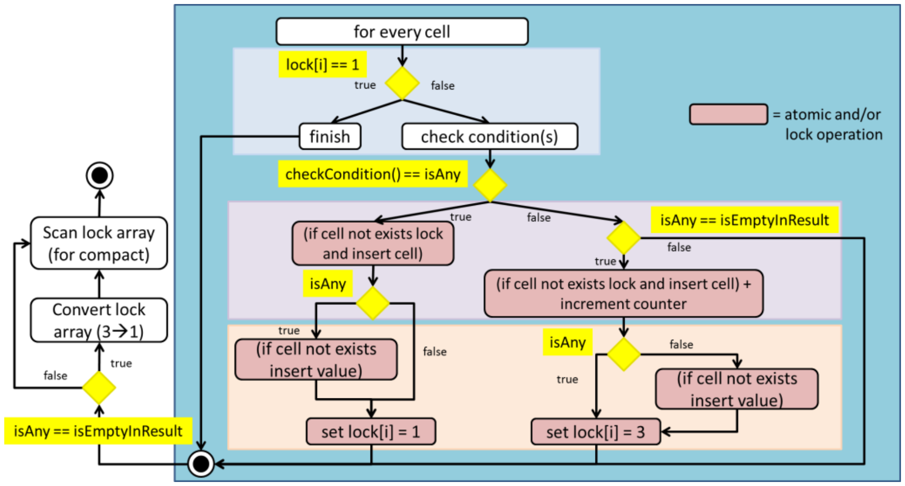

for every cell
lock[i] == 1
true
false
finish
check condition(s)
= atomic and/or lock operation
checkCondition() == isAny
true
false
isAny == isEmptyInResult
(if cell not exists lock and insert cell)
false
true
(if cell not exists lock and insert cell) + increment counter
isAny
true
false
(if cell not exists insert value)
isAny
false
true
(if cell not exists insert value)
set lock[i] = 1
set lock[i] = 3
Scan lock array (for compact)
Convert lock array (3→1)
true
false
isAny == isEmptyInResult

**Fig. 11.** Flow graph of optimized CUDA kernel minimizing locks and global memory accesses

The variable used for locking the access to a hash table entry is used at the same time as a flag to mark targets that have already been decided. If a thread discovers the flag is set for a target, it can immediately finish and continue with the next target. For targets that could still change, a different value for the lock is used. Actual locking is required only for the operations shown in pink boxes.

## 5 Performance Comparison

Our GPU algorithm can greatly speed up the computation of quantification filters. We compared the times of the standard (sequential) CPU algorithm (run on an Intel Xeon E5-2643) with our CUDA implementation on the same system with 2x Nvidia Tesla K40c GPUs, using a data model from Wikipedia.

The example data consists of the Wikipedia page access statistics of the year 2013. The main cube has over 1.8 million page names stored for every hour on every day in 2013. It also consists of different languages, projects and measures and sums up in over 300 million cells, corresponding to about 4.8 GB of compressed data in the GPU cube. We had three test cases for our test scenario:

(1) Popular Wikipedia pages, where we filtered out the top ranked pages depending on the page access count.
(2) Peak search, where we added a virtual measure “peak factor” which consists of the page access count of a given month divided by the page access count of a constant month (e.g. current month / August). The element is then a value less than 1 if the current month has a smaller page access count then the constant month, and a value greater than 1 if it has a higher page access count. By filtering on that measure, the top pages for every month can be obtained.
(3) “What’s new” combines the peak search with an additional filter such that the result will be those pages that have a low peak factor since the selected month, and a high peak factor from then on until the end of the year. So we are searching not for a peak but for a constant high after a constant low.

All examples have to calculate the input of the quantification filter first. In this step the source size as well as the target size can be rather big (the page dimension consists of over 1.8 million elements) and could not be handled with the target driven approach (cf. section 3.1). So for all these quantification filters the source driven approach of our algorithm is used.

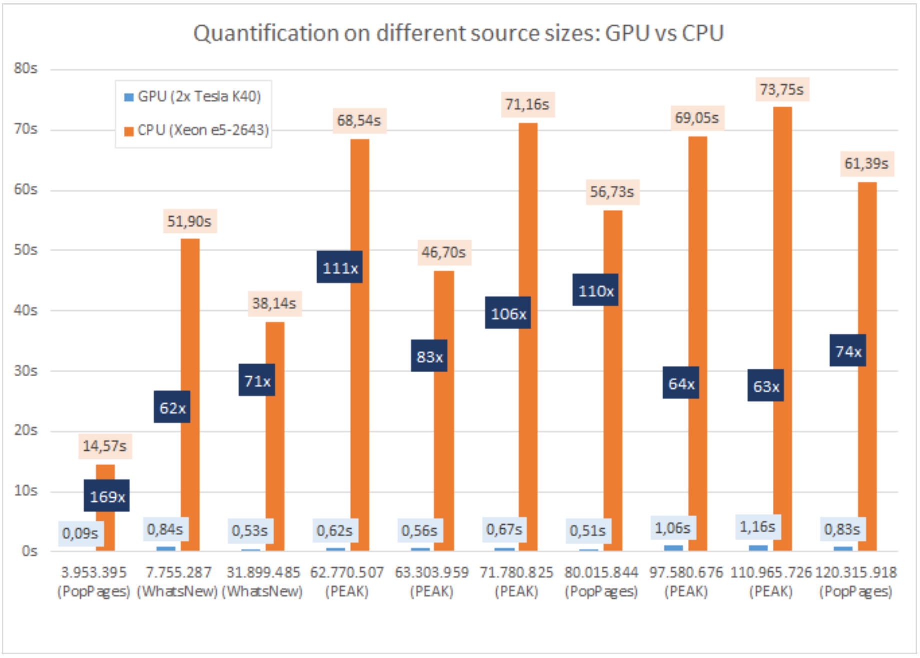


**Fig. 12.** Performance of quantification filters on GPU | CPU on Wikipedia example with different source sizes

In Fig. 12 the performance comparison of our GPU approach versus the CPU version is shown. The results are ordered by their source size from 3.9 million to 120 million (~6.25 million cells = 100 MB). As can be seen, the performance does not only depend on the input size but also on the different test scenarios. The blue boxes show the speedup factor of GPU versus CPU. With our new GPU approach it is now possible to obtain the results of data filters on big data sources in seconds instead of minutes and we gained speedups of about 60-100 times compared to the sequential CPU times.

Because the current maximum amount of GPU memory is 12 GB on a single Tesla K40, our GPU approach is limited to the amount of K40 card memory in the used system. With our test system (2x K40) we are able to store about 1 billion cells (16 GB) and then do complex filtering on the data which also requires temporary space on the GPU. High end systems can easily extend the amount of Tesla K40 cards to 4 or even 8 so that about 64 GB or 4 billion cells can be handled with our approach.

## 6 Conclusion

In this paper, we have presented a method for computing large and sparse areas of aggregated values for OLAP and other analytical database queries on GPU. In the first step we highlighted the differences of our source-driven approach to an earlier target-driven algorithm and described improvements to reduce contention related to atomic memory operations. In the second step we described our highly optimized approach of massive parallel quantification filters on big data sources and the excellent speedup compared to the CPU algorithms.

While first tests showed the usefulness of our approach in practice, the next step is to provide mechanisms and criteria which allow the decision on the best available algorithm (CPU, GPU target-driven, GPU source-driven) for a given query both effectively and efficiently, such as the method proposed in [2].

## References

1. Alcantara, D.: Efficient Hash Tables on the GPU. PhD dissertation, University of California Davis (2011)
2. Breß, S., Beier, F., Rauhe, H., Sattler, K.-U., Schallehn, E., Saake, G.: Efficient Co-Processor Utilization in Database Query Processing. Information Systems 38(8), 1084–1096 (2013)
3. Wilt, N.: The CUDA Handbook (ch. 12: ”Reduction”). Addison-Wesley (2013)
4. Govindaraju, N.K., Lloyd, B., Wang, W., Lin, M., Manocha, M.D.: Fast computation of database operations using graphics processors. In: Proceedings of SIGMOD, Paris, France, pp. 206–217. ACM, New York (2004)
5. He, B., Lu, M., Yang, K., Fang, R., Govindaraju, N.K., Luo, Q., Sander, P.V.: Relational query coprocessing on graphics processors. In: Transactions on Database Systems, vol. 34(4). ACM, New York (2009)

6. Lauer, T., Datta, A., Khadikov, Z., Anselm, C.: Exploring Graphics Processing Units as Parallel Coprocessors for Online Aggregation. In: Proceedings of DOLAP 2010, Toronto, Canada (October 2010)
7. Mostak, T.: An Overview of Map-D (Massively Parallel Database) Online whitepaper (2013), `http://www.map-d.com/docs/mapd-whitepaper.pdf`
8. Wu, H., Diamos, G., Sheard, T., Aref, M., Baxter, S., Garland, M., Yalamanchili, S.: Red Fox: An Execution Environment for Relational Query Processing on GPUs. In: International Symposium on Code Generation and Optimization (CGO) (February 2014)
9. Ye, Y., Ross, K.A., Vesdapunt, N.: Scalable Aggregation on Multicore Processors. In: Proceedings of the Seventh International Workshop on Data Management on New Hardware (DaMoN 2011), Athens, Greece. ACM (2011)
10. Ghodsnia, P.: An In-GPU-Memory Column-Oriented Database for Processing Analytical Workloads. In: VLDB 12 PhD Workshop, Istanbul, Turkey. ACM (August 2012)
11. Cognos TM1, `http://www-03.ibm.com/software/products/en/cognostm1`
12. Infor B, `http://www.infor.com/content/brochures/infor10ionbicomprehensivebi.pdf`
13. Jedox OLAP, `http://www.jedox.com/en/products/jedox-premium/jedox-olap.html`

# Part VIII
# OAIS 2014 Workshop

# Linked Open Data for Medical Institutions and Drug Availability Lists in Macedonia

Milos Jovanovik, Bojan Najdenov, Gjorgji Strezoski, and Dimitar Trajanov

Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
{firstname.lastname}@finki.ukim.mk

**Abstract.** One of the most active fields of research in the past decade has been data representation, storage and retrieval. With the vast amount of data available on the Web, this field has initiated the development of data management techniques for distributed datasets over the existing infrastructure of the Web. The Linked Data paradigm is one of them, and it aims to provide common practices for publishing and linking data on the Web with the use of Semantic Web technologies. This allows for a transformation of the Web from a web of documents, to a web of data. With this, the Web becomes a distributed network for data access, usable by software agents and machines. The interlinked nature of the distributed datasets provides new use-case scenarios for the end users, scenarios which are unavailable over isolated datasets. In this paper, we are describing the process of generating Linked Open Data from the public data of the Health Insurance Fund along with data from the Associated Pharmacies of Macedonia. With this we generate and publish an interlinked RDF dataset in a machine-readable format. We also provide examples of newly available use-case scenarios which exploit the Linked Data format of the data. These use-cases can be used by applications and services for providing relevant information to the end-users.

**Keywords:** Linked Data, Open Data, Health Care, Medical Institutions, Pharmacies, Drugs, Macedonia.

## 1 Introduction

As we explore and discover new technologies we thrive towards bettering our lives in a variety ways. One of the most active fields of research in the past decade has been the field of data management  representation, storage and retrieval. With the vast amounts of data available on the Web today, this field has initiated the development of data management techniques for distributed datasets over the existing infrastructure of the Web. The Linked Data paradigm is one of them, and it aims to provide common practices for publishing and linking data on the Web with the use of Semantic Web technologies [1]. This allows for a transformation of the Web from a web of documents, to a web of data, envisioned in the original Semantic Web idea. With this, the Web becomes a distributed network for data access, usable by software agents and machines. The interlinked nature of the distributed datasets provides new use-case scenarios for the end users, scenarios which are unavailable over isolated datasets.

The adoption of the Linked Data best practices has led to the extension of the Web into a global data space, interconnecting data from diverse domains, such as people, companies, books, scientific publications, films, music, television and radio programs, genes, proteins, drugs and clinical trials, online communities, statistical and scientific data, etc. This web of data has been embodied into the Linked Open Data (LOD) Cloud[1], a vast network of datasets published and interlinked according to the Linked Data principles, and available for access over the existing infrastructure of the Web (Fig. 1). The LOD Cloud data can be traversed with the use of the technologies of the Semantic Web, i.e. by using the SPARQL query language and the SPARQL federation functionality. This allows for data access and retrieval over distributed data sources on the Web, in a manner similar to accessing a local database.

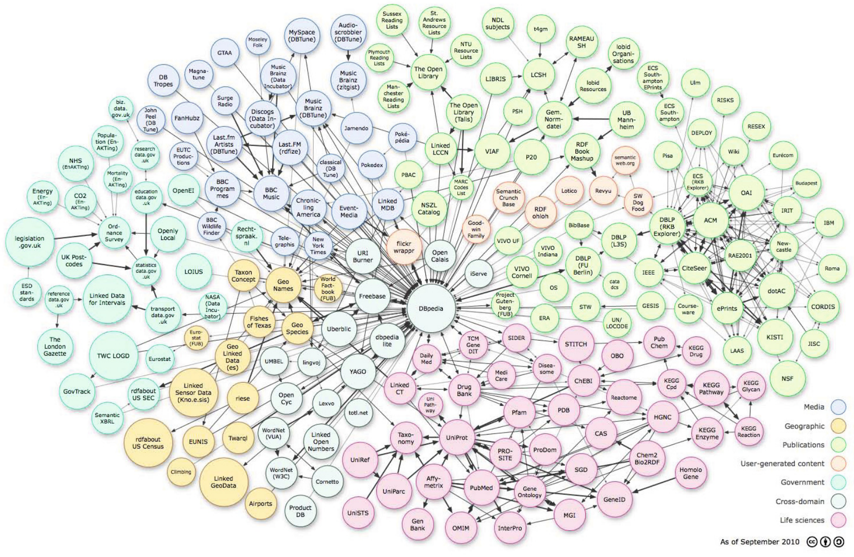


**Fig. 1.** The LOD Cloud, as of September 2011

The LOD Cloud also provides new possibilities for both domain-specific and cross-domain applications. Unlike Web 2.0 mashups which work against a fixed set of data sources, Linked Data applications operate on top of an unbounded, global data space. This enables them to deliver more complete answers as new data sources appear on the Web [1].

In this paper, we are describing the process of generating Linked Open Data from the public data of the Health Insurance Fund along with data from the

[1] `http://lod-cloud.net/`

Associated Pharmacies of Macedonia. With this we generate and publish an interlinked RDF dataset in a machine-readable format. We also provide examples of newly available use-case scenarios which exploit the Linked Data format of the data. These use-cases can be used by applications and services for providing relevant information to the end-users.

## 2 Related Work

Many organizations are now exploring the use of Semantic Web technologies in the hope of relaxing the costs of data integration. The benefits these technologies provide include integration of heterogeneous data using explicit semantics, simplified annotation and sharing, rich explicit models for data representation, aggregation and search, easier re-use of data, and the application of logic to infer additional information.

The World Wide Web Consortium (W3C) has established the Semantic Web for Health Care and Life Sciences Interest Group (HCLS IG)[2] to help organizations from the health domain in their adoption of the Semantic Web technologies.

The NeuroCommons Project[3] has a prototype knowledge base which represents a demonstration intended to show the feasibility and benefits of Semantic Web technologies in the biomedical domain, for assembling and querying biomedical knowledge from multiple sources and disciplines [2]. The prototype allows the exploration of the scalability of current Semantic Web tools and methods for creating a biomedical knowledgebase, and to reveal issues that will need to be addressed in order to further expand its scope and use. The utility of the knowledge base is demonstrated by reviewing a few example queries which provide answers to precise questions relevant to the understanding of the disease.

Other notable projects from this field include the LODD Project[4], which is focused on interlinking drug datasets already present on the Web [3][4], the DrugBank Project[5], which provides RDF data about drugs (chemical, pharmacological and pharmaceutical information) from an open, but non-RDF database[6] [5], then LinkedCT[7] [6], OBO Foundry[8], etc.

In Macedonia, our research team has already worked in applying the Linked Data principles and the technologies of the Semantic Web in the health care domain. We have transformed and published the drug data from the Health Insurance Fund of Macedonia as 5-star Linked Data[9], connected to the LODD and LOD Cloud datasets, via the DrugBank dataset [7].

---

[2] `http://www.w3.org/blog/hcls/`

[3] `http://neurocommons.org/`

[4] `http://www.w3.org/wiki/HCLSIG/LODD`

[5] `http://wifo5-03.informatik.uni-mannheim.de/drugbank/`

[6] `http://drugbank.ca/`

[7] `http://linkedct.org/`

[8] `http://obofoundry.org/`

[9] `http://5stardata.info/`

## 3 Creating Linked Open Data for Medical Institutions and Drug Availability Lists

The Health Insurance Fund of Macedonia (HIFM) is a government operated central organization that coordinates all the medical institutions in the country, from pharmacies, to specialist and hospital healthcare. HIFM has started to regularly publish public data on their website[10], making it available to the public for insight, analysis and reuse. However, this data from HIFMs website is mostly of 1-star or 2-star data quality. This means that the files available on their website are mostly PDFs, images or plain text documents, which are not machine-readable, at least not easily.

But, most of this data can easily be transformed into machine-readable formats, such as CSV or Excel spreadsheets, which have the potential of being directly used from applications, or being further transformed into 4-star and 5-star data. This means annotating them with ontology concepts in RDF format, and interlinking them with other entities from Linked Data datasets from the LOD Cloud.

The public data from the HIFM website can support many use-case scenarios in applications and services for the general public, as we have already suggested and demonstrated in [7]. This time, we chose to work with data about the duty hours of the medical institutions in Macedonia, their geographical coordinates, their medical category and the drugs available for purchase in the pharmacies from ZAM[11], one of the pharmacy associations in Macedonia.

Since the lists of available drugs for individual pharmacies are not yet publicly available, the pharmacies from the Neven-Pharm chain have agreed to give us access to their lists, in order to provide us with grounds for demonstrating the various use-case scenarios which can be supported with this type of data.

Also, the geographical coordinates of the medical institutions in Macedonia is not part of the data published by HIFM. Using the address information available from the HIFM websites, the Eco-Informatics team at our Faculty was able to geo locate each of the medical institutions, and derive its latitude and longitude values. We added them to the HIFM medical institutions dataset, and used it for the next steps of transformation and interlinking of the data.

In the following sections, we present the processes of mapping, conversion and interlinking of the data.

### 3.1 Identifying and Obtaining the Source Datasets

**Medical Institutions, their Geo Coordinates and Schedules.** The Health Insurance Fund publishes 2-star data with category and contact information about all medical institutions in Macedonia (Table 1). Also, there is a dataset with the working date and time of each medical institution (Table 2). These

[10] `http://www.fzo.org.mk/`
[11] `http://www.zam.mk/`

datasets are available as XML and as Excel spreadsheets, respectively. We transformed the structured data into an RDF graph using already existing ontologies and by defining some properties of our own, following the ontology design best practices. The process of conversion and mapping will be explained further in the paper.

**Table 1.** Properties of the Medical Institutions Geo Location Dataset

| Property | Description |
|---|---|
| Id | The id of the medical institution. |
| Code | The unique identifier in the Health Insurance Fund of Macedonia. |
| Name | The name of the medical institution. |
| Field | The field of work of the medical institution. |
| City | The city in which the medical institution is located. |
| Address | The address of the medical institution. |
| Latitude | Latitude coordinates of its location. |
| Longitude | Longitude coordinates of its location. |

In order to annotate the data and transform it into RDF data, we needed suiting ontologies. The first ontologies which come to mind when describing an organization or institution are the vCard[12] and the W3C Geospatial[13] ontologies. The W3C Geospatial ontology is the base ontology we used for mapping the geographical characteristics of the medical institutions. This ontology provides us with the possibility to describe geographical entities in great detail. The vCard ontology offers properties which describe details about telephone numbers, emails, websites, etc., for a company. Since both ontologies are largely used in datasets from the LOD Cloud, this allows further interlinking with data annotated with the same ontology concepts and properties.

**Table 2.** Properties of the Duty Schedule Dataset

| Property | Description |
|---|---|
| Id | The id of the assigned duty day for the medical institution. |
| City | The city in which the medical institution is located. |
| Name | The name of the medical institution. |
| Date | The date in question. |
| Phone | Telephone number of the medical institution. |
| Note | Extra notes (mostly work hours). |

[12] `http://www.w3.org/TR/vcard-rdf/`

[13] `http://www.w3.org/2005/Incubator/geo/XGR-geo/`

Since the entries from the two datasets are not directly linked, i.e. they dont refer to the same medical institution ID, we needed to link the data from the two datasets by matching the medical institution names. Since the data published by HIFM was raw, we used OpenRefine[14] to clean the data first. This allowed us to create links between the datasets in the process of transformation into RDF, when we used the same IDs for data referring to the same medical institution in the two separate datasets.

**Drug Availability Lists for Pharmacies.** Most of the pharmacies in Macedonia have contracts with HIFM, which means that each month they get a shipment of HIFM-regulated drugs by special prices for the people that have health insurance. The list of drugs shipped to the pharmacies depends on the order that the pharmacy makes every month. This order is made via a special interface on the HIFM website, intended for pharmacies. This gives us a hint that HIFM has information about the amount of drugs available in each pharmacy, at the beginning of each month. Unfortunately, this data is not yet publicly available. However, if a proper FOI request is submitted to HIGM, access to this data will be granted. Therefore, the pharmacies from the Never-Pharm pharmacy chain decided to share their drug availability lists with us, for the purpose of this research.

The drug availability dataset was initially in an Excel format, so we transformed it into CSV for further mapping into RDF and later into Linked Data (Table 3).

**Table 3.** Properties of the Drug Availability Dataset

| Property | Description |
|---|---|
| Id | The id of the specified drug. |
| Name | The name of the drug. |
| Type | The type of drug delivery system. |
| Dosage | Dose of the drug in each unit. |
| Manufacturer | The manufacturing company. |
| Quantity | Quantity available. |

In order to start the conversion to RDF, we needed an ontology for describing the drugs contained in the drug availability dataset. For this purpose, we chose to use the DrugBank[15] ontology, the Schema.org[16] vocabulary, and HIFM[17] ontology [7] since they completely cover the properties we need and offer us easier interlinking possibilities for further transformation. Since we knew in advance

[14] `http://openrefine.org/`

[15] `http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/`

[16] `http://schema.org/`

[17] `http://purl.org/net/hifm/ontology#`

which pharmacies (medical institutions) was the list of drugs referring to, we added the medical institution ID in the dataset, allowing us to create a link from an entity in this dataset (a drug) to the medical institution (pharmacy) which its availability is referring to.

### 3.2 Transformation from 2-Star to 5-Star Linked Open Data

**Transformation from CSV to RDF.** After obtaining the three datasets, we first needed to transform them into 4-star data. For the mapping purposes, we used a Virtuoso Universal Server[18] instance. We loaded the CSV data into a Virtuoso instance, where they are stored in a traditional relational database. We then used R2RML[19], a language for mappings from relational databases to RDF datasets, and custom mapper files in order to make the transformations. This transformation resulted in creating RDF Views over the relational database tables, which we then used to create one single RDF graph, containing the data from all three input datasets. This was made possible by having the medical ID information in all three datasets, so the entities from all three source datasets are referring to the same medical institution entity in the RDF graph.

**Transformation into Linked Data.** In order to transform our data into 5-star Linked Open Data, a link to outside entities must be made. We already have a publicly available Linked Drug Data dataset[20], with drugs from HIFM, which is interlinked with the LODD and LOD Clouds. Therefore, since we have drug information in our new RDF dataset, we decided to interlink these two datasets. For this purpose, we used matching of drugs by their name, dosage form and manufacturer, and added the ‘owl:sameAs’ relation between the drug IDs. This interconnection allows us to create use-case scenarios which will provide us with detailed information about a specific drug, information which is not available in the original HIFM source data. These detailed information can include food interactions of the drug, its chemical formula, other manufacturers, form factors and brand names, drug-drug interactions, etc.

## 4 Use-Case Scenarios

The concept of Linked Data allows the users to traverse large amounts of diverse data located on distributed locations on the Web, by starting at one single point. This allows creation of complex use-case scenarios which provide the end-users with additional information and services, previously unavailable over the isolated dataset. In this section, we will provide example use-case scenarios which have the purpose to illustrate the capabilities the Linked Data nature of the datasets can provide.

[18] `http://virtuoso.openlinksw.com/`
[19] `http://www.w3.org/TR/r2rml/`
[20] `http://purl.org/net/hifm/data#`

One possible use-case scenario would be to find a pharmacy which works a 24-hour shift on a specific date, in a specific town. This query can be used from a mobile application, for instance, to locate the nearest pharmacy which is on 24-hour duty in the current day, based on the location of the user.

Here is one example SPARQL query which retrieves the pharmacy which works overnight on the 2nd February 2014, in Bitola (prefixes omitted for brevity):

```
select ?pharmacyName ?lat ?lng ?phone ?notes
from <http://linkeddata.finki.ukim.mk/lod/data/hifmpharm#>
where {
  ?pharmacy rdf:type schema:MedicalOrganization;
          hifm:medicalFacilityName ?pharmacyName;
          geo:city ?city;
          geo:latitude ?lat;
          geo:longitude ?lng;
          hifm:dateOnDuties ?date;
          vcard:Phone ?phone;
          vcard:Notes ?notes.
  filter(contains(lcase(?city),'bitola') && str(?date)='2/22/2014')
}
```

The result of the query is given in Table 4.

**Table 4.** Results from the SPARQL query

| Property | Value |
|---|---|
| Pharmacy Name | PZU Apteka Medika Karta |
| Latitude | 41.02503967285156 |
| Longitude | 21.31836891174316 |
| Phone | 047/225-285 |
| Notes | from 23:00h to 07:00h |

Another use-case scenario made available from the three source datasets we use, is the retrieval of information about the drugs available for purchase in a specific pharmacy. This scenario can be used in drug and pharmacy applications and services, which could provide the end-users with the information whether a given drug is available in a pharmacy, or possibly locating the nearest pharmacy which has the necessary drug. Here is an example SPARQL query which retrieves a shortened list of drugs available in the pharmacy with ID= 25046508 (prefixes omitted for brevity):

```
select ?label ?dosage ?strength ?quantity
from <http://linkeddata.finki.ukim.mk/lod/data/hifmpharm#>
where {
  ?pharmacy hifm:pharmacyID '25046508';
            hifm:hasAvaliableMedicine ?drug.
  ?drug hifm:available ?quantity;
        hifm:dosageForm ?dosage;
        hifm:strength ?strength;
        rdfs:label ?label.
} limit 10
```

The results of the query are given in Table 5.

**Table 5.** Results from the SPARQL query

| Drug Name | Dosage | Strength | Quantity |
|---|---|---|---|
| FURAL | Capsules | 30X100MG | 2 |
| FURAL | Suspension | 90ML | 2 |
| GENTAMICIN | AMP. | 10X40MG | 2 |
| MENDILEX | Tablets | 2MG.X50 | 2 |
| NIFLAM | Tablets | 20X200MG | 2 |
| PROCULIN | TEARS SOL. | 10ML | 2 |
| REGLAN | Syrup | 120ML. | 2 |
| SUMETRIN | Tablets | 50MGX3 | 2 |
| TIMOLOL | Eye drops | 0.5% 5ML | 2 |
| VASOFLEX | Tablets | 30X1MG | 2 |

Since we have links in our RDF dataset to the DrugBank dataset, and it is already connected to a large number of datasets from the LODD and LOD Clouds, we can access and retrieve information from those datasets as well. One such possible scenario would be to obtain the description of a particular drug the user is interested in. Since this data is not available from the original HIFM dataset, we can traverse the LOD Cloud and obtain a short description from DrugBank or the DBpedia dataset[21]. Here is an example SPARQL query which provides both descriptions for a drug available in the pharmacy with ID=25046508 (prefixes omitted for brevity):

```
select distinct ?name str(?dbdesc) str(?dpdesc)
where {
  graph <http://linkeddata.finki.ukim.mk/lod/data/hifmpharm#> {
    ?pharmacy hifm:pharmacyID '25046508';
              hifm:hasAvaliableMedicine ?drug.
    ?drug owl:sameAs ?hifmDrug.
  }
  graph <http://linkeddata.finki.ukim.mk/lod/data/hifm#> {
```

[21] http://dbpedia.org/

```
    ?hifmDrug rdfs:seeAlso ?dbdrug ;
              drugbank:genericName ?name.
  }
  service <http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql> {
    ?dbdrug drugbank:description ?dbdesc ;
            owl:sameAs ?dpdrug.
  }
  service <http://dbpedia.org/sparql> {
    ?dpdrug dbpedia-owl:abstract ?dpdesc.
    filter langMatches( lang(?dpdesc), "EN" )
  }
}
```

The result of the query are given in Table 6.

**Table 6.** Results from the SPARQL query

| Drug Name | DrugBank Description | DBpedia Description (partial result) |
|---|---|---|
| Amoxicillin | A broad-spectrum semisynthetic antibiotic similar to ampicillin except that its resistance to gastric acid permits higher serum levels with oral administration. [PubChem] | Amoxicillin, formerly amoxycillin, and abbreviated amox, is a moderate-spectrum, bacteriolytic, -lactam antibiotic used to treat bacterial infections caused by susceptible microorganisms... |

All of these use-cases can easily be implemented in mobile or web applications, since the SPARQL endpoint[22] is available for use as a REST service:

`http://linkeddata.finki.ukim.mk/sparql?query=SPARQLQUERY&format=FORMAT`

Here, 'SPARQLQUERY' represents the SPARQL query, as the ones shown above, and 'FORMAT' represents the format of the response, such as HTML, XML, JSON, CSV, RDF/XML, N3, Turtle, JSON-LD, etc. This would allow application developers access to additional information, previously unavailable over the isolated datasets obtained from the HIFM website.

## 5 Conclusion and Future Work

The Linked Data concept provides new ways of publishing and connecting data from different distributed sources and with this it provides a new spectrum of use-case scenarios, by both businesses and independent developers, for developing innovative applications and services. The opportunities that lie with the creation

[22] `http://linkeddata.finki.ukim.mk/sparql`

of new use-case scenarios are a field whose potential is becoming increasingly recognized [8].

As we already noted, the healthcare domain is already adopting the Semantic Web technologies and the Linked Data principles, enabling important information retrieval in health fields which are faced with separated data stores [9].

In this paper, we identify the field of healthcare as an important field for end-users which would greatly benefit from the opportunities presented by the Linked Data principles. Therefore, we transformed the 2-star data available on the HIFM website and a chain of pharmacies, into 5-star Linked Open Data. This enabled us to provide example use-case scenarios which can be used when from businesses and independent developers when developing applications and services in the healthcare domain.

Since an important part of our source datasets has been obtained directly by our research team, and was not publicly available, we wanted to show through the use-cases and the paper in general what the benefits of opening up this data would be for the general public. We hope that the presented example use-cases would be a reason for initiating the process of publishing drug availability lists as open, public data.

**Acknowledgment.** The work in this paper was partially financed by the Faculty of Computer Science and Engineering, at the Ss. Cyril and Methodius University in Skopje, as part of the research project Semantic Sky 2.0: Enterprise Knowledge Management.

## References

1. Heath, T.: C. Bizer Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology 1(1), 1–136 (2011)
2. Ruttenberg, A., Rees, J.A., Samwald, M., Marshall, S.: Life Sciences on the Semantic Web: the Neurocommons and Beyond. Briefings in Bioinformatics 004 (2009)
3. Jentzsch, A., Zhao, J., Hassanzadeh, O., Cheung, K.-H., Samwald, M., Andersson, B.: Linking Open Drug Data. In: I-SEMANTICS (2009)
4. Samwald, M., Jentzsch, A., Bouton, C., Kallesøe, C.S., Willighagen, E., Hajagos, J., Marshall, S., Prud'hommeaux, E., Hassanzadeh, O., Pichler, E.: Linked Open Drug Data for Pharmaceutical Research and Development. Journal of Cheminformatics 3(1), 19 (2011)
5. Law, V., Knox, C., Djoumbou, Y., Jewison, T., Guo, A.C., Liu, Y., Maciejewski, A., Arndt, D., Wilson, M., Neveu, V.: Drugbank 4.0: Shedding New Light on Drug Metabolism. Nucleic Acids Research 42(D1), D1091–D1097 (2014)
6. Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R.J., Wang, M.: LinkedCT: A Linked Data Space for Clinical Trials. arXiv preprint arXiv:0908.0567 (2009)
7. Jovanovik, M., Najdenov, B., Trajanov, D.: Linked Open Drug Data from the Health Insurance Fund of Macedonia. In: Proceedings of the 10th Conference on Informatics and Information Technology, pp. 56–61 (2013)

8. Kundra, V.: Digital Fuel of the 21st Century: Innovation through Open Data and the Network Effect. Joan Shorenstein Center on the Press, Politics and Public Policy (2012)
9. Cheung, K.-H., Prudhommeaux, E., Wang, Y., Stephens, S.: Semantic Web for Health Care and Life Sciences: a Review of the State of the Art. Briefings in Bioinformatics 10(2), 111–113 (2009)

# Integrating Multi-viewpoints Paradigm in Ontology Using Ontology Design Patterns

Soumaya Kasri and Fouzia Benchikha

Département d'informatique, université 20 Aot 1955 Skikda, Algérie
Laboratoire LIRE , université Constantine 2, Constantine, Algérie
{kasri.soumaya,f_Benchikha}@yahoo.fr

**Abstract.** Everyone has his own perception or viewpoint of real worlds objects. In this paper, a multi-viewpoints paradigm is integrated in ontology with using the ontology design patterns. Ontology development and use constitute an important research area for semantic web. Ontology has been established for knowledge sharing and is widely used as a means for conceptually structuring domains of interest. An ontology that supports multi-views representation of its concepts is the key to achieve the objective of allowing each user to build his own view according to his perception of the world / his needs and independently of other users. This paper describes a new ontology design pattern called multi-viewpoints pattern for integrating viewpoints notion in ontology. The multi-view pattern is evaluated by an extended formal concept analysis.

**Keywords:** Multi-Viewpoints Notion, Ontology, Ontology Design Pattern, Pattern Evaluation, Formal Concept Analysis, Pattern Structures.

## 1 Introduction

In the literature, there are many definitions of ontology. The most cited is Gruber's definition "Ontology is a formal explicit specification of a shared conceptualization" [18]. Shared refers to the notion that an ontolo-gy captures consensual knowledge. Indeed, several ontologies that represent particu-lars or partials viewpoint are developed separately and coexist with the risk of inconsistency associated with them. Each ontology engineer or ontology group engineers has a different viewpoint of the world to the others. In opposition to the monolithic vision of knowledge representation where the universe of discourse is unique and all observers perceive it in the same way, the multi-viewpoints approach allows us to model the same reality with different viewpoints. Several research teams have been working for several years on the integration of the viewpoint notion in their fields [2] [4] [23]. Ontology Design patterns constitute an important advance in the ontology design by offering the quality and reuse. The patterns are used in many areas, they were first mentioned in the field of architecture by Christopher Alexander in 1977[1]. In software engineering, a design pattern has been successfully applied to the object oriented (OO) paradigm to address recurring design problems. They are formalized for the first time in

1995 in the book "Gang of Four" [14]. The ontology engineering adopts the design patterns in the entities representation according to the generic and the reusable models. We define a ontology design pattern as an appropriate solution for many conceptualizations, according to several ontology engineers, in a similar problem. A design pattern is independent of the domain and ontology language. However, the current efforts to catalog the patterns relate only to OWL language (implanted patterns in OWL) through the ontology design patterns community portal [22]. In this paper, we introduce a new pattern to describe the multi-view point concepts in an ontology. The multi-viewpoints pattern is designed to provide a model to solve a recurring problem when the objective is the creation of a multi-viewpoint ontology. The extending formal concept analysis patterns structure is used to evaluate it. The paper is organized as follows. Section 2 presents the related work. In section 3, we present how to integrate the viewpoint notion on ontology. We also present the proposed pattern, it validation with extending formal concept analysis (pattern structures) and how to use it by imitation operation. Section 4 concludes our work and presents our research perspectives.

## 2 Related Work

The integration of viewpoint notion in the ontology design consists to: (i) elaborate a shareable ontological model and accessible from several viewpoints; (ii) align the local ontologies to link the different views. The first approach is close to the multi-representation notion of the same spacial objects from different viewpoints and at different levels of resolution in the spatial systems [25]. Benslimane et al [6] define a contextual ontology language to support multiple representations of ontologies. The underlying key idea of their work is to adapt the stamping mechanism proposed in [25]. Falquet and Mottaz [13] present a building multi-viewpoints ontology where concepts are associated to many formal definitions corresponding to different view-points of concepts in question. The second approach preserves the independence of each local ontology. The integration of the viewpoint notion and multiple viewpoints consists to put semantic bridges to link the local ontologies that present different views [3]. Bouquet et al [8] propose C-OWL as an extension of OWL language for the representation of contextual ontologies. In C-OWL, knowledge is contained in a set of contexts, called space. Each context of this space is an OWL ontology. The alignments are expressed with "bridges". In ontology engineering, ontology design patterns (ODPs) constitute one of the most promising means for the design and creation of ontologies. The ODPs are classi-fied into six families (Structural, Correspondence, Content, Reasoning, Presentation, Lexico-Syntactic [16]. Several works have been aimed to improve ontology design patterns aspects. Whitehead et al use the ontology design pattern to describe transport phenomena [25]. The core Semantic Transport ODP is deliberately simplified to essential elements, to be applicable to a wide range of use cases in the physical sciences. In [20], the authors present a geo-ontology design pattern for semantic trajectories and demonstrated its applicability. vb [26] proposes to use patterns to obtain better automatic matching algorithms. In this paper, we propose a content pattern for inte-gration the viewpoint notion in ontology. Such pattern, called multi-viewpoint pattern. As ontologies domain, Formal Concept Analysis (FCA) aim at modeling concepts. It regroups a set of objects sharing the same set of attributes in the same concept based in Galois theory. Haav [19] was the first that introduce to use the FCA com-bined with a rule-based language to extract and design an ontology. Cimiano et al [10] in their article present an approach to the automatic acquisition of taxonomies or concept hierarchies from a text corpus. It is based on FCA. Bendouad et al [5] used FCA and Relational Concept Analysis (RCA) to construct an ontology in PACTOLE system. Batrice et al [7] propose to use FCA and RCA for extracting and completing an ODP from a UML schema. We are however the first to propose to use the pattern structures, an extension of FCA for ontology design patterns evaluation. The work that we present in this article is to the first of our knowledge attempt to evaluate ontology design patterns.

## 3 Integrating Viewpoints Paradigm in Ontology Using ODPs

To represent the multi-viewpoints concept, it can create a new language or extend a language as C-OWL but this increases the cost of ontology development, the time spent to learn and use these languages and the effort spent to adapt or create new editors, reasoners ... etc. for this, it is highly desirable to save the same language, formalism, tools to manipulate (express, operate, reason) the multi-viewpoints concept.. For this reason, we introduce a new pattern from which each user will be able to interpret his own view on a given domain.

**Multi-Viewpoints Ontology Design Pattern.** We describe our pattern by name, problem solving, proposed solution, consequences of its use.

1. **Name:** the multi-viewpoints pattern
2. **Problem:** To represent the multi-viewpoints concepts
3. **Solution:** We informally define the key concepts of multi-viewpoints pattern(shown in figure 1). The main features of the pattern are:
   - Multi-viewpoints concept has two relationships to encapsulate the unique identi-fication (global concept) of an individual and attach one or more views( local concepts).
   - Global concept: Also called global view, a global concept contains the shared knowledge among different views. This concept refers to local concepts.
   - Local concept: Also called local view, local concepts represent the different views of the same object.
   - AbstractView concept: The root of all local hierarchy, it is the abstract concept and the father of all local concepts. This concept is used to hide views. A user sees only the views which interest.

- SpaceName (Multi-viewpoints:: View) : Adding a level that represents the roots of the local hierarchies. This level is useful to regroup the same views in the same hierarchy. The main advantage of a namespace is to allow the thorough research from the root.
- has_globalView relationship: Relationship between the multi-viewpoints con-cept and the global concept that encapsulates its identification.
- has_localViews relationship: Relationship between the multi-viewpoints con-cept and its local view concepts. It can provide constraints on the type of in-stances that participate in the relationship
- is-A relaionship: The different concepts are organized by a subsumption rela-tionship according to order of generalization or specialization.
- with_Constraints axiom: This axiom has several rules that act on the consistency of a multi-viewpoints concept during its creation from the views linked by the bridges presented below.
- with_Bridge axiom: Axiom is an interaction that expresses a relationship be-tween view concepts. A bridge symbolizes both the separation between two concepts of the same view or distinct views and the connection between them. In our pattern, we use the bridges in the same view hierarchy "intra-view" and between views "inter-views".
  The bridges are represented by rules that affirm the direction, the possibility or impossibility of passing from a view concept to another. In what follows, we present the bridges that we have built into our pattern. We put:
- – MVP a multi-viewpoints concept that can be linked to multiple views $Vj \in \{V1, V2, V3..\}$, $j = 1...n$;
- – S,K,M are the view concepts and i is an individual of one of these concepts.
- – We adopt the following notation to present the namespace in which we can create the multi-viewpoints concepts: (MVP ::Vj ::view concept)
- Total inclusion bridge: It is a full inclusion between two concepts and it translates that all individuals of the first concept are also individuals of the second concept. This bridge can be described by the rule: $\forall i \in MPV :: Vj :: S \Rightarrow i \in MPV :: Vj :: K$
- Partial inclusion bridge: It translates that few individuals of the first concept are individuals of the second concept. This bridge can be described by the rule:$\exists i \in MPV :: Vj :: S \Rightarrow i \in MPV :: Vj :: K$
- Exclusion bridge: It reflects that an individual can belong to either the first concept or the second concept but not both. This bridge is described by the rule: $\forall i \in MPV :: Vj :: S \Rightarrow i \notin MPV :: Vj :: K$

4. **Consequences**

- The pattern allows to create a multi-viewpoints instance with the same pattern structure. This coexistence of multiple views at the same time allows to express constraints inter-views.
- The presence of AbstractView concept as root of local hierarchies gives a scalability to pattern. It can dynamically add or delete a local concept or a full view without changing the pattern structure.

- The constraints on has_localViews relationship allow a more accurate representation and prohibit inconsistent cohabitation of views.

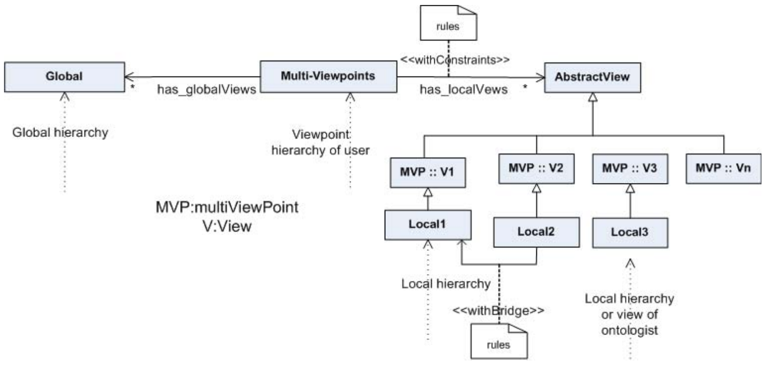


**Fig. 1.** Multi-viewpoints pattern with UML representation [9]

**Pattern Reuse.** To apply the pattern in a given context, imitation is the simplest operation to use a pattern in the oriented object approach [11]. Imitation is to duplicate the solution and adapt it to a given context, is defined as follows [24]:

Imitate = to adapt the proposed solution in pattern

Adapt = (rename / add / remove property)*

For this, we illustrate our pattern using the imitation on a small example concerning rental / housing system. In this example, we adapt our pattern to support a five views for the housing concept: "Size view" contains properties that relate, for example, the number of housing parts, "Localization view" takes into account the housing envi-ronment, "Type view" that decomposes the housing according to type (house or apartment), "occupation duration view" housing for temporary duration or permanent and "Financial view " deals the rental costs of housing. In addition, the housing concept has a global view which could regroup all the common properties between the different views (see figure 2). The axioms can be expressed with the bridges, for example, if we know that all big housing is expensive. We can express this inter-views knowledge(Size and Financial) for housing concept as a total inclusion bridge from Big_Housing concept to Expensive_Housing concept as follows : $\forall i \in$ Housing::Size::Big_Housing $\Rightarrow i \in$ Housing::Financial::Housing_Expensive. We also know that a studio will never be a home. This intra-view knowledge (type) can be expressed for housing concept as exclusion bridge as follows: $\forall i \in$ Housing::Size::Studio $\Rightarrow i \notin$ Housing::Size::Home.

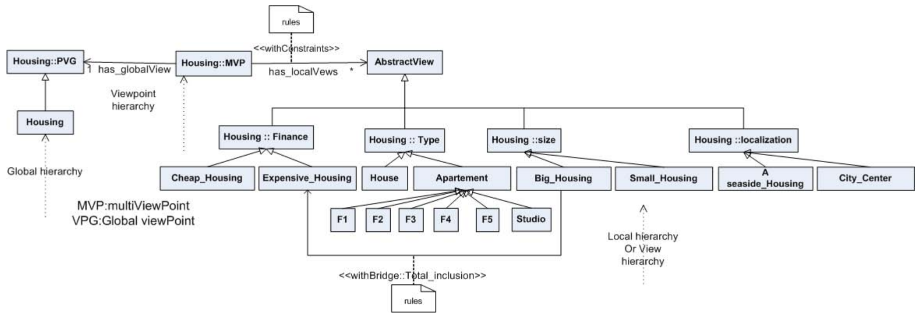


**Fig. 2.** Imitation of multi-viewpoints pattern in rental / sale of housing system

**Evaluation with Pattern Structures.** There are not yet rigorous methodologies to evaluate the Ontology Design Patterns (ODPs) but only certain criteria, such as reuse, to evaluate the success of ODPs [15]. In this evaluation, We use an extension of the formal concepts analysis FCA (pattern structures) to complete the pattern with a formal specification. This formal specification plays a great role to check if the implementation of pattern solves the design problem while respecting its properties. The evaluation result, on a given example, is a lattice which presents a viewpoint of current user. The classic FCA is valid to build a concept lattice when objects are described by bi-nary attributes but not for complex objects such as viewpoint individuals. They are described by a set of view individuals (local and global). For building a concept lattice of viewpoint individuals (user viewpoint), it is necessary to define a partial order of their descriptions. This is the main idea of the pattern structures defined by Ganter & Kuznetsov [17]. Such a structure is formalized by a triple $(G, (D, \sqcap), \delta)$ where G is a finite set of objects. $(D, \sqcap)$ is a meet-semi-lattice of objects description called patterns with infimum (meet) operator $\sqcap$. This operator is idempotent, associative and commutative. $\delta$ is a function which associates to any object g∈G its description $(g)$ in D. The pattern of D are ordered by subsumption relation $\sqsubseteq$ where $c \sqsubseteq d \Leftrightarrow c \sqcap d = c, \forall c, d \in D$. The following operators $(.)^{\Box}$ form a Galois connection between the power set of G and $(D, \sqcap)$ with $A \sqsubseteq G$ et $d \in (D, \delta)$:

$$A^{\Box} = \sqcap_{g \in G} \delta(g); \tag{1}$$

$$d^{\Box} = \{g \in G | d \sqsubseteq \delta(g)\}; \tag{2}$$

Concepts obtained are the pairs (A,d) with $A \in G$ and $d \in (D, \sqcap)$, such that $A^{\Box} = d$ and $A = d^{\Box}$ are ordered by $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \sqsubseteq A_2 (\Leftrightarrow d_2 \sqsubseteq d_1)$ to form a lattice of concepts. We detail in the following the elements needed to build a lattice of viewpoint concepts by using the pattern structures and exploiting the multi-viewpoints pattern.

- **Textual representation of multi-viewpoints pattern**
  The essential concepts represented to make the pattern usable by the pattern structures are: View concept, viewpoint concept, view individual, viewpoint individual.
  - View concept (global/local): We represent by a vector of property and domain pairs. We take the example previously mentioned:
    Global_Housing=<[number:String],[address:String],[floor:Integer],[surface:Integer]>
    Big_housing=<[nbr_Balconies : Integer],[nbr_Rooms$\geq$ 3]>
  - View Individual: It is the instantiation of a view concept. We replace the domain of properties concepts with values. Ho_global=<[number=001],[adress=Skikda],[floor=3],[surface=140]>,Appa=<[annex=park00E]>, Ho_F4=<[nbr_Balconies=3],[nbr_Rooms=4]>
  - Viewpoint concept: It is created by user from multi-viewpoints concept by the inheritance mechanism. We represent viewpoint concept with a set of view concepts presented above, to participate in it building.
    VP1_Housing={<[number:String],[adress:String],[floor:Integer],[surface :Integer]>,<[annex:Park]>,<[nbr_Balconies:Integer],[nbr_Rooms$\geq$ 3]>, <[rent:Integer],[expenses:Integer]>}
  - Viewpoint individual: It is the instantiation of a viewpoint concept. We replace the domain of properties concepts with values
    Ho_VP1=={<[number=001],[adress=Skikda],[floor=3],[surface=90]>, <[annex=Park00E]>,<[nbr_Balconies=3],[nbr_Rooms=4]>, <[rent=60000],[expenses=1000]>}
- **Workspace : instantiation space D**
  From the view of design pattern, instantiation of multi-viewpoints pattern consists in replacing the pattern concepts with those of its application domain. This task commences by creating the viewpoint concept from the multi-viewpoints concept. The instantiation of viewpoint concept consists to create an individual which at least one property of type has_localViews or has_globalView has a value when the type is an instance of a local concept or a global concept. This individual is created according to an operation of instantiation in an instantiation space D. Instantiation space D is composed of the vectors that represent the view concepts constituting the pattern. For our example in Figure 2, the instantiation space is: D={<[number:String],[adress:String],[floor:Integer],[surface:Integer]>,<[ annex:Park]>,<[nbr_Balconies:Integer],[nbr_Rooms$\geq$ 3]>,<[rent:Integer], [expenses:Integer]>...}
- **The set intersection operator $\bigcap$ as a meet operator $\sqcap$**
  Viewpoint individuals noted for VPI are ordered in a meet-semi-lattice (D, $\bigcap$) where D presents the instantiation space D of these individuals. The set-intersection operator $\bigcap$ has the properties of a meet operator $\sqcap$ in a semi-lattice. The VPI individuals are designated by the sets. For this, we use here the set-intersection as an operator giving the meet of $d_i$ patterns in D. So in what follows multi-viewpoints individuals views are ordered in a meet-semi-lattice (D,$\bigcap$).

- **$\delta$, affectation function**
  The pattern structure is formalized by a triplet(VPI,(D, $\bigcap$),$\delta$(vpi)) where VPI is all viewpoint individuals,(D, $\bigcap$) is a semi-lattice of viewpoint individuals and $\delta$ is an affectation function which associates to any vpi $\in$ VPI its description (vpi) in D. This description is given by a set of descriptions of the view individuals. We can compute $\delta$(vpi) by grouping $\delta$ of each view individual constituting vpi. Function $\delta$ of view individual is given by a direct projection in space instantiation D.
- **Building the viewpoint concepts lattice**
  We can classify viewpoint individuals and concepts in order by the two operators $A^{\Box}$ and $d^{\Box}$with A $\subseteq$ VPI and d $\in$ (D,$\bigcap$). For this, we replace $\sqsubseteq$ by= in $d^{\Box}$ because we seek to regroup individuals that are in the same class (concept).
- **Evaluation of pattern implementation in rental/ housing system**
  At first, we assume that the multi-viewpoints ontology of rental/ housing system has already been built by pattern, using any means (presented in future work). In this section, we study the possibility of constructing a viewpoint hierarchy from this ontology. The objective here is to illustrate how to construct the user viewpoint by lattice of concepts. This lattice will be transformed by following to an ontological hierarchy that indicates the viewpoint of current user. In the system, the X user selects the views (view individuals) that are interested to build his viewpoint VP_Housing_userX. We will classify these individuals to construct the lattice of concepts. Let consider, for example instantiation space D:={<[number:Integer],[adress:String],[floor :Integer], [surface:Integer]>,<[annex=Park]>,<[annex=Grage]>,
  <[nbr_Balconies :Integer],[nbr_Rooms$\geq$ 3]>,
  <[nbr_Balconies:Integer],[nbr_Rooms< 3]>,
  <[rent $\geq$ 10000],[expenses:Integer]>,
  <[rent< 10000],[expenses:Integer]>,<[localization:CenterCity]>,
  <[localization:SeaSide]>...... } The descriptions of individuals selected by the user X:

  - Ho_VP1={<[number=001],[adress=Skikda],[floor=3],[surface=140]>,<[ annex=park00E]>,<[nbr_Balconies=3],[nbr_Rooms=4]>,<[rent=20000], [expenses=200]>}
  - Ho_VP2={<[number=002],[adress=Alger],[floor=2],[surface=120]>,<[ annex=park00B1]>,<[nbr_Balconies=2],[nbr_Rooms=3]>, <[rent=33476],[expenses=780]>}
  - Ho_VP3={<[number=003],[adress=Skikda],[floor=3],[surface=140]>,<[ annex=park00H]>,<[nbr_Balconies=3],[nbr_Room=4]>,<[rent=25000], [expenses=200]>,<[localization=citycenter]>}

  For demonstration, we apply the derivation operators on the some individuals of example given above and determine the subsumption relationship between them. In lattice, a viewpoint concept is represented by a pair (A, d) where d is the pattern or the description that represents the concept in instantiation space D and A is the set of view individuals of this concept.

Our process is as follows:

- **First operator:** We take A={Ho_VP1,Ho_VP2 } so $A^{\Box} =$ $\{Ho\_VP1, Ho\_VP2\}^{\Box}$ $\{Ho\_VP1, Ho\_VP2\}^{\Box}$=$\delta(Ho\_VP1) \bigcap \delta(Ho\_VP2)$ $\delta(Ho\_VP1)$={(<[number=001],[adress=Skikda],[floor=3],[surface=140]>), (<[annex=Park00E]>),(<[nbr_Balconies=3],[nbr_Rooms=4]>),(<[rent =20000],[expenses=200]>)}={<[number:Integer],[adress:String],[floor: Integer],[surface:Integer]>,<[annex=Park]>,<[nbr_Balconies:Integer], [nbr_Rooms$\geq$ 3]>,<[rent$\geq$ 10000],[expenses:Integer]>} $\delta(Ho\_VP2)$={(<[number=002],[adress=Alger],[floor=2],[surface=120]>), (<[annex=Park00B1]>),(<[nbr_Balconies=2],[nbr_Rooms=3]>),(<[ rent=33476],[expenses=870]>)}= {<[number:Integer],[adress:String],[floor:Integer],[surface:Integer]>, <[annex=Park]>,<[nbr_Balconies:Integer],[nbr_Rooms$\geq$ 3]>, <[rent $\geq$ 10000],[expenses:Integer]>} so $\{Ho\_VP1, Ho\_VP2\}^{\Box}$ = {<[number:Integer],[adress:String],[ floor:Integer],[surface:Integer]>,<[annex=Park]>,<[nbr_Balconies:Integer],[ nbr_Rooms$\geq$ 3]>,<[rent$\geq$ 10000],[expenses:Integer]>}
- **Second operator:** We take the result of $\{Ho\_VP1, Ho\_VP2\}^{\Box}$ as and we will calculate $d^{\Box}$ $(\{<[number : Integer], [adress : String], [floor : Integer],$ $[surface : Integer]>, <[annex = Park]>, <[nbr\_Balconies : Integer],$ $[nbr\_Rooms \geq 3]>, <[rent \geq 10000], [expenses : Integer]>\})^{\Box}$ ={Ho_VP1,Ho_VP2} We have $\{Ho\_VP1, Ho\_VP2\}^{\Box}$ ={<[number:Integer],[adress:String],[floor:Integer],[surface:Integer]>, <[annex=Park]>,<[nbr_Balconies:Integer],[ nbr_Rooms $\geq$ 3]>,<[rent$\geq$ 10000],[expenses:Integer]>} And$(\{<[number :$ $Integer], [adress : String], [floor : Integer], [surface : Integer]>, <[annex$ $= Park]>, <[nbr\_Balconies : Integer], [nbr\_Rooms \geq 3]>, <[rent \geq 10000],$ $[expenses : Integer]>\})^{\Box}$ ={Ho_VP1,Ho_VP2} so ({ Ho_VP1,Ho_VP2},<[number:Integer],[adress:String],[floor:Integer],[surface :Integer]>,<[annex=Park]>,<[nbr_Balconies:Integer],[nbr_Rooms $\geq$ 3]>, <[rent $\geq$ 10000],[expenses:Integer]>}) is viewpoint formal concept named VP12_housing_userX noted for (A1,d1) because $A^{\Box}$= d and $d^{\Box}$ =A. We have also (A2,d2)=({Ho_VP3},{<[number :Integer],[adress:String],[floor :Integer],[surface:Integer]>,<[annex:Park]>,<[nbr_Balconies:Integer],[nbr_ Rooms$\geq$ 3]>,<[rent $\geq$ 10000],[expenses:Integer]>,<[localization:CenterCity] >})is viewpoint formal concept named VP3_housing_userX noted for(A2,d2).
- **Building the user viewpoint (Building the lattice):** Now, we introduce how to order the viewpoint formal concepts in lattice. The viewpoints concepts are presented by a pair(A, d). According to the textual representation proposed in section 3 A and d are the set of vectors. In our approach, the ordering is based on d description. The two concepts obtained above are ordered by $(A2, d2) \leq (A1, d1)$ because $d1 \sqsubseteq d2$ to form the concepts lattice i.e. the VP3_housing_userX concept is subsumed by VP12_housing_userX concept. The result lattice of the above example is shown in Figure 3:

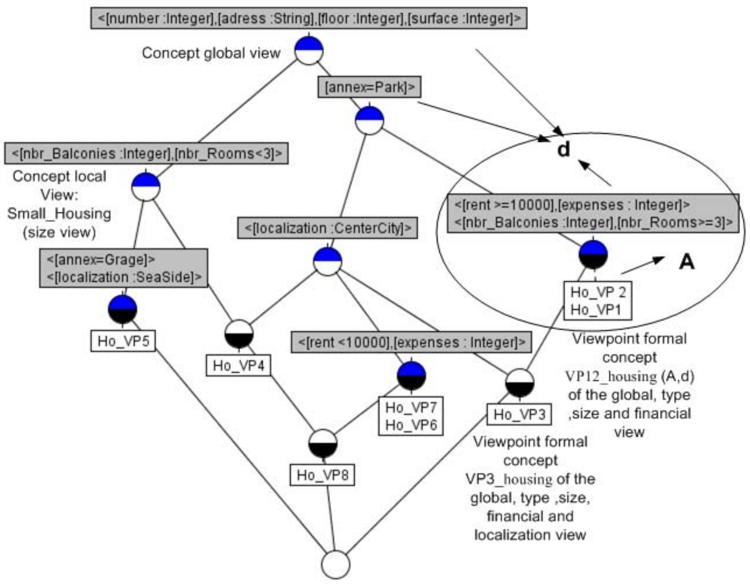

<[number :Integer],[adress :String],[floor :Integer],[surface :Integer]>
Concept global view
[annex=Park]>
d
<[nbr_Balconies :Integer],[nbr_Rooms<3]>
Concept local View: Small_Housing (size view)
<[rent >=10000],[expenses : Integer]>
<[nbr_Balconies :Integer],[nbr_Rooms>=3]>
<[localization :CenterCity]>
A
<[annex=Grage]>
<[localization :SeaSide]>
Ho_VP 2
Ho_VP1
Viewpoint formal concept VP12_housing (A,d) of the global, type ,size and financial view
Ho_VP5
Ho_VP4
<[rent <10000],[expenses : Integer]>
Ho_VP7
Ho_VP6
Ho_VP3
Ho_VP8
Viewpoint formal concept VP3_housing of the global, type ,size, financial and localization view

**Fig. 3.** Viewpoint lattice of VP_Housing_userX

## 4 Conclusion

In this paper, we discussed how to integrate viewpoint notion in ontology with pattern. Our main contribution consists to create a new ontology design patterns, called multi-viewpoints patterns, and its evaluation with pattern structures. For a concrete validation of proposed pattern, we are continuing the work to represent the data in the web of linked open data. The web of linked data is characterized by linking structured data from different sources using equivalence statements, such as owl:sameAs. We will replace the OWL: sameAs by multi-viewpoints pattern.

## References

1. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Oxford Press (1977)
2. Attardi, G., Simi, M.: A Formalization of Viewpoints. Fundamenta Informaticae 23(2/3/4), 149–173 (1995)
3. Bach, T.L.: Construction dun Web Smantique Multi-Points de Vue. Unpublished doctoral dissertation, School of Mines of Paris, Sophia, French (2006)
4. Benchikha, F., Boufaida, M.: The Viewpoint Mechanism for Object oriented Databases Modelling, Distribution and Evolution. Journal of Computing and Information Technology 15, 95–110 (2007)
5. Bendaoud, R., Toussaint, Y., Napoli, A.: LAnalyse Formelle de Concepts au service de la construction et lenrichissement dune ontologie. Revue des Nouvelles Technologies de l'Information, RNTI-E-18, 133–164 (2010)

6. Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P., Gargouri, F.: Contextual Ontologies: Motivations, Challenges, and Solutions. In: Fourth Biennial International Conference on Advances in Information Systems., pp. 168–176 (2006)
7. Batrice, F., Huchard, M., Napoli, A.: Une etude sur la mise en forme de patrons de conception pour les ontologies avec analyse formelle de concepts. Langages et Modles Objets, 83–98 (2010)
8. Bouquet, P., Giunchiglia, F., Van Harmelen, F., Serafn, L., Stuckenschmidt, H.: Contextualizing Ontologies. Journal of Web Semantics 26, 1–19 (2004)
9. Brockmans, S., Volz, S., Eberhart, A., Loeffler, P.: Visual modeling of OWL DL ontologies using UML.Van. In: The Third International Semantic Web Conference, pp. 198–213 (2004)
10. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept anaylsis. Journal of Artificial Intelligence Research 24, 305–339 (2005)
11. Coad, P.: Object-Oriented Patterns. Communication of ACM 35(9) (1992)
12. Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., Villazn-Terrazas, B.: NeOn D2.5.2 Pattern based ontology design. Methodology and Software Support (January 31, 2010 ) http://www.neon-project.org (retrieved)
13. Falquet, G., Mottaz, C.L.: A Model for the Collaborative Design of Multi-Point-of-View Terminological Knowledge Bases. In: Knowledge Management and Organizational Memories, pp. 193–202 (2002)
14. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.: Design Patterns -Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co. (1995)
15. Gangemi, A.: Ontology design patterns for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005)
16. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Handbook of Ontologies, Springer, Berlin (2009)
17. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) ICCS 2001. LNCS (LNAI), vol. 2120, p. 129. Springer, Heidelberg (2001)
18. Gruber, T.R.: A translation approach to portable ontologies. Knowledge Acquisition 5(2), 199–220 (1993)
19. Haav, H.: A Semi-automatic Method to Ontology Design by Using FCA. In: Proceedings of the 2nd International Workshop on Concept Lattices and their Applications, pp. 13–25 (2004)
20. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Tenbrink, T., Stell, J., Galton, A., Wood, Z. (eds.) COSIT 2013. LNCS, vol. 8116, pp. 438–456. Springer, Heidelberg (2013)
21. Marino, O.: Raisonnement classificatoire dans une reprsentation objets multi-points de vue. Unpublished doctoral dissertation, University of Joseph Fourier, Grenoble, French (1993)
22. Presutti V, Daga E, Gangemi A, Salvati A.:(2008), http://www.ontologydesignpatterns.org
23. Ribire, M., Dieng, R.: Introduction of Viewpoints in Conceptual Graph Formalism. In: The 5th International Conference on Conceptual Structures, pp. 168–182 (1997)
24. Rieu, D., Giraudin, J.P., Saint-Marcel, C., Front-Conte, A.: Des oprations et des relations pour les patrons de conception, Congrs INFORSID99, Toulon, Juin (1999)

25. Spaccapietra, S., Parent, C., Vangenot, C.: Gis databases:From multi-scale to multire-presentation. Abstraction, Reformulation, and Approximation 1864,SARA-2000, pp. 57–70. Springer (2000)
26. Vb, O.: Exploiting patterns in Ontology Mapping. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, pp. 950–954 (2007)
27. Whitehead, B., Adams, B., Schildhauer, M., Vardeman, C., Kuhn, W., Shepherd, A., Sinha, K.: Abstracting Transport to an Ontology Design Pattern for the Geosciences. In: 4th Workshop on Ontology and Semantic Web Patterns, Sydney Convention & Exhibition Centre in Darling Harbour, Australia (2013)

# Part IX
# TQMCA 2014 Workshop

# Technologies for Databases Change Management

Kai Jannaschk[1,*] Hannu Jaakkola[2] and Bernhard Thalheim[1]

[1] Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany
{jannaschk,thalheim}@is.informatik.uni-kiel.de
http://www.is.informatik.uni-kiel.de/~jannaschk,
http://www.is.informatik.uni-kiel.de/~thalheim
[2] Tampere University of Technology, P.O. Box 300, FI-28101 Pori, Finland
hannu.jaakkola@tut.fi
http://www.pori.tut.fi/~hj

**Abstract.** Database quality must be maintained during the entire database life span that spans over decades. Quality is typically an issue during development and installation. Later the database system, e.g. the database structure is going to be changed from time to time. This change can be forecasted on the basis of a facilitation methodology. Change management can be based on separation of concern by levels and on abstraction layering techniques. We propose an approach to change-aware database development. The approach is based on an category-problem-cause model that separates change requests into categories which allow a more specific treatment of changes. These specific changes are based on techniques, e.g. restructuring techniques. Category-cause-solution pattern are derived for systematic treatment of changes.

## 1 Introduction

### 1.1 Characteristics of Database System Development

Databases are essential parts of almost all information systems (IS). It can be seen as a critical kernel of it that has to adapt in all changes over the whole life span of the IS. There seems to be tendency towards more and more complex information systems - partially because of growing interoperability requirements between individual applications, but partially also because of longer life span of legacy systems. There are a lot of examples of information systems having their roots in the past of decades ago; this was seen in the year 2000 when big amount of information systems had to be updated to accept the change of millennium in time dependent data handling. Nowadays complex applications handle data from variety of origins. Information about the origin, context, derivation, lineage, ownership or history of some artifact plays a vital role in those applications and is termed as provenance [11].

Information systems can be seen as a construction of layers, in which the layers closest to the user are more tended to radical changes - e.g. user interface technologies are adapted in the trends and technologies. The layers deeper in the structure are staying more stable, but are anyway sensitive for changes having their source either in changing

[*] Corresponding author.

requirements, changing environment or changing other parts of IS. This is the situation with databases, too. They can be seen as a stable part of information system carrying the responsibility of continuing data management over time, but simultaneously has pressure to conceptual, logical and physical change requests. If these requests are not managed properly the quality of data and database is not anymore in an acceptable level.

## 1.2 Data Quality

Data(base) quality as a part of quality management related studies has been on the background reasonable long time. First the focus was given to process quality, which was handled by general purpose quality standards (e.g. ISO 9000) and in Information Technology (IT) context by different software process related applications of it (e.g. ISO 9000-3, ISO 15504, CMM(I)). Simultaneously to this some focus was also transferred to factors related to software product quality. The series of standards ISO 9126 (parts 1-3) defined three product related quality factor categories: internal, external and quality in use. These standards have been the basis for the current work by ISO/IEC JTC1/SC7 having named "Software product Quality Requirements and Evaluation" (SQuaRE), which has widener the scope of product quality standard to cover also data quality in ISO/IEC 25012 [5].

SQuaRE defines fifteen data quality factors from from two different viewpoints: inherent ("the degree to which quality characteristics of data have the intrinsic potential to satisfy stated and implied needs when data is used under specified conditions and system dependent") and system dependent ("the degree to which data quality is attained and preserved within a computer system when data is used under specified conditions"). The standard [5] lists five inherent quality factors (Accuracy, Completeness, Consistency, Credibility and Currentness), and three system dependent quality factors (Availability, Portability and Recoverability). The factors common to both of the views are Accessibility, Compliance, Confidentiality, Efficiency, Precision, Traceability. Moraga et al. in [11] extend this framework to cover quality issues related to web based information systems and list 44 quality factors in four different factor categories.

The above mentioned studies are focused in data quality, which is an essential part of database quality, but not all. Hoxmeier has given additional focus in database (structural) issues in his articles [4] and [3]. He categorizes the database quality factors in four main categories which are further divided in more detailed factors in two level hierarchy: process (the role of development phases, data (quality of data), model (importance of variety of data models), behavior (dynamic properties in data handling). The final list of characteristics covers 34 quality factors in [4].

## 1.3 Towards Systematic Approach for Database Change Management - From Individual Attributes to a Generic Method

The discussion above points out the importance of database change management as a part of IS life cycle and maintenance. To maintain the quality of database and its data has challenges, especially in connection with complex applications handling large amount of diverse data from different sources and from legacy systems having long life cycle. This paper introduces a framework that is planned to support systematic change management in databases. The Category-Problem-Cause (CPC) model applies the principles introduced by Category-Cause-Solution (CCS) method of the authors in [6]. CCS introduces selected change patterns that in software product life cycle context, whereas CPC model is aimed to be used in database context.

The paper is structured as follows. Most challenging task in database change management relates to changes in database structure, which is also in main focus of our paper. The paper is structured as follows. Section 2 introduces the basic elements of our model - facilitation model, separation of concerns and the layered structure of data handling. Section 3 handles the change management in conceptual level by introducing the principles of CPC model. In this Section we also discuss about change policies and methods for evolution of database system models. Section 4 applies the ideas discussed above in database structure change management. Section 5 summarizes the paper.

## 2 Change-Aware Development of Database Systems

Database systems evolve during their lifespan due to some changes in the enterprise or organisation, due to technology used for the system, due to integration of several (sub-)systems, due to the changes in the application area and due to the evolution of the tasks of the system. We might manage changes on the fly, i.e. handling the change whenever it is necessary. We may however also use a framework that can be quickly adopted to the given change and that handles changes already at the development and installation phases. *Change-aware* database development is an open research issue mainly due to the large variety of potential changes.

### 2.1 The Facilitation Model

The Facilitation Model [15] is a methodology model and uses eight stages:

*Inquire - Discovering the symptoms.*
*Investigate - Defining the current state.*
*Vision - Defining the possibilities.*
*Analyse - Generating a list of potential solutions.*
*Qualify- Narrowing solutions down to those with the greatest leverage.*
*Plan - Securing ownership, commitment, permission.*
*Apply - Managing the realisation of the solution(s).*
*Report - Measuring the final outcome and capturing experience.*

This model provides developers with proactive, solution-focused templates. They continuously engender support to both the personnel that face a change and the manager or application engineer. Through this continuous involvement, the Facilitation Model creates awareness, ownership, and commitment to the success of the selected solution(s). *Proactive change handling* can be based on a *general conceptual approach*. [15] uses the following list as a work plan:

(a) *Conceptualisation of change handling solutions.*
(b) *Enhancement of conceptual schemata by change handling templates.*
(c) *Development of control and measurement practices.*

(d) *Development of parameter set reduction and dependence representation techniques.*
(e) *Substantiation of data mining and statistics techniques for performance analysis.*
(f) *Development of a change handling framework.*

## 2.2 Change Management by Separation into Levels

Change management can be based on a separation of concerns. We may systematically separate a number of concerns according to the classical project management framework [6]: 'what' (level 1) provides a specification; 'how' (level 2) defines the way the framework is going to work; 'do' (level 3) prescribes the application of the measures; 'plan' (level 4) provides the schedule for the application; 'manage' (level 5) governs the way of work; 'coordinate' (level 6) integrates the framework into the entire development process; 'optimize' (level 7) revises the project management.

The first four levels for change management are:

1. The *specification* or the *conceptualisation level* is used for a description of the change. The description consists of a specification of the change property, the measurement, and the policies for evaluation. It can be extended by specific policies for various development methods such as agile development, by transformations of change properties into others, and by associations among change properties. Finally, we may derive constraints for the application of the change property.
2. The *control* or *technical level* deals with the application of the change model. It provides guidance for the control procedures such as setting the control management, deriving the scope of control, definition of the control tasks and its actors. The application of the change framework is based on a quality property portfolio.
3. The *application* or *technology level* handles the management of change evaluation within software etc. projects based on the technology of development.
4. The *establishment* or *organisational level* is based on a methodology and may be supported by an change management system.

This four-level framework for change management can be extended by level *five* that provides facilities for handling resolution of change properties and for predicting changes in satisfaction whenever software evolves. Level *six* integrates change management into the optimisation of the software development process. Level *seven* uses experiences gained for the innovation and adaptation of other processes and products that have not yet reached this maturity.

## 2.3 Layering for Change Handling

Software systems and especially database systems are layered systems defined on the basis of a systems architecture. This architecture typically allows to abstract from details such as location of certain operation, infrastructure provided for computation, business user operating the system, or internal structure and functionality of a system. Typically a system architecture allows a number of views such as module or technical architecture, application behaviour architecture, or infrastructure architecture [12] . Moreover, database systems are layered into an external system that communicates with the business through views or public base tables, conceptual system that describes the database system with all its facilities, and internal layer describing the realisation of the system on the basis of a platform or database management system.

This layering and architecturing allows also a multi-layering of system objects into object that are in a ‘good’ (correct and finalised) state, or ‘bad’ (incorrect but known or intermediate) state, or state to be changed. We may thus separate the state of the database system into normal state, intermediate state, final state, and exceptional state. We detected in the previous cooperation project that this separation of concern can be matched with the architecture [7].

Additionally, change handling can be defined on top of business rules. In this case, business transactions and consistency maintenance programs such as triggers or stored procedures can be extended by injecting change handling into these programs. One preliminary prerequisite for such kind of injection is a clear understanding of all possible kinds of changes. This understanding is the main goal for the first task in our framework.

We aim to develop change containers that can be automatically called whenever the system detects or faces some kind of changes. The architecture research in the previous cooperation projects resulted in the discovery how orthogonal schemata may be added to database system conceptual schemata thus forming a way how to combine these schemata. This schema construction approach has been developed in [9] . It allows a multi-shell change handling based on a multi-shell encapsulation of change handlers.

## 3 Towards Conceptualisation for Change Management

### 3.1 The Category-Problem-Cause Model

We can classify and manage changes based on the distinction of their causes. The simplest and obvious cause requiring a change is an *error*, which may relate to design, operation, or organisation. Errors must be corrected. They can occur during application domain description, requirements prescription, software specification as well as during coding. There cannot be any systematic treatment for such. So, we neglect this cause for further consideration. Five main kinds of changes can be distinguished:

- The first cause requiring a change is *incompleteness* (A). A software system operates in an environment (or context), which, following McCarthy [10], we denote by $(w, t)$, where $w$ is a slice of the world at time $t$. Unfortunately it is rarely possible to determine in advance all the components of $w$ that are relevant, and how the relevant components are expected to evolve over time. It is impossible to determine in advance the effect of these components on a computation, which means that changes due to incompleteness require human intervention.
- The second cause requiring a change is based on *insufficiency* (B) to represent the current knowledge about the application, about technology on hand or other issues such as organisational, social and strategic background. Insufficiency typically leads to so-called workarounds that partially patch or repair a situation. This approach causes another stream of changes.
- The third and fourth type corresponds to *deviation from normality*. Users, system developers, and implementers are biased by the ‘normal’ case and do not keep in mind that states different from the normal ones may occur. The system thus operates well in 80% of operating time and suddenly stops normal operating or suddenly behaves in a way that has not been anticipated.

  - Changes are caused when *lifespan changes* (C) are not foreseen.
  - Another kind of change is caused by overestimation of the normal case. *Hidden cases* (D) are overlooked but important.
- The fifth cause requiring a change is *context dependence* (E). Systems are not operating on their own. They share resources with other systems, are used in a combined fashion by users, have different maintenance regimes, have different deployment conditions and thus must be considered to be context dependent. This kind of dependence on the system is observed for all engineering disciplines but not properly handled for software systems.

Architectures of modern computer systems, solutions to application domain tasks, and code developed under these assumptions and environments are interdependent. Change problems are typically observed at the runtime but must be directly tracked back to the systems development and deployment decisions. For instance, the *change category 'space'* with the *problem 'out of space conditions (storage structures)'* can be tracked back to *change causes* in the DBMS Oracle *'poorly forecasted data volumes in physical design'*, *'tablespace fragmentation'*, *'invalid settings for either object space sizes or tablespace object settings'*, or *'not using locally-managed tablespaces'*.

This Category-Problem-Cause Model is the basis for our solution to develop change-aware systems. It is combined with the characterisation by content, motivation, examples, fit criterion, measurements, and considerations.

## 3.2 The Change Policy and Change Patterns

A *change policy* is based on a *problem description* and a chosen *change solution*. Problems are specified by describing a database state in which a problem occurs, by describing properties of better states, by describing potential actions to change states of a system in general, by goal tests that allow to ascertain whether the problem has disappeared and a problem solution controller that traces future states of the database. The change solution is based on a definition of the solution, an illustration of the solution, examples that illustrate the successful solution and patterns that have been used for the solution.

Basic solution patterns specify the required behaviour that should be satisfied under certain conditions, the commanded behaviour that is to be controlled in accordance with commands issued by an operator, the information display for control of the success of the solution states and behaviour information, the simple workpiece used during solution, and the transformation that must be applied to the system for resolving the problem. There are few examples for general solution patterns so far. We defer the development of solution patterns to future research.

We may use a number of *change patterns*. In this paper we provide details for three basic patterns. The first and third pattern correspond to strategies that have been applied for database migration (chicken little and butterfly). A fourth pattern which could be called big bang pattern [8] seems not to be applicable in change management.

**Liquefaction of problem area:** The problem area is completely gathered with all structures, constraints and functions. The problematic elements are collected in a specific notice board. Then the elements are categorised and composed to database types together with the corresponding integrity constraints. The result is a *new database schema*. This new database schema allows us to derive an *extract-transform-load program* for transforming a part of the current database to a new one. In parallel *rewrite rules* for functions are derived. They allow to transform existing functions to new functions. The same kind of rules can also be derived for views. This rule set may be incomplete and for this reason we might have to remain the old part for certain applications. *Backward gateways* and *wrappers* [14] provide a basis for a solution to this problem.

**Change after satisfying observation:** The pattern generalises the butterfly method [1,2,8] to database migration and can be considered to be a lazy change. It uses two change steps. The *intermediate change step* transforms the *initial model* to an *intermediate model* that is tested in application situations and has a validity deadline. If the validity deadline is reached without a transformation success then the initial model and the initial database are enabled and the intermediate model and databases are disabled. The intermediate model is enhanced by *controllers* that monitor and control the success of the changes. The *finalisation change step* transforms the intermediate model to the finalised model and applies this transformation to the application system as well in the case the validity deadline has successfully been reached.

**Transfer after parallel run:** First the current database application is frozen. A solution to the problems is developed and a *new solution* is going to be developed in the *initial step*. A set of *cluster types* [13] is developed for data retrieval. These types allow to retrieve old and new data. The *intermediate step* uses now both systems in parallel. Any new data or any modified data within the change scope is stored in the new system. In the case of change the old data are removed from the old database. In parallel *rewrite rules* and *extract-transform-load programs* are tested within a test environment. In the *finalisation step* the database and the application functions are taken over to the new system.

There are far more patterns known in practice but not yet systematically founded by theory. Pattern are extended by *test portfolio* which are used for check whether the problem has been solved. This change pattern allows us to define specific change templates. For instance, the *change methodology* (generalised from [1]) consists of two evolution steps:

- The *intermediate evolution step* transforms the *initial model* to an *intermediate model* that is tested in application situations and has a validity deadline. If the validity deadline is reached without a transformation success then the initial model and the initial database are enabled and the intermediate model and databases are disabled.
- The *finalisation evolution step* transforms the intermediate model to the finalised model and applies this transformation to the application system as well in the case the validity deadline has successfully been reached.

## 3.3 Methodologies for Evolution of Database System Models

The conceptual model for evolving database system models is based on a formal *evolution methodology* consisting of

- an extended database system model $\mathcal{M}_{DB}$ that supports a standard notion of evolving models (equipped with all the usually model change operators) for which a semantics is provided;
- a collection of interesting reasoning tasks to support the design and management of an evolving model;
- a set of tasks that are combined into a transformation portfolio, a testing portfolio, a data migration portfolio for transformation.

This methodology allows to define specific evolution steps. For instance, the *change methodology* (generalised from [1]) consists of two evolution steps:

- The *intermediate evolution step* transforms the *initial model* to an *intermediate model* that is tested in application situations and has a validity deadline. If the validity deadline is reached without a transformation success then the initial model and the initial database are enabled and the intermediate model and databases are disabled.
- The *finalisation evolution step* transforms the intermediate model to the finalised model and applies this transformation to the application system as well in the case the validity deadline has successfully been reached.

A methodology typically provides a comprehensive set of constructs and rules for their application that serve as the background for constructing applications. We may systematically separate a number of concerns according to the classical project management frame: 'what' (level 1) provides a specification; 'how' (level 2) defines the way the framework is going to work; 'do' (level 3) prescribes the application of the operations and their effect; 'plan' (level 4) provides the methodology for the application; 'manage' (level 5) allows the governance of the change framework; 'coordinate' (level 6) integrates the framework into the entire development process; 'optimise' (level 7) revises the change management.

We are going to elaborate only levels 1, 2 and 3 of the methodology. The *specification level* consists of a specification of model changes. It can be extended by specific policies for various development methods such as agile development. The *control* or *technical level* provides guidance for the control procedures such as setting the control management, deriving the scope of control, definition of the control tasks and its actors. The *application* or *technology level* handles the management of changes.

## 4 Techniques for Database Structure Change Management

### 4.1 Elementary Structural Change Templates

Elementary structural change steps evolve an existing application model and the corresponding database in small steps at a time to improve the quality of the model and the application without changing static and dynamic semantics, functionality and interaction. For instance, the following structural change templates are used within schemata specified in the extended entity-relationship model: apply operation CUD[1] to a type or

[1] Create (add), update (modify), drop (delete); CRUD = CUD + retrieve.

a constraint. An element under change has a scope that leads typically beyond the element. For instance, a modification of an attribute has also an impact on functions, on other attributes, on views, etc.

Structural change templates use the following structure:

***Problem-Cause-Solution:*** explicit and refined statement together with the scope;
***Controller:*** monitor deviations from expected behaviour and evaluate;
***Tradeoff:*** evaluation of the solution after change;
***Change pattern:*** applied pattern with database transformation;
***Function/view/support change mechanics:*** changes to the entire interface system.

We applied this template to the extended entity-relationship model [13]. We could restrict the number of change pattern to two dozen.

*Example 1.* Let us illustrate elementary change steps for insufficiency to represent the current knowledge in the application domain, e.g. bundled complex objects which have rather stable and almost not changing values and constantly changing values for some attributes. We consider the type *Insurance* in the application sketched in Figure 3.

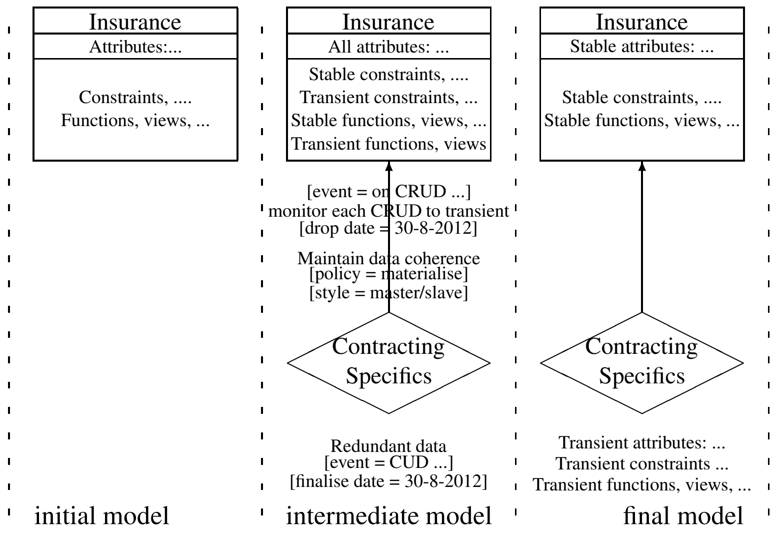


**Fig. 1.** Shifting attributes to subtype in a specialisation hierarchy using unary relationship types

***Problem-Cause-Solution***

(P) *repeated input of data for the type* Insurance*; difficult to maintain coherence in the insurance contracts; maintenance of integrity constraints; code lookup; specific attribute constraints; repeated low-level descriptions;*

(C) *transient data (insurance company details, agent) are combined with general data (insurance company);*

(S) *introduction of a type representing the background data for insurance companies and linking specifics of insurance contracts to the general type; keeping redundant data at new data in a master-slave pattern;*

***Controller:*** *record violations in access to insurance data instead of access to new data; in case of violation send notice and trigger additional changes;*
***Tradeoff:*** *initialisation, changes in performance;*
***Change pattern:*** *change after observation with a change observation period; separate transient and stable data; unary relationship type for IsA relationship;*
***Function/view/support change mechanics:*** *views for combined insurance data; linking facilities for insurance contracts;*

Figure 2 displays corresponding sub-schemata in the HERM [13] notion. $\square^{\mathrm{EoExample}}$

The types under consideration can be attribute types, entity types, relationship types and cluster types. The change operations taxonomy is thus built by combining model language elements, which are subject to change, with elementary modifications, add, drop, and change, which they undergo.

*Elementary change steps.* Elementary change steps evolve an existing application model and the corresponding database in small steps at a time to improve the quality of the model and the application without changing static and dynamic semantics and interaction.

*Test portfolio.* Any modification of an application model must be verified by a full regression test on the system. We must ensure that the application model and the database actually work.

We shall illustrate now structural elementary model changes. We use *control functions* for control of intermediate evolution. Control functions are attached to type structures or their components or to functions. For instance, the controls *[event = on update—insert of invoice]*, *[modification kind = slave]*, *[policy = materialise]* and *[drop date = 1-1-2011]* denote the automatic enforcement of evaluation of a value of a function or value, the master-slave change of a dependent value or function, the explicit storage of results of function application and the deadline of maintenance of the intermediate schema.

*Shift of types within a schema.* Specific instantiations of this transformation are shifts of attributes and functions displayed in Figures 2[2]. The first example shifts a function to the type that requests this function.

## 4.2 Change of a Singleton Model

Similar to the elementary case we may develop a number of category-problem-cause-solution templates for each of the problem-cause cases in (A), ..., (E). These templates are refined to special templates for each case. For instance, the incompleteness with an incomplete coverage might also be caused by incomplete separation of concern. Schemata with such incompleteness suffer from overloaded types. A change in

[2] We use an simplified extended entity-relationship model which type structures are pictured by rectangles (or diamonds) and which type functions are pictured by rectangles with rounded corners attached to the type structure.

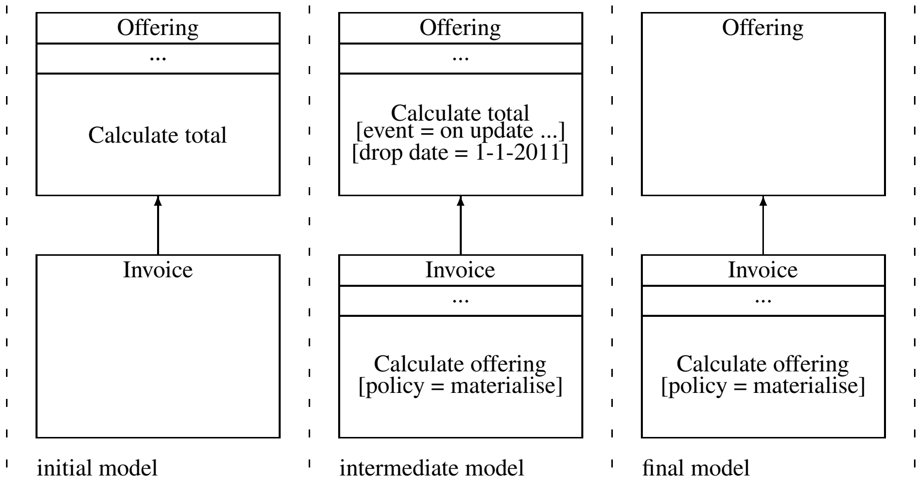


**Fig. 2.** Shifting a function to another type

a schema is seldom applied to a singleton type. It rather uses a subschema which has a border to the rest-schema that remains unchanged and an internal part that is changed. Therefore we start with separating an inner subschema from its border and a rest-schema in a schema.

*Example 2.* Let us consider a simplified application that might be used for managing transportation data which is depicted in Figure 3. We assume that cars which are owned by owners are used to transport goods from one depository (store) belonging to a supplier to another store (retail store) belonging to a market. The first choice could be a complex relationship type on entity types *Car* (incorporating ownership data), *Transport_Company, Marketing_Company, Retail_Store, Insurance_Company, Good* (incorporating supplier data), *Depository_Store* and *Billing_Company*. We abstract from all attributes, e.g., from *Time* for *Transport.* $\square^{\text{EoExample}}$

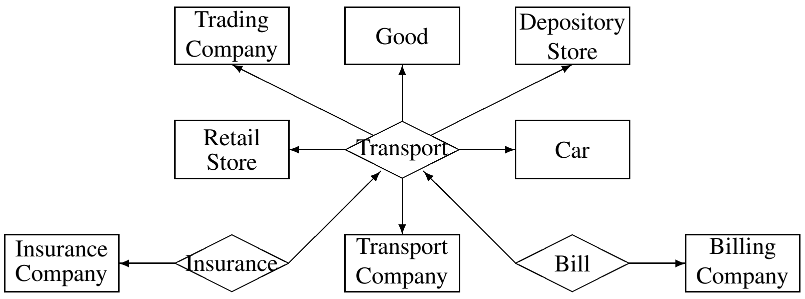


**Fig. 3.** Decomposable independent concepts

Schemata can be analysed based on the quality criteria in the HERM book [13]. Quality optimisation is multi-criteria optimisation and should take into account critical sub-schemata. For instance, a *spider subschema* has a clique structure in the graph and uses at the same time general cardinality constraints that cannot be expressed by simple lookup and participation cardinality constraints. There is no systematic study on criticality yet. Currently we use heuristical rules for it.

*Example 3.* The schema in Figure 3 has a number of critical points:
- Insurances are issued for cars and companies. The transport application is restricted to this kind of insurance.
- Billing is typically applied to the entire transportation event.
- Cars and owners from the first side, markets and their retail stores from the second side and goods and their depository location are relatively independent from each other. This relative independence is not complete.
□EoExample

The change template for structural changes is similar to the one we used in 4.1:

***Problem-Cause-Solution:***
(P) *spider types in a schema with complex semantics;*
(C) *overloaded combined type that simultaneously represent different facet in combined form;*
(S) *decompose the spider type with redevelopment of subschema; detect layering along cardinality constraints; apply pivoting graph grammar rules and use decomposition graph grammar rule [13];*
***Controller:*** *monitor exceptions for irregular semantical cases;*
***Tradeoff:*** *loosing flexibility for irregular cases; concentration on stereotyped treatment;*
***Change pattern:*** *decomposition of a complex type using the schema algebra for HERM [9] including introduction of new types;*
***Function/view/support change mechanics:*** *derive views representing spider type; split CUD for spider type into transactions;*

*Example 4.* The schema in Example 2 and in Figure 3 is now going to be restructured based on the spider type restructuring template.

– We may separate by pivoting rules cars with their owners and insurance from the transport request and transport events.
– Marketing companies have their retail stores (RLocation). Goods are stored at a depository store (DLocation). We thus separate these direct associations by pivoting.
– A transport request relates goods with their current location to markets with their retail stores. We thus reduce transports to transport requests.
– A transport event relates a transport request with a car used for transport. We thus pivot transport events from transport requests.
– Billing applies to the transport event and thus relates to the transporting car and the transport request. It inherits thus the transport request. The transformation to a relational schema that does not use identifier attributes for separate types results

in a relation schema with markets, their retail stores, goods and their depository stores, cars and additional attributes such as time of the transport event. Billing is issued to the market that requested a transport. We do not assume other kinds of billing.

The resulting schema is displayed in Figure 4.

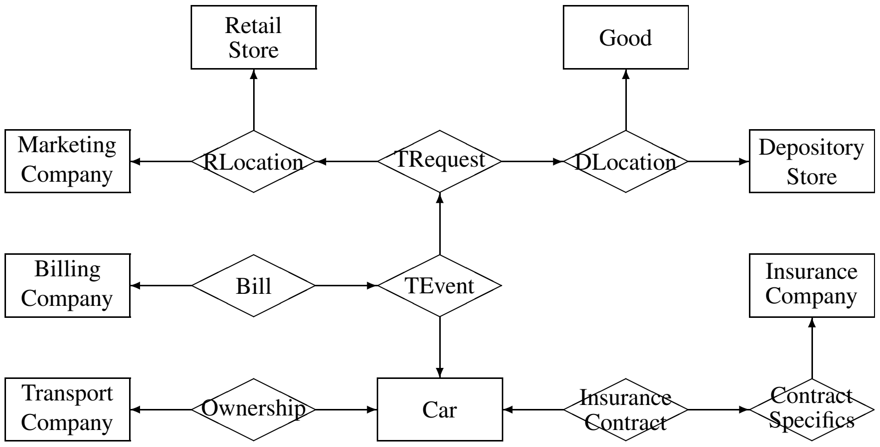


**Fig. 4.** Representation of independent concepts by relationship types

This schema is the result of a sequence of operations which also use results from Subsection 4.1:

(1) *Projection:* Cars and transport companies are associated to each other. We may introduce a new type ownership.
(2) *Shifting:* Transports are carried by cars. The ownership for cars is independent.
(3) *Pivoting:* Transports are completed events based on an issuing event such as requests.
(4) *Multiple shifting:* Insurances are issued for cars and are independent from transport events and transport requests. They are assumed to be independent from ownership.
(5) *Decomposition:* Transport requests are considered to relate markets with their retail stores to goods with the location. Therefore, we introduce new types RLocation and DLocation. These new types form the basis for storing data about transport requests.

□EoExample

## 5 Summary

Change handling is often neglected in database projects and currently not considered in a systematic way. Most database engineering approaches do not take into account change handling. Neglecting in this way is correct whenever the specification is complete, whenever the system is correct, whenever the language for specification and coding covers all potential specifics, whenever no changes occur, whenever there are no hidden cases, and whenever the computational environment is entirely in the hands of the programmer team. It seems however that in almost no application this is true.

We developed a systematic approach to change handling. This approach is based on a categorisation of changes. Categories can be associated to causes and resulting problems. We showed that this category-problem-cause model can be extended by corresponding solutions. To become more systematic a facilitation model must be developed. We may however use the approach given by Tropmann [15] that has been used for sophisticated performance tuning. We show however that by separation into levels the facilitation model is feasible.

There are still many open research issues beside the detailed formalisation of the facilitation model. Open issues are hierarchically structured change sets, general monitors for change sets, tracers and detectors for changes of various categories and changes of changes.

We considered so far only structure or schema changes. Function and feature changes are going to be proposed in a forthcoming paper.

**Acknowledgment.** We would like to thank the Academy of Finland and the German Academic Exchange Service (DAAD) for the support of this research.

## References

1. Ambler, S.W., Sadalage, P.J.: Refactoring databases - Evolutionary database design. Addison-Wesley (2006)
2. Brodie, M.L., Stonebraker, M.: Migrating Legacy Systems - Gateways, Interfaces & The Incremental Approach. Morgan Kaufmann (1995)
3. Hoxmeier, J.A.: A framework for assessing database quality. In: Proceedings of the Workshop on Behaioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling at the International Conference on Conceptual Modeling, Springer (1997)
4. Hoxmeier, J.A.: Typology of database quality factors. Software Quality Journal 7(3/4), 179–193 (1998)
5. International Standardization Organization ISO. Iso/iec-25012: Software engineering - software product quality requirements and evaluation (square) - data quality model. Technical report, ISO/IEC (2008)
6. Jaakkola, H., Thalheim, B.: Framework for high-quality software design and development: a systematic approach. IET Software 4(2), 105–118 (2010)
7. Jaakkola, H., Thalheim, B.: Architecture-driven modelling methodologies. In: Information Modelling and Knowledge Bases, vol. XXII, pp. 97–116. IOS Press (2011)
8. Klettke, M., Thalheim, B.: Evolution and migration of information systems. In: The Handbook of Conceptual Modeling: Its Usage and Its Challenges, ch.12, pp. 381–420. Springer, Berlin (2011)
9. Ma, H., Schewe, K.-D., Thalheim, B.: Modelling and Maintenance of Very Large Database Schemata Using Meta-structures. In: Yang, J., Ginige, A., Mayr, H.C., Kutsche, R.-D. (eds.) Information Systems: Modeling, Development, and Integration. Lecture Notes in Business Information Processing, vol. 20, pp. 17–28. Springer, Heidelberg (2009)

10. McCarthy, J.: Notes on formalizing context. In: 13th Internat. Joint Conf. Artificial Intelligence, pp. 555–560 (1993)
11. Moraga, C., Moraga, M.Á., Calero, C., Caro, A.: SQuaRE-aligned data quality model for web portals. In: QSIC, pp. 117–122. IEEE Computer Society Press (2009)
12. Siedersleben, J.: Moderne Softwarearchitektur. dpunkt-Verlag, Heidelberg (2004)
13. Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)
14. Thiran, P., Hainaut, J.-L., Houben, G.-J., Benslimane, D.: Wrapper-based evolution of legacy information systems. ACM Trans. Softw. Eng. Methodol. 15(4), 329–359 (2006)
15. Tropmann, M., Thalheim, B.: Performance forecasting for perfomance critical huge databases. In: Information Modelling and Knowledge Bases, vol. XXII, pp. 206–225. IOS Press (2011)

# Factors That Influence the Quality of Crowdsourcing

May Al Sohibani, Najd Al Osaimi, Reem Al Ehaidib,
Sarah Al Muhanna, and Ajantha Dahanayake[*]

Dept. of Software Engineering, College of Computer & Information Sciences,
Prince Sultan University - Riyadh, Saudi Arabia
adahanayake@psacw.psu.edu.sa

**Abstract.** Crowdsourcing is a technique that aims to obtain data, ideas, and funds, conduct tasks, or even solve problems with the aid of a group of people. It's a useful technique to save money and time. The quality of data is an issue that confronts crowdsourcing websites; as the data is obtained from the crowd, and how they control the quality of data. In some of the crowdsourcing websites they have implemented mechanisms in order to manage the data quality; such as, rating, reporting, or using specific tools. In this paper, five crowdsourcing websites: Wikipedia, Amazon Mechanical Turk, YouTube, Rally Fighter, and Kickstarter are studied as cases in order to identify the possible quality assurance methods or techniques that are useful to represent crowdsourcing data. A survey is conducted to gather general opinion about the range of reliability of crowdsourcing sites, their passion and contribution to improve the contents of these sites. Combining those to the available knowledge in the crowdsourcing research, the paper highlights the factors that influence the data quality in crowdsourcing.

**Keywords:** Crowdsourcing, Quality, Quality Factors, Crowdsourcing Techniques.

## 1 Introduction

Crowdsourcing was introduced by Jeff Howe in 2006 [1], and was defined as "the process by which the power of the many can be leveraged to accomplish feats that were once the province of the specialized few". However, according to the Merriam Webster's online dictionary definition it means "the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers" [2]. Crowdsourcing is a new way of sourcing, using people who are willing to provide the help or work on some projects. It involves the transfer of available manpower in beneficial manners to help individuals or small business for accomplishing their work.

[*] Corresponding author.

The quality is a characteristic or feature that someone or something has [3]. It is assumed that the data is of high quality when they fit the intended use in operations, decision making, and planning. Alternatively, the data are deemed of high quality if they correctly represent the real-world constructs which they refer to [4].

In the crowdsourcing literature, different researchers have categorized crowdsourcing into different types. Therefore we have summarized them as following:

Daren C. Brabham [5,6] identifies four crowdsourcing types as:

- Knowledge discovery and management approach: crowdsourcing depends on an online community, which produces knowledge or information in well-organized way.
- Broadcast search approach: where it aims to solve problems. When an organization or someone has a problem that is required to be solved, the crowd submits a possible solution.
- Peer-vetted creative production approach: depends on the crowd support, taste, or opinion. It is perfect for designing problems, visual problems, or policy problems.
- Distributed human intelligence tasking: This approach is appropriate for tasks that require human intelligence which it cannot be performed by computers.

Darren Stevens [7] categorized crowdsourcing types as:

- Crowdfunding: when the project is funded by a large group of people, like most charity events.
- Crowdsourced design: when the crowd starts designing different things for different people.
- Crowdwisdom: the type of crowdsourcing when users ask questions to pool of people who are willing to answer.

Thomas Erickson [8] classified another four different types of crowdsourcing based on time and space:

- Audience-centric crowdsourcing (same time, same place): Audience-played games; where audience are divided into subgroups using individual controller such as flight simulator.
- Event-centric crowdsourcing: is a crowdsourcing type that requires a crowd to be hired for a specific event.
- Geocentric crowdsourcing (different times, same place): this type the crowd focuses on the place or region, it enables them to provide tips and information for other users any time such as “Google Maps”.
- Global crowdsourcing (different times, different places): it includes the most commonly known examples of crowdsourcing; it does not require any specific time or place for example “Wikipedia”.

From the many challenges associated when considering crowdsourcing technology, the remote management of the huge amount of participants (crowd) and how to guarantee the quality of data that is received from the crowd are the main worries for crowdsourcing technologies. Therefore, this study hopes to fulfil the gap in research and literature that explores the influence and present knowledge of the crowdsourcing data quality.

This paper aims at identifying factors that influence the quality of crowdsourcing data based on a literature review, a survey, and five case studies. This triangulation approach is promising [34]. Therefore, an informative literature review is carried out in order to produce a concept centric approach for gathering quality defining concepts available within crowdsourcing literature. A survey is designed and carried out describing detailed sampling techniques to show how the survey affects the validity of the concepts. Five case studies are conducted on the widely used and popular crowdsourcing sites and used to justify the identified factors that are derived from literature and the empirical research. The research methodology of this paper is a qualitative study directed by multiple research methods [35].

## 2 Concepts Elicitation through Literature Review

In this section we provide an overview of the quality defining concepts in crowd sourcing literature.

**Quality Control for Real-time Ubiquitous Crowdsourcing** [9]: is about the challenges of controlling the quality of Ubiquitous crowdsourcing, it proposes a technique that lay on the mobility of the crowd to estimate the credibility of data that is provided by the crowd. The information that are provided by the contributors who are not limited to passively-sensor-readings from the device, but also to the productively-generated user's opinions and perspectives, that are processed to offer real-time service. The main challenge in Ubiquitous Crowdsourcing approach is the credibility of the participants, since such approach allows anybody to participate. Furthermore, to understand this challenge the paper has introduced some of the properties of Ubiquitous Crowdsourcing leading to different requirements for controlling the quality of the contributions, such as: Real-time Events; when needed to analyze some collected information to act upon the result, Dynamic Crowd; when the sample of crowd is changing rapidly. The paper has introduced the technique of estimating the quality of contributions based on the contributor's mobility to overcome the above-mentioned challenges. In fact, as a by-product of this research one can learn a lot of information about a human by monitoring their mobility.

**Quantification of YouTube QoE via Crowdsourcing** [10]: demonstrates how it approached successfully to leverage the inherent strengths of crowdsourcing while addressing critical aspects such as the reliability of the obtained experimental data. Here the crowdsourcing is seen as the appropriate model for deriving the Quality of Experience (QoE). A generic subjective QoE assessment methodology is proposed for multimedia applications. The YouTube QoE model takes into account the stalling as a key influence factor based on subjective user studies. It also includes a generic subjective QoE testing methodology for Internet applications like YouTube based on crowdsourcing for efficiently obtaining highly valid and reliable result. The paper quantifies the YouTube QoE for a realistic impairment scenario, where the YouTube video is streamed over a bottleneck link discussing the potential of the crowdsourcing method.

**Assessing Crowdsourcing Quality through Objective Tasks** [11]: investigates the factors that can influence the quality of the results obtained through Amazon's Mechanical Turk crowdsourcing platform. It investigates the impact of different presentation methods (free text versus radio buttons), workers' base (USA versus India as the main bases of M-Turk workers) and payment scale (about $4, $8 and $10 per hour) on the quality of the results. For each run an assessment is made on the results provided by 25 workers on a set of 10 tasks. Two different experiments are run using objective tasks: math and general text questions. In both tasks the answers are unique thereby, eliminates the uncertainty that is usually present in subjective tasks. The reason that may cause this doubt is not figured out: whether the unexpected answer was caused by a lack of worker's motivation, the worker's interpretation of the task or genuine ambiguity. This work presents a comparison of the results and the influence of different factors used. One of the interesting findings is that the results do not confirm to previous studies which concluded that an increase in payment attracts more noise. The country of origin only has an impact in some of the categories and only in general text questions, but there is no significant difference at the top pay.

**Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing** [12]: claims that due to the large number and variety of crowdsourcing users, the quality of the data will be affected. The programmatic gold process is a process that generates gold units automatically with previously known answers. The programmatic gold process increases the accuracy and amount of data. Moreover, it is considered scalable and inexpensive. Research has demonstrated two experiments; the first one compares the effect of manual and programmatic gold, while the other tests the scalability of gold units from 10 to 22,000 units. Discuss mechanisms for controlling the quality through the elimination of suspicious behaviour of fraud, un-ethical or lazy workers by enforcing strategies that assure the quality of the data and work carried out by crowdsourcing. These strategies are: worker screening and inferring workers trust. According to the idea of worker screening, it provides multiple choice questions. Depending on the people's answer they will be accepted and they will perform as required. This strategy is been used in Amazon Mechanical Turk. The inferring workers trust strategy requires several judgments for multiple units of data to estimate the worker accuracy, true answers and worker biases. True responses can be provided by repeating the process several times. This way, it will be possible to categorize trusted workers and reject the rest. For these strategies, both advantages and disadvantages are listed in the research paper. This approach has alluded to Gold-based quality, its challenges, size of gold database, composition of gold units, disguising gold units, how to detect new error types, and how to prevent scammer behavior.

**Analysis of the Key Factors for Software Quality in Crowdsourcing Development: An Empirical Study on TopCoder.com** [13]: provides a detailed explanation on TopCoder.com platform; which is effective software for data development and crowdsourcing platforms. Gives details of a performed experiment analysis - to classify key factors of software quality in crowdsourcing and have discovered 23 factors according to the platform and project. Subsequent application of several mathematical equations of mean, standard deviation, and linear regression the results led to determining 6 factors that have significant impact on the quality. These factors are: number of registered developers, number of contemporary projects, maximum rating of submitted developer, design score, average quality score of the platform, and length of component document. All of the factors mentioned above have positive influence on the quality except the last one. Have proposed four guides to improve the quality of crowdsourcing developments.

**Crowdsourcing Translation: Professional Quality from Non-professionals** [14]: focus on how to improve the quality of translation for non- professional translator in crowdsourcing for a low cost by experiments using crowdsourcing websites to translate sentences from Urdu to English language. To improve and evaluate the quality of that translation several steps are performed. First step is using a professional translations dataset to compare it with non-professional translations sentences. Second step is, collecting low quality translations of sentences from crowdsourcing website Amazon Mechanical Turk. Third step is, in addition to the previous steps, collecting from US-based worker a post-edited version of the translations, and ranking judgments about translations quality. Fourth step is, the selection of the best translation for each sentence. Finally, comparing the selected translations with high quality translations that performed by professional translators in order to evaluate a several selection techniques. According to what is mentioned previously, determined the best quality of data from crowdsourcing website by using a theory. Simple description of that theory is assigning a score to each translation of one sentence to choose the highest score and determine a weight and feature vectors by two different approaches using the “Worker Calibration Feature”; which evaluates the efficiency of each worker by the assigned score to their translations in comparison with reference. Then, evaluate the translation that was obtained from professional translators by calculating the BLEU score [15] for the translator and then compare the BLEU score of translations. Further to determine whether a high quality worker translations exists using two oracle experiments and examine two methods of voting-inspired and assigned BLEU scores to each one. This paper focus on how to improve the quality of translations obtained from crowdsourcing using various methods such as; edit-post, ranking, features function, and scoring.

**Quality Management on Amazon Mechanical Turk** [16]: presents an algorithm for accuracy estimation of workers quality in order to assure the quality of data obtained from crowdsourcing site. It focuses on accurate estimation of workers quality in Amazon Mechanical Turk website in order to assure quality of submitted data. Applies an improved Dwid and Skene [17] algorithm, which returns a confusion matrix (estimated correct results of each task), and error probabilities for each worker. However, the algorithm is not sufficient to measure the quality of workers, because there is an unrecoverable error rate. Therefore, presents an algorithm to separate that error rate from worker's bias in order to obtain reliable quality and performs experiments to examine the accuracy of this algorithm.

## 3 Survey Results

An electronic questionnaire was distributed via social media directed at males and females in Saudi Arabia to determine the factors that influence the quality of crowdsourcing. The questionnaire checks the reliability range of crowdsourcing sites to gauge people's opinion, their enthusiasm to improve the content of these sites, and their participation to improve the data quality. The survey is designed in two parts; first part is about personal information, and the second part on the usage of crowdsourcing sites.

The analysis is based on a sample of 452 responses of which 307 females and 136 males with 9 undetermined. Of the 452 responders, 325 are of above 26 years, 76 are between 21 and 25 years, 41 between 15 and 20 years, 7 are less than 15 years and the age of 3 responders are undetermined. The education background of this sample is made up of 28 PhD holders, 78 Master's degree holders, 277 Bachelor's degree holders, 21 Diploma holders, 2 with elementary education and the education level of 2 responders is undetermined.

Along the influence of usage in crowdsourcing site following questions are evaluated.

- The trust in crowdsourcing sites
- The participation in these sites for the improvement of content
- The participation in the improvement of the quality of the content

The answers according to the reliability range of crowdsourcing sites showed that:

- 300 of them search for information in the Wikipedia, 150 don't, and 2 didn't answer at all.

The responses for whether they trust the information in some of the crowdsourcing sites:

- Wikipedia; 277 yes, 161 no, and 14 no responses
- Google translate: 38 excellent, 139 very good, 134 good, 101 neither good nor bad, 36 bad, and 4 no responses
- The extent of trusting the answers provided by people in Crowd-wisdom sites, such as; yahoo answers (Scores: "5" is reliable to "1" unreliable), 68 of them gave "1", 142 gave "2", 195 gave "3", 32 gave "4", 2 gave "5", and 13 didn't answer.

When asked whether they agree on performing some of their tasks via Internet by anonymous workers who they haven't worked with previously, such as; Amazon Mechanical Turk, or freelancer websites.

- The responses showed 229 answered yes, 216 no, and 7 didn't answer.

On improving the crowdsourcing sites content:

- The respondents' participation in improving the Internet content: 93 of them answered yes, 356 answered no, and 3 didn't answer.
- The participation in answering people's questions in Crowd-wisdom or knowledge sites, such as yahoo answers: 91 of them answered yes, 358 answered no, and 3 didn't answer.
- The improvement of data quality in crowdsourcing sites such as participating in rating the information quality in Wikipedia: 46 yes, 393 answered no, and 13 didn't answer.

On the percentage of higher ratings of crowdsourcing sites influenced the trust in their information: 321 answered yes, 120 answered no, and 11 didn't answer.

Furthermore, textboxes were made available for respondents' comments:

- It is widely acknowledged that the trust in information depends on the website
- The data that is obtained from crowdsourcing is used as knowledge but not for scientific or academic purposes.
- Some of them mentioned that Wikipedia is generally inaccurate, but they used anyhow
- English pages are used more than the Arabic pages

Overall, more than 50% of the respondents indicated that they somehow trust crowdsourcing websites. Unfortunately, most of them don't participate to improve the Internet content or answering questions in Crowd-wisdom sites. They rarely participated in rating the data quality in Wikipedia which is the most common crowdsourcing site. On the other hand, most of them acknowledged that their reliability of the information is affected by higher ratings of data, which means that the general public trusts the quality techniques that are provided by the crowdsourcing sites.

## 4 Case Studies

### 4.1 Amazon Mechanical Turk

Amazon Mechanical Turk provides multiple qualification approaches for the requesters in order to assure the submitted data quality. Some quality approaches related to workers level, and others related to the task requirements. The qualification determined by various criteria that websites provide, which are the number of submitted tasks, how many rejects, and how many approvals that they have accumulated. First type of qualification approaches is allowing a requester to choose worker level by specifying his/her task to Workers, Categorization Masters, or Photo Moderation Masters. The master group is workers with high accuracy qualifications. Second approach is allowing requester to specify location, approval rate for all requesters' tasks, adult content qualification, and number of tasks approved, and customize qualifications [18].

Furthermore, there two types of qualification task that required from worker in order to perform the task. First one is qualification test, which is a rating qualification, and the second one is confidentiality qualification, which is a task agreement that the worker should confirm after reading and agreeing to the terms of qualification agreement [19].

Overall, the qualification methods that Amazon Mechanical Turk use are approval/rejection rating, workers level, workers location, qualification test, and confidentiality qualification.

## 4.2 Wikipedia

Wikipedia is an online multilingual encyclopedia articles that could be edited by the crowd; anyone with internet access could contribute to Wikipedia either by introducing a new article or by editing an existing one. It is the most famous crowdsourcing website. The core idea of Wikipedia is to have a huge website references for the crowd which is provided by the crowd, as well as enabling people to access and contribute anonymously or with their real names. Anybody can access and edit any article but there are exceptions for some of the articles to prevent disruption or vandalism.

Wikipedia has five fundamental principles [20]:

- Wikipedia is an encyclopedia
- Wikipedia is written from a neutral point of view
- Wikipedia is free content that anyone can edit, use, modify, and distribute
- Editors should treat each other with respect and civility
- Wikipedia does not have firm rules

Wikipedia has very ambition contributors who are very alert for any new contribution and keen in editing them and making sure that the articles are free of copyright restriction, and contentious information about living people, and also whether the contribution fit to the Wikipedia's policies [21]. Wikipedia also maintains public/crowd rating of its pages.

## 4.3 YouTube

YouTube provide quality mechanisms about "videos" content that is inappropriate and violates their terms of use. On the other hand, it doesn't focus on the quality of data that is presented. For example, if a video presents erroneous information about specific science, yet YouTube doesn't interfere to eliminate that. However, it provides like/dislike, and comments services to show people's opinion of the video, which we might consider as an evaluation technique. In addition, YouTube provide tools that are useful for a video production process; such as, YouTube Editor, Captions, and Slideshows [22].

YouTube allows crowd to flag videos when there are improper types of videos in order to review and remove it, when that inform is true. The types that are disallowed in YouTube are any video that displays inappropriate behavior that violates YouTube

Community Guidelines [23]. Also, they provide additional reporting mechanisms; such as, Reporting tools (when videos or comments violate YouTube Community Guidelines), Privacy reporting (when videos or comments violate the user's privacy), and Moment of death or critical injury footage (when videos with content viewing moments of death or critical injury footages) [24]. Furthermore, the site provides "Flag for spam" feature in order to allow users to report comments as a spam. After getting enough number of reporting's as a spam, the comments are hidden automatically [25].

In addition, YouTube emphasizes about copyright issues, and it provides approaches to eliminate copyright violations. Those approaches are infringement notification feature, Content ID system, and Content Verification Program [26]. Content ID is a tool which content's owner uses to manage their YouTube's content. When a user uploads a video, the Content ID searches for videos in the database for similar videos submitted as the content owner. If there is a video in the database that matches the uploaded video, the tool applies the policy that determines the content's owner [27]. Moreover, Content Verification Program is a tool for content owners, which is usually used in order to flag via research results or video pages, and report videos that infringe content's owner copyright [28].

Finally, the techniques that YouTube uses to ensure data quality are: flags, reports, notifications, tools. Also, it provides like/dislike and comments services.

## 4.4 Rally Fighter

Local Motors has created the Rally Fighter website, which is the first crowdsourcing application that is specialized in designing cars by the community. Three thousands community members from over 100 countries have submitted more than 35,000 designs. Rally fighter allows members to create their cars by first submitting a design for each piece of the car from the overall design to the electrical systems to the interior, including the name [29]. Then the community members will evaluate the submitted designs by voting for the best. Members can comment on the designs or share ideas to enhance the design. Once a full car design is completed, people can order them online. The team of Local Motors will help the new owner in the actual manufacturing of the car

Local Motors does not require users to register in order to view its products, designs, or user comments. Registration is required to use certain features such as posting comments, evaluating submissions, participating in the forums, submitting designs, and participating in competitions.

Local Motors increased their profits by giving this opportunity to the crowd. The first cars built by rally fighter member costs $3 million where the manufacturing of an airbag for a commercial model costs $6 million [30].

## 4.5 Kickstarter

Kickstarter [31] is a website that helps to collect funding from the crowd for varies reasons such as: rising money for movies, gadgets, games, music, or anything you can think of. As long as Project creators meet the website's guidelines [32], they are fully responsible for their work. They are able to determine the funding goal that they are willing to reach from the backers pledge and the deadline for that project; which doesn't exceed 60 days. In order to be eligible for funding your project, you must reach the goal.

Kickstarter have stated a very detailed terms of use and privacy policy to assure in somehow the quality of crowdsourcing [33]. Here are some of those points:

- To assure the maturity of the user and the ability of taking responsibility they should be over 17 years old.
- To eliminate any fraud of suspicious actions, backers can't reduce or cancel the pledge before 24 hours of the deadline.
- The backers have the ability to request for refund if the project creator is unable to provide the reward to the backer.
- Kickstarter service is only available for personal use not for commercial, unless the project is titled to notify that.
- Details of the project must be shown to assure the growth of that project.
- Kickstarter does not require users to register in order to view the projects.
- Registration is required to use certain features such as posting comments, pledging projects, and submitting projects.
- Any user can report any project that breaks the term of use or guidelines, or any material that violates the copyright of the user by email.

## 5 Analysis

The data quality of crowdsourcing is impacted by various factors. These factors are presented in the Figure 1. Those factors are elicited by the triangulation approach: literature review, case study, and survey. Those quality factors are made up of 4 categories: Company's worker i.e. the workers of the Crowdsourcing platform provider, Crowds, Techniques, and Cost.

The quality factors related to company's worker; who works as the reviewer of received data, are:

- worker's capacity for accuracy: which is affected by worker's biases. For instant in YouTube, the reviewer's nationality might affect the understanding of the videos
- worker's experience level

The quality factors which are related to the crowds:

- using data evaluation methods: when the users don't participate in data evaluation or the unreliable users' evaluation of the data is made, whatever the evaluation methods they use (rating, like/dislike, comments, etc.) in another words, if there is a useful video that matches people's needs in YouTube and gets a high rate of dislike it effects the quality of data of the site.

- user's biases; for example, a requester requires from workers in Amazon Mechanical Turk (M-Turk) to evaluate specific websites, the result task is effected by worker's biasness
- user's accuracy which is effected by several things such as: user's motivation, user's base (the English translation from worker in M-Turk from USA is different from India), the user's interpretation of the task or presented data
- user's location and experiences; which affect the submitted data for example, task performing in M-Turk or design voting in Rally Fighter.
- inappropriate user's behaviour: such as, cheating (using tools to perform task), violation of copyrights
- user's age: this affects the data evaluation, or task's performance
- user's information; the registration in website or present user's information assists to ensure data quality that is provided by user

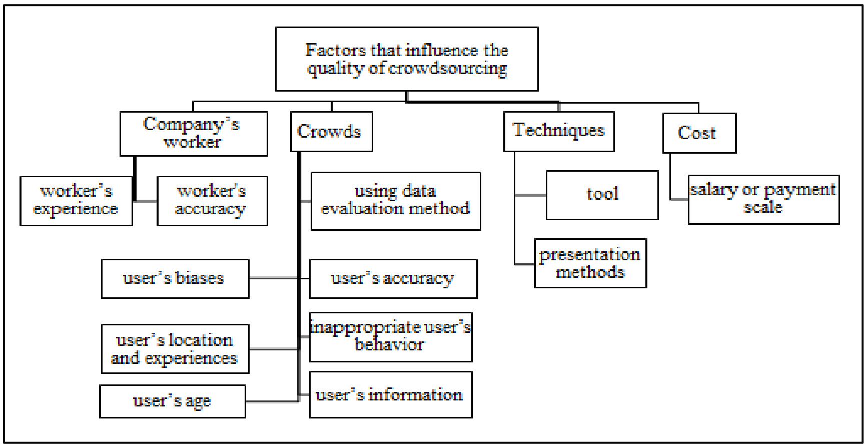


**Fig. 1.** Factors that influence the quality of crowdsourcing

The factors related to techniques that are used in crowdsourcing websites:

- tool: the accuracy result of the tools used by a website affects the data quality such as YouTube provide dispute Content ID tool match in order to solve misidentified video by this tool
- presentation methods: for example sometimes the use of radio buttons are more quality prone than textbox

The factor related to cost is: the salary or payment scale, whether that cost for reviewer (company's worker) or crowd who submit the data.

## 6 Conclusions and Discussions

In this paper we have made a contribution to the quality factors that influence the crowdsourcing applications. The crowdsourcing has become a common word these days and it is important to identify the quality factors in order to define the trust in the data. In the past, the knowledge is acquired from books, articles and other resources, whereas then the knowledge was provided only by specialists'. But, with the evolution of crowdsourcing applications, information can be gained from unknown users with unknown backgrounds making it difficult to trust the data generated by those crowds. Therefore, the content or data must be evaluated to ensure the quality of the information.

During this study we reviewed several crowdsourcing research articles and we represented different techniques that are used by different crowdsourcing applications to control the given information, Further, we conducted a survey in order to identify the extent of people's reliability of crowdsourcing applications, and extent of enthusiasm to improve Internet content and its data quality. Finally, we defined and presented the factors that influence the quality of crowdsourcing.

We faced many challenges while conducting this research. First we attempted to contact crowdsourcing websites in order to gain insights into their data quality assurance techniques, but unfortunately we didn't get any responses. Second we found out in our survey some inaccurate answers or to be honest unreliable answers because there were questions related to each other posing different percentages. So, we decided not to include those questions in the analysis.

In this paper we have made a contribution to the quality factors that influence the crowdsourcing applications. We hope to extend this model towards a quality framework for crowdsourcing applications.

## References

[1] Howe, J.: The Rise of Crowdsourcing. Wired magazine 14 (6), 1-4 (2006)

[2] Webster, M.: `http://www.merriam-webster.com/dictionary/crowdsourcing` (accessed April 19, 2014)

[3] Webster, M.: `http://www.merriam-webster.com/dictionary/quality?show=0&t=1398180177` (accessed April 19, 2014)

[4] Roebuck, K.: Data Quality: High-impact Strategies - What You Need to Know: Definitions Adoptions, Impact, Benefits, Maturity, Vendors. Emereo Publishing (2001)

[5] Brabham, D.C.: Crowdsourcing: A Model for Leveraging Online Communities. In: Delwiche, A., Henderson, J. (eds.) The Routledge handbook of participatory cultures. Routledge, New York (2001)

[6] Aitamurto, T., Leiponen, A., Tee, R.: The Promise of Idea Crowdsourcing – Benefits, Contexts, Limitations. white paper online available (2011)

[7] Soresina, C.: SkipsoLabs, `http://www.skipsolabs.com/en/blog/crowdsourcing/types-ofcrowdsourcing` (accessed March 6, 2014)

[8] Erickson, T.: Geocentric Crowdsourcing and Smarter Cities: Enabling Urban Intelligence in Cities and Regions. Founded in the industry website, Crowdsourcing.org (September 2010)

[9] Mashhadi, A.J., Capra, L.: Quality Control for Real-time Ubiquitous Crowdsourcing. In: Proceedings of the 2nd international workshop on Ubiquitous Crowdsouring, pp. 5–8 (2011)
[10] Hoßfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., Schatz, R.: Quantification of YouTube QoE via Crowdsourcing. In: IEEE International Symposium on Multimedia, pp. 494–499 (2011)
[11] Aker, A., El-Haj, M., Albakour, M.-D., Kruschwitz, U.: Assessing Crowdsourcing Quality through Objective Tasks. In: Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012), Istanbul (2012)
[12] Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., Biewald, L.: Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing, Human Computation. Papers from the 2011 AAAI Workshop (WS-11-11), pp. 43–48 (2011)
[13] Li, K., Xiao, J., Wang, Y., Wang, Q.: Analysis of the Key Factors for Software Quality in Crowdsourcing Development: An Empirical Study on TopCoder.com. In: IEEE 37th Annual Computer Software and Applications Conference, pp. 812–817 (2013)
[14] Zaidan, O.F., Callison-Burch, C.: Crowdsourcing Translation: Professional Quality from Non-Professionals. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011), pp. 1220–1229 (2011)
[15] Papineni, K., Poukos, S., Ward, T., Zhu, W.-J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of ACL, pp. 311–318 (2002)
[16] Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on Amazon Mechanical Turk. In: Proceedings of the Second Human Computation Workshop (KDD-HCOMP 2010), Washington DC, USA (2010)
[17] Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the EM algorithm. J. Roy. Statist. Soc. C (Applied Statistics) 28(1), 20–28 (1979)
[18] QualificationRequirement - Amazon Mechanical Turk, http://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_QualificationRequirementDataStructureArticle.html (accessed March 27, 2014)
[19] SpeechInk, Getting Qualified To Do Jobs (June 26, 2010), http://www.youtube.com/watch?v=yMXlCaH7VcQ (accessed March 27, 2014)
[20] Wikipedia, http://en.wikipedia.org/wiki/Wikipedia:Five_pillars (accessed April 22, 2014)
[21] Wikipedia, http://en.wikipedia.org/wiki/Wikipedia:About (accessed April 22 2014)
[22] Tools - YouTube, YouTube, https://www.youtube.com/yt/creators/tools.html (accessed April 25, 2014)
[23] YouTube Community Guidelines, YouTube, https://www.youtube.com/t/community_guidelines (accessed April 25, 2014)
[24] Other reporting options,YouTube, https://support.google.com/youtube/answer/2802057?hl=en&ref_topic=2803138 (Accessed April 25, 2014)
[25] What does the, Mark as Spam, feature do?, YouTube, https://support.google.com/youtube/answer/128036?hl=en (accessed 25 April 2014)

[26] Submit a copyright infringement notification, YouTube, https://support.google.com/youtube/answer/128036?hl=en (accessed April 25, 2014)
[27] How Content ID works, YouTube, https://support.google.com/youtube/answer/2797370?p=cid_what_is&rd=1 (accessed April 25, 2014)
[28] How to use the YouTube Content Verification Program, YouTube, https://support.google.com/youtube/answer/3010500?hl=en (accessed April 25, 2014)
[29] Rally Fighter, http://localmotors.com/rallyfighter/ (accessed April 17, 2014)
[30] Munoz, J.A.: CNN, How the Internet built a $100,000 race car (March 13, 2013), http://www.cnn.com/2013/03/12/tech/web/crowdsourced-carsxsw/ (accessed April 17, 2014)
[31] Kickstarter, https://www.kickstarter.com/ (accessed April 21, 2014)
[32] kickstarter, https://www.kickstarter.com/help/guidelines (accessed April 21, 2014)
[33] kickstarter, https://www.kickstarter.com/privacy?ref=footer (Accessed April 21, 2014)
[34] Jane, W., Watson, R.T.: Analyzing the past to present, MIS quarterly (2001), https://www.kickstarter.com/privacy?ref=footer
[35] Saunders, M., Lewis, P., Thornhill, A.: Research Methods for Business Students, 6th edn. Prentice Hall/ Pearson Education (2009)

# Framework for Social Media Big Data Quality Analysis

Dua'a Al-Hajjar, Nouf Jaafar, Manal Al-Jadaan, and Reem Alnutaifi

Prince Sultan University – College for Women, King Abdullah Road,
Riyadh 11586 Saudi Arabia

**Abstract.** Unlimited amount of unstructured data is being captured and analyzed over social media. The paper highlights the issue of lack of standard quality control approaches that could be utilized for all social media sites. This is due to the variety of formats of big data acceptable over these sites. The issue reveals a challenge not only in the capture of big data but also in the analysis and yield of valuable data, which affect decision-making. The paper reviews a collection of archived documents in the field of big data and social media. This paper presents a framework identifying the issues of quality analysis of big data on social media, examining current techniques used by social media companies to capture and analyze big data, and mapping social media sites and the appropriate combinations of big data capture and analysis techniques with the data quality control requirements.

**Keywords:** Big data, Social Media, Framework, Quality Analysis.

## 1 Introduction

Nowadays, the amount of data collected and analyzed is increasing enormously. Companies collect a constantly growing amount of data whose size ranges from a few dozens of terabytes to many petabytes of data. Matthew Gold [1] defined the term big data as "data sets whose size is beyond the ability of commonly used software tools to capture, manage and process data within a tolerable time". Unlike regular data, which is primarily consistent, structured, and built on relational database platforms, big data is unstructured and puts a challenging limitation on the relational databases [2]. Initially, they believed that the volume of information collected and available to them outstripped the capabilities of the memories used by their computers for processing. This encouraged scientists in Google and Yahoo to implement Google's MapReduce and Hadoop [3]. Volume is not the only distinguished property of big data. There are four properties associated with big data and known as the four V's of big data[4]:

**A. Volume:** Volume explains the massive amount of data available.

**B. Velocity:** Velocity or speed includes aspects like how fast data is being generated, and how fast it becomes stale or obsolete, and how fast we need to analyze it to make the data meaningful.

**C. Variety:** Big data is a collection of data sets which are in different formats, and combining these data sets together to analyze them is very complex. This makes it difficult to be processed by traditional data processing techniques; therefore, big data problems can't be solved with the computing resources that are available to most organizations.

**D. Veracity:** Veracity can be defined as the degree of accuracy and certainty of data. Data must be truthful and trustworthy, or the decisions made by organizations will not be meaningful and valuable.

Big data is often used in social media and customer sentiments that help business organizations to get customer feedback. This feedback is later used to make decisions. The number of users of social media is increasing enormously. Mayer-Schönberger and Cukier [3] have exemplified the huge amount of big data by providing some numbers and statistics. One example is Facebook which gets 10 million new photos being uploaded every hour. This creates a digital trail that Facebook could analyze to learn about their users' preferences [3]. According to [5], monitoring and analyzing this rich content can yield unprecedentedly valuable information that enables users and organizations to acquire actionable knowledge. The quality of data captured, processed and analyzed over these networks has remained a questionable issue.

In this paper, issues related to quality of data being collected on social media are investigated and current techniques used for data capture, analysis, and processing are assessed and compared. The purpose is to create a framework of quality assessment techniques of big data on social media that suits several social networks depending on needs and specifications. The main objective is to measure to what extent the current big data quality assessment techniques influence collecting, processing, and anazlyzing big data on social media. Section 2 provides an overview of the big data analytics and data quality requirements on social media. Section 3 presents the research method that is used to collect related data about the topic. Section 4 discusses the techniques used to capture and analyze big data in four of the most popular social media services and maps the techniques to the quality aspects resulting in a framework. The framework is presented in Section 5. The paper concludes and provides recommendations in Section 6.

## 2 Related Works

The author in [6] indicated that metadata, which means data about data, influences big data analytics. The author proposed a framework for metadata management in big data analytics that consists of: metadata discovery, metadata collection, metadata governance, metadata storage and metadata distribution. Vemuganti believed that there is a need to create a testing environment to conduct an intensive functional and non-functional testing. Their testing techniques focused on Hadoop ecosystem as a baseline for big data processing. They discussed three main testing steps that could be performed in each big data processing phase. The first phase is the validation of pre-Hadoop processing to tackle issues in data capturing where the unstructured data is loaded into Hadoop Distributed File System (HDFS). Secondly, validation of Hadoop MapReduce process to overcome any problem that might be introduced in data processing phase. Then, validation of data extracts needs to be conducted. Overall, this paper concludes that for each data source, there is a need to establish specific data quality requirements that align the nature of these data sources to tackle the potential data quality issues.

Liang and Dai in [7] proposed a new system architecture that can automatically analyze the sentiments of micro-blogging messages. They combined their proposed system with manually annotated data from Twitter for the task of sentiment analysis. They confirmed that machines will be able to extract a set of messages and determine their sentiment direction when using this architecture.

The authors in [8] explained the ways used in analyzing Twitter data to detect the emotions change and matching it with events that cause that change. Also, they explained the process of analyzing the information used in US elections by using computational methods to detect and extract the most dominated people in the elections. They illustrated ways compare and analyze different sets of topics to detect the bias, common people, and writing styles properties by using Support Vector Machines, which is a machine learning technique. In addition, they described the techniques used in discovering the trend of different set of articles in many languages of different networks from different countries by using machine translation technology combined with other computational methods.

Analyzing the big data on social media is a challenge. The authors in [9] introduced different classification techniques and algorithms that could be used for analyzing large-scale social media data like Latent Dirichlet Allocation, support vector machines and Naïve Bayes classifier. They proposed a workflow for analyzing social media data. Based on the workflow, the tools can be implemented to provide integration between qualitative analysis and the detection algorithms.

## 3 Research Method

This research is designed to propose a framework for quality assurance of big data techniques in certain social media sites. The research is exploratory in nature. The research is conducted to obtain the recent and most useful techniques or strategies that are used in social media big data analysis. These techniques are collected and analyzed from different sources based on qualitative research methods. The following process is followed: first, the collection of archived papers to discuss quality issues of big data in social media. Second, techniques used to capture, process, and analyze social data on four popular sites are discussed. Quality aspects related to these techniques are explored. Then, a framework is developed to map the techniques on studied social media sites and their quality aspects.

## 4 Identifying Quality of Social Media Data Analytics Techniques

The authors choose to survey and analyze the quality analysis techniques of the most popular social media sites. The decision of which social media services to study is based on Alexa website which is a sub-company of Amazon that provides deep analytical insights to benchmark, compare and optimize businesses on the web.

According to Alexa [10], Facebook is globally ranked as the second popular site following the most popular search engine, Google. Twitter comes in the eighth place. Linkedin is ranked as 10. Also, Flickr is explored as one of the first image and video hosting social media sites.

## 4.1 Twitter

Analysis of data on twitter is a challenge because Twitter streams contain large amounts of meaningless messages and polluted content, which may negatively affect the detection performance [5]. Twitter tweets are restricted in length to 140 characters. Twitter Public information includes the following [11] [12]:

A. Tweet Data: Tweet text, Timestamp, and Unique tweet ID
B. Profile Information of the tweet author: user's Twitter handle; e.g. @user1, user's location; e.g. NY, USA, URL, which contains more information about the user on an external website, textual description about the user and his/her interests, user's network activity information on Twitter; e.g. 1 follower, and following 6 friends, number of tweets published by the user; e.g. 3 tweets, profile creation date, verified mark if the identity of the user has been externally verified by Twitter
C. Geo-location Data Specific to Tweet, for example, latitude and longitude coordinates of the mobile location if this feature is enabled by the author
D. Twitter's Entities: Images, URL's, @Mentions, Hashtags

To allow applications to search and download public information: REST (or Search) APIs and Streaming APIs [11] are used. According to [12], REST APIs use the pull strategy for data retrieval while Streaming APIs rely on the push strategy for data retrieval. However, Twitter APIs have certain limitations; the search query is limited to 1000 characters and only up to 6-9 days of historical data can be retrieved. Also, the retrieved data is based on relevance, not completeness, which may lead to some missing data or users [13].There are three licensed resellers on Twitter data that rely on Twitter APIs and have complete access to Twitter Firehose - the complete stream of tweets that users post to the service [14]. The resellers are as follows[11]:

### 4.1.1 DataSift

DataSift is a licensed third party reseller of Twitter data that provides access to past data as well as streaming data. The main solutions used are [15] [16]:

#### 4.1.1.1 Pull Connector

This connector helps customers collect data using the Push delivery mechanism.

#### 4.1.1.2 MongoDB

MongoDB is a NoSQL, document-based, scalable, high-performance, open source database used mainly for big data applications in social media. MongoDB data is organized into a hierarchy as follows: database, collection, and documents. A database is a set of collections while a collection is a set of schema-less JSON-like documents. MongoDB has its own built-in aggregation MapReduce functions that can be used to aggregate large amounts of data [17]. MongoDB relies on REST API. The architecture of MongoDB is designed such that each MongoDB cluster is composed of one or more shards where each shard holds a portion of the total data. Reads and writes are automatically routed to the appropriate shards [18]. A replica set-backs each shard by holding a replication of the shard data. MongoDB has the capability to automatically shard the data between multiple hosts. This allows Mongo to scale horizontally to thousands of servers [18].

#### 4.1.1.3 CouchDB

CouchDB is a NoSQl, document-based database that supports replication and versioning. All operations are performed using a REST API.

#### 4.1.1.4 Zoomdata

Zoodata is a data analysis and visualization tool used by DataSift subscribers.

The following are quality related aspects to DataSift products:

- Twitter sends DataSift a notification whenever one of their users deletes a public tweet. These notifications are passed to DataSift subscribers as part of the stream. However, there are no delete messages from these private accounts.
- The cursor, which is a pointer to the Push queue and is associated with the current Push subscription, may point to data that is not stored in the subscription queue. Using a cursor that points to a batch of interactions that no longer exist in the queue leads to a bad request error.
- A limit of two requests per second is put on Pull Connector's subscribers' requests.
- MongoDB supports horizontal scalability making it easy to copy and deploy one database from one server to another [17].
- A highly scalable, redundant cluster, with no single point of failure results when combining shards with replica sets [18].
- MongoDB data files are unencrypted and no automatic method is provided to automatically encrypt data files. Therefore, any attacker with access to the file system may be able to directly extract information from the data files. To resolve this security issue, the user shall encrypt sensitive information before saving it to the MongoDB database [18].
- In an experiment conducted by Li and Manoharan [19], the researchers proved that MongoDB is amongst the fastest NoSQL techniques in creating database buckets, reading, writing and deleting key-value pairs from the buckets.
- If the data stream generates data at a faster rate than the permitted by the delivery, the buffer is filled up until it reaches to the point where some data is discarded.

### 4.1.2 Gnip

Gnip provides access to streaming (real time) data as well as public archived data. Three common GNIP products used for social data collection are [16]:

#### 4.1.2.1 Data Collector

Data Collector is a solution to collect social data from multiple public APIs. Retrieval requests are optimized for maximum data retrieval and duplications are removed.

#### 4.1.2.2 Rehydration API

This product delivers full Twitter content including the associated metadata and Gnip enrichments.

#### 4.1.2.3 PowerTrack

PowerTrack is a filtering language that gives users the ability to get complete coverage of the data they need and allows filtering on geo-boundaries, keyword and phrase matches, the presence of links or images, and the language of an activity.

Certain quality aspects are related to Gnip products:

- Gnip provides reliable and sustainable access to social media data.
- By tracking the activities sent to users in real-time, Gnip ensures protection from data loss caused by brief disconnects from the user's real time connection. A redundant stream can mitigate the effects of a disconnected stream by providing a second live connection to the stream. The redundant stream can help bridge periods of disconnection and potentially prevent missed data.

### 4.1.3 Topsy

Topsy provides access to past data only. Topsy provides search results on a topic based on the influence of a person's posting about the topic. The influence is measured mainly by how many times the author's tweet has been re-tweeted or reposted by other people and the number of people who follow and read that person's tweets. Topsy analyzes all traffic and then selects the top most important and influencing Tweets, articles, and other media to include [20].

The following aspect affects the quality of Topsy:

- Retrieving results is based on influence rather than completeness and that may affect in turn the quality of the search results.

## 4.2 Flickr

Flickr is one of the dominant social networks in photo sharing services. The analysis process of its data can reveal useful information to support different purposes involving: Personal Network Analysis, E-commerce and Geo-Tagged Applications [21]. Flickr has an API and provides a variety of options to access it by different programming languages to conduct an analysis. Recently, the social data analytics that extracted from Flickr is widely generated by using social network analysis (SNA) tools to automate the analysis process [22]. SNA tools emphasize on extracting information from social data by applying quantifiable metrics and visualization techniques to demonstrate the social interactions patterns [22].

### 4.2.1 NodeXL SNA Tool

NodeXL (Network Overview for Discovery and Exploration in Excel) is an open-source tool developed to perform social networks analysis and plugged easily into

Excel [23]. It supports the workflow of undertaking a basic networks analysis and includes: collecting, storing, analyzing, visualizing and publishing [23]. This tool is useful to obtain insight from social media data that enable the user to interact through photos. Different pictures' elements in terms of tags can be analyzed through this tool including: people tagged in the picture, locations of this picture, dates, comments and the events behind each picture [24].

Capturing and analyzing data can be summarized in the following points:

- Social Network Importer: NodeXL provides several options to import data from different social media including: Twitter, Facebook, YouTube, Flickr, email, blogs and wikis [23]. The data can be imported directly from these sources or by storing the data in separate files such as: text, CSV, or GraphML files [23]. At this stage the imported data is demonstrated in a structured workbook template that includes multiple worksheets to store all the information needed to represent a network graph. Network relationships are represented as an "edge list", which contains all pairs of entities that are connected in the network. While, worksheets contain information about each vertex and cluster [22].
- Social data analyzing: NodeXL involves a collection of measures provided by The Stanford Network Analysis Platform SNAP and integrated within NodeXL: the Betweenness Centrality, Closeness Centrality, Eigenvector Centrality and Page Rank metric. They also include two clustering algorithms: Girvan-Newman and Clauset-Newman-Moore that are responsible of grouping the nodes into collections in the network graph automatically [25].

The following are quality related aspects to NodeXL tool:

**Amount of data:** There is limitation of the amount of data that can be imported since the spreadsheet of NodeXL cannot store more than1048576 rows [24] [26].

**Timeliness:** Provides the ability to generate a pre-scheduled analysis reports to capture the updated data [23].

**Accessibility:** Different types of data can be retrieved from Flickr easily and quickly such as: List of photos with links to their sources, Photo to photo relationship that contains pairs of photos linked by a tag and List of tags by photo[24].Once the data is retrieved it will be directly accessible through the spreadsheet [23]. Moreover, NodeXL is an open-source tool that allow user to modify or commercially apply it [23].

**Usefulness**

- The interface combines both statistics and visualizations facilities into one single view in order to demonstrate the relationship between the data and the associated visualization simultaneously [27].
- NodeXL is plugged into Excel, so it can utilize the existing charts facilities that provided by Excel [27].
- NodeXL enable the novice and expert users to analyze the networks regardless of the simplicity or complexity of the network [23].

- It has the ability to generate automated reports which scheduled on specific basis to get new updates and a summarized description as well as the network graph. Also, these reports can be shared with others through the email or the web [23].
- It supports automation feature in which the user can automate the process of network analysis one a single step [23].
- Most effective algorithm have been implemented on this tool such as: calculating the betweenness centrality, which aims to reduce time and cost [21].
- The data can be imported easily on this tool and the associated graph can be generated quickly to illustrate these data [27].

**Consistency:** The statistical data that is displayed in the spreadsheet is consistent with the information in the associated graph and any changes happened to the data will incorporated automatically in the graph [23].

**Understandability**

- NodeXL provides sophisticated methods to organize the data through the workbook template that consists of different worksheets required to store information that will be used to generate the corresponding network graph [22].
- NodeXL supports an "encode" feature; the user can display different representations of network graph and select multiple properties such as: shape, colour, size, transparency, and location to understand the crucial relationship between graph's elements clearly [22][27].
- NodeXL supports the "connect" feature; the user can easily explore the interrelation between the data displayed in the workbook template and the network graph [27].

Other quality issues and improvement tasks of this tool have been noticed and suggested by different users and can be concluded in the following points [27]:

1. There is a need of more clustering algorithms to group the nodes effectively.
2. The tool cannot be accessed outside Excel.
3. Some users were unable to export the final analysis through this tool.
4. The tool doesn't have undo/redo function to provide traceable histories of exploration and the option of safe mode to save the work if failure occurs.
5. There is a quality issue related to the produced graph, which has a large file size since it has JPEG or BMP formats which are uncompressed. The user can face issue in text legibility when he/she attempts to resize the graph and display it in other applications.

## 4.3 Facebook

Facebook has a large number of active users with high privacy model. Facebook collects three types of data considered as company's assets [28]:

1. Profile data: contains user name, country, photos, interests, contact information, and education.

2. The social graph: represent the relationship between Facebook users. The social graph helps to detect the common interests between the users, identify the friendship between the users, obtain user profile data in efficient ways, and obtain user's friends' data.
3. Traffic Data: related to the users privacy data like IP address, session's information, web browser used to access Facebook, etc.

Several techniques exist; some described below, to extract/analyze the required data from the Facebook:

### 4.3.1 Facebook Query Language FQL

The authors in [28] explained how Facebook allows application developers to extract data by making queries using FQL that is embedded into applications. It is a subtype of the SQL database language. It allows retrieving user's information to gather social graph data. By using FQL, we can easily extract user IDs by typing query like: *Select UID from user where UID in (.....).*It is also possible to retrieve information of the group members by submitting query like (*SELECT uid FROM group_member WHERE gid = G;).*However, it is restricted to return a maximum of 500 group's members. After getting large list of UIDs, then it is possible to extract their friendships connections by typing this query:(*SELECT uid1, uid2 FROM friend WHERE uid1 IN (X,Y, ... Z)AND uid2 IN (U,V, ... W);).*The following explain the quality aspects related to FQL:

- High performance as it is speed and robust tool [29].
- High availability since it is embedded into applications [28].
- There are many flaws detected which allow applications with no registered users to access and collect data of the social graph from Facebook [28].
- There a restriction in amount of data returned since FQL return result doesn't exceed 5k edges for any query. So, the query needs to be broken in many sub queries [29].
- This tool can access information that is might be invisible to the other tools or methods due to the privacy settings. The users need to withdraw from Facebook to hidden their information from the query results [28].

### 4.3.2 Netvizz

Netvizz is a Facebook application written in PHP runs on a server provided by Digital Methods Initiative and used to extract and analyze Facebook data. It is used to collect data and generate data files of many sections of Facebook without the need to do the manual data collection process or develop scripts. Also, it provides raw data for personal networks, pages, and groups. Additionally, it provides data perspectives through comment text extraction. It generates a .gdf file that contains the information of the bulleted sections. Netvizz extracts data from three section of Facebook [30]:

A. **Personal Network:** divided into 2 types:
   - Friendship network: provides a graph where users are represented as nodes and friendship as edges. Other information like gender, language, posts and likes counts are displayed upon request.

- Like network: It works like the above feature as it represents the user and the liked elements as nodes and liking process as edges.

B. **Groups:** work similar to friendship network. Another feature in the group is social graph which represent the interaction between the group members so that an edge created between users when one user makes some activities in another's profile.

C. **Pages:** represented as a bipartite network since the users and posts are nodes and edge created between them when the user like or comment on the post. From this network we can identify the most active users and popular post by the number of comments or likes which represented as edges to be used in statistical analysis.

Netvizz doesn’t contain a visual interface, so it produces the collected information in two types of data file that can be used in analysis process: Network file and Tabular file, which is used in traditional statistical analysis techniques. However, Netvizz connects the data and the various network analysis toolkits provide visual representation that is provided by Facebook such as Gephi. Grephi is a tool used to manipulate and explore the graph by many provided properties. After collecting the required data, it can be visualized by using Gephi as an example through displaying graph that contains nodes and the edges connecting them based on the source of the data: group, page, etc. Moreover, Netvizz includes many metrics used to facilitate the analysis process likeBetweenness Centrality, a graph metric that works based on colouring the nodes. We can demonstrate the node’s social status and its influence to other nodes using this metric. Below are some quality related aspects to Netvizz [30]:

- There is a restriction in the amount of data due to limitation of the number of users that can be retrieved from a group to 5000.
- It supports security as it is one of the Facebook applications. The users need to login with their Facebook account to access the data. Also, Netvizz doesn’t store any type of the retrieved data in the database and the generated files are deleted in regular basis.
- As it is an API type then it seems to have high availability and reliability.
- The displayed data is accurate since it is extracted from the Facebook database and analyzed using robust tool.
- It provides high accessibility as the Netvizz extracts many types of data from different sections like personal network, like network, groups, and pages. The pages feature provided by Netvizz enables extracting the needed data of the users and their uses without access individual accounts. Additionally, Netvizz can be accessed from many visulization tools like Gephi, which allows exporting the analyzed data as svc or pdf.

### 4.4 LinkedIn

LinkedIn is the largest social network on the Internet for business professional with over 277 million members. Generating a multi-dimensional of data by LinkedIn members has been massively growing year by year. Avatara, Voledmort and Hadoop are well-suited techniques for this data. These techniques are used in collecting, processing and analyzing data. According to [31], some libraries can be used with

Hadoop techniques to improve the style and quality of testing like Pig, Kafka and DataFu's. We describe quality issues on LinkedIn feature and techniques in more details in the following subsections.

### 4.4.1 Quality Issues on LinkedIn

The main issue with Skills and Expertise feature is unstructured textual data like abbreviations and contextual. For instance, a member may identify his skills with JSP (a common abbreviation for Java Server Pages). The challenges for this issue are tackled by using standardized corpus that has some classification on it. As a result, the reliability and correctness of data are increased. Using different analytics techniques like natural language processing, text mining, entity extraction and machine learning can perform this.

Another significant feature in LinkedIn is People You May Know. One of the primary issues of this feature is scalability of matching data, matching member to all other 277 million members. Voldemort technique is one of the most suitable solutions for this problem.

### 4.4.2 Techniques

#### 4.4.2.1 Text Mining

As mentioned in [32], text mining technique is used to increase efficiency, accessibility and relevance of data content. Three main steps to mining the text are preprocessing the text by comparing the text with natural language, applied appropriate algorithm in order to process the text such as classification, visualization and clustering and finally analyse the text. The author in [33] demonstrates some related limitations; unstructured data can't analyze directly but it must transpose text into numerical values that can then be linked with structured data in a database. According to [34], ambiguities on natural language text led to flexibility, usability and consistency issues.

#### 4.4.2.2 DataFu's

According to Vaughan [35], Datafu's is an open source collection of useful user-defined functions (UDF) working with large-scale data in Hadoop and Pig. All of the UDF are well-tested library to ensure quality and to help in data mining and statistical task. Let's take the task to counting event in recommendation system as an example. In this system the member will be recommended an item, called an impression, and he can accept, reject or ignore that recommendation. To perform this task, a list will be generated for each member. The list includes all items with a count of how many items were accepted, how many items were rejected and how many items were seen. DataFu technique is used to group all of the data together by the member then count the occurrences of each items using CountEach UDF. Next, it merges all of the separate counts for each type of event together into one tuple per item. Finally, it cleans up the schema and puts some default values in place where member did not take any actions reject or accept. Datafu technique will trigger only one mapreduce job and the percentage of the slowest map and reduce task of each job is 10%. Two main libraries for DataFu techniques are:

- **DataFu Pig** is high-level data flow language that consists of dozen operators and makes it easy to write Hadoop MapReduce jobs. It support UDF's and includes functions for statistics, Bags, sessions, set operations, estimations, link analysis and sampling.
- **DataFu Hourglass** is a library for incrementally processing data using Hadoop MapReduce that was designed to make computations over sliding windows more efficient. As mentioned in [31], this library is utilized in tracking metrics accurately and efficiently by using basic arithmetic to make query incremental rather than scheduling a query that runs on daily basis to gather the data for specific number of days which is time consuming and requires recalculating the stats for that number of days. The main concept of this library is that the input data is partitioned by day and output data is produced from the previous data by adding and subtracting input data that enables processing only the new data. The main two type of incremental processes job are Partition-preserving and Partition-collapsing. The partition-preserving job is performed by consumes partitioned input data and produce partition output data whereas partition-collapsing job is performed by consumes partitioned input data and merges it to produce a single output. The goal of this technique is achieved by reducing the computational resources required and by improving the performance.

## 5 Framework for Quality Analysis of Big Data on Social Media

The table below provides a framework of the satisfied quality factors of big data techniques on social media. We map social networks and their capture and processing techniques to quality factors [16] [17] [18] [19] [20] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] in the framework given below.

**Table 1.** Framework for Quality Analysis of Big Data on Social Media

| Social Network | Technique | Satisfied Quality Factors |
|---|---|---|
| Twitter | DataSift | Scalability, Performance, Reliability, Fast operations, Backup, and Accessibility |
| | Gnip | Sustainability, Reliability, Protection from data loos, Availability, and Accessibility |
| | Topsy | Accessibility |
| Flickr | NodeXL | Timeliness, Accessibility, Usefulness, Consistency, and Understandability |
| Facebook | FQL | Performance, Accessibility, and Availability |
| | Netvizz | Security, Availability, Accuracy, Accessibility, and Reliability |
| LinkedIn | Text mining | Efficiency, Reliability, Correctness, and Accessibility |
| | DataFu Pig | Efficiency, Performance, and Accuracy |
| | DataFu Hourglass | Efficiency, Performance, and Accuracy |

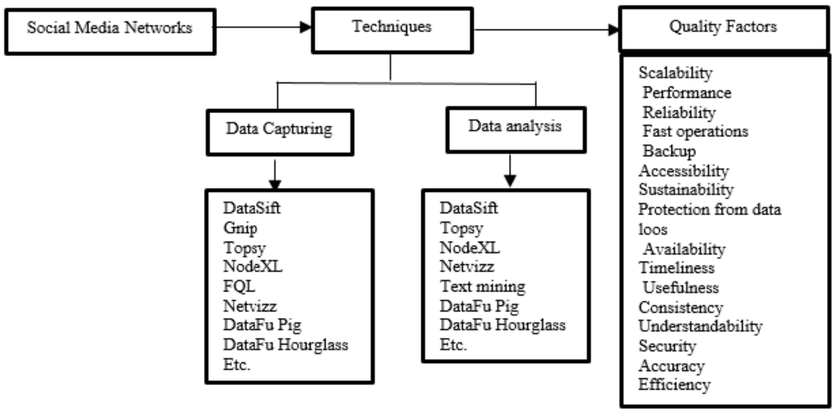


**Fig. 1.** Framework for Quality Analysis of Big Data on Social Media

## 6 Conclusion

This paper provided an overview of quality issues of techniques used in the capture, analysis and processing of big data on social media and it produces a framework for mapping big data analysis techniques with the satisfied quality control factors for several social media sites. Hopefully, this paper could provide suggested areas for business to improve their quality issues. The business must ensure that all the required quality factors, such as performance, efficiency and flexibility, are satisfied in the technique most of the challenges will be controlled.

## References

[1] Gold, M.K.: Debates in the Digital Humanities. Univ of Minnesota Press (2012)

[2] Deters, R., Lomotey, R.K.: RSenter: terms mining tool from unstructured data sources. Int. J. of Business Process Integration and Management 6, 298–311 (2014)

[3] Mayer-Schönberger, V., Cukier, K.: Big Data: A Revolution that Will Transform how We Live, Work, and Think. Eamon Dolan/Houghton Mifflin Harcourt, New York (2013)

[4] Robinson, D.: Big Data – The 4 V's: What Was Old is New Again; Part 1, from Making Data Meaningful (December 3, 2012), http://makingdatameaningful.com/2012/12/03/big-data-the-4-vs-what-was-old-is-new-again-part-1/ (retrieved March 4, 2014)

[5] Atefeh, F., Khreich, W.: A Survey of Techniques For Event Detection in Twitter. Computational Intelligence (September 4, 2013)

[6] Vemuganti, G.: Metadata Management in Big Data. Infosys Labs Briefings (2013)

[7] Liang, P.-W., Dai, B.-R.: Opinion Mining on Social Media Data. In: IEEE 14th International Conference on Mobile Data Management (MDM), Milan, vol. 2, pp. 91–96 (2013)

[8] Flaounas, I., Sudhahar, S., Lansdall-Welfare, T., Hensiger, E., Cristianini, N.: Big Data Analysis of News and Social Media Content (2014), http://www.see-a-pattern.org/sites/default/files/Big%20Data%20Analysis%20of%20News%20and%20Social%20Media%20Content.pdf (retrieved 2014 йил 23-03 from See a pattern)

[9] Xin Chen, M.V.: Mining Social Media Data for Understanding Students' Learning Experiences (2013)

[10] Alexa, Actionable Analytics for the Web, from Alexa (April 5, 2014), http://www.alexa.com/ (retrieved)

[11] Kumar, S., Morstatter, F., Liu, H.: Twitter Data Analytics. Springer (2013)

[12] Small, H., Kasianovitz, K., Blanford, R., Celaya, I.: What Your Tweets Tell Us About You: Identity, Ownership and Privacy of Twitter Data. The International Journal of Digital Curation 7(1), 174–197 (2012)
[13] Chen, X., Madhavan, K., Vorvoreanu, M.: A Web-Based Tool for Collaborative Social Media Data Analysis. In: IEEE Third International Conference on Cloud and Green Computing, pp. 383–388. IEEE Computer Society, Karlsruhe (2013)
[14] Miners, Z., Ribeiro, J.: Apple snaps up Topsy, PrimeSense: acquisitions reflect interest in Twitter access, 3D sensing technology. Macworld 31(3), 24 (2014)
[15] DataSift. Pull. from DataSift Developers (February 10, 2014) (retrieved April 18, 2014 )
[16] Information Management Journal. Search Firms to Mine Tweets. Information Management Journal 46(3), 17 (2012)
[17] Boicea, A., Radulescu, F., Agapin, L.I.: MongoDB vs Oracle - database comparison. In: Third International Conference on Emerging Intelligent Data and Web Technologies, pp. 330–335. IEEE Computer Society, Bucharest (2012)
[18] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., Abramov, J.: Security Issues in NoSQL Databases. In: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 541–547. IEEE Computer Society, Changsha (2011)
[19] Li, Y., Manoharan, S.: A performance comparison of SQL and NoSQL databases. In: 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 15–19. IEEE, Victoria (2013)
[20] Information Today. Topsy introduces alerts and reports. EContent 36(4), 15
[21] Akrouf, S., Meriem, L., Yahia, B., Eddine, M.N.: Social Network Analysis and Information Propagation: A Case Study Using Flickr and YouTube Networks. International Journal of Future Computer and Communication (2013)
[22] Hansen, D.L., Rotman, D., Bonsignore, E., Milic-Frayling, N., Rodrigues, E.M., Smith, M., Shneiderman, B.: Do You Know the Way to SNA?: A Process Model for Analyzing and Visualizing Social Media Network Data. In: 2012 International Conference on Social Informatics (SocialInformatics), Lausanne (2012)
[23] Smith, M.A.: NodeXL: Simple network analysis for social media. In: 2013 International Conference Collaboration Technologies and Systems (CTS), San Diego, CA (2013)
[24] Gómez, J.A., Shneiderman, B.: Understanding social relationships from photo collection tags. Human-Computer Interaction Lab & Department of Computer Science (2011)
[25] Smith, M.M.-F.: NodeXL: a free and open network overview, discovery and exploration add-in for Excel (2007/2010), `http://nodexl.codeplex.com/` (retrieved 2014 йил 20-April from CodePlex)
[26] Microsoft. Excel specifications and limits, `http://office.microsoft.com/en-us/excel-help/excel-specifications-and-limits-HP010073849.aspx` (retrieved 2014 йил 20-April from Microsoft Office)
[27] Bonsignore, E.M., Dunne, C., Rotman, D., Smith, M., Capone, T., Hansen, D.L., Shneiderman, B.: First Steps to Netviz Nirvana: Evaluating Social Network Analysis with NodeXL. In: International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC (2009)
[28] Bonneau, J., Anderson, J.: Prying Data out of a Social Network. Cambridge, UK (2009)
[29] Hogan, B.: Facebook as a data capture site: Techniques, Traps, Terms & Conditions (2011 йил 24-March), `http://www.slideshare.net/primath/dl-tech-talkhogan` (retrieved 2014 йил 18-April from slideshare)
[30] Rieder, B.: Studying Facebook via Data Extraction. The Netvizz, Amesterdam (2013 йил 29-June)
[31] Hayes, M.: DataFu's Hourglass: Incremental Data Processing in Hadoop (October 03, 2013)
[32] Diane, M.: The Value and Benefits of Text Mining
[33] Sukanyal, M., Biruntha, S.: Techniques on Text Mining (2012)
[34] Alfawareh, S.J.: Techniques, Applications and Challenging Issue in Text Mining (2012)
[35] Vaughan, W.: DataFu 1.0 (September 2013)

# Part X
# Doctoral Consortium

# Querying and Managing Complex Data

Luiz Gomes-Jr. and André Santanchè

Institute of Computing, State University of Campinas (UNICAMP)
13083-852, Campinas, SP, Brazil
{gomesjr,santanche}@ic.unicamp.br
http://www.ic.unicamp.br

**Abstract.** Advances in models, algorithms and computing power enabled complex network research to be applied in several areas. From social sciences to the internet, from microscopic to macroscopic phenomena, from natural to man-made networks, network analysis have expanded its application scenarios to most areas of human activity. Although research in complex networks has generated tools that cover a wide range of topics, the analysis is still largely restricted to experts and done in *ad-hoc* settings. Our goal is to develop data querying and management mechanisms for complex networks. We are defining and developing the Complex Data Management System (CDMS), which is intended to provide database-like means to interact with complex networks. Our current query model is based on ranking clauses defined over a Spreading Activation (SA) processing model. SA allows us to compose metrics that can capture several aspects of the dynamics of the underlying network.

**Keywords:** Complex Networks, Graph Data Models, Graph Query Languages, Complex Data.

## 1 Introduction

Data and query models have evolved towards supporting increasingly complex interrelationships. Highly structured models for management of relational data, with precise query semantics and predictable results, preceded semi-structured models that added path indirection and flexible schema. Document management and information retrieval (IR) has to deal with even greater indirections, typically correlating imprecise queries (e.g. keywords) to lists of results ranked according to metrics that capture non-obvious aspects of the underlying data (e.g. PageRank). More recently, graph databases have taken this diversity to new levels, allowing unrestricted correlation of data elements.

As a consequence of the advances in models, algorithms and computing power, new types of applications became possible, such as social networks, recommendation systems and collaborative filtering. Network analysis, usually associated with the complex network field, has become an important resource for supporting these new applications and has been used in diverse areas such as systems biology, neuroscience, communications, transportation, power grids, and economics [1].

Although graph databases and complex network research are fueled by the same increase in data complexity and analysis demands, there is little intersection between the areas. Graph databases are typically based on query languages specialized in matching and retrieving subgraphs while complex network tasks often rely on ad hoc algorithms running on various analysis software. Although graph analysis algorithms can be and are often run over graph databases, it represents little more than a data storage mechanism.

In the complex networks area, non-obvious, topology dependent properties are formalized as network dynamics processes. The analysis of the network dynamics reveals emergent behavior that cannot be expressed as graph isomorphism tasks. To aggregate this type of graph analysis in current graph databases, one typically has to run off-line algorithms and store the results in node properties. After this off-line and time-consuming step, the values can be used to query the database. This set-up provides no flexibility for the user to influence how the graph analysis should be undertaken. In our approach, we focus on providing the user with a query language that can specify subgraphs in which the analysis should be done, select the types of links that are relevant, and combine several graph analysis metrics – all in a declarative fashion and in an online querying scenario.

Supporting these queries in a DBMS (DataBase Management System) brings several new architectural requirements: (i) the data model must support the high level of complexity, (ii) the query language should be flexible enough to allow correlation of data when little is known about how they are linked and organized, (iii) a new abstraction for query evaluation should be able to capture the intended network dynamics while allowing for under-the-hood optimizations, (iv) data management mechanisms must be coherent with the heightened importance and diversity of relationships.

We propose a new definition for the increasingly important type of data we aim at supporting. Inspired by the area of complex networks, we adopted the term *complex data*. Our research aims developing the CDMS (Complex Data Management System), a system suited to query and manage complex data. Our goal is to provide a tighter integration between complex networks and databases, enabling a database-like interaction for complex network tasks and, conversely, incorporating graph analysis capabilities into graph query languages.

## 2 Research Directions

Our solution for complex data querying is based on a property graph data model with weighted relationships. We propose a new query language that allows ranking of elements based on properties of the topology of the graph (the dynamics of the network). The queries are evaluated based on a variation of the spreading activation model, which is the core of the query processor and the main target for query optimization strategies.

Our proposed query language extend current graph query languages by adding a "RANK BY" clause. The clause should enable an arbitrary combination of metrics that expresses the global ranking condition defined by the user. Figure 1 show some query exemples over SPARQL and Cypher[1]. These queries are meant to demonstrate the expressiveness of the approach in a wide range of applications [5]. More details about the language and its design principles can be found in [4].

Figure 1a shows a Cypher query that suggests diagnoses of patients based on their symptoms. It is based on an underlying database containing symptom-diagnosis relationships specified by professionals in a previously unrelated project. The combination of the Connectivity and Relevance metrics showed results comparable to unassisted diagnoses from health professionals (details in [4]).

Figure 1b shows a product recommendation query (SPARQL) that finds products that the client Bob (with uri :bob) has not purchased. The query traverses Bob's friendship network to find products purchased by his friends that might be relevant to him.

The query in Figure 1c (SPARQL) ranks species that play an important role in the food web and are related to the biome of coral reefs. This type of query would identify species that should be main targets for monitoring and preservation efforts.

```
a  START diag=node(*)
     WHERE has(diag.Type) and diag.Type = "Diagnose"
     RETURN diag
   RANK BY
     1 RELEVANCE OF diag TO node($patient) ,
     2 CONNECTIVITY OF diag TO node($patient)
b  SELECT DISTINCT ?product
    WHERE { ?product :type :Product .
      FILTER NOT EXISTS (:bob :purchased ?product) }
   RANK BY RELEVANCE OF ?product TO :bob
    FOLLOW (:friendsWith, :purchased)
    DEPTH 3 DIRECTION BOTH
c  SELECT ?species
    WHERE { ?species :type :MarineSpecies }
   RANK BY
    3 INFLUENCE OF ?species FOLLOW :preysOn
       DIRECTION OUTBOUND,
    1 RELEVANCE OF ?species TO :CoralReefBiome
```

**Fig. 1.** Examples of extended queries (namespaces have been omitted)

The ranking clauses in the queries are based on a set of metrics that we are developing. To capture the underlying network dynamics we employ spreading activation processes (TSA). Spreading activation [2] was developed to infer relationships among nodes in associative networks. The mechanism is based on traversing the network from an initial set of nodes, activating new nodes until certain stop conditions are reached. By controlling several aspects related to this activation flow, it is possible to infer and quantify the relationships of the initial nodes to the reached ones. The SA-based metrics proposed so far are Relevance, Connectivity, Reputation, Influence, Similarity, and Context.

The proposed query language and new type of user interaction envisioned for the CDMS also require changes in architectural elements when compared to

[1] `http://docs.neo4j.org`

traditional database systems. Query processing and data management mechanisms such as query optimizers, Data Manipulation Languages (DMLs), indexes and collection of data statistics must be coherent with the heightened importance and diversity of relationships. We propose several query optimization and approximation strategies to speed up query processing. We also introduce the concept of *mappers* in our DML, which are similar to stored procedures, but are aimed at relationship creation and can be integrated in the query language to allow query-time correlation of data.

## 3 Progress to Date

In our research so far we have elicited the main requirements for a database management system suited for Complex Network tasks. To meet the requirements, we proposed a new query processing model, based on spreading activation, that can capture diverse aspects of network dynamics. The proposed constructs are used to compose ranking metrics that are integrated in a declarative query language. This set-up allows users to write queries based on parameters tailored to their information needs.

We have implemented prototypes of several parts of our proposed system, including a query processor supporting several ranking metrics, query optimization and approximation mechanisms, and data management extensions. We have tested our framework on real data and real applications such as nursing diagnosis. Our preliminary experiments show that our approach is practical in terms of performance and that our language can express complex concepts in real data and application scenarios. Combining the TSA model and a declarative query language offers many opportunities for query optimization (as reported in [3]).

Besides the new query language and model, the CDMS must offer adequate data management mechanism. We have implemented the parts of the mapper mechanism and have integrated them in the query language, enabling a more flexible interaction and data management.

As ongoing and future research, we are working on query optimization mechanisms, analyzing alternative query processing approaches, gathering new datasets and use cases, and devising model extensions to support a wider range of complex network tasks.

**Acknowledgments.** This work was partially financed by the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO Project and PRONEX-FAPESP), INCT in Web Science (CNPq 557.128/2009-9) and CAPES-COFECUB, with individual grants from CAPES and FAPESP (Proc. 2012/15988-9 and 2014/01419-8).

## References

1. Costa, L., Oliveira Jr., O., Travieso, G., Rodrigues, F., Boas, P., Antiqueira, L., Viana, M., Rocha, L.: Analyzing and modeling real-world phenomena with complex networks: A survey of applications. Advances in Physics 60, 329–412 (2011)

2. Crestani, F.: Application of spreading activation techniques in information retrieval. Artif. Intell. Rev. 11(6), 453–482 (1997)
3. Gomes-Jr, L., Costa, L., Santanchè, A.: Querying complex data. Technical Report IC-13-27, Institute of Computing, University of Campinas (October 2013)
4. Gomes-Jr, L., Jensen, R., Santanchè, A.: Towards query model integration: topology-aware, ir-inspired metrics for declarative graph querying. In: GraphQ-EDBT (2013)
5. Gomes-Jr, L., Santanchè, A.: The Web Within: leveraging Web standards and graph analysis to enable application-level integration of institutional data. Technical Report IC-13-01, Institute of Computing, University of Campinas (January 2013)

# Implementation of Generalized Relational Algebraic Operations with AsterixDB BDMS

Nickolay Saveliev

Saint Petersburg State University, St. Petersburg, Russian Federationa
spbu@spbu.ru,
nickolay.saveliev@gmail.com

**Abstract.** Every year data with a large volume and variety to be processed is growing. Such data handling requires special approaches, methodologies and tools. One of the most important parts of manipulating information is a high-level declarative query language that helps us not only to solve a lot of kinds of problems, but gives us great possibility for optimization.

In this paper there will be considered a practical part of approach that introducing an extended relational algebra from [1]. This algebra can be used by query optimizer based on cost models. Implementation of generalized algebraic operations that uses big data management system AsterixDB [2], a cost model for estimation and optimization is the main aim of this work.

## 1 Introduction

Nowadays Big Data is synonymous with large volume of data with different structure and diversity. That's why it is important to have a possibility for fast implementation of information retrieval algorithms for such data. A high-level declarative query language can give as such opportunity. Also a powerful platform for query execution is required.

General stages of query execution in relational databases are described in [7] and consist of query language parser, optimizer, code generator and executor. If we are talking about relational databases, all these components are parts of one system. The minus of such systems is that it is required to store all processed data in some predefined formats. But today the amount of data is very large, so data is stored in different formats. That's why systems to process data from heterogeneous information resources are required.

One of the ways to build such system is to provide user a high-level query language, implement optimizer based on generalized relational algebra, code generators and executors for every informational resource that it contains. To minimize work databases can be treated as heterogeneous informational resources where information is stored and can be queried without developing executors for every data source. But such approach still needs a code generator for every database management system and a cost model for optimizer to choose not only an optimal query plan but also a platform where the plan will be executed with the best efficiency. There is a query processing flowchart on Fig.1.

N. Bassiliades et al. (eds.), *New Trends in Database and Information Systems II*,
Advances in Intelligent Systems and Computing 312, DOI: 10.1007/978-3-319-10518-5_25

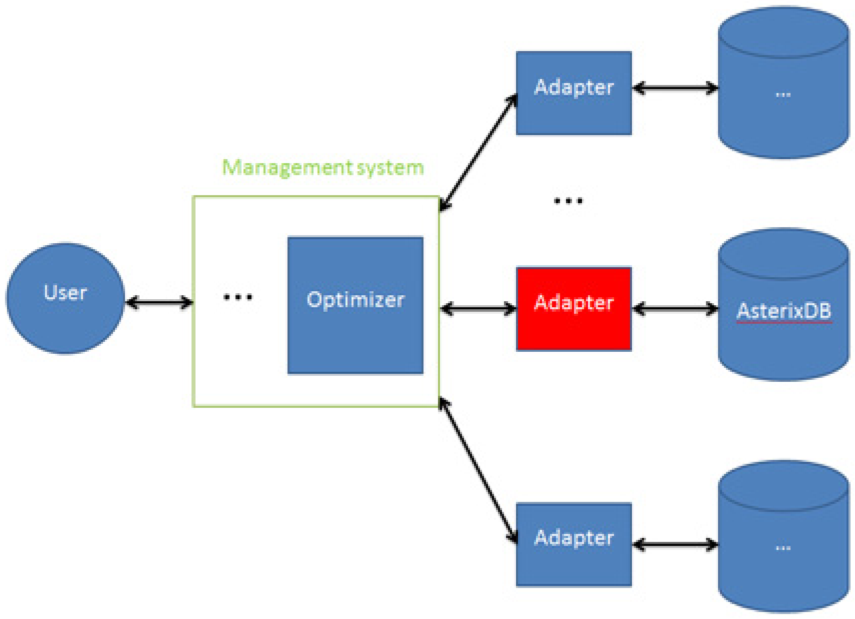


**Fig. 1.** Query execution flowchart

As you can see from pic.1 user has a common interface to query heterogeneous data sources. The optimizer is connected with all data sources through adapters. Every adapter is responsible for generating code for the query plan, computing additional information (scores, estimations, identities, attributes and etc.) for data for correct handling and computing cost model functions to find optimal plan. The main purpose of this work is to implement such adapter for a big data management system AsterixDB that is used as an execution platform.

In the next sections firstly the extended relational algebra from [1] will be described and some features of AsterixDB BDMS [2] that is used in this work will be provided. Next there will be a description of integrating AsterixDB to the optimizer and query code generation for generalized algebraic operations. After that, executed work with some examples and future work with plans and problems will be represented. And finally, related works and projects will be reviewed.

## 2 Extended Relational Algebra

The optimizer of heterogeneous system uses the generalized relational algebra introduced in [1]. This algebra exploits similarity as a matching tool. The central concept of it is a q-set defined as a triple (q, B, S) where:

- q is a query;
- B is a base set of objects for query processing;
- S is scoring function for objects in B.

Q-set encapsulates both the query and the result of its evaluation represented as scoring function. There is no matter how the query (q) will be represented in this model. But it is required for every object from the base set (B) to have an additional field with the score that can represent numerical value of object precision, precision of object's resource or other values. In general case an object score represents relevance of an object to a query. This scoring value can be used for fuzzy queries and an approximate result computing. The scoring function (S) is used to modify object scores and depends on a query.

All generalized algebraic operations from extended relational algebra are defined on q-sets. There are such operations as:

1. filter – some kind of select statement with a sequential evaluating scoring function on all objects of a base set;
2. joins;
3. aggregations;
4. fusion;
5. nest/unnest.

There some examples of such operations:

- `filter(restaurants, 'Italian')` - where a query finds Italian restaurants from base set 'restaurants' without modifying scores;
- `filter(objects, score_function='normalization')` - where object scores will be calibrated and normalized to make objects from different sources comparable;

-
  ```
  group_join (filter (shops, 'has spaghetti'),
       filter (restaurants, 'Italian'),
       'distance < 20 km')
  ```

  The informal query is to find shops with spaghetti located closely to Italian restaurants.

A full list of generalized algebraic operations can be found in [1]. It is important to note that operations can have parameters (algorithms, execution time frames and other) and can use remote execution.

## 3 AsterixDB BDMS

AsterixDB BDMS is a Big Data Management System (BDMS) with a semi structured noSQL style data model resulting from extending JSON with object database ideas and expressive and declarative query language (AQL) that supports a broad range of queries and analysis over semi structured data. A full description of this system is given in [5]. The main reasons to use it as execution platform are powerful query language which can be simply translated to algebraic operators represented in [1], using pipeline for computing query results without materialization and external data processing with support of different information formats.

## 4 Integrating AsterixDB to the Optimizer

One of the ideas of this work is using AsterixDB as an execution platform for optimizer. To integrate it we should provide an adapter to a data source that consists of a code generator (translator) and a cost model. Such module generates code on AsterixDB Query Language for algebraic operators, gives cost model for optimizer to build optimal execution plan and connects optimizer to AsterixDB instance to send queries and receive results.

The query optimizer constructs a query tree where algebraic operations are nodes and information resources are leaves and transforms it to optimize query plan. It does not only use algebraic operator's properties to transform tree, but it also can choose a better execution platform for computing result of some sub tree. That's why we need a cost model for all platforms. The cost model computes its execution time at chosen platform for every sub tree and the optimizer decides on what platform is better to execute sub tree. Also some sub tree that was executed at one processing platform can be treated as an information resource for another platform to handle nodes above this sub tree.

## 5 Query Translation

### 5.1 Q-Set Translation

The main idea of implementing generalized algebraic operations in AsterixDB BDMS is a translation from extended relational algebra operator's syntax to AQL. That's why we need a representation of q-set in AsterixDB BDMS.

The query from tripe (q,B,S) can be represented in any forms. From a practical point of view it can be some skeleton of a query on AsterixDB Query Language (AQL) with prepared fields for the base set (**B**) and the scoring function(**S**). It is important to note that the base set can be another generalized algebraic operation, so we can build queries as a chain of different operations.

AQL is designed according to XQuery – a query and functional programming language for structured and unstructured data.

For example, let's build simple query skeleton for q-set triple (q, B, S):

```
q = "for $item in B return S ($item)"
```

This query only modifies object scores according to the scoring function S. Almost all queries expressed by AQL use FLWOR expression (for, let, where, order by, return). That's why all algebraic operators are written with such expression.

### 5.2 Operation Translation

The query translator operates with query skeletons, base set and scoring function pointers. It performs no nontrivial optimization of generated query, because it will be done by an AsterixDB BDMS optimizer.

There is an example of translation from a generalized algebraic query to AQL:

Input query:

```
Group_join(
  filter(Employee, ‘age <30’),
  filter(Department, *),
  ‘department name’)
```

A query, translated to AQL:

```
for $dep in dataset Departments
  for $emp in dataset Employees
    where $dep.id=$emp.department_id
      and $emp.age <30
group by $department:=$dep.name with $emp
return {
  "department":$department,
  "name":for $t in $emp
  return $t.name
};
```

As you can see, an AQL code generation for algebraic operators is quite clear. The code generator can translate nested operators using FLWOR expressions for each operator.

## 6 Done Work

At the moment some algebraic operator’s translation with a cost model for them are implemented. All information about parts of done work can be found below.

### 6.1 Translator

The main part of an implementation of generalized algebraic operations is a translator. It receives a query sub tree, tranforms it to AQL, computes for optimizer an approximate execution time or sends the resulted query to AsterixDB BDMS for execution.

### 6.2 Q-Set

The q-set has already defined in terms of AsterixDB BDMS. Q-set is a triple (q, B, S) and according to section “5. Query translation”, ‘q’ is a query skeleton and both ‘B’ and ‘S’ are pointers to a data set and scoring function respectively. Also we need to represent scores for every object of data set. The simplest way to do it is to add a new field to every object with a score.

### 6.3 Filter Operator

Filter operation evaluates some predicate or expression and scoring function (or some predicate too) on every object from base set or another operation from extended algebra.

The general translation rule for such operation is specified below:

Input query:

```
filter(BaseSet, Expression(arguments),ScoringFunction);
```

A query, translated to AQL:

```
for $obj in BaseSet
  where Expression (arguments)
return ScoringFunction($obj);
```

As is shown above, "filter" expression evaluates in "where" section and a scoring function computes new scores for every object from base set.

### 6.4 Join Operator

Join operations can be represented in terms of AQL as nested cycles that iterate on the base sets with some join condition.

The general translation rule for precise inner join operation is specified below:

Input query:

```
join(FirstSet, SecondSet, Expression(arguments),
  ScoringFunction);
```

A query, translated to AQL:

```
for $first in FirstSet
  for $second in SecondSet
    where Expression (arguments)
return {
  $first,
  $second,
  ScoringFunction($first, $second)
};
```

As you can see, the query joins two sets of objects with some condition. The result of this operation is a set of new objects where every object consists of two joined objects from base sets with score that are the result of scoring function evaluation.

### 6.5 Cost Model

Some assumptions about AsterixDB BDMS data processing to design cost model were made. AstersixDB has shared-nothing architecture, that's why all of the data stored in it is splitted to all its nodes. $P$ is defined as a number of execution platform working nodes. Also one of the main constants that used by this model is $C_b$ – time to read one data block of base set no matter from disk or from another working node.

**Filter Operation.** Cost function for execution time ($t_f$) of filter operation has a linear complexity and takes in account a number of blocks to read $N_b$ for a base set $N$ ($|N|$ is cardinality of $N$), an execution time of filter expression and a scoring function $C_e$ on every object.

$$t_f = \frac{C_b N_b + C_e |N|}{P} \tag{1}$$

All working nodes use only their own data to compute result of filter operation without broadcasting to other nodes.

**Join Operation.** An assumption that AsterixDB BDMS use divide and broadcast-based parallel join algorithm [9] was made to define time execution cost function for a join operation. This algorithm is composed of two stages: a data partitioning using the “divide and broadcast” method and a local join. The “divide and broadcast” data partitioning method consists of a dividing one base set into multiple disjoint partitions, where each partition is allocated on a working node, and broadcasts the other base set of join operation to all available nodes. Because of AsterixDB has a shared-nothing architecture there is no need in a dividing stage and data blcks of smallest of two base sets should be broadcasted to all working nodes to use a local join.

Local join operation uses three different algorithms. That’s why three cost functions for execution time was designed. All of these functions have such parameters as base sets $N$ and $M$, numbers of base set data blocks $N_b$ and $M_b$, where $N_b$ is smaller or equal to $M_b$, and algorithm specific constant $C_e$.

*Hash join algorithm.* This algorithm has a linear complexity and requires an equijoin predicate. A cost function takes in account time to read base set data blocks and to evaluate hash join algorithm.

$$t_{hj} = \frac{C_b}{P}\big((P-1)N_b + M_b\big) + C_e\left(|N| + \frac{|M|}{P}\right) \tag{2}$$

*Sort-merge join algorithm.* This algorithm is based on sorting base sets and requires a transitive join predicate. A cost function takes in account time to read and sort base set data blocks and to evaluate sort-merge join algorithm.

$$t_{mj} = \frac{C_b}{P}\left(M_b \log\left(\frac{M_b}{P}\right) + N_b \log\left(\frac{N_b}{P}\right) + (P-1)N_b + M_b\right) + C_e\left(\frac{|N| log\left(\frac{|N|}{P}\right)}{P} + \frac{|M| \log\left(\frac{|M|}{P}\right)}{P} + |N| + \frac{|M|}{P}\right) \tag{3}$$

*Nested loop join algorithm.* This algorithm is based on nested loops and can be evaluated with every join predicate. A cost function takes in account a time to read base set data blocks in nested loops and to evaluate join predicate on every object.

$$t_{nj} = \frac{C_b(P-1)N_b M_b}{P^2} + \frac{C_e|N||M|}{P} \tag{4}$$

The first summand of all this functions is a time to read data blocks for local join where a broadcasting data blocks of smallest base set is taken in account and the second one is a time for join algorithm evaluation.

## 7 Future Work

### 7.1 Other Operations

First of all implementation of other generalized algebraic operations, such as group join, aggregation, fusion, nest/unnest and cost functions for them is planned.

### 7.2 Cost Model Quality Assessment

A quality of introduced cost model for execution time estimation needs to be checked on various data sets volumes and different number of AsterixDB working nodes.

### 7.3 Remote Execution

We want to have possibility to call remote function from our operations. That's why we should implement such option for all operations as a parameter. It is planned to use RPC for remote execution, because of AsterixDB BDMS provides such technology.

### 7.4 Approximate Operations

After implementation of exact algorithms for generalized algebraic operations we plan to implement approximate algorithms and add to all operations additional parameter that will specify algorithm for execution.

## 8 Related Work

Querying heterogeneous distributed data is a very important field of informational retrieval. There are a lot of projects and products that deal with handling heterogeneous data sources. Some of them implement an idea of federated databases. Several commercial products are represented in [11, 12, and 14]. All of them combine information from different data sources into a federated database with possibility for complex querying. Oracle Database [11] offers to users an environment for handling data from different sources, such as XML documents, DB2, Informix, MS SQL, Teradata and etc. The Garlic project idea is represented in [13]. This system integrates various data sources to build complex queries.

Optiq [10] is a framework over Apache Hive or Apache Drill for processing heterogeneous and federated data. It based on translation from SQL-based query language to platform-oriented languages. Optiq uses adapters to connect to data sources and provides user unified interface to query it. This framework translates and optimizes queries according to rules and cost estimations.

Different types of extended relational algebras are represented in [1, 3 and 4]. Implementation of generalized algebraic operations from [1] that can be used by heterogeneous system optimizer is considered at this work. There are such operations as projections, joins, aggregations, nest and unnest operators and other. Authors introduce extended relational algebra with a uniform approach to formal representation, optimization, specification and execution of complex queries. A generalization of the classical algebra by introducing fuzzy operations and weights is proposed in [4]. Authors define a declarative query language, which brings together traditional relational domain calculus and handling of fuzzy values. This algebraic framework contains join, product, projection, selection, union, intersection and difference operation. The query language also contains operations for weighting similarity predicates. A similarity algebra based on relational operations on object lists with scores is introduced in [3]. Complex queries with different interpretations of similarity values and algorithms for computing these values can be specified in this algebra. The algebra supports joins, merge, select, map operations, union, intersection, and difference operations.

An optimizer needs a cost model to find optimal query plan. Several approaches to design cost model for extended relational algebraic operators are represented in [3, 6 and 8]. Cost models that use estimation of operation cardinality and selectivity are represented in [3]. Authors of [6] introduce fuzzy cost model for query optimization in multi-database systems based on experience, knowledge, tests and heuristics of the required parameters of optimization. Designing cost models for parallel join algorithms are considered in [8]. Author builds cost models for different parallel join algorithms and compare them with each other.

## 9 Conclusions

A system for processing heterogeneous data sources based on extended relational algebra was represented. An approach to support AsterixDB BDMS as data source at this system was introduced. It consisted of developing a code generator for translating a query plan, which is expressed in terms of algebraic operations to AsterixDB query language and a cost model for query evaluation time prediction.

## References

1. Novikov, B., Vassilieva, A., Yarygina, A.: Querying Big Data. In: Proceedings of the 13th International Conference of Computer Systems and Technologies, CompSysTech 2012, pp. 1–10 (2012)
2. AsterixDBBDMS Web site, `http://asterix.ics.uci.edu/`

3. Fagin, R.: Fuzzy queries in multimedia database systems. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1998, pp. 1–10. ACM, New York (1998)
4. Deshpande, A., Ives, Z.G., Raman, V.: Adaptive query processing. Foundations and Trends in Databases 1(1), 1–140 (2007)
5. Borkar, V.R., Carey, M.J., Li, C.: Inside "Big Data Management": Ogres, Onions, or Parfaits? In: EDBT 2012 (2012)
6. Zhu, Q., Larson, P.-A.: Establishing a fuzzy cost model for query optimization in a multidatabase system. In: Proc. of 27th ACM/IEEE Hawaii Int'l Conf. on Syst. Sci. (February 1994)
7. Ioannidis, Y.E.: Query optimization. ACM Comput. Surv. 28(1), 121–123 (1996)
8. Pigul, A.: Generalized parallel join algorithms and designing cost models. In: Proceedings of the Eighth Spring Researchers Colloquium on Database and Information Systems, SYRCoDIS 2012, pp. 29–40 (2012)
9. Taniar, D., Leung, C.H.C., Rahayu, W., Goel, S.: High-performance parallel database processing and grid databases. A John Wiley & Sons, Inc., Publication, New Jersey (2008)
10. Optiq framework Web site, https://wiki.apache.org/incubator/OptiqProposal
11. Oracle Database Web site, http://www.oracle.com/ru/products/database/overview/index.html
12. IBM Garlic Project Web site, http://www.research.ibm.com/topics/popups/deep/manage/html/garlic.html
13. Tork Roth, M., Arya, M., Haas, L., Carey, M., Cody, W., Fagin, R., Schwarz, P., Thomas, J., Wimmers, E.: The Garlic project. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, SIGMOD 1996, p. 557 (1996)
14. Composite Information Server Web site, http://www.compositesw.com/products-services/information-server/

# Data Warehouse Schema Evolution Perspectives

Danijela Subotić

University of Rijeka, Department of Informatics, Rijeka, Croatia
dsubotic@inf.uniri.hr

**Abstract.** The paper presents a short analysis of research related to the problem of the data warehouse (DW) evolution. The main contributions of the paper are: a) analysis of existing methods and approaches to the DW schema evolution, and b) characterization of the general research idea for the DW schema evolution problem. The general research idea includes a meta-Data Vault (DV) model that will integrate the DW with the master data management (MDM) system. We believe the following issues could be resolved: a) tracking the origin of data, b) tracking the history of changes, c) avoiding loss of data, d) faster and simpler migration and transformation of data, and e) trend projections. Also, due to long-term storage of historical data and tracking the origin of data, the DW could be used as a complete system of records and as the basis for the data governance (MDM integrated with the DW).

**Keywords:** data warehouse evolution, schema evolution, schema versioning, view maintenance, master data management, data vault.

## 1 Introduction

A data warehouse (DW) integrates a number of heterogeneous data sources and provides for a quick and efficient analysis of business needs. However, today's data sources often change their content and structure, which greatly affects the DW - it should contain the latest data to be able to reflect the evolving state of the real world and our evolving understanding of it. Because of this, it is necessary to properly manage all types of changes and appropriately update the DW. With respect to the literature, the DW evolution research can be grouped into research on managing the data and schema changes in the data warehouse, managing the data changes in the data mart and managing the schema changes in the data mart. However, we will observe the DW schema evolution more broadly, through three approaches - schema evolution, schema versioning and view maintenance. The aim of this paper is to present a short analysis of research related to the problem of the DW schema evolution and to briefly describe our general idea for the future research. The paper is organized as follows; in section 2, research related to the DW evolution is analyzed; in section 3, a characterization of the general research idea is presented; and section 4 contains the conclusion and directions for future work.

## 2 Analysis of Related Work

We will mainly observe schema evolution and schema versioning approaches in this paper, as they are the most relevant approaches for our future research. More detailed state of the art will not be presented here, as this is only a short paper. The process of schema evolution [1,2,3,4] and versioning [5,6,7,8,9,10,11] is still demanding in terms of invested time and resources. Perhaps the biggest problem here is the preservation of schema consistency and data integrity (there is still a lack of an integrated system-of-records). Also, migration and transformation of data is still slow and expensive, the loss of data during these processes is still present and there is a lack of effective integration, organization and management of metadata. Although the academic community has made steps towards solving these problems, there is still room for improvement, as well as for defining general solutions and fully effective commercial solutions. Different approaches to solving the DW schema evolution problem are presented in literature, but there is still no widely accepted solution for managing DW schema changes. We believe it is necessary to find a simple and complete solution for preserving schema consistency and data integrity. All this issues will serve as requirements for a new approach in which we will try to make a departure from previous research and try to find a new perspective and solution to the DW schema evolution problem.

## 3 General Research Idea

We observe the problem of the schema evolution as a double issue or a dual problem- from the DW perspective, and from the master data management (MDM) perspective. MDM represents the master and reference data (golden copy) and related metadata, set of policies, governance, standards, processes and tools that define and manage the master and reference data (of an organization) to provide a single point of reference, with the purpose of increasing data and business analysis quality. From the DW perspective every fact that is associated with dimensions is observed and star schemas are standard representation used for visualization. From the MDM perspective every master entity (dimension) that is associated with events (facts) is observed and an inverse star-like schema can be used for visualization [16]. In the case of MDM, as in a DW case, there is the problem of schema evolution after changes in the data sources or user requirements, and as such we will address it together. Our general research idea includes a Data Vault meta-model based modeling approach [12,13] that will integrate the DW with the MDM metadata in a common model to serve as an extension of a generic DBMS catalog. The relational model is largely responsible for physical data independence, but we can conclude that it is not convenient to simply and effectively support the evolution of the logical structure of the DB schema. Data Vault (DV) is a data modeling method that supports design of data warehouses for long-term storage of historical data collected from various data sources. Its main advantage is the separation of the structural data from descriptive attributes, which makes the model flexible to changes in business environment. Also, it highlights the need for tracking the origin of data contained in the database and the history of changes, through empirically defined set of metadata (record_source, load_datetimestamp). Furthermore, any change is implemented in the model as an independent extension of the existing model, which means that the changes do not affect current applications and all versions of the model are a subset of the DV model.

With a development and implementation of our research idea we expect that the following issues will be resolved (mainly through the use of a DV modeling method): a) tracking the origin of data, b) tracking the history of changes, c) avoiding the loss of data, d) slow and expensive migration and transformation of data, and e) trend projections. Due to the long-term storage of historical data and tracking the origin of data, the preservation of DW integrity would be facilitated and the DW would then contain both a "single version of the fact" [12] and a "single version of the truth" [14]. Also, it could be used as a complete and integrated dual solution and system of records [15] with the support for data sources evolution, user requirements evolution and data security evolution.

## 3.1 Characterization of the General Research Idea

Figure 1 shows a diagram of the proposed DW architecture. The proposed architecture consists of four parts: a) data sources, b) enterprise DW, c) reporting DW and d) user analysis. Data sources are usually distinguished (in the literature and practice) by their place of origin and maintenance. Accordingly, we make a distinction between internal and external reference data (which is obtained from internal or external data sources). Reporting DW (RDW) consists of a derivative (summarized, aggregated and computed) data stored in materialized or virtual DM. User analysis is the user side of the system architecture

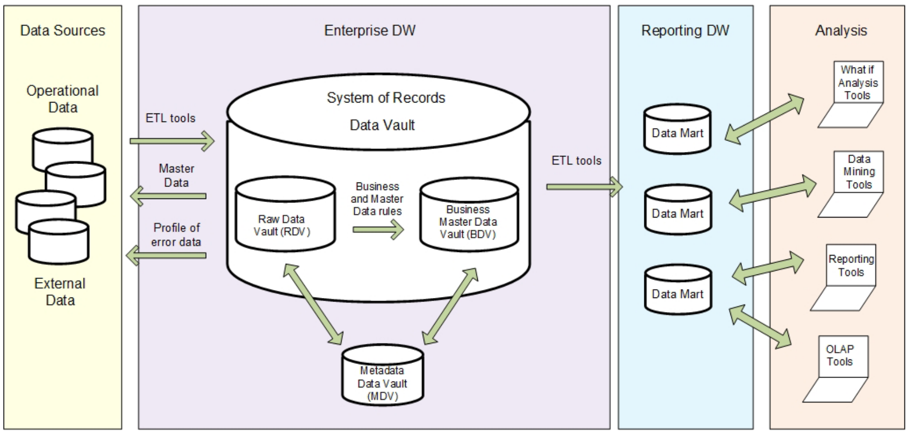


**Fig. 1.** Suggested Data Warehouse Architecture Diagram

where the tools for analysis and reporting are located. With the help of these tools the user is directly accessing the RDW. Our research will mainly focus on the Enterprise DW part of the proposed architecture. Enterprise DW (EDW) includes the raw data vault (RDV) and business master data vault (BDV), which are integrated via metadata data vault (MDV). The RDV is oriented toward data sources. With the help of ETL tools and processes data is extracted and loaded into the RDV. The RDV contains actual copies of the originals from the data sources. Once the data is entered into the RDV, it is no longer deleted (all changes are implemented through additions-only). This means that copies of the originals are kept permanently in RDV. With this persistent preservation of history the loss of data is avoided and a solid basis for the audit process is provided. The BDV is created by upgrading the RDV with the application of standardized master data and business rules, for the purpose of business integration. The RDV and the BDV are shown as separate systems in Figure 1, but that is only a logical representation. Physically it is possible to implement them individually or as a single system. Our EDW will consist of a single DV model which is partially oriented towards the data sources side (RDV), and partially oriented towards the MDM and reporting side (BDV). Also, because they are now physically separated, we can distinguish reversible (light) and irreversible (heavy) transformations [15]. Reversible transformations are used for loading data from a data source into the RDV, and it is possible to reverse their effects, in order to obtain the system-of records. They allow RDV to reach the exact copies of the original from the data source. Irreversible transformations are mainly based on the business and master rules and are usually irreversible. In this case, both the transformations and the original data must be preserved in order to trace exits back to the source and reconstruct them if necessary. The proposed architecture moves irreversible transformations downstream - after RDV, towards BDV and reporting DW (they are loading materialized DMs). Because of those two types of separation (raw/business data and light/heavy transformations), we can target only the needed set of data at a given time which will reduce the time and the cost of data migration and transformation processes, and we can preserve the whole history of changes (of the schema and the data) in the EDW. This is the key idea for getting an integrated EDW system of records which will also serve as the basis for the data governance [15,18]. Finally, MDV is a key component of the proposed architecture, which serves to integrate the RDV and BDV in the EDW system of records (we can say that MDV is the DW on the DW). MDV logical model is based on the DV model. The aim of our research is to define and formalize the MDV model and a final set of change cases for the DW and MDM schema evolution scenarios (DW and MDM – dual solution). We have a preliminary, true working draft version of the MDV for now, but it will not be shown here. However MDV will keep historicized hubs, links, satellites, attributes, domains and reference tables as a hub in a DV model, which will serve as a mechanism for monitoring the data sources evolution. We also plan to resolve (at the meta-level) the transformation from a source into the RDV (via the extraction rules and light transformations), the transformation from the

RDV into the BDV (via the business and master rules and heavy transformations) and finally, the transformation from the integrated RDV/BDV into the DM (also via the business rules and heavy transformations). This way MDV will provide a mechanism for monitoring the user requirements evolution. Furthermore on a meta-level, the security aspect will be resolved, i.e. the user's access rights will be historicized and managed so a mechanism for monitoring the data security evolution will be provided.

## 4 Conclusion

The paper presents a short analysis of previous DW schema evolution research and characterization of our general research idea. The DW evolution process is still quite complex, error prone and requires a lot of time and resources. There is still room for research and improvement regarding the process of: a) transformation and migration of a DB on a new schema (system overload and system downtime are still present, and these processes are still slow and expensive), b) preservation of information during migration (the data is often lost, which affects schema consistency and data integrity), c) rewriting queries and applications in order to run on the new schema (current solutions have high maintenance cost) and d) effective integration, organization and management of metadata. Also the lack of defined mechanism for monitoring the data source model evolution, the lack of support for model relativism and the lack of support for the data security evolution can be noticed. Furthermore, only few of the approaches aim to solve all these DW evolution problems together - the majority focus on just one aspect. We believe that the problem of the DW and MDM schema evolution must be addressed at the general level and that all these problems must be dealt with together - as a dual solution. We are not resolving the problem of the DW and the MDM schema evolution "on the fly" at the operational level (where it occurs), but suggest a permanent general solution situated on a higher (meta) level using the DV model. A key component here is MDV (metadata repository model - metadata data vault) which in this context can be observed as a DW on the DW. MDV will serve to integrate raw and persistent DW with the business aligned MDM in order to obtain one consolidated EDW system of records. We expect the end result to be a flexible, modular, general solution which will track and manage changes in data and metadata, as well as their schemas. We just started our research, after completing state of the art review, and are working on a meta-Data Vault model and general requirements for desired dual solution model. Among directions of ongoing and/or future research are development and formalization of a final set of evolution change cases for the proposed architecture, formalization of a data vault based metadata catalog and incremental development of an Meta-Data Vault implementation prototype with various aspects included (source, rules/transformation, materialized view for DMs, and security) for an empirical evaluation of a proposed solution, as well as an experimental benchmark test based on a completed case study model.

**Acknowledgements.** This paper is based upon work supported by the University of Rijeka under project titled "*Metode i modeli za dizajn i evoluciju skladita podataka*".

## References

1. Hurtado, C.A., Mendelzon, A.O., Vaisman, A.: Maintaining Data Cubes under Dimension Updates. In: Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia, pp. 346–355 (1999)
2. Blaschka, M., Sapia, C., Höfling, G.: On Schema Evolution in Multidimensional Databases. In: Mohania, M., Tjoa, A.M. (eds.) DaWaK 1999. LNCS, vol. 1676, pp. 153–164. Springer, Heidelberg (1999)
3. Marotta, A., Ruggia, R.: Data Warehouse Design: A Schema-Transformation Approach. In: 22nd International Conference of the Chilean Computer Science Society (SCCC), Copiapo, Chile (2002)
4. Fan, H., Poulovassilis, A.: Schema Evolution in Data Warehousing Environments – A Schema Transformation-based Approach. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 639–653. Springer, Heidelberg (2004)
5. Eder, J., Koncilla, C.: Evolution of Dimension Data in Temporal Data Warehouses. Technical Report (2000)
6. Body, M., Miquel, M., Bedard, Y., Tchounikine, A.: A Multidimensional and Multiversion Structure for OLAP Applications. In: 5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2002), McLean, Virginia, USA, pp. 1–6. ACM Press (2002)
7. Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: Schema Versioning in Data Warehouses. In: Wang, S., et al. (eds.) ER Workshops 2004. LNCS, vol. 3289, pp. 415–428. Springer, Heidelberg (2004)
8. Bebel, B., Eder, J., Koncilia, C., Morzy, T., Wrembel, R.: Creation and Management of Versions in Multiversion Data Warehouse. In: 19th ACM Symposium on Applied Computing (SAC 2004), Nicosia, Cyprus, pp. 717–723. ACM Press (2004)
9. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: What-if Analysis for Data Warehouse Evolution. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 23–33. Springer, Heidelberg (2007)
10. Solodovnikova, D.: Data Warehouse Evolution Framework. In: Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems, SYRCoDIS, Moscow, Russia (2007)
11. Malinowski, E., Zimányi, E.: A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models. Data & Knowledge Engineering 64, 101–133 (2008)
12. Linstedt, D.: SuperCharge Your Data Watehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace Independent Publishing Platform, USA (2011)
13. Jovanovic, V., Bojiéic, I.: Conceptual Data Vault Model. In: Proceedings of the Southern Association for Information Systems Conference, Atlanta, USA (2012)
14. Inmon, W.H., Strauss, D., Neushloss, G.: DW 2.0: The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann Publishers, Burlington (2008)
15. Jovanovic, V., Bojiéic, I., Knowles, C., Pavlic, M.: Persistent Staging Area Models For Data Warehouses. Issues in Information Systems 13(1), 121–132 (2012)
16. Berson, A., Dubbov, L.: Master Data management and Data Governance, 2nd edn. McGraw Hill (2011)

# Author Index