DBKDA 2015

The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications

ISBN: 978-1-61208-408-4

GraphSM 2015

The Second International Workshop on Large-scale Graph Storage and Management

May 24 - 29, 2015

Rome, Italy

DBKDA 2015 Editors

Friedrich Laux, Reutlingen University, Germany

Andreas Schmidt, Karlsruhe University of Applied Sciences | Karlsruhe Institute of Technology, Germany

Maria Del Pilar Angeles, Universidad Nacional Autonoma de Mexico - Del Coyoacan, Mexico

Kiyoshi Nitta, Yahoo Japan Research, Japan

Iztok Savnik, Jozef Stefan Institute and University of Primorska, Slovenia

DBKDA 2015

Forward

The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2015), held between May 24-29, 2015 in Rome, Italy, continued a series of international events covering a large spectrum of topics related to advances in fundamentals on databases, evolution of relation between databases and other domains, data base technologies and content processing, as well as specifics in applications domains databases.

Advances in different technologies and domains related to databases triggered substantial improvements for content processing, information indexing, and data, process and knowledge mining. The push came from Web services, artificial intelligence, and agent technologies, as well as from the generalization of the XML adoption.

High-speed communications and computations, large storage capacities, and load-balancing for distributed databases access allow new approaches for content processing with incomplete patterns, advanced ranking algorithms and advanced indexing methods.

Evolution on e-business, e-health and telemedicine, bioinformatics, finance and marketing, geographical positioning systems put pressure on database communities to push the 'de facto' methods to support new requirements in terms of scalability, privacy, performance, indexing, and heterogeneity of both content and technology.

The conference had the following tracks:

- Data query, access, mining, and correlation
- Databases and other domains
- Databases technologies
- Data management
- Knowledge and decision bases
- Databases content processing

The conference also featured the following workshop:

- **GraphSM 2015*, The Second International Workshop on Large-scale Graph Storage and Management***

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the DBKDA 2015 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to

DBKDA 2015. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the DBKDA 2015 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope DBKDA 2015 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of databases, knowledge, and data applications. We also hope that Rome, Italy provided a pleasant environment during the conference and everyone saved some time to enjoy the historic beauty of the city.

DBKDA 2015 Chairs

DBKDA Advisory Chairs

Friedrich Laux, Reutlingen University, Germany
Aris M. Ouksel, The University of Illinois at Chicago, USA
Serge Miranda, Université de Nice Sophia Antipolis, France
Andreas Schmidt, Karlsruhe University of Applied Sciences | Karlsruhe Institute of Technology, Germany
Maribel Yasmina Santos, University of Minho, Portugal
Filip Zavoral, Charles University Prague, Czech Republic
Maria Del Pilar Angeles, Universidad Nacional Autonoma de Mexico - Del Coyoacan, Mexico

GraphSM Chairs

Kiyoshi Nitta, Yahoo Japan Research, Japan
Iztok Savnik, Jozef Stefan Institute and University of Primorska, Slovenia

DBKDA 2015

Committee

DBKDA 2015 Advisory Chairs

Friedrich Laux, Reutlingen University, Germany
Aris M. Ouksel, The University of Illinois at Chicago, USA
Serge Miranda, Université de Nice Sophia Antipolis, France
Andreas Schmidt, Karlsruhe University of Applied Sciences | Karlsruhe Institute of Technology, Germany
Maribel Yasmina Santos, University of Minho, Portugal
Filip Zavoral, Charles University Prague, Czech Republic
Maria Del Pilar Angeles, Universidad Nacional Autonoma de Mexico - Del Coyoacan, Mexico

DBKDA 2015 Technical Program Committee

Nipun Agarwal, Oracle Corporation, USA
Suad Alagic, University of Southern Maine, USA
Toshiyuki Amagasa, University of Tsukuba, Japan
Fabrizio Angiulli, University of Calabria, Italy
Masayoshi Aritsugi, Kumamoto University, Japan
Zeyar Aung, Masdar Institute of Science and Technology - Abu Dhabi, United Arab Emirates
Ana Azevedo, Algoritmi R&D Center/University of Minho & Polytechnic Institute of Porto/ISCAP, Portugal
Gilbert Babin, HEC Montréal, Canada
Orlando Belo, University of Minho, Portugal
Fadila Bentayeb, University of Lyon 2, France
Arnab Bhattacharya, IIT Kanpur, India
Erik Buchmann, Karlsruhe Institute of Technology (KIT), Germany
Martine Cadot, LORIA-Nancy, France
Ricardo Campos, Polytechnic Institute of Tomar / LIAAD - INESCT TEC Porto, Portugal
Michelangelo Ceci, University of Bari, Italy
Chin-Chen Chang, Feng Chia University Taiwan, Taiwan
Chi-Hua Chen, National Chiao Tung University - Taiwan, R.O.C.
Qiming Chen, HP Labs - Palo Alto, USA
Ding-Yuan Cheng, National Chiao Tung University, Taiwan , R.O.C.
Camelia Constantin, UPMC, France
Gabriel David, University of Porto, Portugal
Maria Del Pilar Angeles, Universidad Nacional Autonoma de Mexico - Del Coyoacan, Mexico
Taoufiq Dkaki, IRIT - Toulouse, France

Michele Melchiori, Università degli Studi di Brescia, Italy
Ernestina Menasalvas, Universidad Politécnica de Madrid, Spain
Elisabeth Métais, CEDRIC / CNAM - Paris, France
Cristian Mihaescu, University of Craiova, Romania
Serge Miranda, Université de Nice Sophia Antipolis, France
Mehran Misaghi, Educational Society of Santa Catarina – Joinville, Brazil
Mohamed Mkaouar, Sfax, Tunisia
Jacky Montmain, LGI2P - Ecole des Mines d'Alès, France
Yasuhiko Morimoto, Hiroshima University, Japan
Bela Mutschler, Ravensburg-Weingarten University of Applied Sciences, Germany
Franco Maria Nardini, ISTI-CNR, Italy
Khaled Nagi, Alexandria University, Egypt
Aris M. Ouksel, The University of Illinois at Chicago, USA
George Papastefanatos, Athena Research and Innovation Center, Greece
Alexander Pastwa, Ruhr-Universität Bochum, Germany
Dhaval Patel, IIT-Roorkee, Singapore
Przemyslaw Pawluk, York University - Toronto, Canada
Alexander Peter, AOL Data Warehouse, USA
Alain Pirotte, University of Louvain (Louvain-la-Neuve), Belgium
Pascal Poncelet, LIRMM, France
Ricardo Queirós, ESEIG-IPP, Portugal
Mandar Rahurkar, Yahoo! Labs, USA
Praveen R. Rao, University of Missouri-Kansas City, USA
Peter Revesz, University of Nebraska-Lincoln, USA
Mathieu Roche, TETIS, Cirad, France
Satya Sahoo, Case Western Reserve University, USA
Fatiha Saïs, LRI (CNRS & Paris Sud University), France
Abhishek Sanwaliya, Indian Institute of Technology - Kanpur, India
Ismael Sanz, Universitat Jaume I - Castelló, Spain
M. Saravanan, Ericsson India Pvt. Ltd -Tamil Nadu, India
Idrissa Sarr, Université Cheikh Anta Diop, Senegal
Najla Sassi Jaziri, ISSAT Mahdia, Tunisia
Andreas Schmidt, Karlsruhe University of Applied Sciences | Karlsruhe Institute of Technology, Germany
Yong Shi, Kennesaw State University, USA
Damires Souza, Federal Institute of Education, Science and Technology of Paraiba (IFPB), Brazil
Lubomir Stanchev, University-Purdue University - Fort Wayne, USA
Ahmad Taleb, Najran University, Saudi Arabia
Maguelonne Teisseire, Irstea - UMR TETIS, France
Martin Theobald, Max Planck Institute for Informatics, Germany
Gabriele Tolomei, CNR, Italy
Jose Torres-Jimenez, CINVESTAV 3C, Mexico
Nicolas Travers, CNAM-Paris, France
Thomas Triplet, Computer Research Institute of Montreal (CRIM), Canada

Marian Vajtersic, University of Salzburg, Austria
Genoveva Vargas, Solar | CNRS | LIG-LAFMIA, France
Fan Wang, Microsoft Corporation - Bellevue, USA
Zhihui Wang, Dalian University of Technology, China
Guandong Xu, Victoria University, Australia
Maribel Yasmina Santos, University of Minho, Portugal
Jin Soung Yoo, Indiana University-Purdue University - Fort Wayne, USA
Feng Yu, Youngstown State University, USA
Filip Zavoral , Charles University Prague, Czech Republic
Wei Zhang, Amazon.com, USA

GraphSM 2015 Chairs

Kiyoshi Nitta, Yahoo Japan Research, Japan
Iztok Savnik, Jozef Stefan Institute and University of Primorska, Slovenia

GraphSM 2015 Technical Program Committee

Khaled Ammar, University of Waterloo, Canada
Blaz Fortuna, Jozef Stefan Institute, Slovenia
Dimitar Hristovski, University of Ljubljana, Slovenia
Yasunori Ishihara, Osaka University, Japan
Dejan Lavbic, University of Ljubljana, Slovenia
Yanhua Li, Huawei Noah's Ark Lab, Hong Kong
Khaled Nagi, Alexandria University, Egypt
Kiyoshi Nitta, Yahoo Japan Research, Japan
Elena Ravve, Ort-Braude College - Karmiel, Israel
Iztok Savnik, Jozef Stefan Institute and University of Primorska, Slovenia
Andreas Schmidt, Karlsruhe University of Applied Sciences & Karlsruhe Institute of Technology, Germany
Jim Webber, Neo Technology, USA
Tatjana Wezler, University of Maribor, Slovenia
Kokou Yetongnon, Université de Bourgogne, France

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

An Industry Experience on Dynamic Handling of Multiple Database Types for Data Integration Tool

Badrul Affandy Bin Ahmad Latfi, Anbarasan Kodandaraman, Lee Chee Kiam, and Thong Tong Khin
Software Development Lab
MIMOS Bhd
Kuala Lumpur, Malaysia
e-mails: {badrul.affandy, anbarasan.raman, ck.lee, thong.tkin}@mimos.my

Abstract— **This paper presents the industry experience of implementing a Structured Query Language (SQL) Query Builder component and its interface supporting multiple database types in web-based Data Integration Tool (DIT), which performs Extract-Transform-Load (ETL) and Data Cleansing. Due to the dynamic nature of database connectivity for different types of source and target tables, the usage of direct SQL statement is preferred over Object Relationship Management (ORM). The common problem while handling multiple different databases is that it requires database specific queries for SQL functions, like create table, insert data or query specific range of rows for each type of database. Hence, we developed the SQL Query Builder (QB), which generates specific SQL statements for specific syntax. By standardizing the usage of delimited identifier with American National Standards Institute (ANSI) mode, and schema hierarchy, the QB's common interface will enable to dynamically handle multiple types of databases. With the QB, we created ETL and data cleansing applications compatible with multiple database types.**

Keywords-Data integration; SQL; multiple database type.

I. INTRODUCTION

A metadata-intensive application, such as data integration that performs Extract-Transform-Load (ETL) and data cleansing works on low level Application Programming Interface (API) to produce actions on source and target database. Logical data mapping needs to be done at early stage of ETL to ensure ETL produce quality data [1]. ETL operation could be done through effective usage of SQL to map and transform data [2]. Manual tables mapping with the help of user interface is a slow process [3]; thus, we use the automated mapping of source and target tables with additional supports of multiple database types. The data integration tool that we developed is based on server centric solution, which can be deployed on powerful servers.

We developed a SQL Query Builder component, also referred as Query Builder (QB) in this document, for supporting multiple database connections, providing services to source and target database, and the system resource itself. QB has three benefits: (1) creates SQL based on source/target tables, (2) eases of applying system identification to manage source and target tables, and (3) enforces delimited identifiers that are generic for all database types. The output of QB is SQL statement produced specifically for source or target table based on database type. QB is the main component of Data Integration Tool (DIT). Since the DIT is a web-based solution, whereby table entities classes are loaded during system startup, it is essential for the DIT to execute SQL query to multiple database types dynamically, which cannot be performed by traditional solution, such as Object Relational Management (ORM).

Section II elaborates problems encountered and solutions. Section III covers the QB architecture and design. Section IV presents the results, and finally, Section V concludes QB benefit and future enhancement.

II. PROBLEMS ENCOUNTERED AND SOLUTIONS

A. Database Mapping

Systems or platforms that require accessing multiple database types had been presences many years back. These systems connect database and access its tables through Java connector with configured object mappings via ORM programming tool. ORM is a powerful tool that allows Java objects interacts with relational database. The object/relational metadata are configured prior to Java object instantiations. The object instantiation for all mappings normally happens during application startup, after that application could perform standard operations like create, update, delete and read. If the XML configuration of the ORM is not correct, exceptions occur and the application loading is aborted. Open-source tools, such as Hibernate [4], MyBatis [5], provide object relational mapping with database tables.

For ETL, the source and target databases could be of different database types. The source and target databases are configured via a configuration screen. User may add or remove the configuration as required. User may link up source and target databases to perform ETL process. Once tables are connected, ETL processes extract, transform and then load data into data warehouse. The data warehouse is the primary environment of ETL process. ORM is not fitting the requirement due to the tool need to dynamically connect to multiple data sources (with any number of columns of different type) and data structure during runtime. We use database specific SQL over ORM. Each ETL process would

 ISBN: 978-1-61208-408-4

require a set of dynamic SQL scripts to be generated during runtime for its operations (table creation, alteration and selection).

B. Data Integration Tools

There are a few open source data integration tools available in the IT industry, such as Pentaho Data Integration (Kettle) [6] and Talend Open Studio for Data Integration (TOS) [7]. These tools run job flow on client desktop or on remote server through deployment. The job flow typically bounds specific database tables, so that each ETL job is unique. In our application, an ETL process runs on generic ETL job. A new ETL process with database configurations could be added without application redeployment. Kettle and TOS jobs perform stream-based process, which is per record basis. Our application could not make use these open source tool platforms because it is required to runs multi-threaded jobs with bulk data stored in multiple database types. It is also required to connect with Graphic Processing Unit (GPU) system for data cleansing operation [8]. Our application also has other proprietary data cleansing features, which runs in web-based application.

C. Database syntax

There are many commercial RDBMS in the market where we use three types, namely, Oracle, MySQL and PostgreSQL. We use SQL language to connect with these database types be it at data processing level or user interface level. Problems encountered by the development team of DIT are listed below. Other problems, such as built-in function and dialect, are not considered as they are out of scope.

- **Quoted Identifier**. Many sources recommended avoiding quoted identifier, as it destroys portability of the code and invites poorly constructed names [9]. However, this may not be true if we were to extract data from multiple database types. Using quoted identifier with ANSI [10] setting would improve code maintainability across different database vendors [11][12][13].
- **Schema hierarchy**. The schema name is always supplied in order to avoid confusion especially in PostgreSQL. This problem is re-solved using quoted identifier for schema, so that it works across multiple database types.
- **Record number and limit**. Record limit built-in function normally receive two arguments, namely, start number and number of records. According to MYSQL manual, the LIMIT command in MYSQL limits the return row [14]. For PostgreSQL, the offset value specifies the number of row to skip. For Oracle, it has top N-query processing with ROWNUM [15]. The query has inline view of the target table with full query. The returned records are bigger than start number but less than record number. The next batch of records will be received by incrementing the start and record numbers.
- **Upsert**. Upsert is a function to update and insert data (if data not found in target table). There is no standard command of upsert among multiple database types. Therefore, we develop small SQL functions to build an upsert command for each type of database.
- **Type casting** for Numeric Type. Usually, the data types are same across database types. But, in PostgreSQL, NUMERIC type can save float or integer, this data type has to be considered carefully during data conversion, otherwise SQL exception will occur. Casting numeric data type to float re-solves the issue.
- **Auto increment**. Creation of auto increment constraint in Oracle is not straight forward as in MySQL (AUTO INCREMENT) and PostgreSQL (BIGSERIAL). We have to create a new sequence with id and other details like start and increment value. A trigger with unique id is also created and the "NEXTVAL" of sequence is called on the column where the value has to be incremented.
- **Connection pooling on dynamic connections**. In an ETL application, it is essential to connect to multiple databases and tables in a dynamic way. Since connection pooling is used, pooled connections have to be reused for connecting to the same data source. As we are using DBCP of Apache [16], the pooled connections for the dynamically created BasicDataSource have to be reused to avoid connection leak or to open too many connections. Hence, we need to pool the Apache's DBCP connections into a map. We can make sure that only one connection pool is created for one data source and is pooled in our map. When requested for an existing connection, the map holding the object of the current data source is returned.
- **Auto generates column value in Oracle.** There might be several instances where the auto generated id upon inserting a row into table is required. The prepared statement or statement has the facility to retrieve the generated value. After setting the values, the statement is executed and the result set hold the generated values. For Oracle, since the auto generated values are achieved by using sequence and trigger, the column which holds the auto-generated key is not considered as generated column and hence, it is not returned. It holds the ROWID of the inserted row. This ROWID is used to retrieve the generated key by a separate call, as shown in Figure 1.

```
StringBuilder que = new StringBuild-
er();
tempStat= conn.createStatement();
que.append("SELECT generatedColumn-
Name FROM "+ tablename+("
    WHERE rowid ='"+rowValue+"'");
```

Figure 1. Auto generate key for Oracle

Figure 1 illustrates way of creating a SQL query string to get the auto generate key for Oracle. Executing the SQL query string with Oracle database generated a result set and from the result set, the generated key value is obtained.

III. ARCHITECTURE

A. *Component architecture*

The component architecture for DIT is displayed in Figure 2.

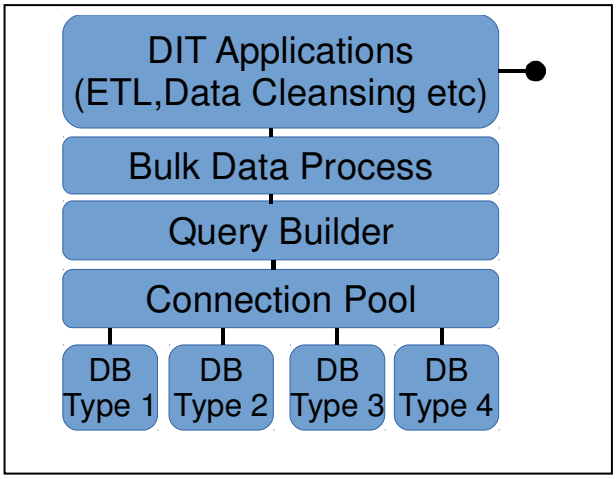


Figure 2. DIT Component Architecture

The application establishes connection with source and target tables through connection pool. For source and target tables, the database connection pool is used for each type of database. The lookup cache for connection pool will provide available connection to the connecting database type. The QB formats SQL syntax according to connecting database types. The business logic in the form of general query arguments is passed to QB to form a valid SQL syntax. The QB component output are read data list and new database structure, such as system columns, system tables and systems data. The bulk data process is a component to process data through multithreaded process whereby its SQLs are also supplied by QB. The applications are also connected to external processes via an interface.

B. *Query Builder interface*

The source and target tables operations rely on the generated SQL statement. Figure 3 shows the QB's interface for Create table and Select query statement.

```
//create a table
ICreate ic= new ICreate(databaseType);
ic.table('edu.address_xd').columns(('NA
ME','string',20),('ADDRESS','string',20
0));
String query= ic.buildQuery();

//select
ISelect is= new ISelect(databaseType);
is.table('edu.address','*').limit(1000,
2000).where('_group_id','001');
String query= is.buildQuery();
```

Figure 3. Query Builder interface

For creating a table, the respective data type semantic (e.g., STRING, INTEGER) are translated into correct syntax (e.g., VARCHAR2 or VARCHAR, NUMBER or INT). Similarly, we have separate interfaces to generate queries for Insert, Update, Delete and Alter with their respective inputs. These are the interfaces for consuming QB component services. The QB's role is to produce the following outputs in relation to source or target tables (based on database type), namely, (1) read statement for paging, (2) create new system tables, (3) create new index and trigger, (4) append new columns, and (5) populate system values.

IV. RESULTS

By implementing the above design and architecture, we are able to create an ETL and data cleansing applications that compatible with MySQL, PostgreSQL and Oracle RDBMS. All SQL issues specific to databases are handled in QB. It is easy to use for user without in depth knowledge of SQL. The DIT's user only needs to configure the function and the target database's table; QB will generate all the SQL statement specified to the target database type. The QB project is loosely coupled it can be easily scaled to more databases without any code change in the main project.

Some of the problems below and their solutions are handled by the QB: (1) the quoted identifier is implemented in QB for all databases without requiring database administrator to configure any database system settings. (2) The schema hierarchy is implemented in QB for all database types as it is mandatory for specific database, i.e., PostgreSQL. (3) QB also handles specific queries that use SQL built-in functions. Specific queries are built when database type is known and its respective built-in functions are included in the queries. (4) The UPSERT command also varies for different database types, which is handled in QB. (5) The auto increment is a standard feature in MySQL and PostgreSQL, but not in Oracle; this is handled in QB by issuing set of dependent queries for different database types.

 ISBN: 978-1-61208-408-4

For application wide problems, the issues and solutions are also being identified: (1) PostgreSQL database has a specific data type called NUMERIC, which can hold decimal and integer, hence, decimal and integer data should be type casted to float type. (2) While using connection pooling on multiple databases, pooled connection instances should be created only once and maintained in a map. This has to be reused to avoid connection leak. (3) Specific JDBC driver like Oracle JDBC doesn't return the auto generated column values by using the function "getGeneratedKeys()". This problem can be overcome by reading the ROWID column to get the generated value.

The DIT has been successfully deployed in User Integration Test (UAT) environment and currently supporting ORACLE, MySQL and PostgreSQL databases.

V. Conclusion And Future Work

This industry experience gives a glimpse of the best practices and the issues faced while using multiple type of database dynamically in a DIT web-based application. The major problems are namely, dynamic database mapping of source and target database, maintaining database connections in application and validation of query syntax for each database types. These problems are difficult for ORM framework as discussed in this paper. Hence, the usage of direct SQL statement is preferred.

The QB is able to generate specific SQL statement for multiple database types. This is an independent component which can be scaled to more databases and plugged to any applications.

A lesson learned during working with the DIT application is that a complex housekeeping is needed when user cancels a job. User could terminate a running job, but as a result, specific SQL commands need to be executed; for example, to drop supporting tables and remove system columns of the target table.

The DIT can work with other RDBMS in future with minimum changes, and the changes are structured and only happen in QB. In future, QB will support version specific SQL queries. Moreover, this service can be extended as web service, which can be consumed by external parties.

References

[1] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit, Indianapolis: Wiley Publishing, 2004.

[2] B. Walek and C. Klimes, "Expert System for data migration between different database management systems," Advances in Data Networks, Communications, Computers and Materials, 2012, pp. 167-172.

[3] N. Vijayendra and M. Lu, "A Web-based ETL Tool for Data Integration," in The 6th International Conference on Human System Interaction (HSI), IEEE Explore, Sopot, Poland, 2013, pp. 434-438.

[4] RedHat, "Hibernate ORM," [Online]. Available: http://hibernate.org/orm/. [retrieved: May, 2015].

[5] Clinton. Begin, "MyBatis," [Online]. Available: http://blog.mybatis.org/. [retrieved: May, 2015].

[6] Pentaho, "Data Integration - Kettle" [Online]. Available: http://community.pentaho.com/projects/data-integration/ [retrieved: May, 2015].

[7] Talend, "Talend Product - Data Integration" [Online]. Available: https://www.talend.com/products/data-integration [retrieved: May, 2015].

[8] E. K. Karuppiah, Y. K. Kok and K. Singh, "A Middleware Framework for Programmable Multi-GPU-Based Big Data Applications" Springer Link, 2015, pp. 187-206.

[9] J. Celko, "Joe Celko's SQL Programming Style," Morgan Kaufmann, 2005.

[10] ANSI X3.135-1992, American National Standard for Information Systems — Database Language — SQL, November, 1992.

[11] PostgreSQL, "Lexical Structure," [Online]. Available: http://www.postgresql.org/docs/9.3/static/sql-syntax-lexical.html. [retrieved: May, 2015].

[12] Oracle, "Schema Object Names," [Online]. Available: http://dev.mysql.com/doc/refman/5.6/en/identifiers.html. [retrieved: May, 2015].

[13] Oracle, "Schema Object Names and Qualifiers," [Online]. Available: http://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements008.htm#SQLRF51109. [retrieved: May, 2015].

[14] Oracle, "MySQL SELECT," [Online]. Available: http://dev.mysql.com/doc/refman/5.0/en/select.html. [retrieved: May, 2015].

[15] Oracle, "Ask Tom ROWNUM," [Online]. Available: http://www.oracle.com/technetwork/issue-archive/2006/06-sep/o56asktom-086197.html. [retrieved: May, 2015].

[16] Apache, "The DBCP Component," Apache, 12 July 2014. [Online]. Available: http://commons.apache.org/proper/commons-dbcp/ [retrieved: May, 2015].

 ISBN: 978-1-61208-408-4

Clustering of Words in Texts Using Fuzzy Neighborhood

Zhang Canlun, Sadaaki Miyamoto
Department of Risk Engineering
University of Tsukuba
Tsukuba, Ibaraki, Japan
{s1320646, miyamoto}@risk.tsukuba.ac.jp

***Abstract*—In this paper, we study the clustering of keywords in documents. We consider a model of fuzzy neighborhood for measuring similarity between words. Fuzzy neighborhoods lead to positive-definite kernels. By using the methods of kernel-based fuzzy c-means and affinity propagation, we show the results of clustering for Chinese documents on the Web. Moreover, Rand index is used to compare robustness of hard and fuzzy c-means algorithms with respect to different initial values.**

Keywords- kernel-based clustering; fuzzy neighborhood; c-means; keywords in texts.

I. INTRODUCTION

With the development of information and communication technology, the Web became a large and complex resource of information. However, by processing of the data from Internet having non-numeric attributes of huge volume, it is difficult to use them effectively and easily, although there is a huge amount of important data being used widely in daily life. In order to provide a better usability of Web data, a technique called text mining has attracted attention of many users. In China, with the spread of Internet, the demand and opportunity to handle a large number of documents, blogs, Twitter, and news Web pages is increasing very rapidly. This has attracted attention for text mining.

The purpose of this study is to show the effectiveness of clustering [1] using a fuzzy neighborhood model [2] [3] that regards a document as a sequence of words/terms. By extracting words of interest from text data on the Web, we calculate the similarity of words based on the neighborhood model and we obtain clusters of words using this similarity. A feature of this model is that the similarity satisfies the property of positive-definite kernel function [4]. This method is applied to a set of Chinese documents on the Web, where the methods of hard c-means, fuzzy c-means [1], and affinity propagation [5] [6] are used. A problem of c-means is the dependence of clusters on initial values. We use Rand index [7] to compare the degrees of dependence by different algorithms.

The rest of this paper is organized as follows: Section 2 describes the methods that are used for clustering, which is the main part of this paper. The methods of kernel-based fuzzy c-means as well as kernel-based affinity propagation are described. Section 3 shows the application of these methods to a Chinese document. Finally, Section 4 concludes the paper.

II. MODEL FOR CLUSTERING

The purpose here is to generate clusters of words or terms in a text. When we have a set of texts, they are handled as a single long text by concatenating them.

A. Fuzzy Neighborhood Model

A fuzzy neighborhood is a model for text data analysis. It does not just consider the numbers of word appearances, but also consider the location of the words in the text. We assume that when two words are near, a similarity between these words is higher.

All the items of fuzzy neighborhood model used in this study are defined as follows:

A word set or term set $T = \{t_1, \dots, t_m\}$ is the objects for clustering.

An occurrence set $o, o', o_i \dots \in O$ is assumed. An occurrence *o* is different from a term *t* in the sense that a term can appear more than once in a same document.

A text is a sequence of occurrences: $oo'o''$ ..., where terms are occurring several or many times; the relation between an occurrence and a term is described by the next relation.

A fuzzy relation between *T* and *O* is assumed: when *t* and *o* imply the same term, $R(t, o) = 1$; when the t and o are different term, $R(t, o) = 0$, when *t* and *o* have fuzzy relation, $0 < R(t, o) < 1$.

D is the distance between occurrences *o* and o' in a text, which is defined as follows:

$$D(o, o') = \{\text{number of occurrences between } o \text{ and } o'\} + 1 \quad (1)$$

Another fuzzy relation *N* of $O \times O$ is defined by using the distance, which is called a *fuzzy neighborhood*. $N(o_i, o_j)$ means the degree of nearness between terms o_i and o_j;

A function f defines a particular form of *N*, $f: \{0,1,2, \dots\} \rightarrow [0,1]$; *f* satisfies the next conditions:

1. $f(0) = 1$,
2. $lim_{x \rightarrow \infty} f(x) = 0$,
3. f is monotone non-increasing.

Using *f* and *D*, *N* is defined as follows:

$$N(o,o') = f(D(o,o')) \ . \quad (2)$$

The fuzzy neighborhood has symmetry and reflectivity, which is shown as follows:

$$N(o,o') = N(o',o) \quad (3)$$

$$N(o,o) = 1. \quad (4)$$

The formula shown below is a typical example of fuzzy neighborhood defined by the distance:

$$N(o,o') = \max\{0, 1 - \frac{D(o,o')}{L}\}, \quad (5)$$

where L is a positive constant.

Figure 1 shows an example of a functional form of fuzzy neighborhood. The center is an occurrence of a term.

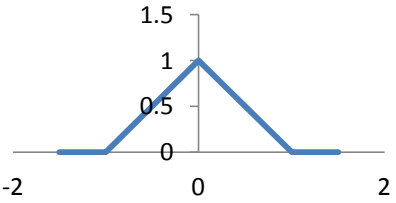


Figure 1. An example of fuzzy neighborhood

A relation $p(t,t')$ is a similarity measure between two words, which is defined as follows:

$$p(t,t') = \sum_{a\in O}\sum_{b\in O} R(t,a)N(a,b)R(t',b) \quad (6)$$

Another relation $s(t,t')$ is a normalized measure derived from $p(t,t')$; $s(t,t')$ is defined as follows:

$$s(t,t') = \frac{p(t,t')}{\sqrt{p(t,t)p(t',t')}} \quad (7)$$

It can be proved that similarity $p(t,t')$ using (5) becomes a positive-definite kernel function [4]. More generally, when function f is convex, $p(t,t')$ is positive-definite. See [2] or [3] for the proof. It is not difficult to see that $s(t,t')$ is also positive-definite.

B. *Kernel fuzzy c-means*

The method of kernel c-means can be applied to data with clusters having nonlinear boundaries; without a kernel, only linear boundaries can be obtained by c-means [1]. In this study, the similarity calculated by fuzzy neighborhood is a kernel function.

Clustering using kernel function uses a high-dimensional feature space, which is the main difference from ordinary fuzzy c-means [1].

The objective function of kernel-fuzzy c-means (K-FCM) is as follows:

$$J_{kfcm}(U,W) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m \|\emptyset(x_k) - W_i\|^2 \quad (8)$$

where W_i is the center of cluster i in a high-dimensional feature space, and can be obtained by minimizing (8) with respect to W_i:

$$W_i = \frac{1}{N_i}\sum_{k=1}^{n}(u_{ik})^m \emptyset(x_k). \quad (9)$$

Note that $\emptyset(x_k)$ is called a high-dimensional mapping. The functional form of $\emptyset(x_k)$ is generally unknown but its inner product is known and given by a kernel function $K(x_j,x_k) = (\emptyset(x_j),\emptyset(x_k))$. Note that either $p(t,t')$ or $s(t,t')$ can be used as the kernel function.

We cannot calculate (9) directly in c-means clustering. Instead, distance d_{ik} is calculated by using the kernel function $K(x_j,x_k)$ as follows:

$$d_{ik} = \|\emptyset(x_k) - W_i\|^2 = \left\|\emptyset(x_k) - \frac{1}{N_i}\sum_{j=1}^{n}(u_{ij})^m \emptyset(x_j)\right\|^2$$
$$= K(x_k,x_k) - \frac{2}{N_i}\sum_{j=1}^{n}(u_{ij})^m K(x_j,x_k)$$
$$+ \frac{1}{(N_i)^2}\sum_{j=1}^{n}\sum_{l=1}^{n}(u_{ij})^m(u_{jl})^m K(x_j,x_l) \quad (10)$$

Fuzzy membership value u_{ik} is given as follows:

$$u_{ik} = \left[\sum_{p=1}^{c}\left(\frac{d_{ik}}{d_{pk}}\right)^{\frac{1}{m-1}}\right]^{-1} \quad (11)$$

The algorithm of Kernel Fuzzy c-Means (K-FCM) is as follows:

K-FCM1. Give an initial value of membership degree $\bar{U}$;

K-FCM2. Calculate the distance d_{ik} of each individual to each cluster center in a high-dimensional space;

K-FCM3. Solve the minimization problem $min_{U\in M_f} J_{kfcm}$ (U, W) and make $\bar{U}$ as an optimal solution;

K-FCM4. If the solution is convergent, stop. Else go to **K-FCM2**.

End of K-FCM.

The hard c-means (K-HCM) has also been well-known [1]. We omit the detail of K-HCM algorithm.

C. *Kernel Affinity Propagation*

A remarkable point of affinity propagation [5] [6] is that the number of clusters is defined by the method itself. In this method, all the data points exchange messages between each other to decide which cluster to subordinate and which point to be the exemplar.

There are two messages exchanged: responsibility $r(i,k)$ and availability $a(i,k)$. Data point i sends a message of responsibility $r(i,k)$ to a candidate exemplar point k to show how point k is suited to serve as the exemplar for point i. The candidate exemplar point k sends the message of availability $a(i,k)$ to point i to reflect the accumulated evidence of the degree of appropriateness for point i so that point k is chosen as its exemplar [5] [6].

Then, responsibility $r(i,k)$ is defined by the following rule:

$$r(i,k) \leftarrow s(i,k) - \max_{k'\neq k}\{a(i,k') + s(i,k')\}. \quad (12)$$

As the above responsibility is updated by (12), availability is updated by the following rule:

$$a(i,k) \leftarrow min\ \{0, r(k,k) + \sum_{i'\notin\{i,k\}} max\{0, r(i',k)\}\}. \quad (13)$$

But availability of point to itself is different:

$$a(i,k) \leftarrow \sum_{i' \notin k} max\{0, r(i',k)\} \quad (14)$$

The similarity $s(i,k)$ between two data points is defined using the kernel function: For point x_i and x_k, we have

$$s(i,k) = -\big(K(x_i,x_i) + K(x_k,x_k) - 2K(x_i,x_k)\big). \quad (15)$$

For data point i, the point k that maximizes $a(i,k) + r(i,k)$ is chosen to be the exemplar of data point i, in other words, data point i belongs to cluster k.

D. Rand Index

The Rand index [7] is a measure of similarity between two data clustering results. The definition is as follows.

Given N points $S = \{X_1, \ldots, X_N\}$, and two clusters of them $Y = \{Y_1, \ldots, Y_s\}$ and $Y' = \{Y'_1, \ldots, Y'_k\}$, we define:

- m: the number of pairs of elements in S that are in the same set in Y and Y',
- m': the number of pairs of elements in S that are in different sets in Y and Y'.

Then the Rand index R is:

$$R = \frac{m+m'}{\binom{N}{2}} \quad (16)$$

Assume that we use K-FCM. The result is fuzzy clusters. In order to use the Rand index, we make clusters hard by using the maximum membership rule: *allocate an object to the cluster where the membership takes its maximum value*.

When we perform M times of **K-FCM** algorithm with different initial values, we have M results. Then, we use the Rand index to calculate each pair of resulting clusters: the combination of all the pairs of results is $\binom{M}{2}$. Then the average of the Rand index value is calculated. It will show the degree of dependence of clusters on the initial values: when we have a larger value of the Rand index, it means less dependency on the initial values. When we compare the averaged Rand indexes for different methods, we can judge which method has the least dependency on the initial values, in other words, robustness with respect to initial values.

In this paper, we will show two Rand index values of two methods of K-HCM and K-FCM.

III. Application to a Document in Chinese

A document < Company Culture Management > on the Web site [8] introduces the importance of company culture and how to management the company culture. The data cannot be easily clustered by human eyes. Nouns and adjectives of which the numbers of occurrences were more than 2 were extracted; the number of objects for clustering was 36.

Figures 2 and 3 show clusters using kernel fuzzy c-means and kernel affinity propagation for this document. The kernel principal component [4] was used for the visualization. The number of clusters for kernel c-means was assumed to be three, while the number of clusters was automatically decided in the affinity propagation. The number of clusters obtained by the latter method can be varied according to the maximum number of iterations, but generally the number is far larger than three.

Investigating the contents of this document, we observe that the meanings of the clusters are as follows:

- Cluster 1 (blue): the role of corporate culture and its development history;
- Cluster 1 (red): excellent cases of business or companies, such as United States and Japan, outstanding business success stories and economic outcomes or impact;
- Cluster 3 (green): the way we should build the socialist modern company culture in the market economy, and the importance of modern corporate culture and its role, or the regulatory requirements to business leaders, the masses of workers and others.

The details of the keywords in these clusters are omitted here, but the three clusters shown by different colors by fuzzy c-means correspond well with the three prolonged groups on the plane in Figure 2, while geometrical separation of clusters is not found in Figure 3. Note that the configuration of points in these figures was derived from the kernel principal component and is independent of clusters.

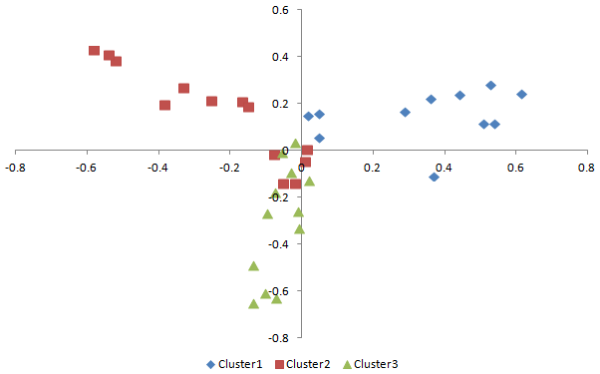


Figure 2. Clusters by kernel fuzzy c-means.

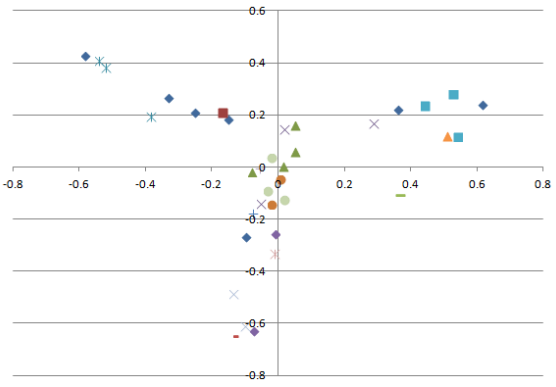


Figure 3. Clusters by kernel Affinity Propagation

TABLE I. RAND INDEX VALUES BY HARD AND FUZZY C-MEANS

	K-HCM	K-FCM
Rand index	0.716	0.830

Table 1 shows the values of the Rand index from K-FCM

and K-HCM. The value of the Rand index was higher when we used K-FCM than that of K-HCM, which means that the fuzzy method was more robust with respect to the difference of initial values than the hard method of c-means clustering.

IV. CONCLUSION

The model of fuzzy neighborhood was shown and the kernel based methods of clustering were developed. The developed methods were hard and fuzzy kernel c-means as well as kernel affinity propagation. The results of application to a Chinese document showed interpretable clusters by fuzzy c-means, while the result of affinity propagation did not show good clusters. Using the Rand index, we showed the method of fuzzy c-means gave more robust results than the hard c-means. To evaluate the combination of fuzzy neighborhood and kernel affinity propagation quantitatively, we need far more examples.

The document is a real Chinese document on the Internet. Although the volume is not enough now, further studies of Chinese documents or Social Networking Service data seem to be promising.

ACKNOWLEDGMENT

This study has partly been supported by the Grant-in-aid for Scientific Research, JSPS, Japan, No. 26330270.

REFERENCES

[1] S. Miyamoto, H. Ichihashi, K. Honda, Algorithms for Fuzzy Clustering, Springer, 2008.

[2] S. Miyamoto, S. Suzuki, “Clustering in Tweets Using a Fuzzy Neighborhood Model”, Proc. of WCCI 2012.

[3] S. Miyamoto, Y. Komazaki, S. Suzuki, S. Takumi, “Analysis of Disaster Information on Twitter Using Different Methods of Clustering Based on A Fuzzy Neighborhood Model”, Proc. of ISCIIA 2012.

[4] B. Scholkopf, A. J. Smola, Learning with Kernels, MIT Press, 2002.

[5] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points”, Science, vol.315, pp.972-977, 2007.

[6] I. E. Givoni, B. J. Frey, “A Binary Variable Model for Affinity Propagation”, Neural Comput., vol.21, pp.1589-1600. doi: 10.1162/neco.2009.05-08-785, 2009.

[7] W. M. Rand, “Objective criteria for the evaluation of clustering methods”, Journal of the American Statistical Association, vol. 66, pp.846–850, 1971.

[8] http://wenku.baidu.com/view/e599acf2f242336c1fb95e36.html

 ISBN: 978-1-61208-408-4

Improving Near Duplicate Data Detection via DSound Phonetic Matching Algorithm: A Solution to Address Typographical Problems

Cihan Varol and Sairam Hari
Department of Computer Science
Sam Houston State University
email: { cxv007@shsu.edu, sxh020@shsu.edu }

***Abstract*—Near duplicate data not only increase the cost for information processing, but also increase the time taken for a decision. Therefore, detecting and eliminating them is vital for business decisions. Shingling algorithm has been used in detecting near duplicates in large-scale text databases. The algorithm is based on the number of common tokens in two or more set of information. In other words, if there is a slight variation of the text, such as misspelling, in one of those documents, the performance of the algorithm decreases. Therefore, in this work, we proposed to embed a new phonetic approximate algorithm, namely DSound, to Shingling algorithm for improving the near duplicate data detection if there is a typographical error. Based on the experiments on real dataset, this newly proposed framework improved the Shingling algorithm's performance by 16 percent.**

Keywords-data cleansing; data quality; duplicate detection; DSound; Shingling

I. INTRODUCTION

Now-a-days, duplicate document is a common problem in big database. Advanced duplicate detection techniques are required not only to process query, but also to filter the redundant (duplicate data) information in large scale document database to improve the search quality. To address this issue, duplicate document detection techniques are used to prevent the search results from including the multiple documents having the same or nearly same content.

The search quality in a data is affected as a result of multiple copies being included in the search results. The process of indexing the data is done by scanning the content of each and every document. When two documents contain identical content, they are regarded as duplicates. There might be some documents with small dissimilarities and are not declared as being "exact duplicates" of each other but are identical to such an extent that they can be declared as near-duplicates [1]. Detecting near duplicates are utmost important to improve the search quality.

Following are examples of near duplicate samples seen in documents [2]:

- Documents containing a few different words - widespread form of near-duplicates.
- Documents with the same content but different formatting – for instance, the documents may contain the same text, but dissimilar fonts, bold type, or italics.
- Documents with the same content but with typographical errors (mistyped words)
- Plagiarized documents and documents with different versions
- Documents with the same content but different file type – for instance, Microsoft Word and PDF.
- Documents providing identical information written by the same author being published in more than one domain.

Relying only on the resemblance of the exact contents of the documents may yield to overlook near identical ones. Thus, this will eventually lead to miss the detection of duplicate documents. Although, all the listed examples are part of the near duplicate problem, in this work, we will particularly address the problems raised with *typographical errors*. *Typographical errors* are a spelling error that can be captured simply because it is mistyped or misspelled [3]. As the name suggests, these are invalid strings, properly identified and isolated as incorrect representations of a valid word [4]. Fat fingering places an important role in this type of misspelling. These errors are made assuming that the writer or typist knows how to spell the word, but may have typed the word hastily resulting in an error [5]. No matter how the word is misspelled, this results in decrease in quality of the document and yield to near identical documents. Therefore, in this paper, we introduce a hybrid approach to improve the widely known near duplicate detection algorithm, namely Shingling algorithm. Particularly we embed phonetic matching technique to Shingling algorithm to improve the outcome of near duplicate detection.

In Section 2, we discuss related work about near duplicate detection. In Section 3, the newly proposed approach will be elaborated. In Sections 4 and 5, the test cases and experimental results will be discussed. At the end, the paper will be finalized with the conclusion section.

II. RELATED WORK

There are few techniques that have been developed to identify near duplicate documents [6]-[10], web page

 ISBN: 978-1-61208-408-4

duplicates [11]–[15], and duplicate database records [16]-[17]. Brin et al. has proposed a Copy Protection System (COPS) system to protect important and intellectual property of digital documents [6]. As a part of the Stanford Digital Library project Kumar et al. developed Stand Copy Analysis Mechanism (SCAM) to identify similar documents in online library [7]. Author in [12] proposed shingling algorithm which keeps the sketch of shingles of each document to compute the resemblance of two documents. Any documents with at least one common shingle are examined and checked whether to see if it exceeds the threshold for resemblance. Broder's shingling method is implemented in AltaVista search engine for duplicate document detection [12]. Lyon et al. investigated the theoretical background to automate plagiarism detection [8]. They observe that independently written texts have a comparatively low level of matching trigrams. The Ferret plagiarism system counts the matching trigrams of a pair of documents [8]-[9]. In [10], the authors offered a new filtering technique by exploiting the ordering information. With this way, they drastically reduced the candidate sizes and hence improved the efficiency of search.

The authors in [11] proposed two approaches to compute near duplicates between all web documents simultaneously. Both of the approaches assume that two documents *di* and *dj* can be near duplicates only when document *di* and *dj* share more than 'm' fingerprints, where 'm' is a predefined threshold [11]. Das et al. proposed a Term Document Weight (TDW) matrix based algorithm with three phases, rendering, filtering, and verification, which receives an input web page and a threshold in its first phase, prefix filtering and positional filtering to reduce the size of record set in the second phase and returns an optimal set of near duplicate web pages in the verification phase by using Minimum Weight Overlapping (MWO) method [15]. In [18], the author showed that rounding algorithms for Linear Programming (LPs) and Semidefinite Programming (SDPs) used in the context of approximation algorithms can be viewed as locality sensitive hashing schemes. Simhash is a fingerprint technique that holds the property that the two documents are near duplicate only if the fingerprints of the documents are identical, since near-duplicates differ only in a small number of bit positions [13] and [18].

As stated above and discussed in [19], most of the techniques are looking for the resemblance of the documents via the exact similarity level or use some form of approximation technique to detect the near similar documents. If the latter approach is used in the whole document the computation overhead will be high. If the first approach is used alone, then the misspelled or slightly incorrect representation will yield to decrease the accuracy rating. Therefore, there is a dire need to combine these two main approaches in a way that it will not only improve the accuracy rating, but also keeps the computational overhead in low levels.

III. METHODOLOGY

In this work, we propose to embed DSound Phonetic Approximate algorithm to Shingling algorithm for detecting near duplicates documents.

Shingling algorithm is a well-known technique used for detecting the near duplicates. DSound is a newly designed phonetic based spelling correction algorithm which returns an equivalent word if there is a misspelling. Both of these methods are explained briefly in below sections.

A. Shingling Algorithm

Shingling algorithm is a well-known technique introduced by Broder to estimate the degree of similarity among pairs of documents [12]. The algorithm does not depend on any linguistic knowledge other than the ability to tokenize documents into a set of words based on the shingle size [12]. In shingling, the string is divided into words and all word sequences of adjacent words are extracted. If two documents contain the same set of shingles they are considered identical and if their sets of shingles appreciably overlap, they are accepted as exceedingly identical [12].

In detail, shingling divides the text document into set of substrings of length *k* (called *k-shingles*). The shingling algorithm is implemented using sliding window method with the window size of *k (shingle size of k)* over the input document character by character and placing the substrings into a set. For example, the string "abcabcac" can be represented by {"abc", "bca", "cab", "abc", "bca", "cac"} if the sliding window is chosen as 3-shingles. Since the algorithm eliminates the duplicate ones, only four shingles {"abc", "bca", "cab", "cac"} will be in the final set. Another example for shingling algorithm is given below:

Given a string S = {a rose is a rose is a rose}
Using shingling algorithm (with window size set at 4) the string can be tokenized as {(a rose is a), (rose is a rose), (is a rose is), (rose is a rose), (a rose is a), (rose is a rose)}.
Removing the duplicates will yield to the following shingles
{ (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is) }

At the end, Jaccard similarity is used for expressing similarity of sets. For sets A and B, it is defined in (1).

$$SIM(A, B) := |A \cap B| / |A \cup B|. \quad (1)$$

The result ranges from 0 (no elements in common) to 1 (identical). It is evident that duplicate documents have a higher similarity to the original than unrelated documents.

B. DSound Phonetic Matching Algorithm

Phonetic matching algorithms are relied and dependent on the phonetic structures of a language. In spite of this restriction, they are proved to be very effective when fixing ill-defined data in English language [20]. Moreover, computation wise, it can be faster than string matching algorithms because of the applied conversion rules can shrink the textual information to a small number of digit

coding. Because of the stated reasons, in this work, we have concentrated on phonetic matching algorithms.

Although it is one of the early designed algorithms for phonetic matching, previously we have shown the effectiveness of Soundex algorithm [20]. The Soundex algorithm keeps the first letter in a string and converts the rest into numbers. All zeros (vowels and 'h', 'w' and 'y') are then removed and sequences of the same number are reduced to one only (e.g., '222' is replaced with '2'). The final code is the original first letter and three numbers (longer codes are cut-off, and shorter codes are extended with zeros). As an example, the Soundex code for 'Rodgers' is 'r326', and 'r262' for 'Rojers'.

Although Soundex is a well-designed solution, a major problem with the algorithm is that it keeps the first letter, thus any error at the beginning of a name will result in a different Soundex code. This can cause to eliminate the valid candidates because of the error in the first letter. Another disadvantage of the algorithm is the lack of transformation rules for special conditions, such as a word containing –dge- letters consecutively, i.e., Rodgers. As shown in the above examples with 'Rodgers' and 'Rojers', although those two text information are very close to each other, the difference between the codings are large enough to not easily come to a conclusion that those text information can be same. Moreover, truncated the coding with only 3 digit and letter coding may yield to miss misspellings occurring late in a long word. Therefore, we have created DSound phonetic matching algorithm based on the deficiencies of the Soundex and created three steps to generate a phonetic coding for a given text information as shown in Table 1.

TABLE I. DSOUND TRANSFORMATION RULES

Step	Letters	Code
1	kn-, gn- pn, ac-, wr-	drop first letter
	x-	change to "s"
	wh-	change to "w"
	otherwise	skip to step 2
2	d	2 (if in –dge- or –dgi- 'Rodgers') – drop 'g'
	g	0 (if in –gh- 'Houghton')
	t	2 (if in -tia- or -tio- 'Attention')
	otherwise	skip to step 3
3	a e h i o u w y	0
	b f p v	1
	c g j k q s x z	2
	d t	3
	l	4
	m n	5
	r	6

In detail, first the initial characters are parsed by the given rules. Second, special cases are handled based on the created conversion system. Third, as a last step, the final text information is converted to digits based on the given rules and only repeated digits are removed from the final coding. For instance, the DSound code for "Rodgers" and "Rogers" are same and will yield to a coding of '602062'

C. Proposed Approach

The proposed approach is a combination of the shingling algorithm and DSound phonetic matching algorithm as shown in Figure 1.

Specifically, two input files are preprocessed and inserted into the system to check if there are any inconsistencies (mistyped words) between those. If there is any, then the proposed approach utilizes the DSound phonetic matching algorithm for finding the closest match for the mistyped data with the help of dictionary, which holds the English words. If there is no perfect match (different DSound codings), then Edit Distance score [21] is calculated between the original data and suggestions. In order to eliminate irrelevant results from the suggestion list a threshold of 2 is set for the distance. Then the closest candidate is selected as the best possible replacement for the mistyped word and is replaced with the ill-defined one in the text document. After the mistyped words are modified according to the combined approach, Shingling algorithm is applied to see the degree of the closeness of the text files. The closeness is calculated by the number of common shingles divided by the total number of shingles in both files. The details of the algorithm are provided in the following pseudo code (Figure 2).

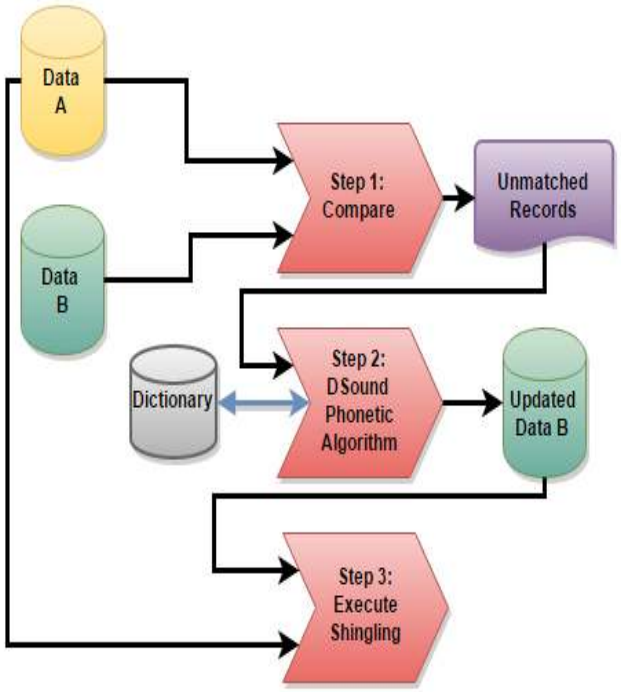


Figure 1. Proposed Hybrid Approach

Algorithm

```
1:  Original File = OF;
2:  Reference File = RF;
3:  Open OF and RF
4:  Run Shingling Algorithm
5:  Compare OF and RF
6:  Run DSound Algorithm on not-matched results
7:      IF: match found = 1
8:          write and replace the word in RF
9:      ELSE
10:         calculate edit distance
11:         write and replace the word in RF with the
              (lowest score) closest one
12:     END IF
13: Open OF and RF
14: Run shingling algorithm
15: Compare OF and RF
```

Figure 2. Pseudo code of the Hybrid Approach

IV. TEST CASE AND RESULTS

The data sets used for testing the algorithm were open to public text files that were obtained from the internet. Two files were obtained for each data set. The first file contained the error-free version of the text, while the second one held the same content but with occasional typographical errors due to mistyping or OCR errors. Overall, two files were duplicates, but because of some mistyped words the degree of closeness was less. The length of the text varied between 119 words to 1108 words. As a proof of concept we executed the proposed approach on the mistyped version of the Abraham Lincoln's Gettysburg address [22]. First, the mistyped words were corrected using the DSound phonetic matching algorithm. Second, Shingling algorithm was applied to the updated data set to evaluate the algorithms' performance (Table II).

TABLE II. FIRST TEST DATA RESULTS

	Shingle size=3	**Shingle size=4**	**Shingle size=5**
Common Shingles w/o Proposed Approach	254	252	247
Not Matching Shingles w/o Proposed Approach	26	34	42
Common Shingles with Proposed Approach	265	263	262
Not Matching Shingles with Proposed Approach	7	11	12

As shown in Table II, before applying the hybrid approach and given a shingle size of 3, the number of common shingles between the two input files was 254 and the number of un-matched shingles was 26, after duplicate shingles were removed. After we applied the hybrid approach to the Shingling algorithm, the number of common shingles increased to 265 and the number of un-matched shingles dropped to 7 with the shingle size of 3, once again after duplicate shingles were removed. The difference between the total number of shingles produced before using the proposed approach and after using the proposed approach is because of the removal of duplicates from the final set of shingles. As reflected from the results, the hybrid approach improved the performance of shingling algorithm and increased the degree of closeness between the two input files with different shingle sizes.

As a second task, we executed the same test on forty documents. Overall, 40 pairs of error free and mistyped versions of the documents were collected from the internet. The error free versions of the documents are used as a reference point and mistyped versions are compared with it. These files found in the internet were mostly because of Optical Character Recognition (OCR) problems caused by scanners [23]-[24]. As shown in Figure 3, the percentage of common shingles to all shingles was improved with an average of 16 percent with different shingle sizes of 3, 4 and 5. Although the average common shingle sizes were increased all over the board, the highest percentage of the common shingles were always seen when the shingle size was selected as 3.

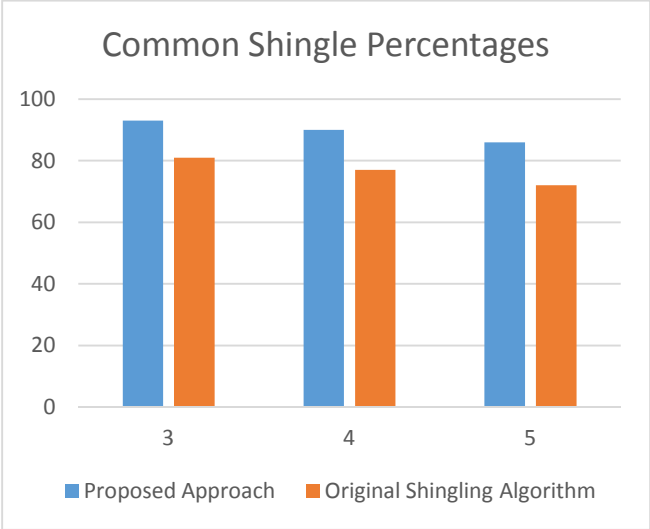


Figure 3. Percentages of Common Shingles with Different Shingle Sizes

The answer to which shingle size is better suited for detecting the near duplicates is vital while applying the Shingling algorithm. Based on the tests we conducted, our observations are reflected in Table III. Overall, the shingle size 3 provides the highest degree of closeness among two near identical documents and less overhead (total number of shingles) in all test data we experimented on. However, as expected if the all misspelled words are corrected, then the accuracy percentage would be same among different shingle sizes and the computation overhead would be very close to each other. Other shingle sizes, such as greater than 5 or less than 3 is omitted, since those window sizes yield to lower shingle scores or require extensive computation.

 ISBN: 978-1-61208-408-4

TABLE III. SELECTION OF SHINGLE SIZE

Scenario	Suited shingle size
When no misspelled words are corrected by Jaro.	3
When only some misspelled words are corrected by Jaro	3
When all misspelled words are corrected by Jaro.	3, 4 or 5

It is also important to note that correctly fixing one ill-defined data with the hybrid approach in most cases yields to substantial decrease in the number of not matching shingles. The main reason is that most of the each fix yields to multiple matching shingles based on the selected window size. Specifically, if a mistyped word is fixed when the window size is 4, then this may result in increase of 4 common shingles between the two documents, unless the mistyped word is not within the first or last three words of the document.

Word similarity and document similarity have been widely studied by researchers. In these works each word or each document are expressed in a vector space, then the similarity in the vector space can be calculated. The major disadvantage of this method is the large space vector and the lack of considering relationship and order between terms. It is computationally inefficient due to the sparse sentence vector. Moreover, it doesn't do any favoring if there is any slight spelling error in the compared text. However, our proposed approach is particularly designed to address the errors caused by misspellings. That is why, we rationale our technique based on the assumption that the misspelled information is close to each other and the context and weights associated with the text is not relevant.

V. PHONETIC MATCHING ALGORITHMS

In this work, we particularly created a new phonetic matching algorithm to increase the efficiency of the near duplicate detection. To prove that the DSound algorithm performs best as the phonetic matching algorithm of this newly proposed hybrid system, we compared its efficiency with other well-known phonetic matching algorithms, such as Soundex, Phonex, Phonix, and Double Metaphone. As mentioned earlier, Soundex is used to correct phonetic misspellings with mapping a string into a key consisting of its first letter followed by a sequence of digits [25]. The encoding algorithm is very fast in practice. However, a major problem with Soundex is that it keeps the first letter, thus any error at the beginning of a name will result in a different Soundex code. Phonex [25] tries to improve the encoding quality by pre-processing names according to their English pronunciation before the encoding. All trailing '*s*' are removed and various rules are applied to the leading part of a name (for example 'kn' is replaced with 'n', and 'wr' with 'r'). Like Soundex, the leading letter of the transformed name string is kept and the remainder is encoded with numbers (1 letter, 3 digits). Phonix algorithm is an improvement for the Phonex and applies more than one hundred transformation rules on groups of letters [26]. The Metaphone algorithm is also a system for transforming words into codes based on phonetic properties [25]. Unlike Soundex, which operates on a letter-by letter scheme, Metaphone analyzes both single consonants and groups of letters called diphthongs according to a set of rules for grouping consonants and then maps groups to metaphone codes.

In order to reduce the overhead, information has to be better organized and should produce relevant results. Two major metrics commonly associated with Information Retrieval Systems are Precision and Recall [27].

Precision can be defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search as shown in (2).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2)$$

Precision measures one aspect of information retrieval overhead for a user performing a particular search. If a search has 90% precision, then 10% of the user effort is the overhead reviewing non-relevant items. Please note that the definition of relevant documents is broadened that the suggestion algorithms provide the correct result as one of the possible candidate for the misspelled word.

Recall is different than precision. Recall can be defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents, or in other words Recall is the percentage of spelling correction rate. Recall gauges how well a system processing a particular query is able to retrieve the relevant items that the user is interested in seeing. The formula of the recall is shown in (3).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (3)$$

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score. F-score is calculated by (4).

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

The discussed five algorithms are tested on the discussed forty data set as shown in Table IV.

TABLE IV. COMPARISON OF PHONETIC MATCHING ALGORITHMS

Algorithm	Precision	Recall	F-Score
DSound	83.7%	61.8%	71.10%
Soundex	78.3%	54.3%	64.13%
Phonex	76.4%	51.8%	61.74%
Phonix	79.6%	57.1%	66.50%
Double Metaphone	79.8%	56.2%	65.95%

 ISBN: 978-1-61208-408-4

As an average, DSound phonetic matching algorithm achieved highest F-Score, Precision and Recall rates. This is the main rationale why DSound phonetic matching algorithm is selected as the part of the hybrid approach

VI. CONCLUSION AND FUTURE WORK

Shingling is an effective and efficient algorithm to detect and eliminate duplicates in large-scale short text databases. However, applying this technique to original data can result in poor ratings if the documents contain mistyped information. Therefore, in this work, a new phonetic matching algorithm, namely DSound, and Shingling algorithm are combined to detect the near duplicate documents. Experiments reflected that using the hybrid approach is an encouraging solution for large-scale short text duplicate detection and increased the performance of shingling algorithm. As a future work, the hybrid approach will be used in various domains, such as in health care data, to improve the detection of near duplicates. Moreover, we will focus on language identification on the text information so that we can create and apply language specific rules for the phonetic matching process.

REFERENCES

[1] K. J. Prasanna and P. Govindarajulu, "Duplicate and Near-duplicate documents detection," European Journal of Scientific Research, Vol.32 No.4, 2009, pp. 514-527.

[2] T. Gupta and L. Banda, "A hybrid model for detection and elimination of Near Duplicates based on Web Provenance for Effective Web Search," International Journal of Advances in Engineering & Technology. Vol. 4, Issue 1, 2012, pp. 192-205.

[3] Levenshtein Edit-Distance Algorithm, http://www.nist.gov/dads/HTML/Levenshtein.html [retrieved: March, 2015].

[4] C. Becchetti and L. P. Ricotti, "Speech Recognition: Theory and C++ Implementation". John Wiley & Sons, 1999.

[5] K. Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Surveys, vol. 24, No. 4, 1992, pp. 377-439.

[6] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," In: Proceedings of the ACM SIGMOD Annual Conference, San Francisco, CA, May 1995, pp. 398-409.

[7] K. N. Shiva and H. Garcia-Molina, "SCAM: A copy detection mechanism for digital document,". In: Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, Austin, Texas, June 1995, pp. 1-13.

[8] C. Lyon, R. Barrett, and J. Malcolm, "A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector," In: Plagiarism: Prevention, Practice and Policies Conference, June 2004, pp. 1-7.

[9] C. Lyon, R. Barrett, and J. Malcolm, "Plagiarism is easy, but also easy to detect," Plagiary: Cross-Disciplinary Studies in Plagiarism, Fabrication, and Falsification Vol.1, No.5, 2006, pp. 1–10.

[10] C. Xiao, W. Wang, and X. Lin, "Efficient Similarity Joins for Near-Duplicate Detection," Proceeding of the 17th International Conference on World Wide Web, April, 2008, pp 131 – 140.

[11] K. N. Shiva and H. Garnia-Molina, "Finding near-replicas of documents on the web," In: Proceedings of Workshop on Web Databases, Valencia, Spain, March 1998, pp 204-212.

[12] A. Broder, "Identifying and Filtering Near-Duplicate Documents," In: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Canada, June, 2000, pp. 1-10.

[13] G. S. Manku, A. Jain, and A. D. Sarma, "Detecting near-duplicates for web crawling," In: Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada, May 2007, pp. 141-149.

[14] M. Henzinger, "Finding near-duplicate web pages: A large-scale evaluation of algorithms," In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, U.S.A, August 2006, pp.284-291.

[15] S. N. Das, M. Mathew, and P. K. Vijayaraghavan, "An Approach for Optimal Feature Subset Selection using a New Term Weighting Scheme and Mutual Information," Proceedings of the International Conference on Advanced Science, Engineering and Information Technology, Malaysia, January 2011, pp. 273-278.

[16] Z. P. Tian, H. J. Lu, and W.Y. Ji, "An n-gram-based approach for detecting approximately duplicate database records," International Journal on Digital Libraries, Vol. 5 No. 3, 2001, pp. 325-331.

[17] M. A. Hernandez and S. J. Stolfo, "The merge/purge problem for large databases," In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, 1995, pp. 127-138.

[18] M. Charikar, "Similarity estimation techniques from rounding algorithms," In: Proceedings of 34th Annual Symposium on Theory of Computing, Montréal, Québec, Canada, May 2002, pp. 380-388.

[19] J. P. Kumar and P. Govindarajulu, "Duplicate and near duplicate documents detection: A review," European Journal of Scientific Research, 32, 2009, 514--527.

[20] C. Varol and C. Bayrak, "Personal Name-based Pattern and Phonetic Matching Techniques: A Survey," ALAR Conference on Applied Research in Information Technology, February 13, 2009, Conway, Arkansas, USA..

[21] Levenshtein VI. "Binary codes capable of correcting deletions, insertions and reversals". Doklady Akademii Nauk SSSR 163 : 845-848, 1965, also {1966) Soviet Physics Doklady 10 : 707-710.

[22] "The Gettysburg Address", http://www2.cs.uregina.ca/~simeon2m/CS210/Assignments/testfile.txt [retrieved: March, 2015].

[23] "Electronic Text Center", http://courses.cs.vt.edu/csonline/AI/Lessons/VisualProcessing/OCRscans.html [retrieved: March, 2015].

[24] "Error Report on ScarWox", http://www.columbia.edu/acis/cria/rosenberg/sample/ [retrieved: March, 2015].

[25] L. Philips, "Hanging on the metaphone," Computer Language, 7(12), 1990, pp. 39-43.

[26] T. Gagg, "PHONIX: The algorithm," Program: automated library and information systems, 24(4), 1990, pp. 363–366.

[27] J. Davis and M. Goadrich: "The relationship between Precision-Recall and ROC curves." ICML 2006: pp. 233-240.

Query-Based ℓ-Diversity

Chittaphone Phonharath
Nara Institute of Science
and Technology
Nara, Japan
Email: chittaphone-p@is.naist.jp

Ryunosuke Takayama, Kenji Hashimoto, Hiroyuki Seki
Nagoya University
Nagoya, Japan
Email: {k-hasimt,seki}@is.nagoya-u.ac.jp

***Abstract*—We propose a new privacy notion called *query-based ℓ-diversity*. A database instance T is ℓ-diverse with respect to given authorized queries if an attacker cannot narrow down the number of possible values of the sensitive information to less than ℓ by inference using the result of the authorized queries on the instance T and the meaning of the queries. We provide two approaches to deciding the query-based ℓ-diversity. In the first approach, a decision algorithm is given by using relational operations, which can be directly implemented by a relational database management system, e.g., Structured Query Language (SQL). The second approach transforms a given input to a logical formula and decides the problem by model counting using a #SAT solver. We discuss the effectiveness and scalability of the two approaches based on the experimental results.**

Keywords–Database Privacy; Diversity; Inference Attack; Relational Databases.

I. INTRODUCTION

Database security is one of the most important challenges to minimize the leakage of the sensitive information over the accesses or the data publishing, which have been growing rapidly and more powerful. Access control is a traditional mechanism for confidentially restricting accesses to a database made by a user by dividing the queries into authorized and unauthorized ones, and restricting the portion of the data that can be retrieved and updated by the user.

Inference attack is a malicious way to infer the sensitive information protected by access control. An attack is conducted by combining the result of authorized queries, the code (the meaning) of the queries and other available external information to obtain the candidate values of the sensitive information, as shown in Example 1. Thus, we need an appropriate quantitative notion of the security of a database against inference attacks.

Example 1: Table I shows a database instance consisting of six tuples. Assume that {Zipcode, Gender, Age} is the quasi-identifier and Diagnosis is the sensitive attribute. Assume that query (1) extracts tuples of {Zipcode, Age} where "Age" ≤ 60 from Table I, as shown in Table II, and query (2) extracts tuples of {Age, Diagnosis}, as shown in Table III. If we combine two results of queries (1) and (2), we can obtain candidate set of the type of "Diagnosis", as shown in Table IV. This cannot guarantee that this database instance is safe against inferencing.

As described in Related Work, there are well-known related notions such as k-anonymity and ℓ-diversity. Intuitively, a database is k-anonymous if for every individual x, there are at least k different records (or tuples in the relational database setting) which cannot be distinguished from the real record for x. A database is ℓ-diverse if for every individual x, there are at least ℓ different values of the sensitive information contained in the records which cannot be distinguished from the real record for x. Also, various methods are reported for transforming a given database into a database satisfying k-anonymity or ℓ-diversity. However, these notions do not take the effect of access control for queries into consideration.

TABLE I. A SAMPLE INSTANCE.

Zipcode	Gender	Age	Diagnosis
123-4567	F	45	A
123-5235	F	44	B
123-4567	F	44	C
378-2102	M	65	A
378-2102	M	62	B
378-2102	F	65	A

TABLE II. A RESULT OF QUERY (1).

Zipcode	Age
123-4567	45
123-5235	44
123-4567	44

TABLE III. A RESULT OF QUERY (2).

Age	Diagnosis
45	A
44	B
44	C

TABLE IV. A RESULT OF QUERIES (1) AND (2).

Zipcode	Age	Diagnosis
123-4567	45	A
123-5235	44	B
123-5235	44	C
123-4567	44	B
123-4567	44	C

The goal of this study is to introduce a notion of the security against inference attack by extending ℓ-diversity in relational databases with access control for queries. More specifically, we propose a new privacy notion called *query-based ℓ-diversity*. A database instance T is ℓ-diverse with respect to given authorized queries if an attacker cannot narrow down the number of possible values of the sensitive information for any individual to less than ℓ by inference based on the result of the authorized queries on the instance T and the queries themselves. We provide two approaches to deciding the query-based ℓ-diversity. In the first approach, a decision algorithm is given by using relational operations, which can

 ISBN: 978-1-61208-408-4

be directly implemented by a relational database management system, e.g., SQL. The second approach transforms a given input to a logical formula and decides the problem by model counting using a #SAT solver. We discuss the effectiveness and scalability of the two approaches based on the experimental results.

A. Related Work

The anonymization technique enforces the preservation of privacy of personal data or sensitive information. Generalization of the data is one of the well-known techniques for anonymizing information accordingly with the domain generalization hierarchy from which the quasi-identifier value can be generalized such as numeric values are generalized to intervals. There are a few well-known notions for database privacy, k-anonymity [11][13], ℓ-diversity [8] and t-closeness [7]. These notions assume the following basic concepts on relational databases. The set of attributes are divided into sensitive and nonsensitive attributes. Also, a subset of the nonsensitive attributes, called the quasi-identifier, is assumed. The value of the quasi-identifier is potentially used to identify the tuple of a target individual by linking the disclosed information with external data.

k-anonymity A database instance satisfies k-anonymity if for any value of the quasi-identifier, there are k or more tuples having that value of the quasi-identifier. A maximal subset of tuples having same values of the quasi-identifier is called an equivalence class. k-anonymity means that the cardinality of each equivalence class is at least k. A transformation of a given instance to another instance satisfying k-anonymity is called a k-anonymization for the original instance. [13] proposed a method for k-anonymization by hiding some information of individuals by generalization and suppression. Generalization replaces a value with less specific but semantically consistent value. While suppression hides the data or does not release the entire data. Various anonymization methods have been reported using clustering, branch-and-bound search, and so on [1][2]. k-anonymity is a simple notion and has been frequently used. As discussed in [8], however, a k-anonymous database may still have some issues because the database may lack the diversity in the sensitive attributes. ℓ-diversity has been proposed by [8] to overcome the weakness of k-anonymity. Though [8] proposes a general definition of ℓ-diversity, we just review a simple and frequently used one, called distinctive non-recursive ℓ-diversity.

ℓ-diversity A database instance satisfies distinctive non-recursive ℓ-diversity (or simply, ℓ-diversity) if for each equivalence class, there are at least ℓ different values of the sensitive attributes.

Example 2: As shown in Table I, all tuples have different values of the quasi-identifier and hence this database instance does not satisfy k-anonymity for any $k \geq 2$. Assume that in this database instance, the lower four digits of the values of "Zipcode" are hidden, the values of "Gender" is hidden and the values of "Age" are generalized to the intervals of ten years. Then we obtain the database instance shown in Table V. This instance consists of two equivalence classes and the number of tuples in each class is three. Hence, this instance satisfies 3-anonymity and the above transformation is a 3-anonymization for the original instance. Also, the first class has three different values of the sensitive attribute and the second class has two different values. Hence, the transformed instance satisfies 2-diversity but does not satisfy ℓ-diversity for any $\ell \geq 3$.

TABLE V. A 3-ANONYMOUS AND 2-DIVERSE INSTANCE.

Zipcode	Gender	Age	Diagnosis
123-****	-	[40,49]	A
123-****	-	[40,49]	B
123-****	-	[40,49]	C
378-****	-	[60,69]	A
378-****	-	[60,69]	B
378-****	-	[60,69]	A

k-secrecy A related but different quantitative notion on database security is given in [5] based on access control on queries. Assume that a database instance T of a schema $\mathbf{R}$, authorized queries $q_1, q_2, \ldots, q_m$ and an unauthorized query q_U are given. An attacker knows $\mathbf{R}$, $q_1, q_2, \ldots, q_m, q_U$ (the meaning of the authorized and unauthorized queries) and $q_1(T), \ldots, q_m(T)$ (the result of the authorized queries), but he does not know T. The goal of the attacker is to obtain $q_U(T)$, which is the result of the unauthorized query q_U on T. For a positive integer k, a database instance T is k-secret with respect to $\mathbf{R}, q_1, \ldots, q_m, q_U$ if the attacker cannot narrow down the number of the candidates of $q_U(T)$ to less than k. T is ∞-secret if the candidates of $q_U(T)$ are infinite. We say that a database schema $\mathbf{R}$ is k-secret with respect to $q_1, \ldots, q_m, q_U$ if every database instance of $\mathbf{R}$ is k-secret. [5] showed that k-secrecy is decidable for XML databases where queries are given as tree transducers in a certain subclass that can use relabeling and deletion. Also, [10] showed that the problem for deciding whether a given XML schema is k-secret is undecidable for any finite k while the problem is decidable when $k = \infty$. Although [5] deals with XML databases, the notion of k-secrecy is general enough for other kinds of databases. More sophisticated notions have been also proposed. For example, [3][9] proposed stronger notions where the probability distribution of possible secrets does not change after observing (authorized) information. The notion of query-based ℓ-diversity proposed in this paper is a combination of k-secrecy and ℓ-diversity in the relational database setting.

The organization of this paper is as follows. Section II provides basic notions and notations on relational database that will be used in the paper. We define the query-based ℓ-diversity in Section III. In Section IV, a decision algorithm based on relational operations as the first approach is given. Section V provides the second approach based on model counting. Experimental results conducted on SQL for the first approach and using a #SAT solver sharpCDCL for the second approach are shown in Section VI. We conclude the paper in Section VII.

II. MODELS

In this section, we introduce a simple relational database model , which will be used in the rest of the paper. A relational database instance (or simply a database) can be seen as a *table*, of which columns are *attributes*. There are two types of attributes, namely *sensitive* and *nonsensitive* attributes. The values of sensitive attributes are considered as secret, that is, the data owner keeps them confidentially and restrictively and protects them from unauthorized accesses.

Definition 1: A *relational database schema* (or simply a schema) is a finite set of attributes. Let $\mathbf{R} = \{A_1, \ldots, A_n\}$ be

 ISBN: 978-1-61208-408-4

a schema. We assume that for each attribute A_i $(1 \leq i \leq n)$, a finite set of values, denoted by $dom(A_i)$ is associated. A *tuple* (or a record) over $\mathbf{R}$ is $t = (d_1, \ldots, d_n)$ where $d_i \in dom(A_i)$ for each $1 \leq i \leq n$. Let $t[A_i] = d_i$, which is called the value of attribute A_i in t. That is, $t = (t[A_1], \ldots, t[A_n])$. A *relational database instance* (or simply an instance) of $\mathbf{R}$ is a finite set of tuples over $\mathbf{R}$. An instance is sometimes called a *table*. Let $I[\mathbf{R}]$ denote the set of all instances of $\mathbf{R}$.

Let $\mathbf{R}$ be a schema. We assume that $\mathbf{R}$ is divided into two disjoint subsets, namely, $\mathcal{S}e$ and $\mathcal{NS}e$, which are the set of *sensitive attributes* and the set of *nonsensitive attributes*, respectively. We furthermore assume that a subset $Qi \subseteq \mathcal{NS}e$ of nonsensitive attributes is given as a *quasi-identifier* of $\mathbf{R}$. We intend that the values of the quasi-identifier can be potentially used to identify the values of the sensitive attributes by linking the attribute values of the quasi-identifier with external data sets.

We define projection, selection and join in the usual way. Let $\mathbf{R}$ be a schema. For a subset of attributes $\alpha = \{A_{j_1}, \ldots, A_{j_m}\} \subseteq \mathbf{R}$ and a tuple t over $\mathbf{R}$, let $\pi_\alpha(t)$ denote the tuple $(t[A_{j_1}], \ldots, t[A_{j_m}])$, which is called the projection of t on α. Also, for an instance $T \in I[\mathbf{R}]$, let $\pi_\alpha(T) = \{\pi_\alpha(t) \mid t \in T\}$. Let $T_1 \in I(\mathbf{R}_1)$ and $T_2 \in I(\mathbf{R}_2)$. For a filtering condition F, and an instance T, let $\sigma_F(T)$ denote the set of tuples in T that satisfy F. The natural join of T_1 and T_2 is the instance obtained by "linking" every possible pair of tuples in T_1 and T_2:

$$T_1 \bowtie T_2 = \{t \textit{ over } \mathbf{R}_1 \cup \mathbf{R}_2 \mid \textit{for some } u \in T_1, \\ w \in T_2,\ t[U] = u \textit{ and } t[W] = w\}. \quad (1)$$

Since the natural join operator is associative and commutative, we sometimes view the natural join as a polyadic operator and write $T_1 \bowtie .... \bowtie T_m$.

III. Proposed Framework

In order to provide the definitions, we need to introduce the candidate set of instances of which results of queries are the same as those of the real instance. For a given instance T and queries $q_1, ..., q_m$, let $cand(q_1, ..., q_m, T)$ be the set consisting of all instances that give the same result as T with respect to all queries $q_1, ..., q_m$:

$$cand(\mathbf{R}, q_1, ..., q_m, T) = \{T' \in I(R) \mid \forall i(1 \leq i \leq m) \cdot \\ q_i(T) = q_i(T')\}. \quad (2)$$

Each $T' \in cand(\mathbf{R}, q_1, ..., q_m, T)$ is called a candidate instance.

Let T be a database instance over schema $\mathbf{R}$. T is query-based ℓ-diverse if for each maximal subset of a candidate instance of which tuples have the same quasi-identifier, there are ℓ or more different values of the sensitive attributes.

Suppose that the following information is available to public: a database schema $\mathbf{R}$, authorized queries $q_1, ..., q_m$, quasi-identifier Qi, sensitive attributes Se and a threshold ℓ (a positive integer). Let T be an instance of $\mathbf{R}$. An attacker infers sensitive information by taking the natural join of the results of the authorized queries $q_1, ..., q_m$ on the instance T to obtain the candidate set of sensitive information. We now show three options for the definition of query-based ℓ-diversity as follows.

Definition 2: An instance $T \in I(\mathbf{R})$ is l-diverse (with respect to $\mathbf{R}, Qi, Se, q_1, ..., q_m, T$)

(Option 1) if for every $t \in \pi_{Qi}(T)$,

$$|\{\pi_{Se}(t') \mid {}^{\exists}T' \in cand(q_1, ..., q_m, T) \cdot \\ (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l, \quad (3)$$

(Option 2) if for every $t \in \pi_{Qi}(T)$, there is an instance $T' \in cand(q_1, ..., q_m, T)$ such that

$$|\{\pi_{Se}(t') \mid (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l, \quad (4)$$

(Option 3) if there is $T' \in cand(q_1, ..., q_m, T)$ such that for every $t \in \pi_{Qi}(T)$,

$$|\{\pi_{Se}(t') \mid (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l. \quad (5)$$

By definition, (5) implies (4), and (4) implies (3).

A conjunctive query consists of projection, selection and join. In our proposed framework, we assume self-join free conjunctive queries.

Definition 3: A query q on $\mathbf{R}$ is *monotonic* if for any $T_1, T_2 \in I(\mathbf{R})$, $T_1 \subseteq T_2$ implies $q(T_1) \subseteq q(T_2)$.

Lemma 1: Every conjunctive query is monotonic.

If we restrict the class of queries to self-join free conjunctive queries, all the three definitions of ℓ-diversity become equivalent as stated in the next theorem.

Theorem 1: If we assume self-join free conjunctive queries, then the three options in definitions 2 become equivalent.

Proof: By the following properties 1 and 2. ■

Property 1: For any instances T_1, T_2 and a self-join free conjunctive query q,

$$q(T_1 \cup T_2) = q(T_1) \cup q(T_2).$$

Proof: Let T_1, T_2 be instances and q be a self-join free conjunctive query. By Lemma 1, q is monotonic and hence $q(T_1 \cup T_2) \supseteq q(T_1) \cup q(T_2)$ holds. Since q does not contain self-join, $q(T_1 \cup T_2) \subseteq q(T_1) \cup q(T_2)$ also holds. ■

Property 2: Let T be an instance and $q_1, \ldots, q_m$ be self-join free conjunctive queries. The largest candidate set in $cand(q_1, ..., q_m, T)$ (with respect to set inclusion) is the union of all instances in $cand(q_1, ..., q_m, T)$.

Proof: Let $T_c = \bigcup_{T' \in cand(q_1, ..., q_m, T)} T'$. By Property 1,

$$\begin{aligned} q_i(T_c) &= \bigcup_{T' \in cand(q_1, ..., q_m, T)} q_i(T') \\ &= \bigcup_{T' \in cand(q_1, ..., q_m, T)} q_i(T) \\ &= q_i(T)\ (1 \leq i \leq m). \end{aligned}$$

Hence, $T_c \in cand(q_1, ..., q_m, T)$. Apparently, T_c is the largest set in $cand(q_1, ..., q_m, T)$. ■

We define the query-based ℓ-diversity problem as follows:

Input : A schema $\mathbf{R}$, an instance $T \in I(\mathbf{R})$, authorized queries $q_1, ..., q_m$, quasi-identifier $Qi \subseteq \mathbf{R}$, sensitive attributes $Se \subseteq \mathbf{R}$, and a threshold $\ell \geq 1$.

Output : T is query-based ℓ-diverse or not with respect to $\mathbf{R}$, Qi, Se, $q_1, ..., q_m$.

IV. VERIFICATION BY RELATIONAL ALGEBRA

In this section, we describe our verification algorithm that solves the query-based ℓ-diversity problem. For simplicity, we only focus on projection queries. The algorithm can be extended to deal with join without self-join. However, selection cannot be allowed. Also, we assume that the set of sensitive attributes is not empty.

We assume that an attacker knows the domain of each attribute in $\mathbf{R}$, specially the domains of the sensitive attributes, so that he can infer a candidate instance by adding values of the sensitive attributes chosen from the domain even if (some of) the sensitive attributes are missing in the result of queries $q_1, ..., q_m$.

Our algorithm consists of four steps as follows:

1) Obtain the candidate set of tuples T' by taking the natural join of all results $q_1(T), ..., q_m(T)$ as follows.
$$T' = q_1(T) \bowtie ... \bowtie q_m(T).$$
2) Let Qi' ($\subseteq Qi$) be the set of quasi-identifier that exist in T'. Compute the subset T_c of T' consisting of tuples whose quasi-identifier value belongs to the original instance T.
$$T_c = T' \bowtie \pi_{Qi'}(T).$$
3) Divide T_c into subsets (equivalence classes) $g_1, \ldots, g_h$ such that
   a) $\pi_{Qi'}(t) = \pi_{Qi'}(t')$ for any $t, t' \in g_i$ $(1 \leq i \leq h)$ and
   b) $\pi_{Qi'}(t) \neq \pi_{Qi'}(t')$ for any $t \in g_i$ $(1 \leq i \leq h)$ and $t' \in g_j$ $(1 \leq j \leq h)$ with $i \neq j$.
4) Let mis_Se be the set of sensitive attributes that does not exist in T'. With mis_Se and the threshold ℓ, decide whether T is ℓ-diverse by examining the following necessary and sufficient condition for ℓ-diversity:
   $\forall g_i$ $(1 \leq i \leq h)$,
$$|g_i| \times \prod_{a \in mis\_Se} | \, dom(a) \, | \geq \ell. \qquad (6)$$

In the last step, the number of different sensitive values in each equivalence class g_i $(1 \leq i \leq h)$ is computed by using the domains of the missing sensitive attributes mis_Se. If for every equivalence class g_i $(1 \leq i \leq h)$, the left-hand side of (6) is greater than or equal to the threshold ℓ, the algorithm answers that the given input is ℓ-diverse. If there is at least one equivalence class g_i such that the left-hand side of (6) is less than ℓ, the algorithm answers that the given input is not ℓ-diverse.

V. VERIFICATION BY MODEL COUNTING

In this section, we provide another method for deciding the query-based ℓ-diversity. The method transforms a given input of the problem to a logical formula, and decides the problem by model counting using a #SAT solver. The advantage of this method is that it can handle self-join free conjunctive queries, consisting of projection, selection and join without self-join. Henceforth, we assume queries in the class.

Before we explain our method, we give some definitions. For a formula Ψ, let $\#models(\Psi)$ denote the number of different models (assignments to variables that make Ψ true). If a formula Ψ contains only variables in Σ, we call Ψ a Σ-formula. For a Σ-formula Ψ and $\Delta \subseteq \Sigma$, let $\Psi_{|\Delta}$ denote the strongest Δ-formula implied by Ψ when considered as a Σ-formula where A is stronger than B if and only if $A \Rightarrow B$ holds. We say that $\Psi_{|\Delta}$ is the projection of Ψ onto Δ.

Assume that a schema $\mathbf{R}$ where $n = |\mathbf{R}|$, an instance $T \in I(\mathbf{R})$, queries $q_1, \ldots, q_m$ on $\mathbf{R}$, quasi-identifies $Qi \subseteq \mathbf{R}$, sensitive attributes $Se \subseteq \mathbf{R}$, and a threshold l are given. For simplicity, suppose that $Qi = \{A_1, \ldots, A_k\} \subseteq \mathbf{R}$, and $Se = \{A_{k+1}, \ldots, A_m\} \subseteq \mathbf{R}$ where $1 \leq k < m < n$. The summary of the method is as follows.

1) Construct a logical formula $\Phi(x_1, \ldots, x_n)$ such that
   $\Phi(c_1, \ldots, c_n)$ is satisfiable
   if and only if $(c_1, \ldots, c_n) \in T_c$ $(*)$
   Note that $\phi(x_1, \ldots, x_n)$ has free variables other than $x_1, \ldots, x_n$ in general.
2) Decide if for all tuple $(c_1, \ldots, c_k) \in \pi_{Qi}(T)$,
$$\#models(\Phi_p(x_{k+1}, \ldots, x_n)_{|Xs}) \geq l.$$
   where $Xs = \{x_{k+1}, \ldots, x_m\}$ and
$$\Phi_p(x_{k+1}, \ldots, x_n) = \Phi(c_1, \ldots, c_k, x_{k+1}, \ldots, x_n).$$

1) Constructing Constraint: Let $n = |\mathbf{R}|$ and $n_i = |\mathbf{R_i}|$ where $\mathbf{R_i}$ is the output schema of q_i $(1 \leq i \leq m)$. To construct a formula $\Phi(x_1, \ldots, x_n)$ satisfying $(*)$, we first construct subformulas ϕ_{q_i} and O_{q_i} for $1 \leq i \leq m$.

(1-i) For $1 \leq i \leq m$, ϕ_{q_i} represents the input-output relation of the query q_i. The formula ϕ_{q_i} contains free variables $x_1, \ldots, x_n, y_1, \ldots, y_{n_i}$ and satisfies:

for any $t = (c_1, \ldots, c_n)$ and $t'_i = (d_1, \ldots, d_{n_i})$, $\phi_{q_i}(c_1, \ldots, c_n, d_1, \ldots, d_{n_i})$ is satisfiable if and only if $q_i(\{t\}) \subseteq \{t'\}$.

(Construction)
If $q = T$ then
$$\phi_q(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigwedge_{i=1}^{n} (x_i = y_i).$$

projection: If $q = \pi_\alpha(q')$ where $\alpha = \{A_{j_1}, \ldots, A_{j_{n'}}\}$,
$$\phi_q(x_1, \ldots, x_{n_I}, z_1, \ldots, z_{n'}) = \phi_{q'}(x_1, \ldots, x_{n_I}, y_1, \ldots, y_{n_O}) \wedge \bigwedge_{i=1}^{n'} (y_{j_i} = z_i).$$

selection: If $q = \sigma_F(q')$,
$$\phi_q(x_1, \ldots, x_{n_I}, z_1, \ldots, z_{n_O}) = \phi_{q'}(x_1, \ldots, x_{n_I}, y_1, \ldots, y_{n_O}) \wedge \left(P_F(y_1, \ldots, y_{n_O}) \Rightarrow \bigwedge_{i=1}^{n_O} (y_i = z_i) \right).$$

where $P_F(y_1, \ldots, y_{n_O})$ is a formula representing the filtering condition F of σ_F.

 ISBN: 978-1-61208-408-4

cross product: If $q = q' * q''$,

$$\begin{aligned}\phi_q(x_1,\ldots,x_{n'_I},x'_1,\ldots,x'_{n''_I},z_1,\ldots,z_{n'_O+n''_O}) \\ = \ \phi_{q'}(x_1,\ldots,x_{n'_I},y_1,\ldots,y_{n'_O}) \\ \wedge\ \phi_{q''}(x'_1,\ldots,x'_{n'_I},y'_1,\ldots,y'_{n''_O}) \\ \wedge \bigwedge_{i=1}^{n'_O}(y_i = z_i) \wedge \bigwedge_{i=1}^{n''_O}(y'_i = z_{n'_O+i}).\end{aligned}$$

(1-ii) O_{q_i} is defined as

$$\begin{aligned}O_{q_i}(y_1,\ldots,y_{n_i}) \\ = \bigvee_{(d_1,\ldots,d_{n_i})\in q_i(T)} ((y_1 = d_1) \wedge \cdots \wedge (y_{n_i} = d_{n_i})).\end{aligned}$$

(1-iii) Finally, Φ is defined as

$$\begin{aligned}\Phi(x_1,\ldots,x_n) \\ = \bigwedge_{i=1}^{m}(\phi_{q_i}(x_1,\ldots,x_n,y_{i,1},\ldots,y_{i,n_i}) \wedge O_{q_i}(y_{i,1},\ldots,y_{i,n_i})).\end{aligned}$$

Remember that in the algorithm of the previous section, we introduce the subsets $g_1,\ldots,g_h$, each of which shares same values of the quasi-identifier. For g_j ($1 \leq j \leq h$), let $(c^j_1,\ldots,c^j_k)$ be the values of the quasi-identifier shared by tuples in g_j. Let $\Phi^j_p(x_{k+1},\ldots,x_n) = \Phi(c^j_1,\ldots,c^j_k,x_{k+1},\ldots,x_n)$. By $(*)$, $\Phi^j_p(c_{k+1},\ldots,c_n)$ is satisfiable if and only if $(c^j_1,\ldots,c^j_k,c_{k+1},\ldots,c_n) \in g_j$. Furthermore, $\Phi^j_p(x_{k+1},\ldots,x_n)_{|Xs}$ is the strongest Xs-formula implied by $\Phi^j_p(x_{k+1},\ldots,x_n)$. Hence, the number of assignments to variables in Xs that make $\Phi^j_p(x_{k+1},\ldots,x_n)_{|Xs}$ true coincides with the number of different values of Se appearing in tuples that belong to g_j. Hence, we obtain the following lemma.

Lemma 2: Let $\mathbf{R}$ be a schema, $Qi, Se \subseteq \mathbf{R}$ be the quasi-quantifier and sensitive attributes, respectively, $q_1,\ldots,q_m$ be self-join free conjunctive queries on $\mathbf{R}$ and $T \in I(\mathbf{R})$ be an instance. Let $g_1,\ldots,g_h$ be the subsets of T_c, each of which shares same values for the quasi-identifier. For each j ($1 \leq j \leq h$), the number of different values of sensitive attributes in g_j is

$$\#models(\Phi^j_p(x_{k+1},\ldots,x_n)_{|Xs}).$$

2) Counting Candidates: To count the different values of sensitive attributes for each g_j ($1 \leq j \leq h$), we transform $\Phi^j_p(x_{k+1},\ldots,x_n)$ to an equivalent propositional formula Φ^j_{cnf} in conjunctive normal form (CNF) by using Sugar [12]. Next, for each $t' = (c_1,\ldots,c_k) \in \pi_{Qi}(T)$, we construct a CNF formula $\psi_{t'}$ that represents $x_1 = c_1 \wedge \cdots \wedge x_k = c_k$, and then count $\#models(\Phi^j_{cnf} \wedge \psi_{t'})_{|P(Xs)}$, where $P(Xs)$ is the set of the propositional variables in Φ^j_{cnf} corresponding to Xs in $\Phi^j_p(x_{k+1},\ldots,x_n)$. We use sharpCDCL [6], which is a #SAT solver (an automatic tool for counting the models of a given propositional formula). Among other #SAT solvers that can count models, the advantage of sharpCDCL is that it can automatically count $\#models(\Psi_{|\Delta})$ only by giving a formula Ψ and a subset Δ of propositional variables. If some $t' \in \pi_{Qi}(T)$ such that $\#models(\Phi^j_{cnf} \wedge \psi_{t'})_{|P(Xs)} < \ell$ is found, we say that T is not ℓ-diverse. Otherwise, T is ℓ-diverse.

VI. EXPERIMENTS

A. Experimental Result of Relational Algebra

The purpose of the experiment was to investigate the scalability of our approach.

1) Setup: Experiment were performed on a 3.33 GHz Intel(R) Core(TM) i7 CPU with 6GB of RAM. The operating system was Microsoft Windows 8.1 Enterprise, and implementation was built and run in MySQL Workbench, version 6.1. We used available dataset, Employees Sample Database [14], Copyright (C) 2007, 2008, MySQL AB, version 1.0.6. The database contains about 300,000 tuples with 2.8 million salary entries. In our experiment, the schema consists of ten attributes, where five attributes $\{Gender,\ DeptName,\ BirthDate,\ HireDate,\ FromDate\}$ were designated as the quasi-identifier and the sensitive attribute is $\{Salary\}$.

2) Datasets and Queries: The proposed algorithm was implemented in MySQL and was performed on three instances (datasets) with $n = 37,500, 75,000, 150,000, 300,000$ tuples. Also, we prepared three queries, each of which is the projection onto the following attributes:

$q_1 : \{EmpNo, LastName, Gender\}$.
$q_2 : \{EmpNo, Salary, HireDate\}$.
$q_3 : \{DeptName\}$.

In the experiment, we used three sets of queries, namely, $\mathbf{Q_A} = \{q_1, q_2\}$, $\mathbf{Q_B} = \{q_3\}$, and $\mathbf{Q_C} = \{q_1, q_2, q_3\}$. For example, for $\mathbf{Q_A}$, the verification algorithm took the natural join of the results of q_1 and q_2 on each of the datasets in Step 1. In Step 3, the algorithm constructed the table from the candidate set T_c obtained in Step 2 by grouping tuples that have same values of the quasi-identifier. Lastly, in Step 4, the algorithm tested ℓ-diversity ($\ell = 2$ in the experiment).

TABLE VI. TOTAL TIME OF VERIFYING 2-DIVERSITY.

Dataset	Cases	Total time
37,500	q_1, q_2	7sec
	q_3	4sec
	q_1, q_2, q_3	11sec
75,000	q_1, q_2	19sec
	q_3	8sec
	q_1, q_2, q_3	31sec
150,000	q_1, q_2	1min 7sec
	q_3	14sec
	q_1, q_2, q_3	1min 47sec
300,000	q_1, q_2	4min 8sec
	q_3	26sec
	q_1, q_2, q_3	8min 48sec

Table VI shows the total running time of our algorithm. For example, for $\mathbf{Q_A}$ and the dataset n = 37,500, the total running time is 7sec. Also the running time of each set of queries, $\mathbf{Q_A}, \mathbf{Q_B}$ and $\mathbf{Q_C}$ on the datasets of size $n = 37,500, 75,000, 150,000$ and $300,000$. We can observe that the decision algorithm is efficient, in general, and also the computation time depends on the size of the datasets.

B. Experimental Result of Model Counting

1) Setup: Experiment were performed on a 3.10 GHz Intel(R) Core(TM) i5 CPU with 8GB of RAM. The operating system was Ubantu 14.04. We performed the experiment on a dataset, having 50,000 tuples.

2) Datasets and Queries: In the experiment, we used two instances T_1 and T_2, having ten and eleven attributes, respectively. Both of T_1 and T_2 have $5,000$ tuples. Actually, T_1 was obtained from T_2 by projecting out one of the eleven attributes. We conducted the experiment on the following two settings:

$D1$: a query $\sigma_{state=Iwate}(T_1)$, $Qi = \{ID, state\}$ and $Se = \{Name\}$.

$D2$: two queries $\Pi_{BirthYear,BirthMonth}(T_2)$, $\Pi_{BirthYear,BirthMonth}(\sigma_{Carrier=SoftBank}(T_2))$, $Qi = \{ID, BirthMonth, Carrier\}$ and $Se = \{BirthYear\}$.

The experimental results for these settings are shown in Table VII where clauses and variables are those in the transformed CNF formula, projected variables are the variables corresponding to sensitive attributes, min count is the minimum number of different values of sensitive attributes among $g_1, \ldots, g_h$. That is, D1 is ℓ-diverse if and only if $\ell \leq 50$ and D2 is ℓ-diverse if and only if $\ell \leq 42$. Next, we

TABLE VII. PERFORMANCE OF MODEL COUNTING METHOD.

	Clauses	Variables	Projected variables	Min count	Time
$D1$	$34,271$	$14,916$	$4,961$	50	6min 15sec
$D2$	$22,941$	$11,121$	96	42	2min 1sec

increased the number of tuples in T_2 to $10,000$, $30,000$ and $50,000$ and examined the scalability of the proposed method by using the setting $D2$. The result is shown in Table VIII. In sharpCDCL, an upperbound U of the model counting can be specified. That is, when sharpCDCL detects that the current number of models reaches U, sharpCDCL terminates. The computation times in Table VIII are those when this upperbound is specified as $U = 20$. The transformation to a

TABLE VIII. SCALABILITY OF MODEL COUNTING METHOD.

Tuples	Clauses	Variables	Projected variables	Time
$5,000$	$22,941$	$11,121$	96	1min 58sec
$10,000$	$42,334$	$20,877$	96	10min 2sec
$30,000$	$117,557$	$58,541$	106	2hrs 28min 30sec
$50,000$	$187,820$	$93,633$	106	8hrs 56min 8sec

CNF formula takes less than one second, and model counting dominates the running time.

VII. CONCLUSION

We have introduced query-based ℓ-diversity as a privacy notion for a realistic database system that assumes access control for queries. This new notion inherits from ℓ-diversity of [8] the quantitative notion for the diversity of sensitive attributes. Also, the notion utilizes k-secrecy of [5] by taking attacker's inference on the authorized information into consideration.

We proposed two approaches to deciding whether a given database instance satisfies query-based ℓ-diversity with respect to given queries. The first approach is based on relational algebra computation that counts the candidate values of the sensitive attributes. The second approach transforms a given input to a logical formula and then decide the problem by counting models of a formula by a #SAT solver. The first approach can directly be implemented by an existing relational database system such as SQL, and the experimental results show that this approach is fairly efficient. The weakness is that it cannot deal with selection queries. The second approach, on the other hand, can deal with selection queries. However, the model counting in a #SAT solver is generally time consuming and we have not yet customized the solver to our problem and hence, the performance is not good compared with the first approach.

Applying the proposed approach to other kind of databases such as object-oriented or XML databases is left as a future study.

REFERENCES

[1] R. J. Bayardo and R. Agrawal, "Data Privacy Through Optimal k-Anonymization," the 10th International Conference on Database Theory (ICDT), 2005, pp. 217-228.

[2] J. W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient k-Anonymization Using Clustering Techniques," the 12th International Conference on Database Systems for Advanced Applications (DASFAA), 2007, pp. 188-200.

[3] A. Deutsch and Y. Papakonstantinou, "Privacy in Database Publishing," the 10th International Conference on Database Theory (ICDT), 2005, pp. 230-245.

[4] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-Preserving Data Publishing: A Survey of Recent Developments," ACM Computing Surveys, vol. 42, 2010, pp. 14:1-53.

[5] K. Hashimoto, K. Sakano, F. Takasuka, Y. Ishihara, and T. Fujiwara, "Verification of the Security Against Inference Attacks on XML Databases," IEICE Transactions on Information and Systems, vol. E92-D, 2009, pp. 1022-1032.

[6] V. Klebanov, N. Manthey, and C. Muise, "SAT-Based Analysis and Quantification of Information Flow in Programs," 10th International Conference on Quantitative Evaluation of Systems (QEST), 2013, pp. 177-192.

[7] N. Li, T. Li and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity, the 23rd IEEE International Conference Data Engineering (ICDE), 2007, pp. 106-115.

[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy Beyond k-Anonymity," 22nd International Conference on Data Engineering (ICDE), 2006, also in ACM Transactions on Knowledge Discovery from Data (TKDD), 2007, vol. 1(3), 52 pages.

[9] G. Miklau and D Suciu, "A Formal Analysis of Information Disclosure in Data Exchange," Journal of Computer and System Sciences, vol. 73, 2007, pp. 507-534.

[10] C. Phonharath, K. Hashimoto, and H. Seki, "Deciding Schema k-Secrecy for XML Databases," IEICE Transactions on Information and Systems, vol. E96-D, 2013, pp. 1268-1277.

[11] P. Samarati, "Protecting Respondents' Identities in Microdata Release," IEEE Transactions on Knowledge and Data Engineering, vol. 13, 2001, pp. 1010-1027.

[12] "A SAT-Based Constraint Solver," URL: http://bach.istc.kobe-u.ac.jp/sugar/[accessed: 2015-01-23]

[13] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol. 10, 2002, pp. 557-570.

[14] F. Wang and C. Zaniolo, "Employees Sample Database," 2008, URL: https://launchpad.net/test-db/[accessed: 2014-11-13].

 ISBN: 978-1-61208-408-4

Exposing the Myth: Object-Relational Impedance Mismatch is a Wicked Problem

Christopher Ireland, David Bowers
Department of Maths and Computing
The Open University
United Kingdom
e-mail: cjireland@btinternet.com, D.S.Bowers@open.ac.uk

***Abstract*—Addressing a problem of software integration is a fact of life for those involved in software development. The popularity of both object and relational technologies means that they will inevitably be used together. However, the combination of these two technologies introduces problems. These problems are referred to collectively as the object-relational impedance mismatch.**

A mismatch is addressed using one or more mapping strategies, typically embodied in a pattern. A strategy is concerned with correspondence between the schema of a relational database and an object-oriented program. Such strategies are employed in mapping tools such as Hibernate and TopLink, and reinforce the received wisdom that the problem of object-relational impedance mismatch has been solved.

In this paper, we observe that it is not clear whether each strategy, as one possible solution, addresses the cause or a symptom of a mismatch. We argue that the problem is not tame and easily resolved; rather it is complex and wicked. We introduce a catalogue of problem themes that demonstrate the complex nature of the problem and provide a way both to talk about the problem and to understand its complexity.

In the future, as software systems become more complex and more connected, it will be important to learn from past endeavours. Our catalogue of problem themes represents a shift, in thinking about the problem of object-relational impedance mismatch, from issues of implementation towards an analysis of cause and effect. Such a shift has implications for those involved in the design of current and future software architectures. Because we have questioned the received wisdom, we are now in a position to work toward an appropriate solution to the problem of object-relational impedance mismatch.

Keywords—object-relational; impedance mismatch; wicked problem; problem theme

I. INTRODUCTION

Addressing a problem of software system integration is a fact of life for those involved in software development [26], p46. Typically, an organization will employ a number of software systems, possibly written using different programming languages, each to a separate design, and running on different operating systems on different hardware platforms. Each software system will support different facets of the organisation's business activities.

An object-relational application is a software system that combines technologies based on the concepts of both " object" and "relation". Object-relational impedance mismatch is the term we use to refer to a difference between the schema of an object-oriented program and the schema of a relational database. Despite the received wisdom that the problem of object-relational mismatch has been "solved", reinforced by technologies such as Hibernate [2], TopLink [3] and LINQ [4], the resolution of a mismatch typically involves some form of object-relational mapping, and costs significant time and effort to address.

In this paper, we explore the nature of object-relational impedance mismatch. We demonstrate that, contrary to the received wisdom, the problem of object-relational impedance mismatch is not tame and easily resolved but, rather, it is wicked and complex. We provide a new way both to talk about the problem and to understand its complexity. Such understanding provides a sound foundation for work toward an appropriate solution to the problem.

This paper is structured as follows: in Section II. we illuminate the received wisdom. In Section III we expose the wicked nature of impedance mismatch. In Section IV we introduce a catalogue of problem themes as a lens through which we can understand the problem. In sections V and VI we use problem themes to demonstrate the complex nature of impedance mismatch and expose relationships between themes. Finally, in Section VII, we set out the limitations of a perspective on impedance mismatch based on problem themes, before presenting our conclusions and proposing future work in Section VIII.

II. OBJECT-RELATIONAL IMPEDANCE MISMATCH

An object-relational mismatch can occur only when an object-oriented program uses a relational database for persistence. A mismatch between an object-oriented program and a relational database does not materialise until a particular mapping strategy is selected. An object-relational mapping strategy (mapping strategy) sets out the correspondence between classes in the schema of an object-oriented program and the schema of a relational database. An object-relational application comprises many such strategies. However it is not always clear what each strategy addresses: the cause of or a symptom of a mismatch.

The problem of object-relational impedance mismatch is important in practice because addressing a mismatch costs both time and effort. Contrary to the suggestion of [1], the decoupling of a program and a database does not resolve a mismatch. Problems still occur at the point where objects and relations are combined, and they do not go away simply because a persistence layer [2][3] or a hybrid language [4] is used. A persistence layer embodies a number of mapping strategies. Such a layer will only address the cause of a mismatch if the mapping strategy employed is an *appropriate* solution.

 ISBN: 978-1-61208-408-4

In contrast we define an *acceptable* solution as one that gives the illusion that a mismatch is solved even if it addresses only a symptom of the mismatch. Such a solution might reinforce a belief that the mismatch has been avoided even though its cause has not been addressed. However, it should not be concluded that a mismatch is inevitable and that there is no alternative but to deal just with the symptoms. In the next section we explore the misconception that underpins this received wisdom by demonstrating the true nature of the problem of object-relational impedance mismatch.

III. A WICKED PROBLEM

Rittel and Webber [5] observe that some problems cannot be resolved in a linear way. They label such problems as wicked. A tame problem is particularly suited to a linear resolution because it is well defined; it has a clear and well-defined stopping point, when a solution is found from a list of possible solutions; and it is possible to choose a solution because there is a set of pre-defined criteria for making such a choice.

A wicked problem is less straightforward. Rittel and Webber describe ten characteristics of a wicked problem. In essence, a wicked problem is a problem that resists resolution because its definition is incomplete, the requirements of multiple stakeholders change, and there is no single definitive and optimal solution, so a choice of solution typically involves a compromise.

The concepts of tame and wicked problems represent the two extremes of a continuum along which a given problem may be positioned, depending on its particular characteristics. The characteristics of a wicked problem were derived from work on planning policy in the 1960s. Conklin [6], p21 subsequently refined them so they could be applied to areas other than planning policy. In Table I, we use each of Conklin's characteristics to explore the extent to which the problem of object-relational impedance mismatch may be considered wicked.

Object-relational impedance mismatch is an exemplar of a wicked problem. There is no single problem or solution. Each problem involves a number of stakeholders both within and outside an organisation, such as programmers, designers, analysts, software vendors and language designers. Furthermore, a problem is not addressed in isolation. The solution to a problem is a mapping strategy but each strategy involves a compromise because data about an object will not fit neatly into the schema of a relational database. Consequently, a choice of a particular strategy may cause another problem.

The search for a solution involves accepting compromises (or satisficing [6], p14). The result is a mapping strategy that produces the best fit rather than the optimal fit, and a solution that is somehow acceptable rather than appropriate. Furthermore, any choice of solution has implications for the design of an object-relational application. Consequently, we can think of an object-relational application as a complex collection of interrelated problems; what Ackoff [11] termed a *mess*.

Thinking about object-relational impedance mismatch as a wicked problem raises new questions about how we understand and address a mismatch. Such questions (Table I) expose issues with the received wisdom that the problem of object-relational impedance mismatch has been solved. We present next a new vocabulary to describe the problem of impedance mismatch. This vocabulary provides a way both to structure the mess and to understand the complex nature of the problem.

TABLE I. OBJECT-RELATIONAL IMPEDANCE MISMATCH FRAMED AS A WICKED PROBLEM

Characteristic of a Wicked Problem [6]	**The problem of Object-Relational Impedance Mismatch**
You don't understand the problem until you have developed a solution. Every solution exposes new aspects of the problem. There is no single definition of the problem instead an interlocking set of issues and constraints from different stakeholders.	There is no mismatch between an object-oriented program and a relational database until a decision is made to use a particular mapping strategy. There are many mismatches and there are many mapping strategies each of which may be a potential solution. Each solution involves a compromise [7]. There are issues such as those of a consistent identity and the preservation of semantics [8]. How do we understand the nature and consequence of a compromise?
Wicked problems have no stopping rule. There is no single definition of the problem and so there is no definitive solution.	A problem does not exist in isolation and a solution to one problem may cause another problem. There are a number of object, relational, and mapping technologies. A solution is chosen based on some criteria [9] but how do we know that these criteria are appropriate for making such a choice?
Solutions are not right or wrong simply better/worse/good enough. Stakeholders each interpret the solution based on their objectives.	Each solution involves a compromise either in the design of a program or in the design of a database. For example, Ambler [7], Chapter 14 lists a number of pros and cons for each mapping strategy. How then do we make an informed choice of an appropriate solution?
Each problem is essentially unique and novel because there are so many factors and conditions.	There are a number of mapping strategies but each must be interpreted in the context of a particular object-relational application. On the surface, defining a mapping strategy appears to be a straightforward activity. For example, a class corresponds to a table and an attribute corresponds to a column, but as [10] observes, a quagmire of issues rapidly develops. How then do we understand and avoid this quagmire?
Every solution is a one-shot operation. It has consequences and changes the context.	A choice of solution impacts the design of a program and the design of a database. Once a particular mapping strategy is implemented in an object-relational application how easy it is to adopt a different strategy?
There are no given alternative solutions. It is a matter of creativity to devise new solutions and a matter of judgement to decide which are valid and worth pursuing.	A one-solution-fits-all approach may not be appropriate but to what extent do we accept the available mapping strategies as a given? Are there other possibilities for a solution outside the code of an object-relational application?

IV. PROBLEM THEMES

Neward [12] refers to the problem of an object-relational impedance mismatch as "a quagmire of issues". In this section we set out to understand the problem of object-relational impedance mismatch. The objective is to make sense of the problem and the different interpretations of object-relational impedance mismatch.

Copeland & Maier [13], Neward [12] and Ambler [7], p105-113 each characterise object-relational impedance mismatch in a different way. Copeland & Maier are concerned with issues of concept and data structure, Neward focuses on problems of implementation and Ambler is concerned with technical and cultural difficulties.

It is not clear how each characterisation relates to the others, whether a particular characterisation refers to the cause of a mismatch or a symptom, whether the list of characterisations is complete, or why each characterisation was chosen. Ambler and Neward consider issues beyond those of technology but it is not clear whether Copeland & Maier, Ambler and Neward describe the cause or a symptom of a mismatch.

Each characterisation draws attention to a collection of mismatches that together represent a particular problem. In this section, we consolidate the characterisations described by Copeland & Maier, Neward and Ambler, along with contributions from others, as a catalogue of problem themes. An early version of this work can be found in [14][15].

A problem theme is defined as a collection of mismatches. A problem theme reflects a particular characterisation, such as the "object-to-table mapping problem" and the "schema ownership problem" described by Neward, or the "cultural impedance mismatch" described by Ambler, and helps to make sense of a collection of mismatches. A mapping strategy is one solution to a mismatch. It follows that a mapping strategy is also (part of) one solution to a problem theme. The relationships between a theme, a mismatch and a mapping strategy are summarised in Figure 1.

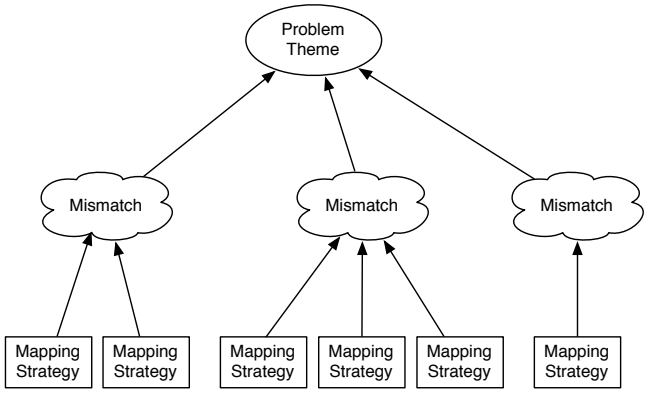


Figure 1. A Problem Theme, Mismatches and Mapping Strategies

A problem theme is important for two reasons. A problem theme provides a way to understand one aspect of object-relational impedance mismatch. It makes it possible to talk about and focus on a specific problem rather than use the general term object-relational impedance mismatch. Table II summarizes the concern of each problem theme.

In summary, a problem of object-relational impedance mismatch displays the characteristics of a number of problem themes. Problem themes provide a way to understand impedance mismatch. Each theme is concerned with a collection of mismatches. In the context of a problem theme it is possible to talk about the problem of a specific subtype of object-relational impedance mismatch. In the next section, we use the catalogue of problem themes to move toward an understanding of the complex nature of impedance mismatch.

TABLE II. PROBLEM THEMES

Problem Theme	Concern
Structure	The structure problem theme is concerned with any difference of data structure between the schema of an object-oriented program and the schema of a relational database, and so adopts a broad interpretation of the notion of structure. The essence of a structure problem is the extent to which an object-oriented data structure can be, and should be, described by a relational data structure. Problems of the structure theme are important because they are concerned with a description of the data processed by an object-relational application.
Instance	The essence of an instance problem theme is, where is the canonical copy of state located? Problems of the instance theme are important because they are concerned with the ownership of and the responsibility for data.
Encapsulation	The principle of encapsulation requires that the state of an object can be determined only by its behaviour, so in an object-oriented program the value of an attribute of an object is accessed via a method. Problems of encapsulation are important because, in a database, the value of a column in a row has no such protection. Consequently, once stored in a database, data may be changed without the protection of the semantics encoded in a method.
Identity	The essence of an identity problem is how to identify uniquely a collection of data values between both object-oriented program and a relational database. Such problems of identity are important to ensure the integrity of data between an object-oriented program and a relational database.
Processing Model	The essence of a processing model problem is how to represent in, maintain and retrieve from a database a sufficient set of objects for processing. Such problems are important because they concern issues of software performance [16].
Schema Ownership	The essence of the schema ownership problem is that the team who design and implement an object-oriented program can be different from the team who design and implement a relational database. Such problems are important because they concern the choices made by those responsible, respectively, for the object-oriented program and the relational database.

V. A COMPLEX MIX OF PROBLEMS

Problem themes classify mismatches that must be addressed during the development of an object-relational application. However, such concerns are not independent. We explore in this section relationships between problem themes. Each relationship is causal; collectively they describe the complex nature of object-relational impedance mismatch.

A structure problem can be the consequence of an ownership problem. A conceptual mismatch and a structural mismatch, as described by Copeland & Maier, are not independent. A conceptual framework determines the semantics of a language, so a language such as Java is referred to as an object-oriented language. Those who implement an object-relational application make a choice of abstraction but can use only the artefacts of a particular language to describe that abstraction.

 ISBN: 978-1-61208-408-4

Neward [12] refers to "the schema ownership conflict" and "the dual schema problem". Each demonstrates a relationship between a schema ownership problem and a structure problem. The schema ownership conflict describes a mismatch of agenda. For example, a performance issue might mean that those responsible for a database have to change a data structure [17]. The dual schema problem occurs when a database must be changed in order to accommodate another application. In this case a structure problem is caused by a solution to a schema ownership problem.

A choice of abstraction made in the design of one application can produce a structure problem in another. Keller [16] helps to reinforce a link between the schema ownership problem and the structure problem. Whilst Neward is concerned with accommodating a new application, for Keller the problem is incorporating an existing data structure, from another application, into the schema of an object-oriented program.

A schema ownership problem can cause an instance problem. Differences in perception between stakeholders of the role of a program and a database in an object-relational application bring into question the location of a canonical copy of state. A solution to a schema ownership problem must reconcile these different perceptions.

Problems of structure and identity are related. A choice of language will decide the data structure to which an identity refers. For example, in Java, an object has an identity whilst in SQL the value of a primary key represents the identity of a row. In order to address an identity problem it is important to be clear about the structure to which an identity refers.

However, a solution to an identity problem can then cause a structure problem. Keller [16] describes a solution to a correspondence of identity between the schema of an object-oriented program and the schema of a relational database. He addresses an identity problem by introducing a surrogate identifier, resulting in the need for a change to the structure of a database schema.

An instance problem can cause an encapsulation problem. Once data has been stored in a database, that data may be modified independently of the logic employed in a program. Such a change can occur if the instance problem is caused by a schema ownership problem whereby those responsible believe that a database maintains the canonical copy of state.

A structure problem can lead to an encapsulation problem. Lodhi [18] observes that the way an association is represented in a relational database can be different from an association between two objects. Consequently the representation of an association between objects as a foreign key in a relational database does not necessarily preserve the encapsulation of an object.

A structure problem can also lead to a processing model problem. The process of normalisation can cause data about an entity to be split across a number of tables. Consequently, in the context of a reference between two objects, in order to retrieve the data for a referenced object it may be necessary to join a number of tables.

An instance problem can be caused by a processing model problem. It may not be necessary to retrieve or store all the data about an object in order to satisfy a request. However it may still be necessary to retrieve all the data for an object in order to create that object. As a result those responsible for an object-oriented program might believe that a program maintains the canonical copy of state.

An encapsulation problem can lead to a processing model problem. In order to reference an object, a program must first create an object. It may not be desirable or practical to load data about all objects in a network from a relational database so a decision must be made at which point to stop. That decision is difficult because the network of references between objects is encapsulated within the objects themselves.

In summary, object-relational impedance mismatch is a complex mix of interrelated problems. Using problem themes we have explored this complexity and demonstrated that solving problems of any particular class can generate problems of another. Consequently each such problem cannot be addressed in isolation. In the next section we use these relationships to understand the mess and to explore the consequences for our understanding of impedance mismatch.

VI. Relationships Between Problem Themes

The previous section demonstrated that problem themes are related. These relationships are summarised in Figure 2.

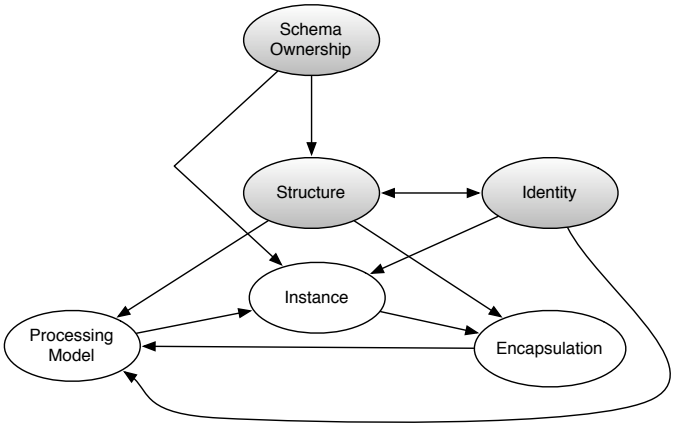


Figure 2. Problem Themes and Relationships

Two themes are related if a solution of one theme leads to a problem of another theme. The arrows on each line in Figure 2 indicate the direction of influence between two problem themes. It is possible to talk about a specific problem such as that of schema ownership, structure or identity and see that such problems are related. For example the line from the problem theme of structure to the problem theme of identity indicates that a solution to a structure problem can have a consequence for an identity problem.

Because it is possible to make a connection between themes it is also possible to explore the complex nature of object-relational impedance mismatch. For example, in order to address an identity problem it might be necessary to first address a structure problem, but a structure problem might be caused by a schema ownership problem. By exposing such relationships between themes it is possible to begin to understand the problem of object-relational impedance mismatch in a systematic way.

Because the solution to one problem can cause another problem, it follows that a problem should not be considered in

 ISBN: 978-1-61208-408-4

isolation. The relationships between problem themes help us to understand problems of object-relational impedance mismatch.

Figure 2 shows that a structure problem can be caused by a schema ownership problem. It is important that a solution to a structure problem involves both those responsible for an object-oriented program and those responsible for a relational database. Because a solution to one problem can lead to another, the relationships between themes provide a way for those responsible for a solution to understand with whom to consult when assessing its consequences.

Figure 2 also shows that a schema ownership problem is related to a structure problem but a structure problem has a consequence for a number of other problem themes. Similarly an identity problem has a consequence for a number of problem themes. The identity problem and the structure problem are also related. An understanding of the structure problem, the schema ownership problem and the identity problem is therefore important to understanding object-relational impedance mismatch.

The importance of a structure problem is reflected in the many mapping strategies between the schema of an object-oriented program and the schema of a relational database. Many authors describe a correspondence of structure between the schema of an object-oriented program and the schema of a relational database. A mapping strategy is based on a perceived correspondence, such as that between a class and a table (for example [18][19][20]); a class hierarchy and a table or a collection of tables (for example [7][21][22][23]); a relationship and a foreign key or a table (for example [7][19][24]); and an aggregation and a table or a column (for example [16][25]). In order to understand such a choice of mapping strategy and whether it results in an appropriate or an acceptable solution first the cause of a mismatch of structure must be understood.

There is a cycle in Figure 2. A solution to a structure problem can cause an encapsulation problem. A solution to an encapsulation problem can cause a processing model problem. A solution to a processing model problem can cause an instance problem. A solution to an instance problem can cause another encapsulation problem. Because there are different solutions to a problem a choice of solution must be made.

A choice of a mapping strategy can break a cycle if that solution does not cause another problem. For example Shadow Information [7], p228 introduces a change in the structure of an object-oriented program that addresses an instance problem. The implication is that it is important to understand the consequences of a mapping strategy as well as the artefacts involved in a correspondence.

In order to address one mismatch it may be necessary to address another problem first. A Synthetic Object Identity [16],p21 is a surrogate identifier used in a number of mapping strategies. In order to address an identity problem a Synthetic Object Identity introduces a change of structure. The question remains whether this change of structure addresses the real cause of a mismatch of identity, and so is an appropriate solution, or whether the change of structure deals with the symptoms and so is an acceptable solution. To answer that question first the cause of a mismatch of identity must be understood.

In summary, relationships between problem themes can be used to visualise the mess, or what Neward referred to as a quagmire. Exploring relationships between problem themes demonstrates that object-relational impedance mismatch is a complex problem, illuminates problems of particular importance, and highlights that there are consequences from a choice of solution. In the next section we highlight the limitations of a perspective based on problem themes.

VII. The Limitations of Problem Themes

The problem themes represent a consolidation of the work of others. However it is not clear whether they identified all possible problems and explored all possible relationships, or whether that was in fact their objective. It is also not clear from the literature whether their categorisations of the problem are simply observations based on experience or an exhaustive search of the problem.

Copeland & Maier talk in general terms of concept and structure, whereas Neward is concerned with specific problems such as retrieving data for an object from a database. Whilst the categorisation of Neward appears more comprehensive than that of Copeland & Maier, because it describes more problems, the level of abstraction can explain such a difference. Consequently the catalogue of problem themes, the relationships between themes, and the categorisations of Copeland & Maier, Neward and Ambler must be considered as partial but illustrative of the problem of object-relational impedance mismatch.

Relationships between problem themes cannot be used to locate the cause of a mismatch. In order to locate the cause of a mismatch it is necessary first to explore the reason for that mismatch between the schema of an object-oriented program and the schema of a relational database. The reason for a mismatch does not lie in a relationship between two problem themes. For example, the answer to an identity problem is not found by understanding that it may be caused by a mismatch of structure. Why there is a mismatch of structure must be understood first.

Choices of transformation in the design of an object-oriented program and a relational database provide the context for a mismatch. One mismatch is that of a data structure. Differences of language and abstraction lead to such mismatches, but a conceptual framework, respectively those of an object and a relation, underpins each language and each abstraction. Using problem themes it is clear that a problem of structure can have consequences for other problem themes, but it not clear whether a choice of abstraction, language or conceptual framework is the root cause of a mismatch of structure. In [15], we describe a framework, based on these choices, for exploring the cause of a mismatch.

VIII. Conclusions and Future Work

Contrary to the received wisdom, we do not know whether a solution addresses the cause or a symptom of an object-relational impedance mismatch. A mapping strategy is simply a pragmatic solution to a problem in the implementation of an object-relational application. Because we do not know the cause of a particular mismatch, we cannot be sure whether such a solution is appropriate or whether it is somehow acceptable.

 ISBN: 978-1-61208-408-4

Object-relational impedance mismatch is a wicked problem, and we introduce problem themes as a way of making sense of such mismatches. Our catalogue of problem themes provides a new vocabulary for describing the problem of object-relational impedance mismatch. Each problem theme focuses attention on a particular aspect of an object-relational impedance mismatch. Problem themes also provide a structure to the problem and demonstrate the complex nature of object-relational impedance mismatch.

Problem themes provide an insight into distinct, but interacting, aspects of object-relational impedance mismatch. Problem themes have implications for those developing an object-relational application. Relationships between themes expose the "quagmire of issues" referred to by Neward and demonstrate that those developing an object-relational application must think about issues of more than one theme in the design and implementation of a mapping strategy.

Our catalogue of problem themes suggests a shift in thinking about the problem of object-relational impedance mismatch from issues of implementation towards an analysis of cause and effect. Because we have questioned the received wisdom, we are in a position to work towards appropriate solutions to problems of object-relational impedance mismatch. Future work might explore also the extent to which such a shift in thinking provides a way to illuminate other issues of software integration.

The problem of object-relational impedance mismatch involves a number of stakeholders. Our own future work will concentrate on identifying a suitable mechanism to engage those responsible for the design and implementation of an object-relational application in an effective dialogue about a problem and its cause.

REFERENCES

[1] M. L. Fussell, "Foundations of Object Relational Mapping," 17th March 2015, 2007; http://www.database-books.us/databasesystems_0003.php.

[2] Hibernate. 17th March 2015; www.hibernate.org.

[3] TopLink. 17th March 2015; http://www.oracle.com/technetwork/middleware/toplink/overview/index.html.

[4] J. Schwartz and M. Desmond, "Looking to LINQ," 17th March 2015, 2007; http://adtmag.com/articles/2007/04/04/looking-to-linq.aspx.

[5] H. Rittel and M. Webber, "Dilemmas in a General Theory of Planning," Policy Sciences, vol. 4, pp. 155-169, 1973.

[6] J. Conklin, Dialogue Mapping - Building Shared Understanding of Wicked Problems, Chichester, England: Wiley, 2006.

[7] S. W. Ambler, Agile Database Techniques - Effective Strategies for the Agile Software Developer: Wiley, 2003.

[8] C. Ireland, "Object-Relationla Impedance Mismatch: A Framework Based Approach," Mathematics and Computing, Open University, Milton Keynes, 2011.

[9] F. Marguerie. "Choosing an object-relational mapping tool," 17th March 2015, 2007; http://madgeek.com/Articles/ORMapping/EN/mapping.htm.

[10] T. Neward, "Avoiding the Quagmire," 17th March 2015; http://www.odbms.org/wp-content/uploads/2007/05/031.02-Neward-Avoiding-the-Quagmire-May-2007.pdf.10

[11] R. Ackoff, "Systems, Messes, and Interactive Planning," Redesigning the Future, New York: Wiley, 1974.

[12] T. Neward, "The Vietnam of Computer Science," 17th March 2015; http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx.

[13] G. Copeland, and D. Maier, "Making Smalltalk a database system," ACM SIGMOD Record, vol. 14, no. 2, June 1984, pp. 316-325.

[14] C. Ireland, D. Bowers, M. Newton, Waugh, K., "A Classification of Object-Relational Impedance Mismatch," Proc. of The First International Conference on Advances in Databases, Knowledge and Data Applications. March 2009, pp. p36-43.

[15] C. Ireland, D. Bowers, M. Newton, Waugh, K., "Understanding Object-Relational Mapping: A Framework Based Approach," International Journal On Advances in Software, vol. 2, no. 2, 2009, pp. 202-216.

[16] W. Keller, "Mapping Objects to Tables: A Pattern Language," Proc. of European Conference on Pattern Languages of Programming Conference (EuroPLoP), 1997

[17] G. L. Sanders, and S. Seungkyoon, "Denormalisation effects in performance of RDBMS," in 34th Annual Hawaii International Conference on System Sciences, Hawaii, January 2001, pp. 9.

[18] F. Lodhi, and M. A. Ghazali, "Design of a simple and effectve object-to-relational mapping technique," Proc. of ACM Symposium on Applied Computing, March 2007, pp. 1445-1449.

[19] K. Brown, and B. G. Whitenack. "Crossing Chasms: A Pattern Language for Object-RDBMS Integration "The Static Patterns"," 30 December 2008; http://www.ksc.com/articles/staticpatterns.htm.

[20] S. Philippi, "Model driven generation and testing of object-relational mappings," Systems and Software, vol. 77, 2005, pp. 193-207.

[21] U. Hohenstein, "Bridging the Gap between C++ and Relational Databases," Proc. of European Conference on Object-Oriented Programming. 1996, pp. 398-420.

[22] L. Cabibbo, and A. Carosi, "Managing Inheritance Hierarchies in Object/Relational Mapping Tools," Lecture Notes in Computer Science, vol. 3520, 2005, pp. 135-150.

[23] M. Pizzo, "An Application-Oriented Model for Relational Data," The Architecture Journal, no. 12, July 2007, pp. 19-25.

[24] L. Cabibbo, and R. Porcelli, "M2ORM2: A Model for the Transparent Management of Relationally Persistent Objects," Proc. of Database Programming Languages: 9th International Workshop. 2003, pp. 166 - 178.

[25] C. Russell, "Bridging the Object-Relational Divide," ACM Queue, vol. 6, no. 3, 2008, pp. 18-28.

[26] K. Roebuck, Object-Relational Mapping: High-impact Strategies - What You Need to Know: Emereo Pty Limited, 2011.

 ISBN: 978-1-61208-408-4

Rule-Based Adaptation of Workflow Patterns for Generic Workflows

Marina Tropmann-Frick, Niklas Sasse
Christian-Albrechts-University of Kiel
Department of Computer Science
Kiel, Germany
Email: {mtr, nsa}@informatik.uni-kiel.de

***Abstract*—This paper focuses on the application of an adaptation rule system for generic workflows based on the basic concepts of rewrite systems in the context of disaster management. Disaster management is one of the challenging, complex and critical application areas dealing with hyper dynamic situation changes, high velocity, voluminous data and organizational heterogeneity. We propose to use generic workflows that satisfy these complex requirements and that provide support for organizing processes and information flow in disaster situations thereby providing decision support to crisis managers and "first responders". We illustrate our approach in a case study for disaster management.**

Keywords–adaptation; generic workflow; workflow pattern; rewrite system; term rewriting; disaster management.

I. INTRODUCTION

Workflow management systems are commonly used to plan, execute and control all kinds of business processes. Workflow management helps to optimize the usage of resources and to provide a view on the business processes of a company, which help to make the right management decisions. Profit can be increased by consequently align the business processes to the customer's needs. This makes a workflow management system the basis for controlling system that allows to steer the success of a company [1].

They are especially applicable for structured processes with sequential or parallel activities which require coordinated processing and involve several actors with different roles. Typical workflow management systems can mainly be used in a static environment with clearly defined organizational structures, completely given business processes and full control. Workflows for such business processes are completely predefined at process design time. Exceptions are part of the workflow. Deviations are often described as separate workflows. They must however be known at modeling time [2] [3]. EPC (Event-driven Process Chain) and BPMN (Business Process Model Notation) support such business processes and their modeling [4] [5].

Generic workflows are flexible and adaptable workflows belonging to the area of process-aware information systems (in particular workflow management systems). Process-aware information systems are going to be used in applications which demand higher flexibility [6]–[12]. For example, M. Reichert and B. Weber [13] survey approaches to manage dynamics in process-aware information systems. We can distinguish between design-time and run-time flexibility. Variability, adaptation, evolution and looseness are the four main categories of flexibility.

Disaster management represents an important, versatile and critical domain. The processes of disaster management can be assigned the looseness category of flexibility. They are non-repeatable (every process instance is different), unpredictable (there is no knowledge existing about situation changes during an event) and emergent (the processes emerge during execution when more information becomes available). The situation specific parameters are unknown in the beginning and might change during process execution. Because of the huge number of parameters and possibilities of process development, dynamic approaches can easily become too complex and incomprehensible for dealing with.

Disaster management processes are highly dynamic and there is no possibility to predefine every exception or variation. Therefore, static approaches are quite ineligible. Although existing dynamic approaches can deal with a certain degree of flexibility, they may fail because of the huge number of parameters and case variations that must be considered.

Our approach is based on the idea of genericity. It allows us to construct an abstract generic workflow which can be adapted to dynamic changes at runtime. We present the structure and basic components of generic workflows and show how our approach satisfies requirements of disaster management in a case study. This work extends our research in [14]–[17], which is going to be used in the EU-project INDYCO [18]. The main goal of this collaborative project is the development of an INtegrated DYnamic decision support system COmponent for disaster management systems.

A. Paper Structure

This paper is structured into four sections. The first section introduces important terms and discusses related work in the context of process aware information systems and disaster management. In Section 2, the structure and components of generic workflows are defined and explained. In Section 3, the essential aspects of term rewriting systems and the approach of their application for the adaptation techniques of generic workflows are described. The following Section 4 involves a case study from the area of disaster management. Afterwards we conclude by summarising and discussing our next steps and future work.

II. GENERIC WORKFLOWS

This section explains the methodical and technical fundamentals of generic workflows. We understand generic workflows as abstract, configurable, mutable and adaptable workflows, which are used to derive the current workflow instance.

 ISBN: 978-1-61208-408-4

The current workflow is developed within the framework of the generic workflow and uses all its services. The current workflow considers the current situation, the current requirements and the current data available.

A. Genericity

The notion of genericity is not new. According to [19], genericity can be described as a quality to be not specific, typifying, applied to or characteristic of all members of a genus, species, class or group.
In our everyday life, we come almost permanently upon generic activities. In science, particularly in computer science genericity is also widely used. A good example is the usage of generic algorithms in context of generic programming [20]. We understand genericity as a capability to describe a group or class of objects on a certain abstraction level. This allows higher adaptation and flexibility.

B. Generic Components

In our previous work [14]–[17], we already described in detail the construction and components of generic workflows. Therefore for the sake of completeness, we give a short overview about the most important parts and refer for more details to our previous work.

1) Generic Functions: The concept of generic functions provided by Bienemann [21] is based on government and binding (GB) approach that was introduced by Chomsky [22]. Chomsky proposed a universal theory of languages. Basic concepts of the theory are the atomic units of the syntax [23], [24].
Consider functions $F, F_1, ..., F_n$ of a chosen function algebra. Generic functions are functions

$$\mathcal{F} = (Dom, \phi, F, \psi, Rng), \tag{1}$$

with free configuration parameters, with predicates ϕ for the domain Dom and ψ for the range Rng of $\mathcal{F}$. A derived function is generated from $F = \theta(F_1, ..., F_n)$ based on expression and instantiation of the configuration parameters and on instantiation of the predicates; θ is here an n-ary operator $\theta \in op^{\mathcal{F}}$ (set of function manipulation operators [21]). In [15], we described the approach of generic functions more detailed.
Generic functions are basic elements for generic workflows. They represent the atomic activities within a generic workflow and are indecomposable.

2) Mini Stories: The next level of abstraction for our generic approach are semantically logical units - mini stories [14]. This are also atomic components, but in an abstract semantic way.
Mini stories represent abstract collections of generic functions which can be dynamically composed at runtime based on parameter initialization. We define a mini story as a quadruple [16]

$$\mathcal{M} = (\mathcal{F}, \mathcal{T}, \mathcal{S}, P), \tag{2}$$

where $\mathcal{F}$ is a set of generic functions as defined above. $\mathcal{T}$ is a set of transitions within a mini story. $\mathcal{S}$ is a set of parameters defining the current state of the mini story. And P is a priority function.
Transitions are given as tuples of the form $T_{ij} = (F_i, F_j) \in \mathcal{T}$ and can be represented as edges of a directed graph with node set $\mathcal{F}$.
The priority function P is defined as follows:

$$P : (\mathcal{F}, \mathcal{S}) \rightarrow \mathbb{R}_{\geq 0}, \tag{3}$$

and assigns each function $F_i \in \mathcal{F}$ depending on the current state $S_j \in \mathcal{S}$ a priority. The function with the highest priority is the best for current situation.
The instantiation of a mini story is performed at runtime depending on the current state (and influencing parameters) during the execution of the priority function.

3) Generic Workflows: As the next level of abstraction for our generic approach we define generic workflows as collections of semantically indecomposable mini stories. In order to specify a generic workflow the composition rules for mini stories are needed. These rules describe the conditions for composition of mini stories within a generic workflow. For example, there can be rules that require the execution of some specific mini story after another or even as a successor of another specific mini story. There can also be rules defined for the prohibition of mini story execution in a specific order.
Mini story composition is a complex issue, that can be characterized by the following three general aspects:

- The order of mini story execution is partly given by the execution order of generic functions. Depending on the priority function generic functions for the next execution step are selected. Therefore, only those mini stories can be executed as next, which contain the selected generic functions and optimally start with one of them.
- On the other hand, some rules for the composition are given by the context, where the execution takes place. The context of disaster management discussed in this paper possesses specific requirements and conditions. For the most disaster categories there are hazard maps, contingency plans or other guidelines existing (e.g., from the natural hazard management or municipality), which partly determine the order of mini story execution.
- The next important part are the influencing parameters. They can be characterized as configuration or control parameters. Some parameters belong only to one mini story and get their values allocated during the instantiation. Another parameters can be shared between various mini stories. As a consequence the instantiation of one mini story reduces the set of mini stories suitable for the next step and sets inevitably limitations for the instantiation of the following mini stories.

C. Adaptation

The general understanding of adaptation in computer science means a process in which an interactive system adjusts its behavior to an individual user by processing context information. Context information is defined as any information that can be used to characterize the situation of entities (whether a person, place or object) [25]. A readjustment of a system is often necessary after gaining new or additional context information. The concept of adaptation can be applied for generic workflows as well. The interactive system is the generic workflow itself and the individual adjustments develop

the unfolded workflow based on the given context information available at the time.

At this point it is necessary to define the scope of adaptation. In this case, adaptation means the rule-based transformation of a generic workflow or rather the composition of different mini stories which will eventually build the concrete and unfolded workflow. The processing of context information parameters, the specification and instantiation of generic tasks within the mini stories is to this effect not part of adaptation but rather a part of workflow refinement and goes beyond the scope of this paper.

The adaptation is performed via rule-based transformations of a generic workflow. Therefore, a system of well-structured rules has to be derived. This system should fulfill some properties to avoid runtime errors or complications between different rules. Since adaptation is heavily dependent on context information it is not possible to derive one abstract and universally valid system of rules. A set of rules is merely applicable for similar situations within an application domain.

In disaster scenarios a lot of action patterns are transferable so that similar sets of adaptation rules can be applied to different disaster situations. In combination with the appropriate refinement methods for generic tasks a decision support system can be developed that allows an automated adaptation for effective response in disastrous situations.

III. Rewrite Systems

To create a reliable system of rules for the adaptation of generic workflows the system has to be well-structured. For that purpose the concept of rewrite systems, especially term rewriting systems are considered to derive suitable properties for the rule-based transformation of generic workflows. Rewriting covers a wide range of methods for replacing subterms of formulas or terms with other subterms using rules.

A. Term Rewriting

Term rewriting systems are sets of directed equations which are used to repeatedly replace subterms of a formula until the simplest form of it is reached. The term rewriting introduced by Gorn [26] can be seen as a nondeterministic Markov algorithm which is used as an effective way to analyze and evaluate algorithms.

A term t consists of a set of function symbols F and a set of variables X [27]. Working with terms often requires replacement of parts of the initial terms with new terms. If at the position p of a term t the following subterm $t|_p$ is replaced with a new term s it is represented as $t[s]_p$. Through this operation a new term u is created which equals t except in position p so that $u[s]_p = t$ is true [27]. The new term u can be seen as an abstraction or a specification of t based on the context leading to the transformation. A substitution σ is a special kind of rewrite relation where terms are assigned to several variables of the initial term $\{x_1 \rightarrow s_1, ..., x_m \rightarrow s_m\}$. For example, there is a substitution $\sigma = \{x/g(y)\}$ for a term $f(x)$ then $f(x)\sigma = f(x\sigma) = f(g(y))$ is true after applying the substitution.

In the scope of term rewriting systems we find mainly binary relations $\rightarrow$ which are used on sets of terms T. They are called term rewriting relations. The general idea in term rewriting is the usage of directional equations. A term rewriting rule of a set T is an ordered pair $< l, r >$ of terms that stands in a directional binary relation $l \rightarrow r$. A finite or infinite set of term rewriting rules R over T is called term rewriting system. For a given term rewriting system R and the terms s and t of the set of terms T, s should be replaced by t, then $s \rightarrow_R t$ is true if $s|_p = l\sigma$ and $t = s[r\sigma]_p$ comply for a rule $l \rightarrow r$ in R with position p and substitution σ [27].

Term rewriting systems should be applied to a term as long as it is possible. If no term t exists for a subterm s in T so that $s \rightarrow_R t$ is applicable, the term s is no longer reducible and is in a normal form. A derivation in R is every possible sequence $t_0 \rightarrow_R t_1 \rightarrow_R ... \rightarrow_R t_i \rightarrow_R ...$ which develops from applying term rewriting rules. If every derivation leads to at least one normal form, the term rewriting system is called normalizing. If independent of the derivation the same normal form is always reached from an initial term, the term rewriting system is called canonical [27].

The basic concepts of term rewriting systems can be applied for generic workflows. In this case, the set of terms is comparable with a generic workflow that can be represented as a graph with a tree structure. The different paths are derivations of the generic workflow and possible compositions of mini stories. When a leaf is reached the term is in normal form. That means the generic workflow is concrete and unfolded. The navigation that leads the flow of the generic workflow is based on rules similar to term rewriting systems. The substitutions or rather abstractions or specifications are based on context information instead of matching variables. Therefore, the left side of an adaptation rule represents context information and the right side the mini story that has to be executed if the current context matches the rules context. By following this procedure and repeatedly applying the adaptation rule system the generic workflow will more and more unfold until a normal form is reached and the concrete workflow is complete.

In order to create a system that matches a broad variety of situations given through context information a lot of adaptation rules need to be derived. Furthermore, additional properties have to be fulfilled or at least need to be considered to prevent unexpected problems by application of an adaptation rule system to a generic workflow.

B. Termination

One of the most important characteristics of the term rewriting systems is the termination property. A binary relation $\rightarrow$ on a set T terminates if there are no endless chains $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow ...$ of elements generated from T. For a term rewriting system R this implies that it terminates if there are no endless derivations $t_1 \rightarrow_R t_2 \rightarrow_R ...$ developed by the application of R on the set of terms T. That means, every term in T has at least one normal form. In case of term rewriting systems termination can be proved by using term orders, e.g., *Knuth-Bendix-Order*, *Recursive-Path-Order* or *Polynomial-Order* [28] [29].

For generic workflows termination implies that no endless sequence of mini stories is generated by applying the adaptation rule system. In practice it is obvious that a generic workflow cannot be endless because at some point every business process comes to an end, especially if dealing with disaster scenarios. Some problems arise based on the flexibility of generic workflows. E.g., it is possible and required to repeat the same mini story more than once. This can lead to cycles

 ISBN: 978-1-61208-408-4

within the workflow and for this reason to endless activity chains. Therefore, it is necessary to define assumptions to maintain the termination property.

For this problem we can use the mini stories as labels on the nodes of our workflow graph, so that there can be many repetitions of the same mini story on different nodes between which we can distinguish. In order to monitor the whole process the adaptation rule system must be controlled by a higher instance. For that purpose a controller must be implemented which monitors the workflow during runtime. Its main priority is then to prevent runtime errors and exceptions by using appropriate abort criteria.

C. Church-Rosser Property

Presumably the set of term rewriting rules is finite in a term rewriting system. The *Church-Rosser* property is used to describe, whether for the terms s and t of the set T the statement $s =_R t$ is true. Therefore, the term rewriting system R is applied to both terms to check whether the results are identical. If derivations differ it is possible that different normal forms are reached during the usage of the term rewriting system. This problem does not exist if in every situation (for all derivations of the initial term) a term exists where they can be reunited. This means that every derivation of a term leads exactly to one single normal form. This characteristic is described as Church-Rosser property [30].

A binary relation $\rightarrow$ on a set of terms T is called Church-Rosser relation if its reflexive-transitive-symmetric closure $\leftrightarrow^*$ is maintained in the junction relation $\rightarrow^* \circ \leftarrow^*$. That means for every term t_1 and t_2 in a set of terms T, if $t_1 \leftrightarrow^* t_2$ is true, there exists another term s in T, for which $t_1 \rightarrow^* s$ and $t_2 \rightarrow^* s$ is true as well [27].

The Church-Rosser property is equivalent to the slightly simpler property of confluence [31]. A binary relation $\rightarrow$ on a set of terms T is confluent if the reflexive-transitive-symmetric closure of the relation $\leftarrow^* \circ \rightarrow^*$ is maintained in the junction relation $\rightarrow^* \circ \leftarrow^*$. That means for every term u, t_1 and t_2 in a set of terms T with $u \rightarrow^* t_1$ and $u \rightarrow^* t_2$ there exists another term s in T, so that $t_1 \rightarrow^* s$ and $t_2 \rightarrow^* s$ are true as depicted in Figure 1 [27]. So the confluence of a term rewriting system R on a set of terms T implies the impossibility of more than one normal form for every particular term $t_1, ..., t_i$ of T. But at the same time it does not guarantee the existence of a normal form for every term $t_1, ..., t_i$ of T because of the termination property that must be fulfilled as well in this case.

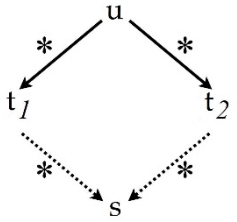


FIGURE 1. CONFLUENCE

Regarding the adaptation rule system for generic workflows the fulfillment of confluence is not crucial because for the concept of generic workflows it isn't important that the resulting unfolded workflow is in every case the same. In reality, it is rather the opposite. This results from the fact that after every application of an adaptation rule new context information is processed. Therefore, it is possible that the priority for the execution of different mini stories may change over time and affect the finally concrete workflow. However the important effect that can be shown through confluence is the prevention of execution for some mini stories by application of certain adaptation rules.

D. Critical Pairs

The consideration of critical pairs is an extension of the Church-Rosser property and the confluence. In general confluence is a nondeterministic criteria [32] but with special methods it is possible to force confluence for finite and terminating term rewriting systems.

Two different term rewriting rules $l \rightarrow r$ and $s \rightarrow t$ overlap if they are both applicable on the same term. In this case, a decision is necessary to determine which rule shall be executed. The proof of *Knuth & Bendix* [33] considers all possible positions of terms and subterms on which term rewriting rules can be applied and it shows that every critical pair can be solved by using the superposition test. Thereby confluence can be forced for a finite and terminating term rewriting system.

The handling of critical pairs is a very important part by creating an adaptation rule system for generic workflows because there occur a lot of them, especially when dealing with disaster scenarios where many activities are performed simultaneously by various actors and the amount of available resources is limited.

To determine which adaptation rule should be executed if more than one rule applies at the same time for the given context of a generic workflow, it is necessary to prioritize the tasks of the mini stories. The processing of the prioritization is also part of the controller which monitors the generic workflow during runtime. The controller has to consider different actors and the amount of available resources before it decides which action alternatives should be performed. The goal of prioritization is the effective usage of all resources and appropriate selection of mini stories possible for a given context. Thus the concrete, unfolded workflow is the most efficient way to carry out the treated business process.

IV. CASE STUDY

By the case study we want to illustrate how our ideas can be used in a practical way. In order to demonstrate our approach and create a first adaptation rule system for generic workflows we selected two scenarios from the disaster management area. The scenarios were reviewed to deviate rules and to gather information that the controller can use for evaluation of different situations. To have the ability to compare both regarded scenarios are flooding events, so that rules and information deviated from the first scenario can directly be evaluated with the second. In both cases activities on the micro level and context information is supposed to be given so that the description is more on the abstraction level of mini stories.

The first scenario is an emergency plan that describes the actions which have to be performed mainly by a task force during a flooding disaster. It is an abstract scenario

 ISBN: 978-1-61208-408-4

and refers to a region in Austria near a huge European river. The plan distinguishes different levels of escalation. At first all measures are described which will help dealing with the situation without taking any severe damage or encounter exceptional circumstances. These measures contain among other things description of how the information flow works and of allocations of actors to their respective roles. Furthermore, the plan shows how to handle some critical situations, e.g., the burst of a dam or a complete system failure. For these exceptional escalations suitable countermeasures are specified.

After analyzing the emergency plan adaptation rules for the generic workflow were derived. For the deriving process it was important to consider the level of abstraction. On the one hand the rules cannot be too specific because they should also be applicable for similar disaster scenarios. On the other hand, if they are too abstract they are of little help in guiding the sequence of the generic workflow at all. Based on the given information first adaptation rule system was formed containing several rules which describe and process the activities during a flooding event. the rules were prioritized and divided into three different categories so that the controller should be able to decide between rules if necessary.

The derived adaptation rule system was then evaluated with the second disaster scenario in order to prove whether the rules and the prioritization are applicable to control a generic workflow during similar situations. The second scenario deals with a real flooding disaster which took place in Germany in 2010. It was triggered by heavy and ongoing rainfall and led to a quick escalation of the situation. Similar to the emergency plan the response actions were performed mainly by a task force, the information flow and the involved actors are accurately documented as well.

During evaluation it was tested if the derived adaptation rule system is also applicable to a real scenario. In general most of the adaptation rules were suitable for the second scenario and the prioritization of the performed actions and selected mini stories was mostly correct. Minor changes and additional rules which could be derived from this scenario were then added to the adaptation rule system. Therefore, it could be shown that the system is applicable to similar scenarios.

However the current system isn't able to control the generic workflow during a disaster scenario by itself. Much more information has to be gathered through processing a lot of disaster situations, so that the adaptation rule system can handle any given situation which is described through context information. Especially the functionality of the controller is important to give sufficient support for making intelligent decisions and to learn from new situations.

V. Conclusion and Future Work

In this contribution we discuss an adaptation rule system approach for generic workflows and its application for disaster management. Most actions during a disaster response are not predictable and cannot be planned completely beforehand. So the corresponding processes cannot be prespecified and handled in a standard way. We intend to use generic workflows for coordination of disaster management processes. They allow accurate, fast and dynamic activity guidance and information coordination in complex situations.

The theoretically elaborated concepts were tested positively while they were applied to a real world scenario. Therefore, it is possible to integrate the adaptation rule system into the framework of generic workflows. This concept should help to provide a software system to support decision making and workflow control in disastrous situations.

Our next step will be to specify a refinement mechanism for generic workflows that could fit to our overall concept. We intend also to develop a toolkit which allows information gathering about past disaster management processes and preparation of the information for generic workflows.

References

[1] L. Fischer, Ed., Workflow Handbook 2003. Future Strategies Inc., Published in association with the Workflow Management Coalition, 2003.

[2] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," in Distributed and Parallel Databases, 1995, pp. 119–153.

[3] N. Russell, A. H. Hofstede, D. Edmond, and W. van der Aalst, "Workflow Data Patterns: Identification, Representation and Tool Support," in 24th International Conference on Conceptual Modeling, L. M. L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, and O. Pastor, Eds. Klagenfurt, Austria: Springer, 2005, pp. 353–368.

[4] L.Fischer, Ed., BPMN 2.0 Handbook Second Edition. Future Strategies Inc., Published in collaboration with the Workflow Management Coalition (WfMC), 2012.

[5] D. M. Stephen A. White, BPMN Modeling and Reference Guide. Future Strategies Inc., 2008.

[6] C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic change within workflow systems," in Proceedings of conference on Organizational computing systems, ser. COCS '95. New York, NY, USA: ACM, 1995, pp. 10–21.

[7] Y. Han and A. Sheth, "On Adaptive Workflow Modeling," in Proceedings of the 4th International Conference on Information Systems Analysis and Synthesis, Orlando, Florida, July 1998, pp. 108–116.

[8] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke, "A Comprehensive Approach to Flexibility in Workflow Management Systems," in WACC99, Work Activities Coordination and Collaboration, ACM Press, San Francisco, USA, February 1999, pp. 79–88.

[9] M. Momotko and K. Subieta, "Dynamic change of Workflow Participant Assignment," in ADBIS 2002. LNCS. Springer, 2002.

[10] J. Klingemann, "Controlled Flexibility in Workflow Management," in Proceedings of the 12th International Conference on Advanced Information Systems Engineering, ser. CAiSE '00. London, UK, UK: Springer-Verlag, 2000, pp. 126–141.

[11] W. M. P. van der Aalst, "How To Handle Dynamic Change and Capture Management Information? An Approach Based on Generic Workflow Models," Comput. Syst. Sci. Eng., vol. 16, no. 5, 2001, pp. 295–318.

[12] R. Müller, U. Greiner, and E. Rahm, "AgentWork: a Workflow System Supporting Rule-Based Workflow Adaptation," Data and Knowledge Engineering, vol. 51, no. 2, 2004, pp. 223 – 256.

[13] M. Reichert and B. Weber, Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Berlin-Heidelberg: Springer, 2012.

[14] B. Thalheim and M. Tropmann-Frick, "Mini Story Composition for Generic Workflows in Support of Disaster Management," in Proceedings of the 24th international workshop on Database and Expert Systems Applications, ser. DEXA 2013. IEEE Computer Society, 2013, pp. 36–40.

[15] B. Thalheim, M. Tropmann-Frick, and T. Ziebermayr, "Application of Generic Workflows for Disaster Management," in Proceedings of the 23rd European-Japanese Conference on Information Modelling and Knowledge Bases, ser. Information Modeling and Knowledge Bases XXIII, Y. Kiyoki, T. Tokuda, and N. Yoshida, Eds. IOS Press, 2013, pp. 68–85.

[16] M. Tropmann-Frick, B. Thalheim, D. Leber, C. Liehr, and G. Czech, "Generic Workflows - A Utility to Govern Disastrous Situations," in Proceedings of the 24th European-Japanese Conference on Information Modelling and Knowledge Bases, ser. Information Modeling and

 ISBN: 978-1-61208-408-4

Knowledge Bases XXIV, Y. Kiyoki, T. Tokuda, and N. Yoshida, Eds. IOS Press, 2014, pp. 473–485.

[17] M. Tropmann-Frick and T. Ziebermayr, “Generic approach for dynamic disaster management system component,” in Proceedings of the 25th international workshop on Database and Expert Systems Applications, ser. DEXA 2014. IEEE Computer Society, Sept 2014, pp. 160–164.

[18] INDYCO, “http://www.is.informatik.uni-kiel.de/ project/indyco/en/,” accessed 06.04.2015.

[19] Webster’s Third New International Dictionary, 1993.

[20] D. R. Musser and A. A. Stepanov, “Generic Programming,” in Lecture Notes in Computer Science 358. Springer Verlag, 1989, pp. 13–25.

[21] A. Bienemann, Context-Driven Generation of Specifications for Interactive Information Systems, ser. Dissertationen zu Datenbanken und Informationssystemen. AKA, 2008.

[22] N. Chomsky, Some Concepts and Consequences of the Theory of Government and Binding, ser. Linguistic Inquiry Monographs. MIT Press, 1982.

[23] N. Chomsky, The minimalist program. Cambridge: MIT Press, 1995.

[24] E. Stabler, “Derivational Minimalism,” in Logical aspects of computational linguistics, C. Retore, Ed., vol. LNCS 1328. Springer, 1998, pp. 68–95.

[25] D. Crestani, A. Jean-Marie, and C. Coves, “Petri Nets Analysis: Complexity and Finite Coverability Graph in Modular Design,” Studies in Informatics and Control, vol. 14, no. 1, 2005, pp. 55–64.

[26] S. Gorn, J. Hart, and S. Takasu, “Explicit Definitions and Linguistic Dominoes,” in Systems and Computer Science. University of Toronto Press, 1967, pp. 77–105.

[27] N. Dershowitz and J.-P. Jouannaud, “Handbook of Theoretical Computer Science (Vol. B),” J. van Leeuwen, Ed. MIT Press, 1990, pp. 243–320.

[28] N. Dershowitz, “Termination,” in Proceedings of the First International Conference on Rewriting Techniques and Applications (Dijon, France), ser. Lecture Notes in Computer Science, vol. 202. Springer-Verlag, Berlin, 1985, pp. 180–224.

[29] ——, “Orderings for Term-Rewriting Systems,” in Theoretical Computer Science, vol. 17, no. 3, 1982, pp. 279–301.

[30] A. Church and J. B. Rosser, “Some Properties of Conversion,” in Transactions of the American Mathematical Society, vol. 39, no. 3, 1936, pp. 472–482.

[31] M. H. A. Newman, “On Theories with a Combinatorial Definition of ”Equivalence”,” in Annals of Mathematics, vol. 43, no. 2, 1942, pp. 223–243.

[32] G. Huet, “Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems: Abstract Properties and Applications to Term Rewriting Systems,” J. ACM, vol. 27, no. 4, Oct. 1980, pp. 797–821.

[33] D. E. Knuth and P. B. Bendix, “Simple Word Problems in Universal Algebras,” in Automation of Reasoning. Springer-Verlag, Berlin, 1983, pp. 342–376.

 ISBN: 978-1-61208-408-4

JAebXR: a Java API for ebXML Registries for Federated Health Information Systems

Antonio Messina, Pietro Storniolo and Alfonso Urso
ICAR - CNR
Palermo, Italy
Email: {messina, storniolo, urso}@pa.icar.cnr.it

***Abstract*—The traditional Java API for Registries (JAXR) provides a useful way for Java developers to use a single simple abstraction API to access a variety of registries. The unified JAXR information model (Infomodel), which describes content and metadata within registries, provides a simple way to access registries information. However, when a JAXR registry provider implements the OASIS (Organization for the Advancement of Structured Information)/ebXML Registry Services Specification, it internally works with ebXML Registry Information Model (RIM) objects which should be exposed as JAXR Infomodel objects. Furthermore, any application using ebXML RIM objects, before the access to the registry via a JAXR provider, has to convert them to the Infomodel counterpart. To deal with this problem, in this paper we suggest a new tool for the ebXML software development, which focuses on an extension of the traditional JAXR layer, providing the native use of ebXML RIM objects in client side applications with a direct access to the ebXML RIM SOAP service, and avoiding any conversion to/from JAXR Infomodel objects. The proposed approach reduces the developers duty, allowing them to focus exclusively on the ebXML objects use and it considerably speeds up the interactions with every ebXML registries.**

Keywords–ebXML, Registry Services, Registry Client, SOAP, JAXR, JAVA.

I. INTRODUCTION

In recent years, we have been involved in the development of some software components within the project *OpenInFSE*, an experimental interoperability infrastructure based on the InFSE architecture [1], which represents a multi-level service-oriented architecture for sharing medical data among federated *Health Information Systems* (HIS) [2]. This infrastructure is based on the extensive use of Web Service technology and XML data exchanged as Simple Object Access Protocol (SOAP) messages in accordance to the Health Level 7 (HL7) [3] Clinical Document Architecture (CDA) standard.

All of the software components included in the InFSE *Component layer* interact with the *Federated Index Registry*, which enables the query of medical data, managed by several HIS, to a federated system of regional registries in Italy, each of them able to localize the data achieved in the regional repositories.

Currently, there are two preeminent registry standards: Universal Description, Discovery, and Integration (UDDI) [4] and Electronic Business using eXtensible Markup Language (ebXML) [5] [6]. Using one of these registry standards, it is possible to publish a set of Web services, enabling internal or external business partners to discover them.

A registry typically works as electronic Yellow Pages, where information about businesses, and the products and services they offer are published and discovered. A registry can also serve as a database or as a storage of shared informations.

A registry can also work as an electronic bulletin board in which the partners share information in a dynamic and ad hoc manner.

Submission and storing of shared information are important operations performed by registry-service clients. These clients also need to complete various registry management operations, such as identifying, naming, describing, classifying, associating, grouping, and annotating registry metadata. Finally, clients also must be able to query, discover, and retrieve shared information from the registry so that they expect that a typical registry supports most of these operations.

The ebXML is designed to create a global electronic market place where enterprises of any size, anywhere, can find each other electronically and conduct business using exchange of XML messages according to standard business process sequences and mutually agreed trading partner protocol agreements. This standard overcomes the limitations of existing business-to-business frameworks and technologies, because, for example, UDDI does not provide repository capability for business objects, SOAP in its basic form does not provide reliable and secure message delivery, and the Web Service Definition Language (WSDL) does not address business collaboration.

The Java API for XML Registries (JAXR) [7] provides a standard *Application Programming Interface* (API) for publication and discovery of Web services through underlying registries. JAXR does not define a new registry standard. Instead, this standard Java API performs registry operations over a various set of registries and defines a unified information model to describe registry contents.

The JAXR specification defines a general-purpose API, allowing any JAXR client to access and interoperate with any business registry accessible via a JAXR provider. In this sense, JAXR provides a Write Once, Run Anywhere API for registry operations, simplifying Web services development, integration, and portability.

During the development of OpenInFSE project, JAXR has been initially adopted to access to underlying ebXML registries. Even though JAXR can be used for the access to ebXML registries, it can be considered unsuitable because it heavily depends on the registry implementation, it involves too many data conversion and it may lead to a serious general inefficiency of the entire infrastructure.

 ISBN: 978-1-61208-408-4

In this paper, the development of JAebXR, which is a JAXR extension, is proposed. This extension can be viewed as a complementary API to offer an ebXML provider implementation and to complete the current JAXR reference implementation.

The paper is organized as follow. Section 2 presents the JAXR architecture, also illustrating some typical registry operation. Section 3 presents the motivations for this work. Section 4 presents our proposed approach to assist the ebXML client code development. Section 5 introduces the open source ebXML registry service developed as natural counterpart of JAebXR within the OpenInFSE Project. Finally, conclusions and the presentation of some future works are reported.

II. JAXR ARCHITECTURE

The JAXR architecture defines three important architectural actors:

- A registry provider implements an existing registry standard, such as the Organization for the Advancement of Structured Information (OASIS)/ebXML Registry Services Specification 3.0.
- A JAXR provider offers an implementation of the JAXR specification approved by the Java Community Process (JCP) in May 2002. You can implement a JAXR provider as its own JAXR-compliant registry provider. However, you would more likely implement a JAXR provider as an interface to an existing registry provider, like UDDI or ebXML type. Currently, the JAXR reference implementation 1.0 offers a JAXR UDDI provider implementation.
- A JAXR client is a Java program that uses JAXR to access the registry provider via a JAXR provider. A JAXR client can be either a standalone Java 2 Platform Standard Edition (J2SE) application or Java 2 Platform Enterprise Edition (J2EE) components, such as Enterprise JavaBeans (EJBs), Java Servlets, or JavaServer Pages (JSs). The JAXR reference implementation also supplies one form of a JAXR client, a Swing-based registry browser application.

Because JAXR offers a standard API for accessing various registry providers and a unified information model to describe registry contents, JAXR clients, whether HTML browsers, J2EE components, or standalone J2SE applications, can uniformly perform registry operations over various registry providers.

Figure 1 shows a high-level view of the current JAXR architecture.

Note that the JAXR client connects with the JAXR provider, not the registry provider. The JAXR provider acts as a proxy on the client's behalf, directing and invoking methods on the appropriate registry provider. The connection maintains client state. In addition, the JAXR client dynamically sets its authentication information and communication preference on the connection any time during the connection's lifetime.

After the JAXR client invokes JAXR capability-level methods, the JAXR provider transforms these methods into registry-specific methods and executes requests to the underlying registry providers.

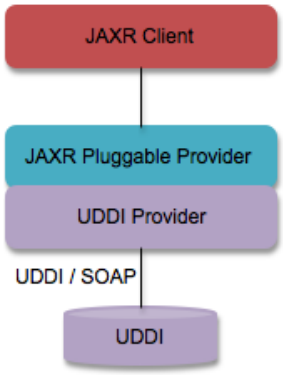


Figure 1. Current available JAXR architecture.

The communication protocol between a JAXR provider and a registry provider depends on the registry type and it is transparent to the JAXR client. A JAXR provider communicates with the UDDI registry provider by exchanging basic SOAP messages, while the JAXR provider should communicate with an ebXML registry provider through SOAP messaging or ebXML message service.

A. Capability profiles

Because among registry provider capabilities some diversity exists, the JAXR expert group decided to add multilayer API abstractions through capability profiles. A capability level is assigned to each method of a JAXR interface, and those JAXR methods with the same capability level define the JAXR provider capability profile.

Currently, JAXR defines only two capability profiles: level 0 profile for basic features and level 1 profile for advanced features. Level 0's basic features support the so-called business-focused APIs, while level 1's advanced features support generic APIs. At the minimum, all JAXR providers must implement a level 0 profile. A JAXR client application using only those methods of the level 0 profile can access any JAXR provider in a portable manner. JAXR providers for UDDI must be level 0 compliant.

JAXR providers can optionally support the level 1 profile. The methods assigned to this profile provide more advanced registry capabilities needed by more demanding JAXR clients. Support for the level 1 profile also implies full support for the level 0 profile. JAXR providers for ebXML must be level 1 compliant.

B. JAXR information model

Invoking life-cycle and query management methods on the JAXR provider requires the JAXR client to create and use the JAXR information model objects.

The JAXR information model resembles the one defined in the ebXML Registry Information Model 2.0, but also accommodates the data types defined in the UDDI Data Structure Specification.

Although developers familiar with the UDDI information model might face a slight learning curve, once understood, the JAXR information model will provide a more intuitive and natural interface to most developers.

 ISBN: 978-1-61208-408-4

C. JAXR interations

Figure 2 shows the JAXR interactions in a client-server communication.

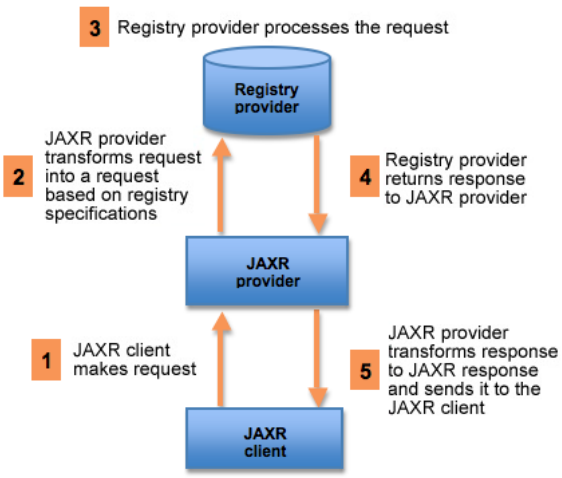


Figure 2. JAXR interactions.

1) A JAXR client uses JAXR interfaces and classes to request access to a registry. The client sends the request to a JAXR provider.
2) When a JAXR provider receives a request from a JAXR client, it transforms the request into an equivalent request that is based on the specifications of the target registry. The JAXR provider then passes this transformed request to a registry provider.
3) The registry provider receives a request from a JAXR provider and processes it. The process is then reversed.
4) The registry provider returns a response to the JAXR provider, which transforms it to an equivalent JAXR response.
5) The JAXR provider sends the JAXR response to the JAXR client.

III. MOTIVATION

In the development of applications or software layers that need to interact with ebXML registries there are some alternative ways:

1) direct use of the ebXML SOAP service exposed by a standard ebXML registry;
2) direct use of registries proprietary APIs;
3) JAXR APIs.

Because the first is well documented but not very practical and a developer should try to avoid the second for portability reasons, JAXR seems to be an obvious choice.

JAXR was our first choice but immediately we have obtained unsatisfactory results for the following reasons:

- Both the request from a JAXR client to a JAXR provider and the JAXR response sent back to the JAXR client carry Infomodel objects. It means that the JAXR client has to *manage* such objects.
- As mentioned in previous section, the current JAXR reference implementation offers only a JAXR UDDI provider implementation. It means that when we develop some ebXML-related piece of software, we need to map our ebXML objects to JAXR Infomodel objects for the requests and viceversa for the responses.
- Avaliable open-source ebXML registry providers, i.e., Omar [8], present a JAXR side-server interface but they internally work on ebXML objects. It means that external JAXR requests are translated to ebXML requests and the ebXML responses are translated back to JAXR responses.

Figure 3 shows the ebXML-over-JAXR interactions in a client-server communication.

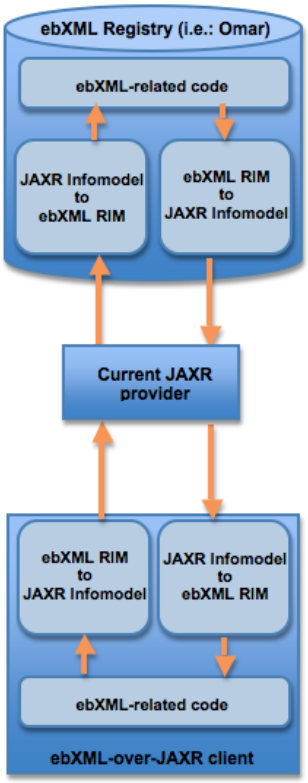


Figure 3. ebXML-over-JAXR interactions.

We refer interested readers to [9] for more details.

The reasons stated above involve considerable complexity in the code development and a substantial performance degradation.

In the following section, we propose an extension of the traditional JAXR layer, providing the native use of ebXML RIM objects in client side applications with a direct access to the ebXML RIM SOAP service, and avoiding any conversion to/from JAXR Infomodel objects.

The proposed approach reduces the developer's duty, allowing them to focus exclusively on the ebXML object's use and it may considerably speed up the interactions with every ebXML registries.

 ISBN: 978-1-61208-408-4

IV. JAXR EXTENSION: JAEBXR

The Java API for ebXML Registries (JAebXR) library has been developed in order to facilitate, standardize and optimize interactions with ebXML registries.

Its architecture is directly derived from JAXR, incorporating and extending its functionality to ensure support to the types of objects and services defined by OASIS ebXML RegRep 3.0 specifications.

For this reason JAebXR can be considered as a JAXR Provider Level 1 implementation, because it also supports UDDI registries type (level 0).

Figure 4 shows a high-level view of the JAebXR architecture.

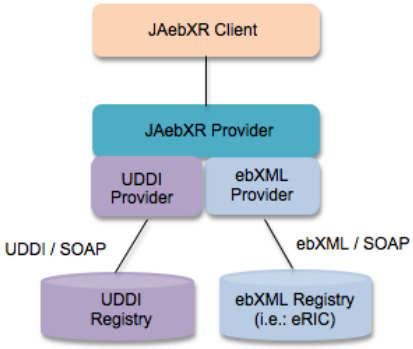


Figure 4. JAebXR architecture.

The API is independent of the particular implementations of ebXML registries. Interactions with ebXML registries occur, in fact, simply by invoking the SOAP web services exposed, as per OASIS specifications. In such circumstances, queries and transactions exclusively refer to ebXML RIM objects, properly encapsulated using the Java Architecture for XML Binding (JAXB) [10].

To speed up operations, in particular queries on registry objects, the API uses the in-memory object cache implementation provided by *cache2k* [11].

A. Implementation

The package *javax.ebxml.registry*, like the package *javax.xml.registry* in JAXR, contains the definition and implementation of the interfaces to register access, as shown in Figure 5.

The classes *BusinessLifeCycleManager*, *BusinessQueryManager*, *Connection*, *ConnectionFactory*, *DeclarativeQueryManager*, *LifeCycleManager*, *QueryManager*, and *RegistryService* are implementations of the homonymous JAXR interface classes, extended by specific methods for the ebXML RIM objects.

The sub packages *javax.ebxml.registry.security* and *javax.ebxml.registry.soap* handle the secure client-server SOAP interactions using Apache WSS4J [12], which implements the primary security standards for web services, namely the OASIS Web Services Security (WS-Security) specifications [13].

In the following subsections, a more detailed description of some of the main classes is reported.

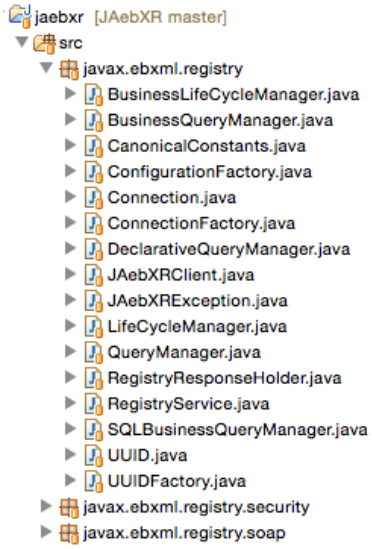


Figure 5. The JAebXR Java package.

B. ConfigurationFactory

Configuration issues are managed by the *ConfigurationFactory* class, which was built as a singleton: there is just a private constructor and a static getter method that returns an instance of the class, creating it in advance or at the first invocation of the method, by storing the reference in a static private attribute.

The class expects configuration directives given in a text properties file called *ebxml.properties* and, once instantiated, reads that file to set the required attributes to instantiate a *SOAPMessenger* object to invoke the ebXML registry web service in authenticated mode.

Figure 6 shows a sample configuration file:

```
connectionFactoryClass = it.cnr.icar.eric.client.
    xml.registry.ConnectionFactoryImpl

aliasName = urn:freebxml:registry:predefinedusers:
    registryoperator
aliasPass = urn:freebxml:registry:predefinedusers:
    registryoperator

keystorePath =
    /opt/eric/3.2/data/security/keystore.jks
keystorePass = ebxmlrr

certificateType = JKS

registryURL =
    http://localhost:8080/eric/registry/soap

cacheExpirationTime = 15
cacheMaxSize = 500

sqlQueries = false
```

Figure 6. Sample ebxml.properties

The supported attributes are:

- *connectionFactoryClass*: fully qualified name of the registry class, which implements the JAXR ConnectionFactory interface. It is required only when JAebXR

is used in a JAXR-way, as better specified in the next subsection;

- *aliasName* and *aliasPass*: registry user credential;
- *keystorePath*, *keystorePass* and *certificateType*: keystore related parameters to access to the user certificate;
- *cacheExpirationTime*: time duration in minutes after an entry expires. A value of 0 disables the cache;
- *cacheMaxSize*: maximum cache size limit in KBytes;
- *sqlQueries*: enable the use of optional SQL-92 [14] queries instead of standard ebXML filter queries.

C. ConnectionFactory

The *ConnectionFactory* class is the base class essential for the creation of a JAXR connection, which can be done by searching via the Java Naming and Directory Interface (JNDI) [15] or directly, through the use of the static method *newInstance()*.

We have chosen the second way, which requires the instantiation of the class that implements the interface *ConnectionFactory*.

The registry connection is also completely managed in terms of authentication, using the user credentials contained in a specific keystore. That means that an invocation of the method *createConnection()* returns a ready-to-use *Connection* object.

D. LifeCycleManager and BusinessLifeCycleManager

The LifeCycleManager interface class in JAXR is devoted essentially to the management of Infomodel objects by the declaration of several abstract methods.

The LifeCycleManager class in JAebXR fully implements the interface javax.xml.registry.LifeCycleManager and extends it to support all the ebXML RIM objects by methods named using the *Type* suffix. It means, for example, that the creation of a RIM registry object is realized by the *createRegistryObjectType()* method, the creation of a RIM classification scheme is realized by the *createClassificationSchemeType()* method, and so on.

Object storage and cancellation are achieved by the methods *saveObjectTypes()* and *deleteObjectTypes()*. Both of these methods call the generic method *submitObjectTypes()*, which actually implements the above operations using SOAP messages sent directly to the ebXML registry.

The BusinessLifeCycleManager class extends LifeCycleManager and implements the ebXML version of traditional JAXR methods.

E. RegistryService

The RegistryService class is the principal interface implemented by a JAXR provider and it can be obtained from a Connection to a registry.

In JAebXR, a RegistryService object is instantiated even when there is no JAXR connection. Therefore you can always use it to obtain the references to the fundamental BusinessLifeCycleManager, DeclarativeQueryManager and BusinessQueryManager objects.

F. DeclarativeQueryManager

The class fully implements the interface *javax.xml.registry.DeclarativeQueryManager* and extends it to support all the ebXML query types, such as:

- ebXML Registy Services Filter queries;
- SQL-92 queries;
- Stored queries;

The queries are incapsulated in an *AdhocQueryRequest* object before they are sent via SOAP. Then the results are returned as a complex type *RegistryResponseType* object.

G. BusinessQueryManager and SQLBusinessQueryManager

The BusinessQueryManager class, which is exposed by the Registry Service, implements the business style query interface. It supports all the standard JAXR methods and it also provides several new methods to interact with ebXML registries. Queries are done using the mandatory ebXML query types.

The derived class SQLBusinessQueryManager implements some of the business queries rewriting them using SQL-92, optionally supported by the OASIS ebXML standards.

H. JAebXRClient

This is an auxiliary class, designed as a singleton, provided to develop ebXML clients simply and quickly. Once instantiated, it immediately sets up all that is necessary and it creates all the main objects able to interact with an ebXML registry:

- *lcm*: BusinessLifeCycleManager;
- *dqm*: DeclarativeQueryManager;
- *bqm*: BusinessQueryManager (or SQLBusinessQueryManager, according to the sqlQueries parameter's value in configuration file).

Their values are available to the outside via the getter methods *getBusinessLifeCycleManager()*, *getDeclarativeQueryManager()* and *getBusinessQueryManager()*, as shown in the figure below.

```
public class ClientTest {

   JAebXRClient ebc = JAebXRClient.getInstance();
   BusinessLifeCycleManager lcm =
       ebc.getLifeCycleManager();
   BusinessQueryManager bqm =
       ebc.getBusinessQueryManager();
   DeclarativeQueryManager dqm =
       ebc.getDeclarativeQueryManager();

   public ClientTest() {
     ...
   }

   ...
}
```

Figure 7. JAebXRClient sample use

V. TEST CASE

The use of the library is pretty simple: first of all the configuration file is prepared and then, as reported in the sample code shown in Figure 7, an instance of JAebXRClient class is created.

All our tests have been done using the *ebXML Registry by ICAR CNR* (eRIC) version 3.2 [16].

To test the ebXML specifications compliance, we have chosen the eRIC JAXR client test suite [17] as starting point and we have rewritten it in a JAebXR way. The library passed all the tests.

Finally, we have put together various pieces of code from previous tests to execute a sort of benchmark to also check the performances. We did the same with the JAXR version of the source code.

We have grouped the tests in the following four macro-categories:

- RIM
  - creation, addition of attributes, storage, load and deletion of a Person object and a User object;
  - creation, storage, load and deletion of ClassificationScheme object and of some its multi-level ClassificationNode childs;
  - creation, storage, modification, load and deletion of a RegistryPackage and associated Classification;
  - nested Classifications update;
- ebXML:
  - creation and deletion of an Association between two objects;
  - creation, storage and deletion of an ExtrinsicObject;
  - simple AdhocQuery execution;
- BusinessQueryManager:
  - search of services by organization, by name pattern, by organization and name;
  - creation of a ServiceBinding with a specification object that is a ClassificationNode with no parent and then search of a service bindings by specification object;
- LifeCycleManager:
  - test of SetStatusOnObject extension protocol, which allows setting status of an object to any ClassificationNode within canonical StatusType ClassificationScheme;
  - invocations of various API methods with null parameter;
  - implicit storage of true-composed objects;
  - implicit storage of associations and associated objects.

Figure 8 shows the tests results, where execution times are in milliseconds. Also note that the JAebXR version was executed two times: one with the configuration parameter *sqlQueries* set to *false*, the other with a value of *true*.

If you take into account that the JAebXR code is actually just a research project and it isn't ready for production use, tests results can be considered very interesting, namely:

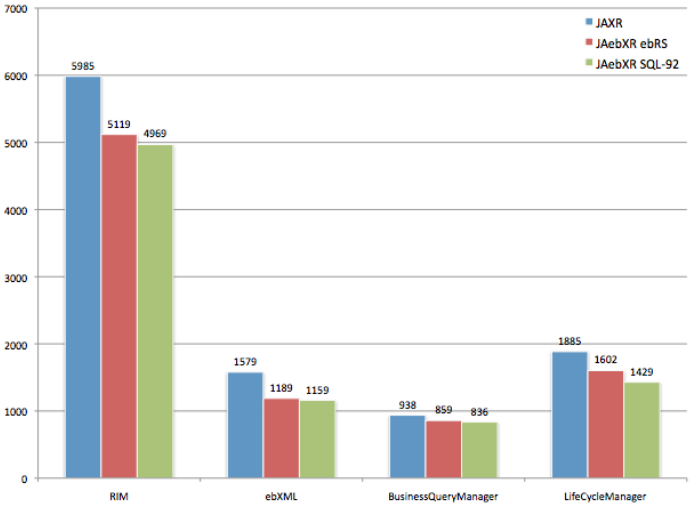


Figure 8. JAebXR Benchmarks

- with standard ebXML filter queries, we saw an increase in performance equal to 8.42% minimum to a maximum equal to 24.69%;
- with SQL-92 mode enabled, the improvement was even more relevant: minimum equal to 10.87% and maximum equal to 26,59%.

VI. CONCLUSION AND FUTURE WORK

In this work, we have highlighted the constraints and challenges faced by developers for producing Java code able to handle ebXML RIM objects in a JAXR environment.

We have explained how to use the JAXR API and methods of implementation of related source code.

Lack of an efficient handling of ebXML RIM objects was our motivation for the design and implementation of a new assistance layer for a better use of standard ebXML registriy services.

The result of current work is a simple easy-to-use API which extends JAXR to manage ebXML objects in a efficient way. Moreover, although the main scope of this work was related to a HIS project, the API is absolutely independent of health information.

Future work to improve this layer, along with the eRIC Registry, includes full support of the latest ebXML specifications published by the OASIS Consortium, ebXML RegRep v4.0 [18].

REFERENCES

[1] M. Ciampi, G. De Pietro, C. Esposito, M. Sicuranza, and P. Donzelli, "On Federating Health Information Systems," International Conference on Green and Ubiquitous Technology, Jul. 2012, pp. 139–143, ISBN: 978-1-4577-2172-4.

[2] R. Hauxe, "Health information systems," International Journal of Medical Informatics, vol. 75(3), Mar. 2012, pp. 268–281.

[3] D. F. Sittig, G. J. Kuperman, and J. M. Teich, "WWW-based interfaces to clinical information systems: the state of the art," Proceedings of the AMIA Annual Fall Symposium, 1996, pp. 694–698.

[4] OASIS UDDI Specification Technical Committee, "Universal Description, Discovery and Integration v3.0.2 (UDDI)," 2004, URL: https://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm [accessed: 2015-01-22].

 ISBN: 978-1-61208-408-4

[5] OASIS ebXML Registry Technical Committee, “ebXML Registry Information Model (RIM) v3.0,” 2005, URL: http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf [accessed: 2015-01-22].

[6] OASIS ebXML Registry Technical Committee, “ebXML Registry Services and Protocols v3.0,” 2005, URL: http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf [accessed: 2015-01-22].

[7] Java Community Process, “JSR 93: Java API for XML Registries 1.0 (JAXR),” 2002, URL: https://jcp.org/ja/jsr/detail?id=93 [accessed: 2015-01-22].

[8] “Omar: OASIS ebXML Registry Reference Implementation Project (ebxmlrr),” 2007, URL: http://ebxmlrr.sourceforge.net [accessed: 2015-01-22].

[9] M. Topolnik, D. Pintar and I. Matasic, “Implementation of the ebXML Registry Client for the ebXML Registry Services,” Proceedings of the 7th International Conference on Telecommunication, Zagreb, Croatia, Jun. 2003, pp. 551–556, ISBN: 953-184-052-0.

[10] Java Community Process, “JSR 222: Java Architecture for XML Binding (JAXB) 2.0,” 2009, URL: https://jcp.org/en/jsr/detail?id=222 [accessed: 2015-01-22].

[11] headissue GmbH, “cache2k - High Performance Java Caching,” 2015, URL: http://cache2k.org [accessed: 2015-03-19].

[12] Apache Software Foundation, “Apache WSS4J - Web Services Security for Java,” 2015, URL: https://ws.apache.org/wss4j/ [accessed: 2015-03-19].

[13] OASIS Web Services Security Technical Committee, “WS-Security OASIS Standard 1.1,” 2006, URL: https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf [accessed: 2015-03-19].

[14] ISO ANSI, “Database Language SQL ISO/IEC 9075:1992,” , Jul. 1992.

[15] Oracle Corporation, “Java Naming and Directory Interface 1.2,” 1999, URL: http://www.oracle.com/technetwork/java/jndi-150206.pdf [accessed: 2015-01-22].

[16] A. Messina, “eRIC: ebXML Registry by ICAR CNR,” 2014, URL: https://github.com/IcarPA-TBlab/eRIC [accessed: 2015-03-02].

[17] A. Messina, “eRIC Test suite,” 2014, URL: https://github.com/IcarPA-TBlab/eRIC/tree/master/eric-test-3.2 [accessed: 2015-03-02].

[18] OASIS ebXML Registry Technical Committee, “ebXML RegRep v4.0,” 2012, URL: http://docs.oasis-open.org/regrep/regrep-core/v4.0/os/regrep-core-v4.0-os.zip [accessed: 2015-01-22].

 ISBN: 978-1-61208-408-4

On the Detection of Nontrivial and Cross Language Plagiarisms

Andreas Schmidt*† and Sören Bühler*
* Department of Computer Science and Business Information Systems,
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
Email: andreas.schmidt@hs-karlsruhe.de, soeren.buehler@gmail.com
† Institute for Applied Computer Science
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: andreas.schmidt@kit.edu

***Abstract*—In this paper, we present a new approach of plagiarism detection for strongly obfuscated or even translated plagiarism, which are difficult to detect. Based on our concept, we build a prototype system and show the efficiency of our approach on different real world scenarios like wikipedia, automatically translated documents, as well as human translated books from the Gutenberg project.**

Keywords–Plagiarism detection; obfuscated plagiarism; cross language plagiarism; compresse bitvector; taxonomies.

I. INTRODUCTION

Plagiarism can occur on different levels. First of all, there is the one to one copy of the original text, without any further manipulation. This is also known as copy & paste plagiarism. Next, we have the syntactical obfuscated plagiarism. In this case, the words in a sentence are rearranged, or sentences are merged or splitted. On the next higher level, semantical relations between words are used. So, for example, single words can be replaced by their synonyms or hyperonyms. Another possibility is the translation of a document into another language. The last possibility is the theft of ideas. In this case, the author of a plagiarism uses foreign mental effort and claim it as its own. In this case, the concepts plagiarized were formulated in the plagiarists own words.

In our research, we focus on highly obfuscated plagiarism types. In former work [1], we developed a concept for plagiarism detection based on compressed bitvectors, word taxonomies and a graphical representation of the results using heatmaps. In the present work [2], these concepts have been concretized and extended and a prototype has been developed, which impressive shows the efficiency of our concept dealing with highly obfuscated plagiarisms.

The rest of the paper is structured as follows. In Section II the key concepts, as already described in [1] and extended in [2] are presented. Then, in Section III a number of significiant results are presented. Starting from the simpliest case of a plagiarism, the copy & paste plagiarism, different levels of obfuscation, inclusive the "theft of idea" plagiarism are presented, all based on real world data, which show the applicability of our approach.

II. KEY CONCEPTS

A. Representation of Text Fragments

Every text fragment is represented as a bitvector. The number of bits is determined by the size of the language (every bit represents a word from the language). But the size is not a critical value, because the bitvector can be compressed very well, as we have shown in [1]. One critical point is the order of the words. By sorting the words according to their frequency, we can later identify different regions of interest.

B. Size of Text Fragments

To determine a good size of our text fragments, we run a number of experiments with different text fragment sizes. To control the result, we use the PAN (**P**lagiarism Analysis, **A**uthorship Identification, and **N**ear-Duplicate Detection) document collection [3]. This collection is an evaluation framework for plagiarism detection with a known set of different plagiarisms. With this background information we run our plagiarism detection system on the PAN corpus with different sizes for the text fragments. Additionally, we use different similarity measures [4] like Jaccard, Dice, Overlap and Cosine coefficient. The best results were obtained by using a fixed fragment size of 55 words for all the used simlarity measures.

C. Similarity measure

So far, we used the pure Jaccard measure to calculate the similarity between text fragments. Because we also want to incorporate synonyms and hyperonyms in our systems, we must adapt the similarity measure accordingly. This can be done by introducing an additional bitvector A', which handles all the synonymes and hyperonyms in A. The modified Jaccard coefficient looks like in (1):

$$Jaccard'(A,B) = \frac{|A \cap B| + |A' \cap B|}{|A \cup B|}. \qquad (1)$$

The synonyms and hyperonyms were determined with the help of the WordNet library [5].

D. Integration of a Weighting Factor

As described before, the ordering of the words in the bitvector results from the frequency of the words in the language. We additionally introduce a weighting, by splitting the bitvector of every text fragment into multiple parts. Thereby, we are able to give the words which occur rarely, but are therefore more relevant, a higher weight compared to more frequent words with lower relevance. Finally, we build five categories of words, with increasing weight for the words in the higher categories. The detailed formula is presented and explained in [2].

 ISBN: 978-1-61208-408-4

E. Visualization

Instead of facing text fragments from the suspicious document and text fragments from the candidate set, which seem to be similar in some sense, we provide a graphical representation, abstracting from textual representation. In contrast, we are using heatmaps to express the similarity between documents. A heatmap is a two dimensional representation (a matrix), where the x and y-axis represent the text fragments from the suspicious document and a candidate document. Each field in the matrix now represents the calculated similarity value between two text fragments. The value is presented by a color gradient from white to red. A white color means, that there is no similarity between the text fragments, a red value indicates a high similarity. Figures 1 and 2 show examples of such heatmaps.

III. Experimental Results

In this section, we present a number of results. We start with the simples plagiarism, the copy & paste and continue to the more complex ones.

A. Copy & Paste Plagiarism

In this example, we use the speech of president Barrack Obama about "net neutrality" [6] as source for our plagiarism, take two fragments out of this document and inserted them into another document from the White House about the NSA scandal. The result is shown in Figure 1 on the left side. The two inserted fragments can easily be identified by so called "plagiarism lines". If you take a closer look at the heatmap, you can see, that not only the points on the well visible diagonal lines have a much higher similarity values, compared to the rest of the document, but also the values which are close to the plagiarism line. The reason for this behaviour is a partial overlapping of text from adjacent fragments. Nevertheless, the plagiarized parts can easily be identified.

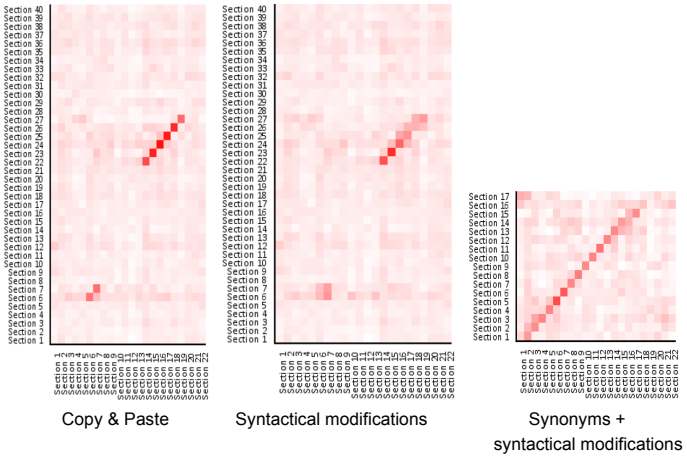


Figure 1. Results for experiments 1-3.

B. Syntactical Changes

Next, we modified the previous used copy & paste plagiarism and manipulated the plagiarized parts manually by changing the syntax (reconstruction of sentences). The result is shown in the middle of Figure 1. Compared to the simple copy & paste plagiarism the plagiarism lines are more blurred, but still very easy to identify.

Figure 2. Heatmaps of experiments 4 and 5.

C. Use of Synonyms & Syntacical Changes

In this example, we also exchange words by their synonyms. Concretely, we again take the document about "net neutrality" and use the Google translator [7] to make a number of consecutive translations into different languages. So, in this case, we start with Obama's speech about "net neutrality" and translate it first to German, then the result to French, Spanish and in a last step back to English. Then we compare the original document and compare it with the document we achieved by a number of successive translation steps. The result can be seen on the right side of Figure 1. An inspection of the orginal and the translated text shows, that a big number of differences exist. So, for example the final text only consists of 17 text fragments compared to 22 fragments of the original text. Nevertheless, our algorithm is able to clearly detect the plagiarism.

D. Cross Language Plagiarism

In contrast to the last experiment, where an automatic transation was generated, we use a human translation in this experiment to detect cross language plagiarisms. We use books, which were available in multiple languages, from the Gutenberg Project [8]. So, in a first step we translated the German version of the book *"The adventures of Tom Sawyer"* from Marc Twain with the Google translator into English and than compared the translated version with the original English version of the book. On the left side of Figure 2 you can see the result. Also in this case, where the result of a human translator is compared with an automatic translation the plagiarism line is clearly visible.

E. Theft of Ideas

In our last example we want to test our algoithm against "Theft of Idea" plagiates. This is the most difficult plagiarism format to detect. A prerequisite was, that the two documents must be written from different authors and don't be translated one from the other. As test documents, we choose the description (first 17 episodes) of the sitcom "Big Bang Theory" from wikipedia and compare it with the correponding description found from the Internet Movie database (IMDb).

So, both authors describe the content from their perspective, with the common background that they have seen the same 17 episodes. The result of the comparision can be seen on the right side of Figure 2. Here, we can't see any more a clear line in our heatmap, but still an diagonal area with higher values.

 ISBN: 978-1-61208-408-4

IV. Related Work

Gottron described in [9] an approach which can be compared with our visualization technology. He also used a two dimensional representation to compare a suspicious document with another document from the candidate set. In contrast to our work, only exact matches can be visualized. Like in our approach, the appearance of diagonal lines indicate the presence of a plagiarism. The "Bag of Word" model for storing text fragments is quite common in text retrieval and plagiarism detection. This is typically done with the vectorspace model [4]. In contrast (or extension) to this approach, we further relax the information about the content of a text fragment, by only storing the information if a word appears in a fragment, but not how often it appears. This relaxation makes our approach robust against syntactical changes and allows us to use compressed bitvectors instead of integer arrays as typically used in the vectorspace approach.

V. Conclusion

We demonstrated the efficiency of our algorithm to detect obfuscated plagiarisms. The algorithm is based on the concept of storing text fragments in form of a compressed bitvector and perform similarity operations on these bitvector using an adapted version of the Jaccard coefficient. The coefficient was adapted to also support the inclusion of synonyms and hyperonyms as well as to allow a weighting of the words in the vocabularity.

Based on our tests, we identified a text fragment length of 55 as optimal. This can probably be extended by also incorporating the concept of sentences and allowing to vary the length of text fragments in a range between 40 and 60.

Another interesting research direction is the use of our approach for the "Source Retrieval" part of a plagiarism detection system. In this case, a much larger size of the text fragments should be used and also sophisticated indexes to limit the number of similarity tests must be established.

References

[1] A. Schmidt, R. Becker, D. Kimmig, R. Senger, and S. Scholz, "A concept for plagiarism detection based on compressed bitmaps," in DBKDA'14: Procceedings of the Sixth International Conference on Advances in Databases, Knowledge, and Data Applications. IARIA, 2014, pp. 30–34.

[2] S. Bühler, "Konzeption und Realisierung eines Systems zur Erkennung verschleierter Plagiate (english title: Conception and Implementation of a System to Detected Obfuscated Plagiarisms)," Bachelor's Thesis, Department of Informatics and Business Information Systems, University of Applied Sciences, Karlsruhe, Jan 2015.

[3] M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso, "An evaluation framework for plagiarism detection," in Proceedings of the 23rd International Conference on Computational Linguistics: Posters, ser. COLING '10, 2010, pp. 997–1005.

[4] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. New York, NY, USA: Cambridge University Press, 2008.

[5] "JWNL - Java WordNet Library - Dev Guide (14.12.2007)," http://jwordnet.sourceforge.net/handbook.html, [accessed 2-April-2015].

[6] B. Obama, "Speech about Net Neutrality," Nov 2014, http://www.whitehouse.gov/net-neutrality, [accessed 2-April-2015].

[7] 2015, https://translate.google.com.

[8] "Gutenberg project," https://www.gutenberg.org, [accessed 24-Jan-2015].

[9] T. Gottron, "External plagiarism detection based on standard ir technology and fast recognition of common subsequences - lab report for pan at clef 2010." in CLEF (Notebook Papers/LABs/Workshops), M. Braschler, D. Harman, and E. Pianta, Eds., 2010.

 ISBN: 978-1-61208-408-4

A New Spatial Database Management System Using an Hyperbolic Tree

Telesphore Tiendrebeogo
Polytechnic University of Bobo-Dioulasso
Bobo-Dioulasso, Burkina Faso
tetiendreb@gmail.com

***Abstract*—Spatial information processing has been a focus of research in the past decade. In spatial databases, data are associated with spatial coordinates and extents, and are retrieved based on spatial proximity. A formidable number of spatial indexes have been proposed to facilitate spatial data retrieval. In this paper, we propose a new scalable, reliable and consistent architecture for spatial database indexing based on hyperbolic topology. This structure uses a strategy that enables to avoid such an exhaustive search that is based on the use of a centralized index. Our architecture is comparable to the well-known R-tree structure, but uses the hyperbolic geometry properties with specific model called "Poincaré disk model". It uses a distributed balanced Q-degree spatial tree that scales with spatial data objects insertions into potentially any number of storage servers through virtual hyperbolic coordinates taken in hyperbolic space. In other words, we propose a new model based on structured Distributed Hash Table (DHT) in which a user/application of each server is considered as a client node. Thus, a database server can connect to the system through a random database server that already addresses the tree with a free address given by one other database server of the system. Database servers addresses are associated with virtual hyperbolic coordinates computed by the system. In this paper, we consider a 3-regular Hyperbolic-tree in the hyperbolic plane; so, except for the root of this tree, others servers have in the best cases two addresses to give a new one (server's father uses one address). In our work, we perform simulations and we analyze the practicability and reliability compared with other DHT, such as Chord, Pastry, Kademlia used in other spatial databases. The results show that our spatial architecture based on a hyperbolic tree is viable, consistent and has acceptable performances given the scalability and flexibility it could provide to distributed database applications.**

Keywords–Spatial Database; Management; Hyperbolic-Tree; Query; Storage.

I. INTRODUCTION

The typical notion of a database is that of a system in which one stores data about a fixed enterprise that one has modelled in some way. Quite often, the modelling is constrained in ways that help to reduce the complexity to a manageable level. The usual kinds of uses to which that data are put include fairly straightforward applications, like inventory management, shipping systems, and payroll systems. However, research in databases has advanced the state-of-the-art sufficiently that we are now moving into non-traditional applications that could not have been foreseen even ten years ago. This move has been largely demand-driven. For example, the emergence of spatial databases was initially driven by the demand for managing large volume of spatial data used in the geosciences and Computer-Aided Design (CAD). Also, we notice to a collection of spatial data that increasing as never seen.

Thus, in this paper, we aim at indexing large datasets of spatial objects, each uniquely identified by an Object IDentifier (OID) and stored in the hyperbolic-tree with a given address. We define a scalable and reliable index that generalizes R-tree structure for a Spatial Database System (SDS) [1]. Our architecture has conformed to the general principles of a Seamless Data Distribution System (SDDS) [2]:

- No central directory is used for data addressing,
- Servers are dynamically added to the system when needed,
- Servers address the structure through coordinates.

Our model permits redundancy of object references, like MSPastry [3], Chord [4], and Kademlia [5]. The fundamental principle of our systems is to map a large database object identifier space onto a set of servers localisation in a deterministic and distributed way. Roughly, given an object key, the system is able to obtain the locations of database servers where the corresponding values are stored. The existing issues consist mainly of how they store and look up key-value pairs of spatial data. These two primitives remain a challenging problem in distributed data despite the considerable research efforts that have been made. A spatial information system is a more general term to describe an SDS that has application customized tools for analysing spatial properties of the data. It has to model reality, integrate spatial data acquired from different sources and by different means, and support processing and analysis of data on demand. Without being exhaustive, we outline here the key issues that we address. Besides, for instance, store and lookup queries are messages forwarded in a progressive manner across cluster paths to the adequate server that contain response. Also, the underlying routing process thus requires that each database server maintains the status of its connections to other databases servers increasing drastically the number of messages exchanged, and this may constitute a severe scaling limitation. Furthermore, each database server's routing table must always be kept as small as possible in order to both reduce the store/lookup query latency and not affect the performances of applications built over our spatial database system. The same applies to the number of routing hops that must not grow too fast with the number of database servers in the system [6]. Moreover, most of spatial database

 ISBN: 978-1-61208-408-4

systems suffer from the lack of flexibility in storage queries (i.e., values placement) involving consequently a heavy lookup traffic load on the lookup paths to the underlying servers. We argue that we can address and overcome simultaneously all the aforementioned issues by maintaining a good trade off between robustness, efficiency and system complexity. To this end, we promote an indexing system for spatial database relying on hyperbolic geometry. In this paper, we make the following contributions:

- We provide a new architecture for indexation in the distributed database system without any constraints. Thus, at first, the database servers can connect arbitrarily to each other during his phase of connection to the system. In addition, the data objects can be inserted, updated or deleted out of the system efficiently, without the cost of maintaining any global knowledge. Our approach is based on the ground breaking work of Kleinberg [7] that we have enhanced in order to manage a dynamic topology which is able to grow and shrink over time.
- We propose a specific key-based routing system for storing the mapping of database object identifier to addresses over hyperbolic system in a flexible and efficient way. Values are stored in a way to avoid overloading a particular zone of the spatial database system. Furthermore, storage and retrieving queries can be solved within $O(logN)$ hops.
- To improve database object availability and access performance, our system that is based on a Distributed Hash Table (DHT) system embeds a redundancy and caching scheme that can be parametrized to get a good trade off between reliability and storage consumption.
- We have carried out simulations, firstly, to demonstrate the interest in terms of feasibility to build a spatial database system over architecture based on the hyperbolic plane, and secondly, to compare our solution with others solutions existing based on DHT.

The remainder of this paper is organized as follows. Section II gives a brief overview of the related previous work. Section III highlights some properties of the hyperbolic plane when represented by the Poincaré disk model. Section IV defines the local addressing and greedy routing algorithms of the distributed database system. Section V defines the binding algorithm of our spatial database system. Section VI presents the results of our practicability evaluation obtained by simulations and we conclude in Section VII.

II. Related work

Spatial databases generally refer to the collection of data which have spatial coordinates and are defined within a space. Until recently, in the most of the spatial indexing design, objects in SDS are of irregular shape, and the irregular shape of these objects is the main cause for expensive spatial operator computation. Consider one of the simplest operators, such as intersection. The intersection of two polyhedra requires testing all points of one polyhedron against the other. The intersection of two polyhedra objects is not always a polyhedron; the intersection may sometimes consist of a set of polyhedra. The efficiency of these operations depends on the representation of the data [8], storage of objects [9][10] and retrieval of relevant data for computation. When data objects are stored in disks, the semantics of the data must be captured so that they can be reconstructed correctly and efficiently. Representation schemes developed for geometric modelling [3][4][5] are suitable for spatial objects. In all that precedes, they have used an Euclidean space splitting or ring, which differs from our case; we are interested in a tiling of the hyperbolic space, generally, and more particularly, to the hyperbolic plan. An elementary property of the Euclidean space is the impossibility to create more than two half planes without they intersect. Our embedding is based on the geometric property of the hyperbolic plane which allows to create distinct areas called half planes. As explained by Miquel [11], in the hyperbolic plane, we can create n half spaces pair wise disjoint whatever n. This property is the base of our embedded algorithm (red line Figure 1). Another important property is that we can tile the hyperbolic plane with polygons of any size, called p-gons. Besides the Merkle Hash Tree (MH-tree) [12] is a main-memory of Advanced Data System (ADS) that has influenced several authenticated processing techniques. It is a binary tree that hierarchically organizes hash1 values (or digests). We use the similar principle except that the key serves to calculate virtual coordinates of the hyperbolic space for more flexibility. The VB-tree [13] is a disk-based ADS that establishes the soundness, but not the completeness, in term of security of the obtained result. For keyword-based retrieval, they have integrated R-tree [14] with spatial index and signature file [15]. By combining R-tree and signature, they have developed a structure called the IR2-tree [15]. IR2-tree has merits of both R-trees and signature files. Our system, like IR2-tree, preserves objectś spatial proximity, which is important for solving spatial queries; but furthermore, it permits a reliable replication, like in Chord, MSPastry and Kademlia.

III. Hyperbolic geometry

The model that we use in our system to represent the hyperbolic plane is called the Poincaré disk model. In the Poincaré disk model, the hyperbolic plane is represented by the open unit disk of radius 1 centered at the origin. In this specific model:

- Points are represented by points within this open unit disk.
- Lines are represented by arcs of circles intersecting the disk and meeting its boundaries at right angles.

In this model, we refer to points by using complex coordinates.

An important property is that we can tile the hyperbolic plane with polygons of any sizes, called p-gons. Each tessellation is represented by a notation of the form $\{p, q\}$, where each polygon has p sides with q of them at each vertex. There exists a hyperbolic tessellation $\{p, q\}$ for every couple $\{p, q\}$ obeying $(p-2)*(q-2) > 4$. In a tiling, p is the number of sides of the polygons of the *primal* (the black edges and green vertices in Figure 1) and q is the number of sides of the polygons of the *dual* (the red triangles in Figure 1). Our purpose is to partition the plane and address each node uniquely. We set p to infinity, thus transforming the primal into a regular tree of degree q. The dual is then tessellated with an infinite number

 ISBN: 978-1-61208-408-4

of q-gons. This particular tiling splits the hyperbolic plane in distinct spaces and constructs an embedded tree that we use to assign unique addresses to the nodes. An example of such a hyperbolic tree with $q = 3$ is shown in Figure 1.

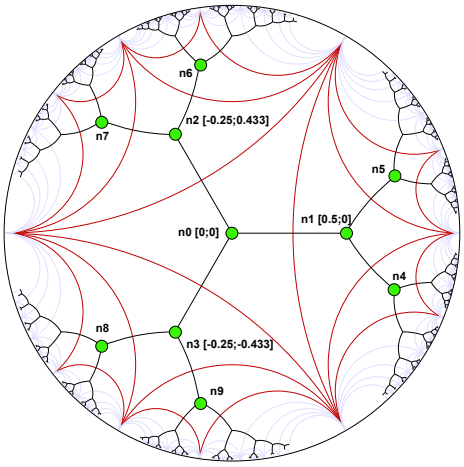


Figure 1. 3-regular tree in the hyperbolic plane.

In the Poincaré disk model, the distances between any two points z and w are given by curves minimizing the distance between these two points and are called geodesics of the hyperbolic plane. To compute the length of a geodesic between two points z and w and thus obtain their hyperbolic distance $d_{\mathbb{H}}$, we use the Poincaré metric which is an isometric invariant given by the formula:

$$d_{\mathbb{H}}(z,w) = argcosh(1 + 2 \times \frac{|z-w|^2}{(1-|z|^2)(1-|w|^2)}) \quad (1)$$

This formula is used by the greedy routing algorithm shown in the next section.

IV. Naming and Binding Technical in the Hyperbolic Plane

We now explain how we create the hyperbolic addressing tree for spatial database servers joins and how queries can be routed in our SDS. We propose here a dynamic and scalable hyperbolic greedy routing algorithm [16]. The first step in the creation of our SDS based on hyperbolic-tree of servers nodes is to start the first database server and to choose the degree of the addressing tree.

We recall that the hyperbolic coordinates (i.e., a complex number) of a server node of the addressing tree are used as the address of the corresponding database server in the SDS. A server node of the tree can give the addresses corresponding to its children in the hyperbolic-tree. The degree determines how many addresses each database server will be able to give for news nodes servers connexions. The degree of the hyperbolic-tree is defined at the beginning for all the lifetime of the SDS. The SDS is then built incrementally, with each new data server joining one or more existing data servers. Over time, the data servers will leave the overlay until there is no server left which is the end of the SDS. So, for every data object that must be stored in the system, an OID is associated with it and map then in key-value pair. The key will allow to determine in which data servers the object will be stored (like in the Section V). Furthermore, when a data object is deleted, the system must be able to update this operation in all the system by forwarding query through the latter. This method is scalable because unlike [17], we do not have to make a two-pass algorithm over the whole spatial system to find its highest degree. Also, in our system, a server can connect to any other server at any time in order to obtain an address.

The first step is thus to define the degree of the tree because it allows building the *dual*, namely the regular $q - gon$. We nail the root of the tree at the origin of the *primal* and we begin the tiling at the origin of the disk in function of q. Each splitting of the space in order to create disjoint subspaces is ensured once the half spaces are tangent; hence, the *primal* is an infinite q-regular tree. We use the theoretical infinite q-regular tree to construct the greedy embedding of our q-regular tree. So, the regular degree of the tree is the number of sides of the polygon used to build the *dual* (see Figure 1). In other words, the space is allocated for q child database servers. Each database server repeats the computation for its own half space. In half space, the space is again allocated for $q - 1$ children. Each child can distribute its addresses in its half space. Algorithm 1 shows how to compute the addresses that can be given to the children of a database server. The first database server takes the hyperbolic address (0;0) and is the root of the tree. The root can assign q addresses.

Algorithm 1 Calculating the coordinates of a database server's children.

```
1:  procedure CalcChildrenCoords(server, q)
2:      step ← argcosh(1/sin(π/q))
3:      angle ← 2π/q
4:      childCoords ← server.Coords
5:      for i ← 1, q do
6:          ChildCoords.rotationLeft(angle)
7:          ChildCoords.translation(step)
8:          ChildCoords.rotationRight(π)
9:          if ChildCoords ≠ server.ParentCoords then
10:             StoreChildCoords(ChildCoords)
11:         end if
12:     end for
13: end procedure
```

This distributed algorithm ensures that the database servers are contained in distinct spaces and have unique coordinates. All the steps of the presented algorithm are suitable for distributed and asynchronous computation. This algorithm allows the assignment of addresses as coordinates in dynamic topologies. As the global knowledge of the SDS is not necessary, a new server can obtain coordinates simply by asking an existing server to be its parent and to give it an address for itself. If the asked server has already given all its addresses, the new server must ask an address to another existing database server. When a new server obtains an address, it computes the addresses (i.e., hyperbolic coordinates) of its future children(The new database servers which shall connect to the SDS). The addressing hyperbolic-tree is thus incrementally built at the same time than the SDS.

When a new database server has connected to database servers already inside the SDS and has obtained an address

 ISBN: 978-1-61208-408-4

from one of those database servers, it can start sending requests to store or lookup database object in the SDS. The routing process is done on each database server on the path (starting from the sender) by using the greedy algorithm 2 based on the hyperbolic distances between the servers. When a query is received by a database server, the database server computes the distance from each of its neighbors to the destination and forwards the query to its neighbor which is the closest to the destination (destination database server computing is given in the Section V). If no neighbor is closer than the server itself, the query has reached a local minima and is dropped.

Algorithm 2 Routing a query in the distributed database system.

function GETNEXTHOP($dataserver$, $query$) **return** dataserver
  $w = query.destinationServerCoords$
  $m = server.Coords$
  $d_{min} = argcosh\left(1 + 2\frac{|m-w|^2}{(1-|m|^2)(1-|w|^2)}\right)$
  $p_{min} = server$
  for all $neighbor \in server.Neighbors$ **do**
    $n = neighbor.Coords$
    $d = argcosh\left(1 + 2\frac{|n-w|^2}{(1-|n|^2)(1-|w|^2)}\right)$
    if $d < d_{min}$ **then**
      $d_{min} = d$
      $p_{min} = neighbor$
    end if
  end for
  return p_{min}
end function

In a real network environment, link and server failures are expected to happen often. If the addressing hyperbolic-tree is broken by the failure of a database server or link, we flush the addresses attributed to the servers beyond the failed server or link and reassign new addresses to those servers (some servers may have first to reconnect to other servers in order to restore connectivity). But, this solution is not developed in this paper.

V. HYPERBOLIC ADRESSING AND DESTINATION SPATIAL DATABASE SERVERS COMPUTING

In this section, we explain how our SDS computes the destination database servers addresses for stores and retrieves queries. Indeed, the first server contacted by a client (prime server) for sending a query in the system considers this latter as a data object that can be stored or looked up. Thus, this server generates an OID associate to the data object and this latter is mapped into hyperbolic addresses corresponding to destination database servers addresses in the hyperbolic-tree. Our solution of SDS is a structured DHT system that uses the local addressing and the greedy routing algorithms presented in Section IV. On start-up, each new client query is associated with the data object with OID corresponding to the name of the query and that identifies the query it runs on. This name will be kept by the data object during all the lifetime of the SDS. When the prime database server computes some specific addresses of database servers, when it is about a storage query, it stores the name (OID) and value of query in these specific addresses of SDS, thus the data object in the DHT; when it is about a retrieving query, it contacts database servers whose addresses have been computed. In our spatial system, the name is used as a key by a mathematical transformation. If the same name is already stored in the SDS, an error message is sent back to the prime server (Server to whom the client is directly bound) in order to generate another name. Thus, the SDS structure itself ensures that OIDs are unique.

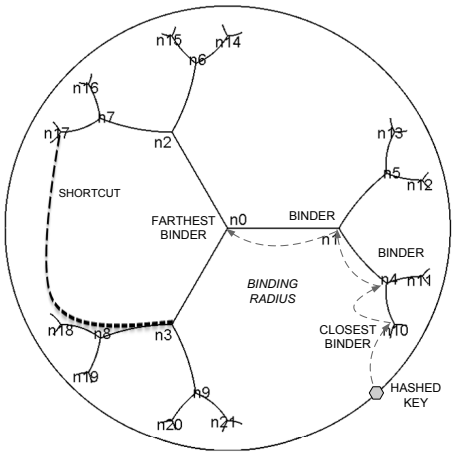


Figure 2. Hyperbolic DHT system.

A (OID, value) pair, with the name acting as a key by mapping is called a *binding*. Figure 2 shows how and where a given binding is stored in the SDS. A binder is any database server that stores these pairs. The depth of a server in the addressing hyperbolic-tree is defined as the number of parent servers to go through for reaching the root of the hyperbolic-tree (including the root itself). When the distributed database system is created, a maximum depth for the potential binders is chosen. This value is defined as the *binding hyperbolic-tree depth*. To ensure a load balancing of the system, the depth d is chosen such d minimize the following in-equation :

$$p \times (\frac{(1-(p-1)^d)}{2-p}) + 1 \geq N \quad (2)$$

with d the depth, p ($p \geq 3$) the degree and N the number of database servers.

When a new database server joins the SDS by connecting to other servers, it obtains an address from one of these servers. Next, the server stores its own binding in the system. So, during his life, each database server tries to join others by sending a join query. Each server cannot accept that a limited number of join queries independently of the degree of the Hyperbolic-tree. The new connections serve as shortcuts during the phases of storage and retrieving of data objects. We call these connections, shortened links, as indicated in Figure 2.

A. Storage query process

When a client wants to send a storage query (insert, etc.), the first server with whom it is connected considers a query as an object (thus generates an OID) and creates a key by hashing its name with the SHA-512 algorithm. It divides the 512-bit key into 16 equally sized 32-bit sub keys (for redundant

storage). The server selects the first sub key and maps it to an angle by a linear transformation.

The angle is given by:

$$\alpha = 2\pi \times \frac{\texttt{32-bit subkey}}{\texttt{0xFFFFFFFF}} \quad (3)$$

The database server then computes a virtual point v on the unit circle by using this angle:

$$v(x,y) \text{ with} \begin{cases} x = cos(\alpha) \\ y = sin(\alpha) \end{cases} \quad (4)$$

Next, the database server determines the coordinates of the closest binder to the computed virtual point above by using the given *binding tree depth*.

In Figure 2, we set the *binding Hyperbolic-tree depth* to three to avoid cluttering the figure. It is important to note that this closest binder may not really exist if no database server is currently owning this address. The database server then sends a storage query to this closest database server. This query is routed inside the SDS by using the greedy algorithm of the Section IV. If the query fails because the binder does not exist or because of database server/link failures, it is redirected to the next closest binder which is the father of the computed binder. This process continues until the query reaches an existing binder database server which can be any database server on the path from the computed closest binder to the center database server.

Upon reaching an existing binder, the query is stored in that binder. The query can thus go up the addressing Hyperbolic-tree to the center database server having the address (0;0) which is the farthest binder. The path from the computed closest binder to the farthest binder is defined as the binding radius. This process ensures that the queries are always stored first in the binders closer to the unit circle and last in the binders closer to the disk center. However, to avoid overloading the farthest binder particularly and to keep a load balancing, we limit the number of storages S like follow:

$$S \leq \lfloor \frac{1}{2} \times \frac{log(N)}{log(q)} \rfloor \quad (5)$$

with N equal to number of distributed database servers, q to degree of hyperbolic-tree.

Furthermore, if addressing hyperbolic-tree is unbalanced (because of servers leaving or failing in the system), many queries may be stored in database servers close to the center thus overloading them. Besides the previous solution, any binder will be able to set a maximum number of stored queries and any new database server to store will be refused and the query redirected as above. Besides, to provide redundancy and so ensure the availability and reduce the latency period in the lookup process, the database server does the storage process described above for each of the other 15 sub keys. Thus 16 differents binding radiuses will be used at the most and this will improve the even distribution of the pairs (key-value).

In addition to this and still for redundancy purposes, a pair key-value of the data object may be stored in more than one database server of the binding radius. A binder could store a data object and still redirect its query for storage in other ancestor binders. The number of stored copies of a key-value pair along the binding radius may be an arbitrary value set at the SDS creation. Similarly, the division of the key in 16 sub keys is arbitrary and could be increased or reduced depending on the redundancy needed. To conclude, we can define two redundancy mechanisms for storage copies of a given binding:

1) We can use more than one binding radius by creating several uniformly distributed subkeys.
2) We can store the data object key-value pair in more than one binder in the same binding radius.

Algorithm 3 Storage algorithm in the general context

```
1: function StoreProcess(PrimeDataServer Query)
   return 0
2:     OID ← Query.GetOID()
3:     Key ← Hash(OID)
4:     for A doll r ∈ R_Circular
5:         d ← P_Max
6:         i ← 1
7:         while i ≤ ⌊½ × log(N)/log(q)⌋ && d ≥ 0 do
8:             SubKey[r][d] ← C_Subkey(Key)[r][d]
9:             TgServAd[r][d] ← C_Ad(SubKey[r][d])
10:            TgServ ← GetTg(TgServAd[r][d])
11:            if route(Query, TgServer) then
12:                i++
13:                put(OID, Query)
14:            end if
15:            d−−
16:        end while
17:    end for
18:    return 0
19: end function
```

These mechanisms enable our SDS to cope with a non-uniform growth of the database servers and they ensure that a data object will be stored in a redundant way that will maximize the success rate of its retrieval.

The numbers of sub keys and the numbers of copies in a radius are parameters that can be set at the creation of the SDS. Increasing them leads to a trade-off between improved reliability and lost storage space in binders.

Our solution has the property of consistent hashing: if one database server fails, only its keys are lost but the other binders are not impacted and the whole system remains coherent.

As in many existing systems, pairs (key-value) will be stored by following a hybrid soft and the hard state strategy. Every database server has to verify every X unit of time that his neighbour is alive. When the database server leaves the system, his neighbor tries to connect with his father for keeping the hyperbolic-tree stable. But, we do not detail this dynamic process in our paper. Algorithm 3 illustrates this mechanism.

B. Lookup query process

Now, if the client wants to lookup a data object in the connected and use to store the data objects distributed SDS, a prime server is contacted and this latter generates an OID for the client query. Here again, OID is mapped into a key by SHA-512 algorithm, thus the 512 bits key is divided into 16

sub keys. Each sub key by the process describe in the Section V-A, will be transform into address that represent address of the database server where data object is stored. This latter is contacted by prime database server for update, delete or select the value associated.

Algorithm 4 Lookup algorithm in general context for inserting, deleting and updating of data object

```
1: function LookupProcess(PrimeDataServer, Query) return Value
2:     OID ← Tg.GetQueryOID()
3:     Key ← Hash(QueryOID)
4:     for A doll r ∈ R_Circulare
5:         d ← P_Max
6:         i ← 1
7:         while i ≤ ⌊ 1/2 × log(N)/log(q) ⌋ && d ≥ 0 do
8:             TgServAd[r][d] ← GetValue(Key)
9:             Value ← GetValue(TgSerAd[r][d], OID)
10:            if Value != null then
11:                if Query == delete then
12:                    delete(OID)
13:                end if
14:                if (Query == update) then
15:                    update(OID)
16:                end if
17:                if Query == select then
18:                    return Value
19:                    break
20:                end if
21:                i++
22:            end if
23:            d−−
24:        end while
25:    end for
26:    return 0
27: end function
```

When the redundancy mechanism has been used to store the data object, lookup process repeats the latter process of lookup for any sub key, thus, the operation will be performed on all database servers that contain the data object. Our SDS ensures then the coherence of data objects of the spatial database. The Algorithm 4 illustrates this mechanism.

VI. Experimental evaluation

We performed experiments evaluating the behaviour of our SDS over large datasets of an hyperbolic plane. Furthermore, we consider that the system is dynamic (there is join or leave of database servers during the simulation) with rates of churn variables. We use a simulator Peersim [18] for SDS simulation and it allows to obtain dataset OIDs by generation following the uniform distribution. The study involved the following parameters of our SDS :

- number of database servers connected and used to store the distributed data objects. Here we have considered 10000 at the beginning;
- we try to store 6 million data objects in our SDS following an exponential distribution with a median equal to 10 minutes;
- we perform simulation during 2 hours and, we fixed the capacity of each server to 6000 objects.
- we consider that the rate of churn varies between 10% and 60%.

We studied the behaviour of our architecture for the data objects storage and retrieving in the system. So, we are interested by certain properties.

A. Balance of the Hyperbolic-tree

Figure 3 shows an experimental distribution of points corresponding to the scatter plot of the distribution of database server in our system. Thus, we can mark that our hyperbolic-tree is balanced. Indeed, we can noticed from part and others around the unit circle which we have database servers. This has an almost uniform distribution around the root, what implies that our system builds a well-balanced tree what will more easily allow to reach a load balancing of storage.

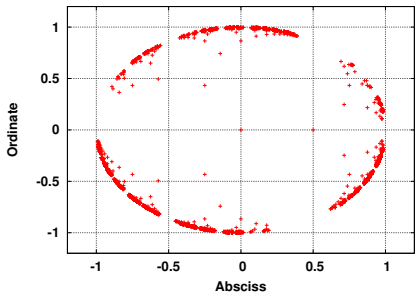


Figure 3. Scatter plot corresponding to the distributed database servers.

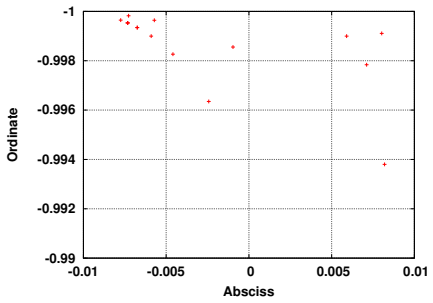


Figure 4. Distribution of database servers in the neighborhood on the edge of the unit circle.

Figure 4 shows correspondingly Poincaré disk model that no address of database server belongs on the edge of the unit circle. Indeed, the addresses of database server were obtained by projection of the tree of the hyperbolic plane in a circle of the Euclidean plan of radius1 and of center of coordinate (0; 0).

The distances between any two points z and w in the Poincaré disk model given in equation 6 can also be written

like follows :

$$d_{\mathbb{H}}(z,w) = argcosh(1+2\Delta) \quad (6)$$

with:

$$\Delta = \frac{|z-w|^2}{(1-|z|^2)(1-|w|^2)} \quad (7)$$

For more details on the Poincaré metric, we refer the reader to the proof in [19]. The hyperbolic distance $d_{\mathbb{H}}(z,w)$ is additive along geodesics and is a Riemannian metric. Thus, we consider the formulae 6, the distance between O (0;0) and any point on edge to the circle so much towards the infinite. All the computed addresses associated to the database servers should be inside the unit circle. This result shows that our distributed database system can grow in the infinite, in theory. In practice of other parameters than we have not evoke in this paper have to intervene. Our system, besides ensuring the security of the data object by hashing them, the availability by the replication of the storage so the coherence of the data object by the lookup for all the occurrences of an OID for the update, the deletion, allows a passage to the scale in term of the number of database servers interconnected in the system.

B. Load balancing of the Hyperbolic-tree

Figure 5 shows a plot of the evaluation of the average number of objects stored by database server in time. This figure shows a regular growth of this number of data objects stored in function of time. Indeed, 293.27 data objects on average are stored by database server after 10 minutes vs 620.4 after 2 hours. What is interesting to notice is that the standard deviations seem low, approximately 10 % of the average. This indicates a low dispersal of the number of objects stored around of average during the simulation.

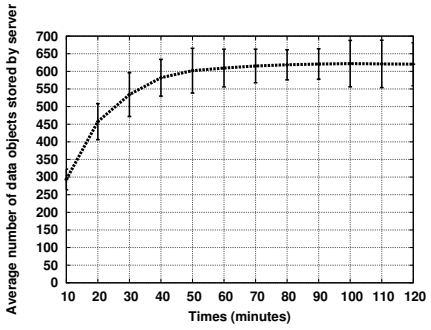


Figure 5. Distribution of the spatial database servers on the hyperbolic-tree.

Indeed, if we use our results to build the confidence interval, we can say that after 10 minutes of simulation, 68.2 % of the database servers stores between 263.69 and 322.71 data objects and 95 % stores it between 234.18 and 352.22 against 68.2 % of the database servers who stores between 560.18 and 681.58 data objects after 2 hours and 95 % of the database servers who stores between 497.95 and 742.84 data objects after 2 hours. In view of these , we can say that our system maintains a good enough load balancing between database servers which is to ensure the stability of our SDS.

C. Storage and retrieval path length in Hyperbolic-tree

Figures 6 and 7 show that during the simulation, queries in both cases can be answered within O (log N) or N equal to the number of database server building the system. Indeed, the standard deviation being very low (less than 5 % of the average for storage et retrieving), we did not represent it. In the worst case, queries contacts less than 4 database servers in a system which in account 10000, for either to store, or to retrieving a data object.

Besides, what is also interesting to note is that the plot decreases slowly to become stationary after around 100 minutes in both cases. It can be explained because during the simulation, the database servers create shortcuts as indicated in the Section V. These shortcuts allow to reach their target in fewer hops. The situation of stationarity is understandable by the fact that after a while, all the database servers reached their maximum of shortcuts create and most part of the queries is processed on average in less than 3.75 hops in both cases.

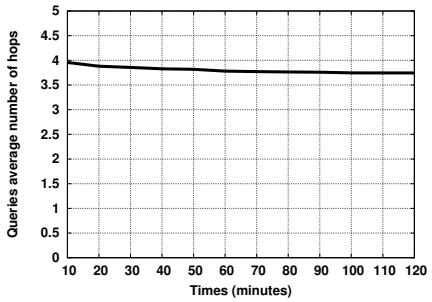


Figure 6. Path length of the storage's queries in Hyperbolic-tree.

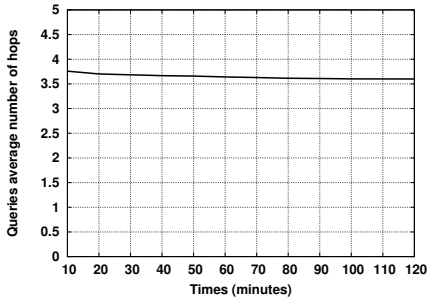


Figure 7. Path length of the lookup's queries in Hyperbolic-tree.

D. Analyses compared by the performances

Figure 8 shows that in a context where churn phenomenon is equal to 10%, we can notice that all the successes rates are between 83% and 88% when the churn rate becomes 60%, according to the number of replication, the success rates is between 18% and 67%. This result indicates that the replication strategy permits to reduce impact of the churn phenomenon on our spatial database performance.

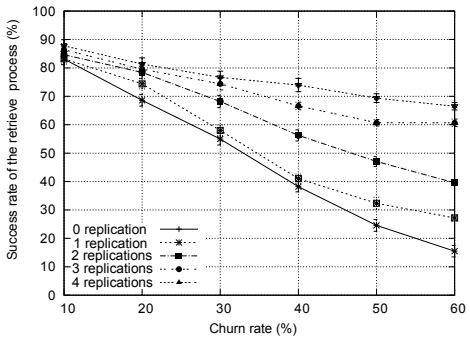


Figure 8. Evaluation of resilient strategy against the churn phenomenon.

Figure 9 indicates that when the churn rate is between 10% and 60%, the average number of hops to send query from source server to target server (server that contains the data object searched) is approximatively the same. Thus, through this result, we show that our system behaves well with compared with the others such as Chord, MSPastry and Kademlia. So, our spatial database that is based on this DHT algorithm has a good performance.

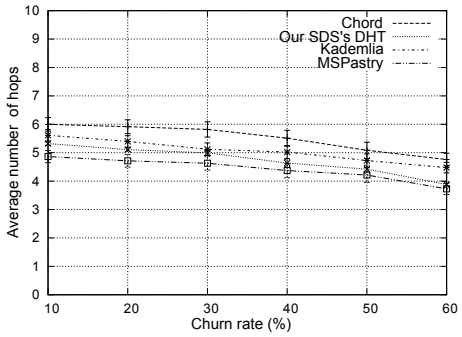


Figure 9. Comparison of the success rate depending to DHTs algorithms.

VII. Conclusion and Future Work

In this paper, we presented a new structure based on the Poincaré disk model. The hyperbolic tree which is used presents some properties that allow us to propose a consistent system of distributed database servers using the virtual address (hyperbolic coordinates). Next, we evaluated the performances of our system according to some parameters.

We showed that our system was scalable in terms of the number of database servers that we can connect as well as in term of number of hops to solve the queries. We also showed that the arrangement of the different database servers of the model allows us to keep a well-balanced tree. Furthermore, we showed that our system maintains a storage load balancing. Furthermore, this model based on Disk of Poincaré present a certain number of properties that permit him to be more consistent and hardness than other existing solution. All these results are encouraging and show us that our system is viable in a dynamic context. In the continuation of our work, we envisage to study our SDS in a realistic context and improve on Churn-resilient replication strategy.

References

[1] R. H. Güting, "An introduction to spatial database systems," The VLDB Journal, vol. 3, no. 4, Oct. 1994, pp. 357–399.

[2] S. Jajodia, W. Litwin, and T. J. E. Schwarz, "Lh*re: A scalable distributed data structure with recoverable encryption." IEEE, 2010, pp. 354–361.

[3] M. Castro, M. Costa, and A. I. T. Rowstron, "Performance and dependability of structured peer-to-peer overlays," in 2004 International Conference on Dependable Systems and Networks (DSN 2004), 28 June - 1 July 2004, Florence, Italy, Proceedings, 2004, pp. 9–18.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." New York, NY, USA: ACM, 2001, pp. 149–160.

[5] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric." London, UK, UK: Springer-Verlag, 2002, pp. 53–65.

[6] S. A. Idowu and S. O. Maitanmi, "Transactions- distributed database systems: Issues and challenges," IJACSCE., vol. 2, no. 1, Mar. 2014.

[7] R. Kleinberg, "Geographic routing using hyperbolic space," in in Proceedings of the 26th Annual Joint Conference of INFOCOM. IEEE Computer and Communications Societies, 2007, pp. 1902–1909.

[8] O. Günther, Efficient Structures for Geometric Data Management, ser. Lecture Notes in Computer Science. Berlin / Heidelberg / New York: Springer-Verlag, 1988, vol. 337.

[9] H. Lu, B. C. Ooi, A. DŠouza, and C. C. Low, "Storage management in geographic information systems," ser. Lecture Notes in Computer Science, vol. 525. Springer, 1991, pp. 451–470.

[10] T. Tiendrebeogo, D. Ahmat, and D. Magoni, "Reliable and scalable distributed hash tables harnessing hyperbolic coordinates," in NTMS'12, 2012, pp. 1–6.

[11] A. Miquel, "Un afficheur générique d'arbres à l'aide de la géométrie hyperbolique," in In Journées francophones des langages applicatifs (JFLA), 2000, pp. 49–62.

[12] R. C. Merkle, "A certified digital signature," in Proceedings on Advances in Cryptology, ser. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 218–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=118209.118230

[13] H. Pang and K.-L. Tan, "Authenticating query results in edge computing," in Proceedings of the 20th International Conference on Data Engineering, ser. ICDE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 560–. [Online]. Available: http://dl.acm.org/citation.cfm?id=977401.978163

[14] A. Guttman, "R-trees: A dynamic index structure for spatial searching," SIGMOD Rec., vol. 14, no. 2, Jun. 1984, pp. 47–57. [Online]. Available: http://doi.acm.org/10.1145/971697.602266

[15] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," ACM Trans. Database Syst., vol. 24, no. 2, Jun. 1999, pp. 265–318. [Online]. Available: http://doi.acm.org/10.1145/320248.320255

[16] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in Proceedings of the 29th Conference on Information Communications, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 2973–2981. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833893

[17] V. Gaede and O. Günther, "Multidimensional access methods," ACM Comput. Surv., vol. 30, no. 2, Jun. 1998, pp. 170–231. [Online]. Available: http://doi.acm.org/10.1145/280277.280279

[18] I. Kazmi and S. F. Y. Bukhari, "Peersim: An efficient & scalable testbed for heterogeneous cluster-based p2p network protocols." Washington, DC, USA: IEEE Computer Society, 2011, pp. 420–425. [Online]. Available: http://dx.doi.org/10.1109/UKSIM.2011.86

[19] A. F. Beardon and D. Minda, "The hyperbolic metric and geometric function theory," vol. 956. International Workshop on Quasiconformal Mappings And Their Applications, 2007, pp. 9–57.

 ISBN: 978-1-61208-408-4

Extending PostgreSQL with Column Store Indexes

Minoru Nakamura, Tsugichika Tabaru, Yoshifumi Ujibashi, Takushi Hashida, Motoyuki Kawaba, Lilian Harada
Computer Systems Laboratories
Fujitsu Laboratories Ltd.
Kawasaki, Japan
{nminoru, tabaru, ujibashi, hashida, kawaba, harada.lilian}@jp.fujitsu.com

***Abstract* — The importance of database systems to support mixed online transaction processing (OLTP) and online analytical processing (OLAP) workloads, the so-called OLXP workloads, has attracted much attention recently. Some research projects and also commercial database systems with focus on OLXP have appeared in the last few years. In this paper, we present our work aiming at extending the PostgreSQL OSS database system to efficiently handle OLXP workloads. Besides PostgreSQL's traditional OLTP-oriented row data store, a new OLAP-oriented column data store is added in the form of a new index. Unlike previous works focusing on OLXP, our column store index has no restrictions concerning data size and/or updatability. Therefore, transactional data inserted to PostgreSQL row data store become immediately available for efficient query processing using this new column store index.**

Keywords - PostgreSQL; columnar data; OLTP; OLAP; OLXP.

I. INTRODUCTION

Historically, database systems were mainly used for online transaction processing (OLTP) where transactions access and process only few rows of the data tables. Then, a new usage of database systems where queries access substantial portions of the data tables in order to aggregate few columns have evolved into the so-called online analytical processing (OLAP). The execution of OLAP query processing led to resource contentions and severely hurt mission-critical OLTP. Therefore, the data staging architecture where a dedicated OLTP database system whose changes are extracted, transformed and loaded into a separate OLAP database system was used for decades [1].

However, new real-time/operational business intelligence applications require OLAP queries on the current, up-to-date state of the transactional OLTP data [2] [3]. This OLXP workload, i.e., mixed workloads of OLTP and OLAP on the same database, has been extensively addressed recently. Some approaches adopt the columnar data store for both OLTP and OLAP [4] but it is still to be proven that columnar data stores can efficiently support mission-critical OLTP applications. Other hybrid approaches allow a data table to be represented simultaneously in both formats: row data for OLTP and columnar data for OLAP but with some constraints in the columnar data size and its updatability/synchronization with row data [5] [6].

In this work, we propose a new approach to enhance PostgreSQL [7] so that it can effectively handle OLXP workloads without any constraints on data sizes nor updatability. In our approach, columnar data are created as indexes of data tables with no size restrictions and are completely synchronized with row data for any insertions/updates/deletions in the data tables. There are other works that extend PostgreSQL to support columnar data stores for time-series data [8] and OLAP [9]. However, to the best of our knowledge, this is the first work extending the PostgreSQL OSS database to support OLXP.

In section II we describe how data are stored and queries executed using the proposed column store index. Section III shows some preliminary evaluation results. Some concluding remarks are presented in section IV.

II. COLUMN STORE INDEX

In the following, we give a brief description of how we are extending PostgreSQL to efficiently support a column store index.

A. Data store

Updates to PostgreSQL's row data store have to be immediately reflected to the column store index without degradation of the performance of OLTP transactions. Some previous works adopt the Write-Optimized-Storage (WOS) /Read-Optimized-Storage (ROS) approach where the updates are first buffered at WOS in row format and then asynchronously transferred to ROS in columnar format [10]. However, unlike previous approaches, as illustrated in Fig. 1, instead of writing all the row data into WOS, we only write PostgreSQL's original Tuple Identifiers (TIDs) into the Insert List (InsL). Only when transferring data from InsL to ROS, the columnar data values identified by the TIDs are used. For performance reasons, deletions are not in-place, but a Deletion List (DelL) and a ROS delete-vector are used to immediately hide data that are physically deleted later. Data in ROS are grouped in units, called extents, for data management.

We use the same Multi-Version Concurrency Control (MVCC) used in PostgreSQL to guarantee data consistency when transferring data from InsL to ROS (the same data cannot be found at InsL and ROS), and to handle uncommitted transactions (only insert/update/delete of committed transactions are reflected to ROS).

 ISBN: 978-1-61208-408-4

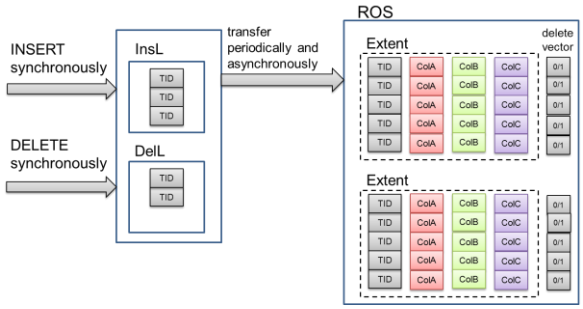


Figure 1. InsL/DelL and ROS Configuration

B. *Query Execution*

When the Query Optimizer chooses to execute a query/subquery using the columnar data instead of the traditional row data, the execution is handled by our new columnar data engine. For each query, the necessary portion of data in InsL is temporarily converted to a columnar data format (called Local ROS) and merged with the ROS data for processing, as shown in Fig. 2.

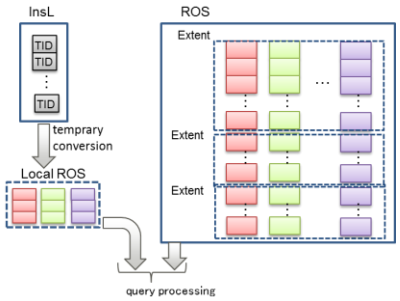


Figure 2. Combining Local ROS and ROS for Query Processing

The extents are processed in parallel by PostgreSQL's Dynamic Background Workers. For an efficient parallel processing, a mechanism where data containing pointers can be shared among the multiple processes is necessary. However, using PostgreSQL's Dynamic Shared Memory, the mapping location is different between the processes and thus it doesn't allow the straightforward sharing of pointers. Aiming at solving this problem, we designed a new shared memory mechanism called Shared Memory Context (SMC). SMC interface is compatible with PostgreSQL's memory context interface. Therefore, pre-existing PostgreSQL's routines can be called, and memory that is newly allocated within those routines is mapped to SMC space. As illustrated in Fig. 3, using SMC, the Backend Process and the Dynamic Background Workers can share the necessary data for an efficient processing of queries in parallel.

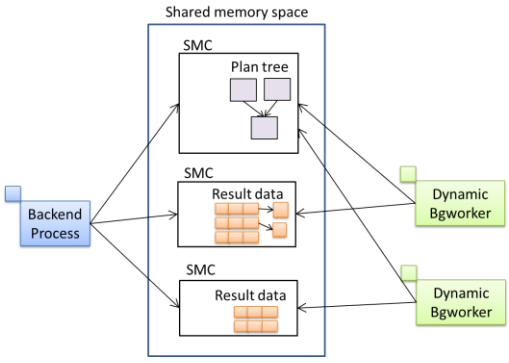


Figure 3. Parallel Processing using SMC

III. Performance Evaluation

Although we are at a preliminary stage of evaluation using the DBT-3 benchmark [11], the results are promising. For instance, for query 1 of DBT-3, a speed-up ratio of 50 was achieved, when using the column store index against PostgreSQL's original row data, on the same server (a 2-CPU machine with 16 cores).

IV. Conclusion and Future Work

In this paper, we have briefly presented our work in progress on extending the PostgreSQL OSS database system with a column store index to handle OLXP workloads. We have introduced new mechanisms to efficiently synchronize the inserts/updates/deletes of row data with the column store indexes, and to efficiently process them in parallel using a new shared memory mechanism that is fully compatible with PostgreSQL's memory context interface.

We are now planning to evaluate the extensions we have introduced to PostgreSQL by using the CH-benCHmark [12] and some real applications.

References

[1] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", Proc. VLDB, 1997, pp.65-74.

[2] A. Kemper and T. Neumann, "Hyper: A Hybrid OLTP&OLAP Main Memory Database System Based on Virtual Memory Snapshots", Proc. IEEE ICDE, 2011, pp.195-206.

[3] H. Plattner, "The Impact of Columnar In-MemoryDatabases on Enterprise Systems", Proc. VLDB, 2014, pp.1722-1729.

[4] V. Sikka, F. Färber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhövd, "Efficient Transaction Processing in SAP HANA Database: The End of a Column Store Myth", Proc. ACM SIGMOD, 2012, pp.731-741.

[5] "Oracle Database In-Memory", Oracle White Paper, October 2014, Available from: http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html [retrieved: March, 2015].

[6] P. Larson, et al., "Enhancements to SQL Server Column Stores", Proc. ACM SIGMOD, 2013, pp.1159-1168.

[7] "PostgreSQL", Available from: http://www.postgresql.org/ [retrieved: March, 2015].

[8] K. Knizhnik, "In-Memory Columnar Store extension for PostgreSQL", Available from: http://www.pgcon.org/2014/schedule/events/643.en.html [retrieved: March, 2015].

[9] "PostgreSQL Columnar Store for Analytics Workloads", Citusdata, Available from: http://www.citusdata.com/blog/76-postgresql-columnar-store-for-analytics.

[10] Mike Stonebraker, et al., "C-store: a column-oriented DBMS", Proc. VLDB, 2005, pp.553-564.

[11] Database Test Suite, Available from: http://sourceforge.net/projects/osdldbt/files/dbt3 [retrieved: March, 2015]..

[12] R. Cole, et al., "The mixed workload CH-bencCHmark", Proc. DBTest, 2011, article no. 8.

 ISBN: 978-1-61208-408-4

A Proposed Architecture to Support the Processing and Analysis of Structured and Unstructured Massive Data Sets in the Brazilian Army

Marcio de Carvalho Victorino
Computer Science Department, University of Brasília
Brasília, Brazil
e-mail: mcvictorino@cic.unb.br

Danilo Rodrigues Azeredo Silva
Computer Science Department, University of Brasília
Brasília, Brazil
e-mail: danilo@draconsultoria.com

Marçal de Lima Hokama
Science and Technology Department, Brazilian Army
Brasília, Brazil
e-mail: lima@cds.eb.mil.br

***Abstract*— In recent years, the demand for processing a great volume of data of different formats has increased considerably. In order to address such a demand, a number of new data models have been proposed, which are different from relational models. The databases that implement such models are commonly known as Non-Structured Query Language (NoSQL). NoSQL databases have become excellent persistence devices for the Big Data environment because they provide structural flexibility, horizontal scalability, unstructured data support and distributed processing. However, these databases are not suitable to support ad-hoc analysis, very common to the conventional Online Analytical Processing (OLAP) architecture. On the other hand, the OLAP architecture has a multidimensional data repository, called Data Warehouse (DW), not able to support the new data storage demand. This work presents a preliminary study within the Brazilian Army to establish a new architecture that integrates the OLAP and Big Data environments to provide structured and unstructured data usage for decision support.**

Keywords-NoSQL; OLAP; Big Data.

I. INTRODUCTION

The Relational Database Management System (RDBMS) has become the most widely used Database Management System (DBMS) since the 1980s, due to its simplicity of representation and the data independence it provides.

However, the significant increase in the quantity and complexity of services offered on the web in the last decade has generated massive volumes of data in varying formats, demanding greater processing distribution and high flexibility of these data storage structures.

To support these new demands, a new database generation without rigid schemes arose, based on non-relational models, called NoSQL databases.

Even though the term "NoSQL" was used for the first time in 1990 by Carlos Strozzi [1], it was only in 2011 that this term began representing the database family that may be categorized in the following models [2][3]: graphs, documents, column family and key-value pair.

Based on these application domains, in which there are massive volumes of data, in a high variety of formats, which require processing in an appropriated velocity (3V's), the term "Big Data" was coined [4].

In 2012, a project that accounts for this context of the 3V's (volume, variety and velocity) was implemented in the Brazilian Army – the Integrated System of Border Monitoring (in Portuguese, Sistema Integrado de Monitoramento de Fronteiras, SISFRON) [19]. SISFRON uses sensoring, communication and information technology resources to cover 16.886 Kilometers of the Brazilian border, monitoring nearly 27% of the Brazilian national territory, and generating a huge data volume in various formats. These data will become an important source for decision support. Its cost is estimated at approximately US$ 4 billion (R$ 12 billion) to be invested over a 10 year period.

Notably, from the great volume and variety of data analysis, this project may also create opportunities that have a positive impact not only for the Brazilian Army, but also for the greater society. However, in this initiative, it is important that the organization chooses appropriate technologies to build its Big Data ecosystems [5].

The NoSQL DBMSs are commonly used in Big Data applications. However, these databases are not appropriate for supporting ad-hoc analysis, very common in Decision Support Systems (DSS) [6].

On the other hand, the OLAP architecture, proposed in the 90s [7], provides broad support for ad-hoc analysis; however, it cannot support all the demands of a Big Data environment [8].

Subsequently, with regard to decision support, the Brazilian Army has already deployed a system that uses OLAP architecture. However, it is clear that there is a need to extend this architecture to provide the processing of new data sources, from new systems such as SISFRON, in differing formats.

In this paper, we present the analysis of possible architectures to be adopted by the Brazilian Army, that integrate the OLAP and Big Data characteristics. The aim is to provide the support for the processing and analysis of

 ISBN: 978-1-61208-408-4

massive structured and unstructured data in the Brazilian Army.

The paper is organized as follows: Section II presents the concept of Big Data. Section III presents the Decision Support Environment in the Brazilian Army. Section IV presents proposals of architecture for Big Data Analysis environments. Finally, conclusions are included in Section V.

II. BIG DATA

There are various definitions and understandings for the term "Big Data". One of the most widely accepted is the '3Vs' definition presented by Doug Laney [9] in 2001 and ratified by Gartner [10] in 2012: "Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."

For tackling these new challenges, a new generation of technologies has been proposed, such as Hadoop [11]. Created by Doug Cutting, Hadoop was inspired by BigTable, which is Google's data storage system; by the Google File System; and by MapReduce [11]. Hadoop is a framework based on Java and a heterogeneous platform with open source license. This framework includes a distributed file system; a data storage platform; and a management parallel computation layer, a work flow and configuration administration. The Hadoop Distributed File System (HDFS) is executed in all the nodes of a Hadoop cluster and connects the file systems in many input and output data nodes, to turn them into a unique large file system.

MapReduce [12], presented by Google [13], has a processing approach that divides the Big Data complex problems into small work units and processes them in parallel.

The term "Hadoop" is also used to designate a family of related projects that are under the infrastructure of distributed computation and data processing in large scale, such as Common, Avro, MapReduce, HDFS, Pig, Hive, HBase, ZooKeeper, Sqoop, among others [11].

III. DECISION SUPPORT IN BRAZILIAN ARMY

The Brazilian Army has many computerized systems in various administrative and operational areas, supporting their respective processes.

Such systems generate and store a large quantity of data related to the processes. However, these data are not related to each other. The data, produced and stored by the Brazilian Army, are assets to those sectors that produce them, but this raw data cannot be used as a strategic resource because it is not integrated.

Aiming to minimize this problem, in 2008, the Brazilian Army began the DSS initiative, called Management Integrated System (MIS). This system is organized according to the OLAP architecture proposed by Kimball [7], comprised of basically four layers: the layer created by relational data sources; the extraction, transformation and load (ETL) layer; the storage layer; and the presentation layer.

In this architecture, the data that come from the source systems go through the ETL processing to be loaded in multidimensional repositories called, Data Warehouse (DW). The goal is to integrate the data that come from various sources and transform them into information.

The MIS is comprised of one Data Mart (DM) in each activity area, initiated by: the Human Resources in 2009, Military Service in 2010, Logistics in 2011, Personal Assessment in 2012 and Finances in 2013. The group formed by these five DM was consolidated in a Brazilian Army DW. However, with new data sources from SISFRON this conventional OLAP architecture has become unable to support the huge data volume and unstructured data from this system.

The SISFRON systemic architecture is composed of the following subsystems: sensoring, decision support, communication and information technology, information security, simulation and logistic. The sensoring subsystem is composed of terrestrial and aerial surveillance radars, weather stations and radars, and electromagnetic and optical sensors. Its main function is to obtain data to provide surveillance and event detection. The decision support subsystem has the aim of treating the data collected by the sensors for following visualization.

For supporting SISFRON data source, the next step of the MIS project will consist in providing unstructured data processing and analysis in the Brazilian Army. However, Kimball and Ross [8] claim that the DW based on RDBMS is not able to store the large data variety available in big organizations, making it necessary to create a Big Data environment.

IV. ARCHITECTURE PROPOSAL

The NoSQL databases have become excellent persistence devices in Big Data environments, for providing structural flexibility, high horizontal scalability, unstructured data support and distributed processing [3].

These databases were created to support operational processes that manipulate a large volume of data in various formats in an appropriate time period, but not to provide more elaborated analysis for decision support [3].

In this context, Big Data Analysis is understood as a procedure set executed on a large scale, on large data repositories, in which the main goal is knowledge extraction [14].

Another important aspect to be observed in data analysis in critical environments is the fact that there are few query tools able to access data repositories available to Hadoop, which is the main framework used in Big Data environments [3]. Examples of these tools are Hive and Pig. Besides the small number of tools, the few that exist require deep programming knowledge to generate analytical queries about the NoSQL databases or other databases.

The OLAP report tools, on the other hand, provide complete support to ad-hoc analysis in which the final user may access the dimensionally organized data in a DW to generate the sheets or graphics without creating any code line.

 ISBN: 978-1-61208-408-4

There are many issues when trying to enable the use of OLAP architecture with DW to support decision-making in a Big Data environment. The main limiting issues for the joint use of these two technologies are the high volume and data complexity available in Big Data environments [15].

The great challenge is to provide architecture that makes it possible to use the structured and unstructured data together in Big Data environments to support decision-making processes.

Davenport [16] presents a proposal to merge the two technologies OLAP and Big Data in an integrated environment. Figure 1 presents conventional OLAP architecture [7]. In the figure, we see that the data sources, mainly those that are structured, are submitted to an ETL process to be stored in DW and, after this, presented to the final users by the OLAP report tools.

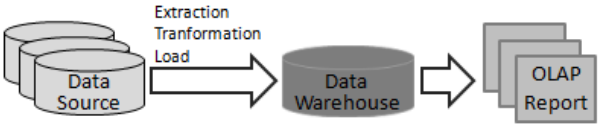


Figure 1. Conventional OLAP architecture [7].

Figure 2 presents an extension of this conventional architecture according to Davenport [16] to provide its use in Big Data environments.

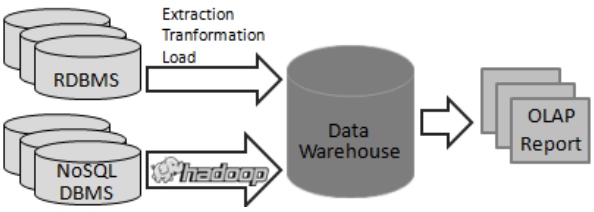


Figure 2. Extended OLAP architecture [16].

In the architecture presented in Figure 2, the data sources are organized in various formats such as relational tables, graphs, documents, column family and key-value pair. In this architecture, the Hadoop framework performs two roles. It may work as a data presentation tool for the operational queries, and may perform the ETL tool role for the stored data in NoSQL databases, since it transports the data from their sources to the DW and executes the transformation process. In this architecture, all the analytic data are stored in a DW verifying an integrated analysis from the information crossing from many areas, generating the result through OLAP reports.

Thus, the various DBMS that compose the SISFRON, such as MySQL and Postgres (relational), Riak (key-value), MongoDB (documents), Cassandra (column family) and Neo4J (graphs) will be used as resources to support decisions.

However, this architecture is limited, since the current OLAP environment is not able to support the high data volume and variety in a Big Data environment [8].

Gartner [17] presents some proposals for architectures in Big Data Analytics environments. Figure 3 presents an adaptation of one of these proposals.

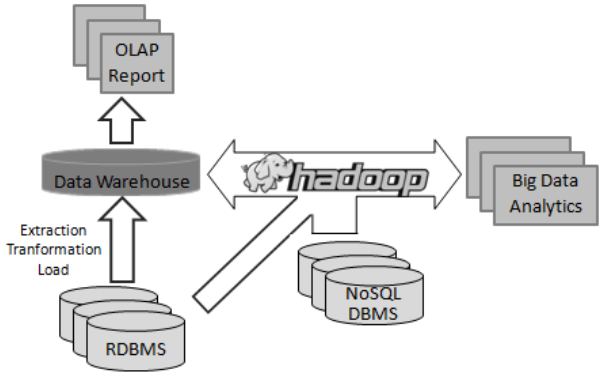


Figure 3. Gartner [17] adapated Big Data Analytics Architecture.

The architecture presented in Figure 3 is broader than the one presented in Figure 2. It provides the generation of analytical reports from the DW through the traditional OLAP tools, or directly from the data repositories that comprise the Big Data (RDBMS and NoSQL DBMS) through the analytical query tools that are part of the Hadoop framework.

In this architecture, the DW stores the data from the RDBMS and part of the data that comes from NoSQL DBMS extracted through the Hadoop, since the DW has storage limitations. On the other hand, as the Hadoop doesn't have limitations related to distributed data source access, it becomes possible to analyze high data volume and variety compared to the OLAP architecture. However, it is worthy to note that the analysis tools available for the Hadoop are not as mature as the OLAP tools.

Hence, as the Brazilian Army already has the OLAP architecture presented in Figure 1, after a thorough study of many architectural options presented in [7] [16] [17], it was concluded that the Brazilian Army will focus efforts, initially, on the architecture presented in Figure 2. However, due to the fact that this architecture has limitations, it is intended, in a second phase, to migrate to a broader architecture presented in Figure 3, bearing in mind that the architecture presented in Figure 3 is an extension of the one presented in Figure 2. No work will be wasted during the transition.

Due to the fact that the architecture initially slated for adoption by the Brazilian Army cannot support all the volume and variety of the existing data in the institution, there is a clear need to consider the following aspects as subsidies to the process of data extraction of the Big Data, stored in NoSQL DBMS to load in DW:

A. Selectivity

Since the DW of the OLAP architecture is not able to store all the data volume of the Big Data, it is necessary to prioritize the activity areas of the observed domain. Subsequently, the data extraction of the Big Data for the greater load in the DW must start with the data of the most important areas.

 ISBN: 978-1-61208-408-4

B. Summarization

Inmon [18] advises storing data in the DW, or, more precisely, in the fact table – the fact in the lowest level of granularity. However, this procedure is not viable in the proposed architecture presented in Figure 2, because, as already mentioned, the DW is not able to store all the data of the Big Data environment. Hence, it is necessary to summarize the data to create important aggregates for the observed domain, decreasing the space needed for data storage in DW. The main drawback of this line of action is to limit the drill down operation, since the level of detailing of the fact will depend on the lowest level of granularity of the aggregates generated and stored in the DW. Even so, this procedure is necessary.

C. Iterative ETL

The iterative ETL refers to the act of executing the ETL processes in an iterative manner, i.e., the employing of one finite sequence of ETL processing, in which the purpose of each one of the steps is to add data to the result of the previous step. Hence, the step sequence will be limited to the DW storage capacity.

Right after the consolidation of the architecture presented in Figure 2, the efforts for the implementation of the architecture presented in Figure 3 will be initiated. In this step, the critical issues for this environment quoted by Cuzzocrea et al. [15] will be considered, among them are: the data volume and complexity; new project methodologies; hardware; query performances; usability and data quality.

A thorough study about Big Data Analytics tools available for the Hadoop framework will carried out. As soon as the tools are mature enough, the architecture of Figure 3 will substitute the one presented in the Figure 2.

V. CONCLUSION

The growing sophistication of the available applications on the web has generated massive volumes of data in various formats that require processing at an appropriate velocity. This environment is called “Big Data”.

The RDBMS have become unable to attend to all of the Big Data environment demands. In this context, users began employing NoSQL DBMS as data persistence devices.

It has become a great challenge to apply advanced analytical techniques to Big Data data sets, Big Data Analytics, since the main framework for Big Data environments, the Hadoop, does not have mature ad-hoc query analytical tools.

On the other hand, OLAP architecture has many mature ad-hoc query analytical tools, but its repository, the DW, is not able to store all the data from Big Data.

Thus, we have presented an initial Brazilian Army study to adopt an architecture capable of providing data extraction from NoSQL data sources for further load in a DW, providing the availability of a Big Data Analytics environment with OLAP architecture in this institution.

From our research results and discussion here, it is possible to conclude that the Brazilian Army plans initially to extend its OLAP architecture to deal with unstructured data for subsequently migrating to a broader architecture.

REFERENCES

[1] NoSQL: a non-SQL RDBMS. [retrieved: 2014, 11]. [Online]. Available: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page

[2] P. J. Sadalage and M. Fowler, “NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence”, New Jersey, IN: Pearson Education, Inc., 2012.

[3] M. Indrawan-Santiago. “Database Research: Are We at a Crossroad? Reflection on NoSQL’. Fifteenth International Conference on Network-Based Information Systems, 2012, pp. 45–51.

[4] S. Singh and N. Singh, "Big Data Analytics", 2012 International Conference on Communication, Information & Computing Technology Mumbai India, IEEE, October 2011.

[5] N. Heudecker and H. Lehong. Applying the Big Data Ecosystem. [retrieved: 2014, 07]. [Online]. Available: http://www.gartner.com/document/code/252014

[6] M. Indrawan-Santiago. “Database Research: Are We at a Crossroad? Reflection on NoSQL’. Fifteenth International Conference on Network-Based Information Systems, 2012, pp. 45–5.

[7] R. Kimball, L. Reeves, M. Ross and W. Thornthwaite. “The Date Warehouse Lifecycle Toolkit”, New York, IN: John Wiley & Sons, 1998.

[8] R. Kimball and M. Ross, “The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling”, 3nd ed., Indiana, IN: John Wiley & Sons, 2013.

[9] D. Laney, "Application Delivery Strategies. Meta Group, Retrieved." [retrieved: 2014, 10]. [Online]. Available: http://blogs.gartner.com/doug- laney/files/2012/01/ad949-3D-Data-Management-Controlling- Data-Volume-Velocity-and-Variety.pdf

[10] M. A. Beyer and D. Laney, “The Importance of ‘Big Data’: A Definition” [retrieved: 2014, 08]. [Online] Available: https://www.gartner.com/doc/2057415/importance-big-data-definition

[11] T. White, "Hadoop: The Definitive Guide", 3nd. ed, IN: O'Reilly Media, 2012.

[12] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processingon Large Clusters” Google, Inc OSDI 2004.

[13] S. Sagiroglu and D. Sinanc, “Big Data: A review, Collaboration Technologies and Systems (CTS)”, 2013 International Conference, 2013, pp. 42 - 47, IEEE Conference Publications.

[14] A. Cuzzocrea, “Analytics over Big Data: Exploring the Convergence of DataWarehousing, OLAP and Data-Intensive Cloud Infrastructures”, Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, 2013, pp. 481 - 483, IEEE Conference Publications.

[15] A. Cuzzocrea, L. Bellatreche, and Il-Yeol Song, “Data warehousing and OLAP over Big Data: current challenges and future research directions”, In Proceedings of the sixteenth international workshop on Data warehousing and OLAP (DOLAP '13). ACM, New York, NY, USA, 2013, pp. 67-70.

[16] T. H. Davenport, “Big Data at Work: Dispelling the Myths”, Harvard Business Press, 2014.

[17] S. Sicular, "Hadoop Architecture: Multiple Choices — Which One Is Right?". Lecture Notes in Gartner Catalyst Conference, Brasília, Brazil, 2014.

[18] W. H. Inmon, “Building the Data Warehouse”, 4nd ed., Indiana, IN: Wiley Publishing, Inc., 2005.

[19] Brazilian Army, "O Sistema Integrado de Monitoramento de Fronteiras (SISFRON)" [retrieved: 2014, 05]. [Online]. Available: http://www.ccomgex.eb.mil.br/index.php/pt_br/sisfron/arquitetura

Analysis of String Comparison Methods During De-Duplication Process

Maria del Pilar Angeles, Francisco Javier García-Ugalde,
Ricardo Valencia, Arturo Nava
Facultad de Ingeniería
Universidad Nacional Autónoma de México
México, D.F.
Email: pilarang@unam.mx,fgarciau@unam.mx,
ricardofdk8@hotmail.com, arturoshox@hotmail.com

***Abstract*—This paper presents three comparison algorithms in terms of computational resources utilized during record linkage process. The comparison algorithms are Monge-Elkan, Bag Distance and Edit Distance. The Monge-Elkan method meets all the requirements to be implemented and to obtain reliable results characteristics. Besides, the method falls within the average execution time efficiency.**

Keywords–data matching; de-duplication; record linkage.

I. INTRODUCTION

During the integration process of a number of heterogeneous databases, the identification and resolution of extensional inconsistencies is one of the main problems to deal with [1]. The data matching process, is focused on joining records from disparated data sources describing the same real world entity. This process requires data standardization, indexing of records for reducing the number of records comparison, field and record comparison, the identification of matched records, not matched records and possible matched records, and finally the evaluation of classification of records. Therefore, the data matching process grows exponentially as the databases to be matched get larger. In real-world data matching applications, the true status of two records that are matched across two databases is not known. Thus, accurately assessing data matching quality and completeness is challenging [2].

We focused on the integration of the Freely Extensible Biomedical Record Linkage prototype (FEBRL) system [3] to any database from any Database Management System (DBMS) by querying the native data dictionary; the research proposal is aimed to the enhancement and addition of further standardization, indexing, and classification algorithms for data matching. We have called our prototype Universal Evaluation System Data Quality (SEUCAD), which nowadays supports six DBMS. The present work is related to the open issues on the comparison of algorithms that reduce the quadratic complexity of the naive process of pair-wise comparing each record from one database with all records in the other database.

The present paper is organized as follows: The next section briefly explains the data matching process. Sections III, IV and V explain the string comparison functions Monge-Elkan, Bag-Distance and Levenshtein distance respectively, along with their role within de process of data matching. Section VI presents the experiments carried out. Section VII analyses the results. Finally, the last section concludes the main topics achieved regarding the performance of the comparison functions and the future work to be done.

II. RELATED WORK

The data matching process is mainly concerned with the record comparison among databases in order to determine if a pair of records corresponds to the same entity or not. This process, in general terms, consists of the following tasks:

- A standardization process [4], which refers to the conversion of input data from multiple databases into a format that allows correct and efficient record correspondence between two data sources.
- The indexing process aims to reduce those pairs of records that are unlikely to correspond to the same real world entity and retaining those records that probably would correspond in the same block for comparison; consequently, reducing the number of record comparisons. The record similarity depends on their data types because they can be phonetically, numerically or textually similar. Some of the methods implemented within FEBRL are for instance, Soundex [5], Phonex [2], Phonix [2], the New York State Identification and Intelligence System Phonetic Code (NYSIIS) [6], Double metaphone [7], QGrams.
- Field and record comparison methods provide degrees of similarity and define thresholds depending on their semantics or data types. In the prototype, the algorithms Qgram, Jaro - Winkler Distance [8] [9], Longest common substring comparison are already implemented.
- The classification of pairs of records grouped and compared during previous steps is mainly based on the similarity values were already obtained, since it is assumed that the more similar two records are, there is more probability that these records belong to the same entity of the real world. The records are classified into matches, not matches or possible matches, the classification of records can be unsupervised or supervised.

The unsupervised process classifies pairs or groups of records based on the similarities between them without having access to more information about the characteristics of those records. The supervised process requires training based on data identified as similar or not similar. In this case, comparison vectors with an associated value that determines whether records correspond or not are required. In the case of potentially corresponding records, the duplicates detection may be performed manually. Within FEBRL, there are methods based

 ISBN: 978-1-61208-408-4

on thresholds, probabilistic methods, costs based methods or rule-based methods; The evaluation of data matching refers to how many of the classified matches correspond to true real-world entities, and how many of the real-world entities that appear in both databases were correctly matched [10]. The present research assessed the algorithms in terms of computational resources which are detailed in Section VIII.

III. THE MONGE-ELKAN ALGORITHM

Monge and Elkan proposed in [11] a simple but effective method for measuring the similarity between two strings containing multiple tokens, using an internal similarity sim (a, b) capable of measuring the similarity between two individual tokens a and b. Given two texts A, B being their respective number of tokens $|A|$ and $|B|$, the Monge-Elkan algorithm measures the average of the similarity values between pairs of more similar tokens within texts A and B. The Monge-Elkan similarity formula is as follows:

$$simMongeElkan(A,B) = \frac{1}{|A|}\sum_{i=1}^{|A|} max\{sim'(A_i,B_j)\}_{i=1}^{|B|} \quad (1)$$

In order to compare two strings, the Monge Elkan algorithm requires a similarity function; we will utilize the Jaro similarity function, which is detailed as follows: The Jaro similarity function was developed by Matthew Jaro in [8], this function was designed specifically for comparing short length strings, such as names, and is given by the following formula:

$$simjaro(s1,s2) = \frac{1}{3}(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}) \quad (2)$$

The Jaro similarity function counts the number of characters that match, where c is the number of coincident characters and t is half the number of transpositions (two adyacent characters that are interchanged en both strings, such as 'pe' and 'ep'). For instance, considering two strings S1 = 'mario alfonso' and S2 = 'Marian alonso'. Applying the Jaro similarity function, the results are as follows: Jaro('alfonso','Marian') = 0.6190; Jaro('alfonso','Alonso') = 0.9523; Jaro('mario', 'Marian') = 0.9047; Jaro('mario','Alonso') = 0.5777. Given the similarity between tokens, the two best matches are selected for computing the Monge-Elkan comparison. Considering the Monge-Elkan formula given in (1), $|A| = 2$ as the number of tokens s1, $|B| = 2$ as the number of tokens of s2, and the Jaro similarity function already calculated Sim(0.9523,0.9047). Therefore, the Monge-Elkan comparison is been given by:

$$monge - elkan(s1,s2) = 1/2(0.95 + 0.90) = 0.928 \quad (3)$$

IV. THE BAG DISTANCE ALGORITHM

This method obtains the non-common characters between two strings s1 and s2. The characters of s1 and s2 are divided and ordered to obtain two charsets X and Y, and theri corresponding differences X-Y and Y-X. The largest difference between s1 as s2 is computed by:

$$bagdistance(s1,s2) = max(|x-y|,|y-x|) \quad (4)$$

The bag distance similarity function is given by the following formula:

$$simbag(s1,s2) = 1 - (\frac{bagdistance(S1,S2)}{max(|S1|,|S2|)}) \quad (5)$$

For instance, let be s1 = maria; s2 = mariano, with $|s1| = 5$, $|s2| = 7$, then X = a,a,i,m,r and Y = a,a,i,m,n,o,r. Therefore, $|x-y| = |\{\varnothing\}| = 0$, $|y-x| = |\{n,o\}| = 2$. The difference is identified as $bagdistance(s1,s2) = 1.0 - (\frac{2}{7}) = 0.71$

V. LEVENSHTEIN DISTANCE ALGORITHM

The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. In order to compare two strings s1 and s2 with lengths $|s1|$ and $|s2|$ requires a matrix of length $|s1|+1, |s2+1|$. The matrix is filled according to the number of deletion, insertion and substitution operations required. Consequently, the levenstain distance between s1 and s2 is obtained as the value of the lower left cell. The Levenshtein similarity is computed by the following formula:

$$levensthein(s1,s2) = 1 - (\frac{levenshteindist(S1,S2)}{max(|S1|,|S2|)}) \quad (6)$$

where the numerator is the value of the lower right corner and the denominator the greater length between strings s1 and s2. For instance, let be strings s1 = raul and s2 = ramon. The intersection of (ra-r) has a value of 1 because it is required to perform one operation to reach the same state. In the case of different characters, a value of 1 indicates, since it is a replacement operation. An example of the Levensteing Matrix when comparing the strings "ramon" and "raul" are shown in Table I In our case, the formula is replaced in the

TABLE I. LEVENSTEING MATRIX

		r	a	m	o	n
	r	0	1	2	3	4
	a	1	0	1	2	3
	u	2	1	1	2	3
	l	3	2	2	2	3

following manner: $levensthein(s1,s2) = 1 - (\frac{3}{5} = 0.4)$ The Comparative analysis of the Monge-Elkan, Bag Distance and Edit distance methods is presented in the following section.

VI. ANALYSIS OF THE MONGE-ELKAN METHOD WITH BAG-DISTANCE AND EDIT-DISTANCE

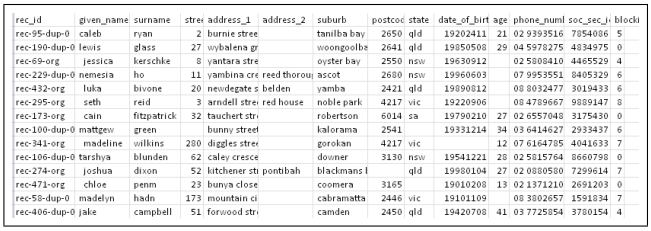
This section shows a comparative analysis of the Monge-Elkan, Bag-Distance and Edit Distance methods in terms of pairs records comparison time, memory used, etc. We have tested the three already mentioned methods with a number of input files generated by the SEUCAD prototype, one of them will be presented as follows:

A. Exploring the file

The de-duplication process initiates with the exploration of the input file. The file called data1.csv was generated with a total length of 1000 records, where 500 records were original and 500 duplicated records.

 ISBN: 978-1-61208-408-4

From statistics we can analyse the file in terms of the number of unique data values, most frequent field values, minimun and maximum value lenghts. Such information allow us to identify which fields are suitable for indexing or the method to apply that corresponds to the data type and length the data source contains. The structure of the data file is presented in Figure 1.



rec_id	given_name	surname	stree	address_1	address_2	suburb	postcod	state	date_of_birt	age	phone_numl	soc_sec_i	blocki
rec-95-dup-0	caleb	ryan	2	burnie stree		tanilba bay	2650	qld	19202411	21	02 9393516	7854086	5
rec-190-dup-0	lewis	glass	27	wybalena gr		woongoolba	2641	qld	19850508	29	04 5978275	4834975	0
rec-69-org	jessica	kerschke	8	yantara stre		oyster bay	2550	nsw	19630912		02 5808410	4465529	4
rec-229-dup-0	nemesia	ho	11	yambina cre	reed thorou	ascot	2680	nsw	19960603		07 9953551	8405329	6
rec-432-org	luka	bivone	20	newdegate s	belden	yamba	2421	qld	19890812		08 8032477	3019433	6
rec-295-org	seth	reid	3	arndell stre	red house	noble park	4217	vic	19220906		08 4789667	9889147	8
rec-173-org	cain	fitzpatrick	32	tauchert str		robertson	6014	sa	19790210	27	02 6557048	3175430	0
rec-100-dup-0	mattgew	green		bunny street		kalorama	2541		19331214	34	03 6414627	2933437	6
rec-341-org	madeline	wilkins	280	diggles stre		gorokan	4217	vic		12	07 6164785	4041633	7
rec-106-dup-0	tarshya	blunden	62	caley cresce		downer	3130	nsw	19541221	28	02 5815764	8660798	0
rec-274-org	joshua	dixon	52	kitchener sti	pontibah	blackmans l		qld	19980104	27	02 0880580	7299614	7
rec-471-org	chloe	penm	23	bunya close		coomera	3165		19010208	13	02 1371210	2691203	0
rec-58-dup-0	madelyn	hadn	173	mountain ci		cabramatta	2446	vic	19101109		08 3002657	1591834	7
rec-406-dup-0	jake	campbell	51	forwood str		camden	2450	qld	19420708	41	03 7725854	3780154	4

Figure 1. File structure

B. Selecting the indexing method

- Indexing method: The QGramIndex method was selected, with a length of parameter Q of 2, to generate bigrams.
- Threshold: This parameter value should be set in the range of 0.0 and 1.0, if the value is 1.0 only records that have the same definition of indexing will be compared. Therefore, in order to establish a more accurate indexing and comparison of records, the threshold value we have defined for this experiment was 0.75.
- Padded: As this parameter sets whether the input strings are set to (q-1) grams of characters or not. It was enabled for a better analysis and more accurate the indexing.
- Blocking key: We have chosen the field ”surname”. We did not set a maximum number of characters for the definition of indexing, otherwise large values will be truncated. As the fields to be compared contain more than one word.
- Sort words: This option was not enable in order to avoid the division and ordering of each word.
- Reverse: The reverse parameter was disabled because otherwise the input string will be reversed and in the case of surname field would not be a representative indexing definition.
- Encoding function: The encoding function selected was ”Soundex”. As we required the whole word to be encoded, this value was not set.

The previous configuration for indexing is presented in Figure 2.

C. Selecting the comparison method

The following parameters correspond to the comparison method. As we focused on the execution of three comparison methods, the parameters are set equally in the three of them.

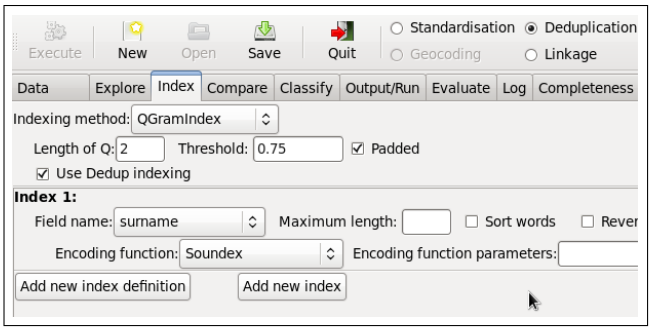


Figure 2. Indexing configuration

- Field name A: The name of the first field of comparison. Since the given name is a relevant field to identify people, and the comparison field is recommended to be of datatype String, the field ”given_name” was selected as the first option for comparison purposes.
- Field name B: The name of the second field of comparison. Once again the field ”given_name” was selected in order to compare those records according to the given name.
- Cache comparisons: Indicate whether the calculation of the similarity of values can not take place on memory. It is recommended when data values are large, complex, or there is limited number of fields. As ”given_name” field is not complex nor large, the option of ”Cache comparisons” will be disabled. Thus the calculations will not be performed in memory.
- Maximum cache size: This value is limited to a certain number of pairs of fields. If not selected (None), then all the comparisons will be made. Since it is desired that the comparisons of all pairs of fields are made, the option to ”Maximum cache size” will default to ”None”.
- Missing value weight: The value to be given in the event that one or both fields have no value. Its default value is 0.0 and its value must be within the range of ”value Disagreeing weight” and ”weight Agreeing value.” For comparison operations are more accurate when one or both fields have no value, the value of ”Missing value weight” will be ”0.0”.
- Agreeing value weight: The value to be given when the similarity is entirely accurate. By default the value is 1.0.
- Disagreeing value weight: The value to be given when the similarity is entirely different. By default the value is 0.0. This value must be less than ”Agreeing value weight”. Like the previous value, the value of ”Disagreeing value weight” as ”0.0” will be defined when the similarity of two strings is totally different.
- Threshold: This value should be set in the range of 0.0 and 1.0, and will determine a better level of accuracy. If the calculation of approximate similarity method is higher than indicated in this field (threshold), then the similarity value will be calculated. If the approximate

 ISBN: 978-1-61208-408-4

similarity is less than that indicated in this field (threshold), then the similarity value will correspond to the ”Disagreeing value weight” parameter.
The selected attributes per comparison method are shown in Table II.

TABLE II. CONFIGURATION MATRIX

Test	*Process*	*Method*
1	Comparison	Monge-Elkan (given_name)
2	Comparison	Bag-Distance (given_name)
3	Comparison	Edit-distance (given_name)
	Coding	soundex(surname)

D. Selecting the method of classification

This section is aimed to present the configuration parameters established for the execution of the classification method.

- Weight vector classification method: The Fellegi and Sunter method will be selected since it has been broadly used and tested.
- Lower threshold: The lowest threshold value to be considered for the classification of records. Since only the comparison of ”given_name” field, place the sum of similarities remain in the range of 0.0 to 1.0. Thus, an acceptable and considered for the lower threshold value is ”0.5”.
- Upper threshold: The higher threshold value to be considered for the classification of records. Since only the comparison of ”given_name” field, place the sum of similarities remain in the range of 0.0 to 1.0. Thus, an acceptable and considered for the higher threshold value is ”0.98”.

Therefore the values of the minor similarities to 0.5 will be classified as ”Non match”, which are greater than 0.9 will be classified as ”Match” and remaining in the range of 0.5 and 0.9, will be classified as ”Potential Match”. Figure 3 shows the classification settings.

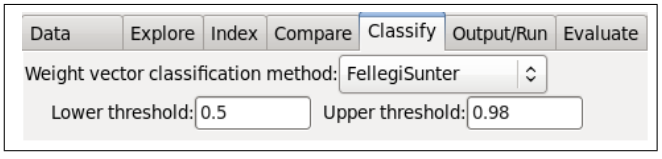


Figure 3. Configuration settings for classification of records.

E. Select the output characteristics and execution

The SEUCAD prototype is able to store the configuration selected for the de-duplication process in a python file. Figure 4 shows the output execution settings. The prototype allows the user to save the weight vector, histogram, match status and match data files.

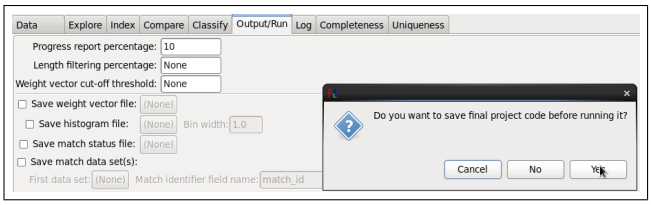


Figure 4. Configuration settings for execution of de-duplication process.

VII. RESULTS

The results of the data matching process are presented in this section. The test file named data1.csv contained 1000 total number of records, considering rec_id as the record identifier, given_name as the fields of comparison, surname as indexing field for Qgram as an indexing method. The number of pairs of records compared were of 13598 and Fellegi and Sunter as a classifier method with lower threshold of 0.5 and an upper threshold of 0.98. The use of memory and time resources per comparison method are shown in Table III.

TABLE III. TIME AND MEMORY UTILIZED PER METHOD

Comparison method	*Total time*	*avg time per pair*	*total memory*	*resident memory*
[1ex] Monge-Elkan	1.64s	0.12ms	11264KB	7.72MB
Bag-Distance	1.22s	0.09ms	11264KB	7.968MB
Edit-Distance	2.38s	0.18ms	11264KB	7.751MB

The quality metrics per method are shown in Table IV.

TABLE IV. QUALITY METRICS PER METHOD

Comparison method	Matches vectors	True positives	no matches	possible matches
Monge-Elkan	474	474	8036	5088
Bag-Distance	437	415	11518	1643
Edit-Distance	432	413	12875	291

A. Monge-Elkan

In the case of Monge-Elkan, the total time taken for the comparison process was 1.64 seconds with a comparison average time per pair of records of 0.12 milliseconds. The total memory usage was of 11264 Kbytes, 7.71875 MB of resident memory, 474 vectors classified as ”match”, and 474 true positives, 8036 vectors classified as ”no match” and 5088 vectors classified as ”possible-match”: 5088. The amount of weight partial acceptance value of 0.0 was of 13166 and the amount of weight partial acceptance of 1.0 were of 432. Figure 5 shows the outcome results considering Monge-Elkan as comparison method.

B. BagDistance

In the case of BagDistance, the total time taken for the comparison process was of 1.22 seconds with a comparison average time per pair of records of 0.09 milliseconds. The total memory usage was of 11264 KB and 7968 of resident memory, 437 vectors classified as ”match”, and 415 true positives, 11518 vectors classified as ”no match”, and 1643 vectors classified as ”possible-match”. The amount of weight

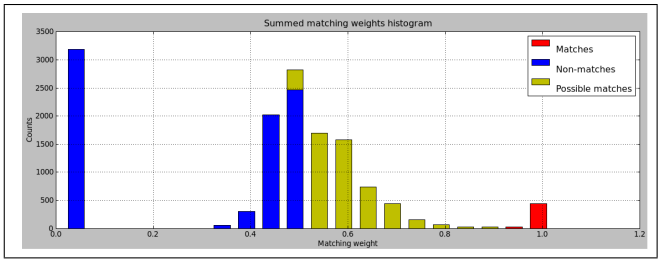


Figure 5. Classification of records with Monge-Elkan as comparison method.

partial acceptance value of 0.0 was of 11518 and the amount of weight partial acceptance of 1.0 were 437. Figure 6 shows the outcome results with Bag-Distance as comparison method.

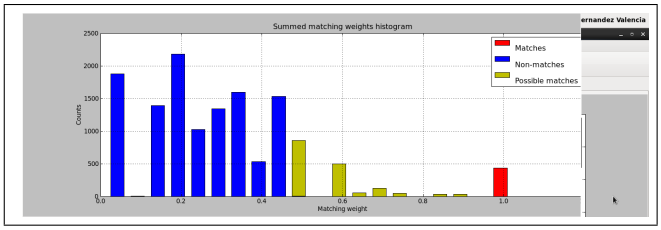


Figure 6. Classification of records with Bag-Distance as comparison method

C. Edit-Distance

In the case of Edit-Distance, the total time taken for the comparison process was of 2.38 seconds with a comparison average time per pair of records of 0.18 milliseconds. The total memory usage was of 11264 KB and 7.7578125 MB of resident memory resident, 432 vectors classified as ”match”, , and 413 true positives, 12875 vectors classified as ”no match”, and 291 vectors classified as ”possible-match”. The amount of weight partial acceptance value of 0.0 was of 13166 and the amount of weight partial acceptance of 1.0 were 432. Figure 2 shows the results. Figure 7 shows the outcome results with Edit-Distance as comparison method.

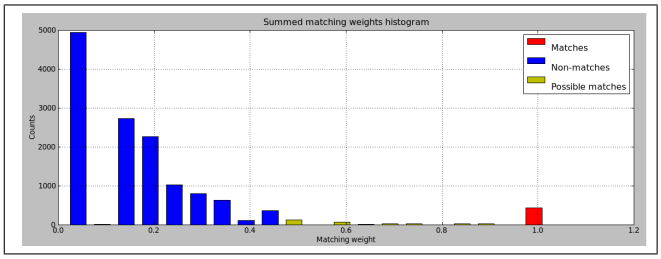


Figure 7. Classification of records with Edit-Distance as comparison method

According to the experiment results, we can make some conclusions about the resources used by Monge-Elkan, Edit Distance and Bag Distance methods. Considering the execution time, the Bag-Distance method is a comparison that performs operations quickly as is shown in Figure 8. Regarding the amount of memory used by the Monge-Elkan method, it has

Figure 8. Pair records comparison time (seconds)

been in the same range as in the rest of the comparison methods. There is a small variation when the number of records and comparisons increases, as the Monge-Elkan method is more efficient by using less memory, as is shown in Figure 9. After performing the relevant comparisons, the classification of

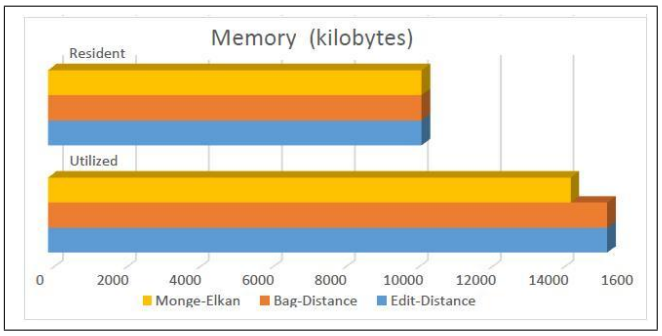


Figure 9. Used Memory and Resident Memory

data has been carried out. Thus, the given records classified as ”match” we can observe that comparisons of methods Bag-distance, Edit-Distance and Monge-Elkan result in similar classification. As for the records classified as ”non-match” and ”possible-match”, it can be concluded that there are a large number of variations in the results, as each algorithm comparison is based on different characteristics, such as: string length, number of similar tokens, etc. Finally, measuring the number of equivalent weights of 0.0 and 1.0 acceptance were practically the same, only small variations were 3 to 4 records. Thus, the level of certainty of the new method of comparison is very similar to those already implemented methods. The Monge-Elkan method meets all the requirements to be implemented and to obtain reliable results characteristics. Besides, the method falls within the average execution time efficiency.

VIII. Conclusion and Future Work

The Monge-Elkan method meets all the requirements to be implemented and to obtain reliable results characteristics. Besides, the method falls within the average execution time efficiency. However, there was no consideration of the quality

 ISBN: 978-1-61208-408-4

of classification, we have utilized synthetic test data sets from which we are able to obtain the number of candidate record pairs generated, or the measures of reduction ratio, pairs quality, and pairs completeness. The corresponding measurements allow to compute the effectiveness of the data matching system, which is a part of our future work.

As we have used synthetic test data sets to evaluate the comparison methods, then it is important to be aware of the limitations of such data, and the results achieved with them should not be generalised, because the performance of a method is dependent on the type and characteristics of the data that are matched.

ACKNOWLEDGMENT

This work is being supported by a grant from Research Projects and Technology Innovation Support Program (Programa de Apoyo a Proyectos de Investigacin e Innovacin Tecnolgica, PAPIIT, UNAM Project IN114413 named Universal Evaluation System Data Quality (Sistema Evaluador Universal de Calidad de Datos).

REFERENCES

[1] A. Motro and P. Anokhin, "Fusionplex: Resolution of data inconsistencies in the integration of heterogeneous information sources," Information Fusion, vol. 7, no. 2, 2006, pp. 176 – 196. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253504000867

[2] P. Christen, Data Matching: Concepts and TEchniques for Record Linkage, Entity Resolution and Duplicate Detection. Springer Data-Centric Systems and Applications, 2012.

[3] P.Christen, "Febrl a freely available record linkage system with a graphical user interface," Second Australasian Workshop on Health Data and Knowledge Management (HDKM 2008), vol. 80, 2008, pp. 17–25.

[4] T. Churches, P. Christen, K. Lim, and J. X. Zhu, "Preparation of name and address data for record linkage using hidden markov models." BMC Medical Informatics and Decision Making, vol. 2, no. 1, 2002, p. 9.

[5] M. Odell and R. Russell, "The soundex coding system," American Patent 1 261 167, 1918.

[6] C. L. Borgman and S. L. Siegfried, "Gettys synonametm and its cousins: A survey of applications of personal name-matching algorithms," Journal of the American Society for Information Science, vol. 43, no. (7), 1992, pp. 459–476.

[7] L. Philips, "The double metaphone search algorithm," C/C++ Users J, vol. 18, no. 6, 2000, pp. 38–43.

[8] M. A. Jaro, "Advances in record-linkage methodology applied to matching the 1985 census of tampa, florida," Journal of the American Statistical Association, vol. 84, 1989, pp. 414–420.

[9] W. Winkler, "String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage," Proceedings of the Section on Survey Research Methods, American Statistical Association., 1990, pp. 354–359.

[10] D. Barone, A. Maurino, F. Stella, and C. Batini, "A privacy-preserving framework for accuracy and completeness quality assessment," Emerging Paradigms in Informatics, Systems and Communication, 2009, p. 83.

[11] A. Monge and C. Elkan, "An efficient domain-independent algorithm for detecting approximately duplicate datadata records." 1997.

 ISBN: 978-1-61208-408-4

Comparison of methods Hamming Distance, Jaro, and Monge-Elkan

Maria del Pilar Angeles, Adrian Espino-Gamez
Facultad de Ingenieria
Universidad Nacional Autonoma de Mexico
Mexico, D.F.
Email: pilarang@unam.mx, adriespingam@gmail.com

***Abstract*—The present paper shows the implementation of a more strict comparison algorithm Hamming Distance, which has been enhanced because it not only determines similarity among substrings, but also takes into consideration their corresponding order. Furthermore, we have carried out an evaluation of quality data matching through the string similarity functions Hamming distance, Jaro distance, and Monge-Elkan distance in terms of precision-recall, f-measure, and execution time.**

Keywords–data matching; de-duplication; record linkage.

I. INTRODUCTION

The task of data matching has been approached by a number of disciplines, such as artificial intelligence, statistics, and databases, with a different perspective, and different techniques proposed. In our case, data matching is approached from the perspective of Integration of Heterogeneous Databases, especially on how to improve the accuracy of data matching, and how to scale data matching to very large databases that contain many millions of records. The experiments presented here have only been evaluated on small data sets. However, there is still work to do in publishing surveys that compare the various data matching and deduplication techniques that have been developed in different research fields.

We have developed a prototype named Universal Evaluation System Data Quality (SEUCAD) [1] on the basis of the Freely Extensible Biomedical Record Linkage prototype (FEBRL) [2] for cleaning, deduplication and record linkage. The process of data matching is executed within three stages: Indexing: In order to reduce the number of comparisons of pairs of records, the information is segmented according to certain fields (blocked index) and coded function. Comparison: Is a function that identifies the similarity between pairs of records, which returns a numeric value between 0 for total dissimilarity and 1 in case of two identical strings. Classification: There are methods of supervised and unsupervised classification, considering a weight vector and the comparisons made in the previous step, the classification determines false positives, false negatives, true positives and true negatives.

The present paper is organized as follows: Section 2 is focused on similar approaches and their lack of some edit similarity functions comparison. We briefly explain the three comparison methods and their role within de process of data matching. Section 3 details the process of evaluation of data matching to be executed in order to obtain the performance of the comparison methods. Section 4 shows the experimentation plan and the four scenarios considered. Section 5 analyses the performance of the comparison methods from the experiment results and Section 6 concludes the main topics achieved and the future work to be done.

II. RELATED WORK

The present section briefly explains what has been done in terms of comparison of string similarity functions and why we propose the Hamming-distance, Jaro, and Monge-Elkan comparison methods to be tested during the process of data matching [3]. There has been a small number of comparisons of string metrics for data matching. In [4] there was a comparison between the following distance functions: Jaro, Jaro-Winkler, Smith Waterman, and Monge-Elkan. In such research, the results showed that on average, Monge-Elkan method performed best of the edit-distance-like methods in terms of recall.

Michelle Cheatham et al. [5] compare a number of string similarity functions for Ontology alignment. Among the similarity functions the Jaro Winkler, and Monge-Elkan methods were analysed. The outcomes regarding this two functions were that Jaro Winker performed better than Monge-Elkan in terms of precision and recall. In the case of legal case management systems in [6] the performance of a number of name matching algorithms was evaluated such as Exact-Match, Nsoundex, Palmer, Approximate matching, etc. However, the similarity functions proposed in this research work were not considered. Even though human reading seems to be unimpressed by framed permutations ambiguous cases might arise, such as with/whit and expcet/expect then the hamming distance would determine the interpretation.

As our intention is to determine which similarity function works best in terms of quality of data matching, we have carried out a number of comparisons considering different string edit distances, and token edit distances, but their publication still on process. The present paper is aimed to show the comparison of three already mentioned similarity functions as part of a our work.

A. Hamming distance

The Hamming distance [7] is named after Richard Hamming, who introduced it in his fundamental paper on Hamming codes error detecting and error correcting codes in 1950. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. Therefore, it measures the minimum number of substitutions required to change one string into the other, or the number of errors that transformed one string into the other.

We have enhanced the comparison algorithm Hamming distance, because it not only finds the order of the letters, but also takes into consideration the content of the words regardless the order and size of both strings. For instance, let be to strings S1, S2, the first step is to identify which is the largest string, regardless spaces. Suppose S2 is the longest

 ISBN: 978-1-61208-408-4

string. Then, the distance is the difference between the longest string minus the smallest string.

$$distance = length(S2) - length(S1) \quad (1)$$

S1 (the smallest string) will be scanned letter by letter and in case a letter that does not correspond with S2, the variable distance will be added by one. The next step is to obtain the factor of distance, which is been given in the following formula:

$$distance\_factor = \frac{(length(S2) - distance)}{length(S2string)} \quad (2)$$

The Hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used. It is thus equivalent to the Hamming distance from the all-zero string of the same length.

B. Jaro distance

The Jaro similarity function was developed by Matthew Jaro in [8]. This function was designed specifically for comparing short length strings, such as names, and is given by the following formula:

$$simjaro(s1, s2) = \frac{1}{3}(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}) \quad (3)$$

The Jaro similarity function counts the number of characters that match, where c is the number of coincident characters and t is half the number of transpositions (two adyacent characters that are interchanged en both strings, such as 'pe' and 'ep'). For instance, considering two strings S1 = 'mario alfonso' and S2 = 'Marian alonso'. Applying the Jaro similarity function, the results are as follows: Jaro('alfonso','Marian') = 0.6190; Jaro('alfonso','Alonso') = 0.9523; Jaro('mario', 'Marian') = 0.9047; Jaro('mario','Alonso') = 0.5777.

C. Monge-Elkan distance

Monge and Elkan proposed in [9] a simple but effective method for measuring the similarity between two strings containing multiple tokens, using an internal similarity sim (a, b) capable of measuring the similarity between two individual tokens a and b. Given two texts A, B being their respective number of tokens $|A|$ and $|B|$, the Monge-Elkan algorithm measures the average of the similarity values between pairs of more similar tokens within texts A and B. The Monge-Elkan similarity formula is as follows:

$$MonElkan(A, B) = \frac{1}{|A|}\sum_{i=1}^{|A|} max\{sim'(A_i, B_j)\}_{i=1}^{|B|} \quad (4)$$

III. Evaluation of Matching

Matching quality refers to how many of the classified matches correspond to true real-world entities, while matching completeness is concerned with how many of the real-world entities that appear in both databases were correctly matched [10]. Each of the record pair corresponds to one of the following categories according to [3].

- True positives (TP). These are the record pairs that have been classified as matches and that are true matches. These are the pairs where both records refer to the same entity.
- False positives (FP). These are the record pairs that have been classified as matches, but they are not true matches. The two records in these pairs refer to two different entities. The classifier has made a wrong decision with these record pairs. These pairs are also known as false matches.
- True negative (TN). These are the record pairs that have been classified as non-matches, and they are true non-matches. The two records in pairs in this category do refer to two different real-world entities.
- False negatives (FN). These are the record pairs that have been classified as non-matches, but they are actually true matches. The two records in these pairs refer to the same entity. The classifier has made a wrong decision with these record pairs. These pairs are also known as false non-matches.
- Precision calculates the proportion of how many of the classified matches (TP + FP) have been correctly classified as true matches. It thus measures how precise a classifier is in classifying true matches [11]. It is calculated as: precision= TP/(TP+FP)
- Recall measures how many of the actual true matching record pairs have been correctly classified as matches [11]. It is calculated as: recall= TP/(TP+FN).

An ideal outcome of a data matching project is to correctly classify as many of the true matches as true positives, while keeping both the number of false positives and false negatives small. Based on the number of TP, TN, FP and FN, different quality measures can be calculated. However, most classification techniques require one or several parameters that can be modified and depending upon the values of such parameters, a classifier will have a different performance according to the number of false positives and negatives. The outcomes can be visualized in different ways to illustrate the performance of several classification tecniques.

- Precision-recall graph. In this visualisation the values of precision and recall are plotted against each other as generated by a classifier with different parameter settings. Recall is plotted along the horizontal axis (or x-axis) of the graph, while precision is plotted against the vertical axis (or y-axis). As parameter values are changed, the resulting precision and recall values generally change as well. Therefore, in Precision-recall graphs there is often a curve starting in the upper left corner moving down to the lower right corner. Ideally, a classifier should achieve both high recall and high precision and therefore the curve should be as high up in the upper right corner as possible.
- F-measure graph. An alternative is to plot the values of one or several measures with regard to the setting of a certain parameter, such as a single threshold used to classify candidate records according to thieir summed comparison vectors, as the threshold is increased, the number of record pairs classified as non-matches increases (and thus the number of TN and FN increases), while the number of TP and FP decreases.

The present work has evaluated the data matching outcomes using synthetically generated data. Consequently, true match status of record pairs was already known. Therefore,

 ISBN: 978-1-61208-408-4

we have developed a set of experiments in order to compare the performance of the following distance algorithms: Hamming, Jaro and Monge-Elkan through the evaluation of the data matching process by computing Precision, Recall and F-measure metrics varying the comparison method only. The set of experiments will be detailed in the following section.

IV. EXPERIMENTATION

The set of experiments correspond to four scenarios, one data file per scenario, which has been indexed, compared and classified three times each. In the case of indexing and classification processes, the corresponding methods and parameters remained the same. However, in the case of comparison process the method has been changed to Hamming-distance, Jaro distance, and Monge-Elkan method.

1) Common configuration for Indexing: Data de-duplication can generally operate at the file or block level. The process of de-duplication by scanning the entire file is not a very efficient means of deduplication. In the case of Block de-duplication, it looks within a file and saves unique iterations of each block. Each chunk of data sorted according to an specific index key, and the comparison process is executed on each block.

- Indexing method: Rather than deduplicate the entire file we have selected Blocking index.
- Blocking key: We have chosen three fields as blocking key or index key ”surname”, given_name and suburb, on each case, we did not set a maximum number of characters for the definition of indexing, otherwise large values will be truncated, and the fields to be compared contain more than one word.
- Sort words: This option was not enable in order to avoid the division and ordering of each word.
- Reverse: The reverse parameter was disabled because otherwise the input string will be reversed and in the case of surname field would not be a representative indexing definition.
- Encoding function: The encoding function selected was ”Soundex” .

The configuration for indexing is presented in Figure 1.

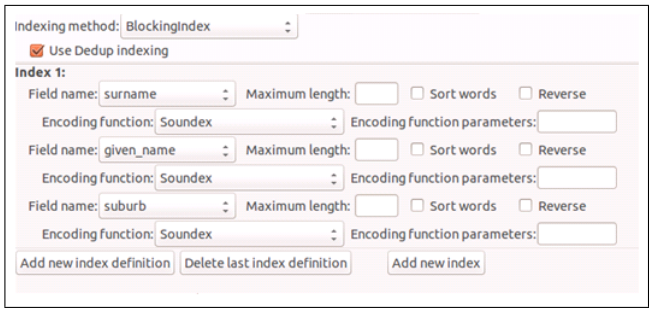


Figure 1. Indexing configuration

2) Common configuration for Classification: This section is aimed to present the configuration parameters established for the execution of the classification method.

- Weight vector classification method: The Fellegi Sunter method has been selected. We have enhaced our prototype in order to compute the data matching metrics even in the case of using a non exact classification method. Otherwise we would have to use for instance, String-exact classification method. It is assumed that the true match status for all compared record pairs is known in the case of supervised classification.

Figure 2 shows the classification settings for the experiments.

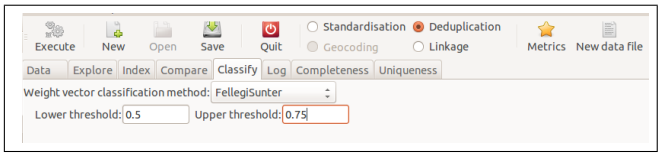


Figure 2. Configuration settings for classification of records.

3) Common configuration for Comparison: The following parameters were chosen for all the comparison cases above described.

- Field name A and Field B: Correspond to the name the fields to be compared against each other. We have chosen ”given_name” field, since the given name is relevant to identify people, and the comparison field is recommended to be of datatype String.
- Cache comparisons: Indicate whether the calculation of the similarity of values can not take place on memory. It is recommended when data values are large, complex, or there is limited number of fields. As ”given_name” field is not complex nor large, the option of ”Cache comparisons” will be disabled. Thus, the calculations will not be performed in memory.
- Maximum cache size: This value is limited to a certain number of pairs of fields. Since it is desired that the comparisons of all pairs of fields are made, the option to ”Maximum cache size” will default to ”None”.
- Missing value weight: The value to be given in the event that one or both fields have no value. Its default value is 0.0 and its value must be within the range of ”value Disagreeing weight” and ”weight Agreeing value.” For comparison operations are more accurate when one or both fields have no value, the value of ”Missing value weight” will be ”0.0”.
- Agreeing value weight: The value to be given when the similarity is entirely accurate. By default the value is 1.0.
- Disagreeing value weight: The value to be given when the similarity is entirely different. By default the value is 0.0. This value must be less than ”Agreeing value weight”. Like the previous value, the value of ”Disagreeing value weight” as ”0.0” will be defined when the similarity of two strings is totally different. In the case of the second comparison Dist-Hamming, Jaro or Monge-Elkan, a parameter threshold was required.
- Threshold: This value should be set in the range of 0.0 and 1.0, and will determine a better level of accuracy. If the calculation of approximate similarity method is higher than indicated in this field (threshold), then the similarity value will be calculated. If the approximate

 ISBN: 978-1-61208-408-4

similarity is less than that indicated in this field (threshold), then the similarity value will correspond to the "Disagreeing value weight" parameter. Therefore, we have chosen 0.25.

A. First Scenario

The file called data_comparacion1.csv was generated with a total length of 1000 records, 500 original records, 500 duplicated records, and only one changed field per record. In the case of classification with Fellegi-Sunter, the lower threshold was 0.5 and the upper threshold was set to 0.75.

B. Analysis of Results of the first scenario

As the number of duplicated records was 500, a total of 418 pairs of records were detected and evaluated on each comparison method.

1) Hamming Distance: In the case of Hamming distance, 406 record pairs were classified as matches with 391 record pairs with a sum weight of 0.9, 11 record pairs with a sum weight of 0.8 (above of threshold, which was set to 0.75), 7 record pairs with sum weight of 0.7, 4 record pairs were classified as possible matches and 8 record pairs were classified as non matches, being completely discarded with a sum weight of 0.0,1 record pair obtained a sum weight of 0.60. The 406 record pairs were true positives. The total time taken for the process was of 337 miliseconds.

2) Jaro: In the case of Jaro, 410 record pairs were classified as matches, 403 pairs obtained a sum weight of 0.9,7 record pairs obtained a sum weight of 0.8 (above of threshold, which was set to 0.75), 8 record pairs were completely discarded with a sum weight of 0.0. The 410 record pairs were true positives, and 8 record pairs were false negatives, no true negatives, no false positives. Regarding the quality properties calculated, recall was 0.9808, F-measure 0.9903 and Precision 1. The total time taken during the process was of 357 miliseconds.

3) Monge-Elkan: In the case of Monge-Elkan method, 410 were classified as matches with 403 with a sum weight of 0.9 and 7 record pairs with a sum weight of 0.8 (above of threshold, which was set to 0.75) 8 record pairs were completely discarded with a sum weight of 0.0. 410 record pairs were true positives, and 8 record pairs were false negatives, no true negatives, no false positives. We can observe from the experiments results that Monge-Elkan and Jaro had the same quality of data matching. However, regarding the Quality properties calculated, recall was 0.9808, F-measure 0.9903 and Precision 1. The total time taken for the process was of 336 miliseconds. Thus, the Monge-Elkan method was the fastest.

We observed that the three methods obtained practically the same scores for the quality metrics, Hamming distance presented a more specific values on the sum weights of the comparisons, compared to Jaro and Monge-Elkan, because the Hamming distance comparison was the strictest in order to assign a high value, but this higher level of precision is a disadvantage because is more sensible to the threshold value. Hamming distance was the only comparison method that classified 4 record pairs as possible matches, Jaro and Monge-Elkan classified the same 4 record pairs as matched and they were true possitives. On one hand, the Monge-Elkan method presented better results than Hamming distance taking practically the same time. On the other hand, Monge-Elkan method had the same results as Jaro, but with a difference of 21 miliseconds.

C. Second Scenario

The file called "data_comparacion2.csv" was generated with a total length of 500 records, 300 original records, 200 duplicated records, only one record duplicated of original record as maximum, and four field changes per record. In the case of classification with Fellegi-Sunter, the lower threshold was 0.4 and the upper threshold was set to 0.75.

D. Analysis of Results of second scenario

As the number of duplicated records was 200, a total of 114 pairs of records were detected and evaluated on each comparison method. During the execution of the second scenario, there were classified 114 records as matches, The 114 record pairs were true positives. The data matching quality metrics Precision, Recall and F-measure obtained a score of 1.

1) Hamming Distance: In the case of Hamming function string, 6 record pairs obtained a sum weight of 0.80, 2 record pairs obtained a sum weight of 1.40, 8 record pairs obtained a sum weight of 1.6, and 98 record pairs obtained a sum weight of 1.80. the total time taken for the process was of 289 miliseconds. Figure 3 shows the results.

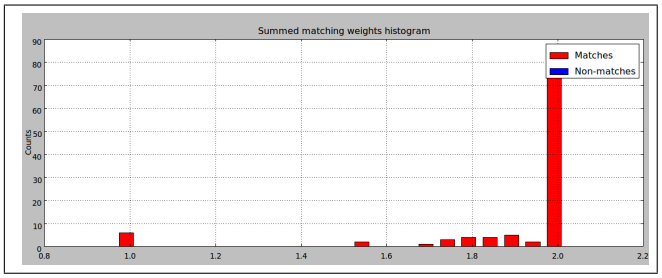


Figure 3. Classification of records with Hamming as comparison method

2) Jaro: In the case of the Jaro function string, from the 114 record pairs classified as true matches, 6 record pairs obtained a sum weight of 0.80, and 108 record pairs obtained a sum weight of 1.8. The total time taken for the process was of 286 miliseconds, being the fastest on the second scenario. Figure 4 shows the results.

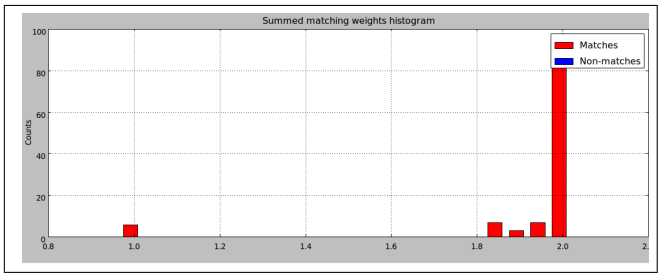


Figure 4. Classification of records with Jaro as comparison method

3) Monge-Elkan : In the case of the Monge-Elkan method from the 114 record pairs classified as true matches, 6 record pairs obtained a sum weight of 0.80, and 108 record pairs obtained a sum weight of 1.80. The time was the longest with 294 miliseconds. Figure 5 shows the results. The three methods obtained the same results. There was practically no difference.

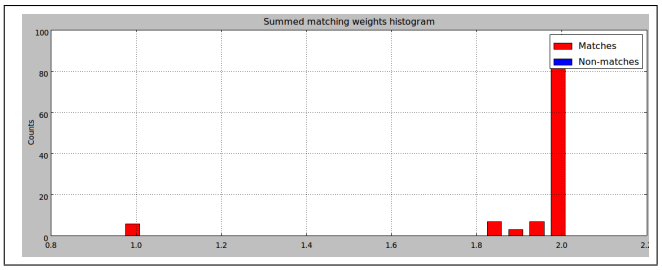


Figure 5. Classification of records with Monge-Elkan as comparison method

E. Third Scenario

The file called "data_comparacion3.csv" was generated with a total length of 1000 records, 800 original records, 200 duplicated records, with two records duplicated of original record as maximum, and three changes per record. In the case of classification with Fellegi-Sunter, the lower threshold was 0.5 and the upper threshold was set to 0.85.

F. Analysis of Results of third scenario

As the number of duplicated records was 200, a total of 141 pairs of records were detected and evaluated on each comparison method.

1) Hamming Distance: In the case of Hamming distance, 122 were classified as matches, 8 record pairs were completely discarded as non matches with 0.0 as a summed weight, 11 record pairs were classified as possible matched. There were 3 record pairs with a sum weight of 0.60, 4 record pairs with a sum weight of 0.70, 5 record pairs with a sum weight of .8 and 121 record pairs with a sum weight of .90. The 122 record pairs classified as matches were true positives. However, there were 8 record pairs classified as false negatives. Precision was 1, recall was of .9384 and f-measure was of .9682. The total time taken for the process was of 328 miliseconds. Fig. 6 shows the results.

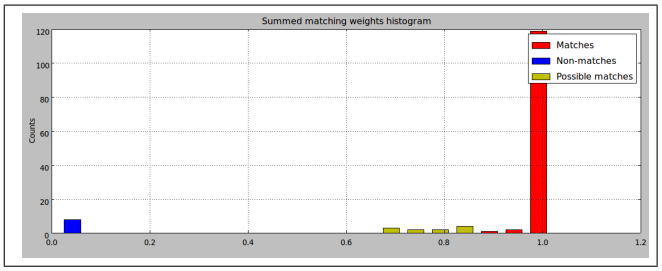


Figure 6. Classification of records with Hamming as comparison method

2) Jaro: In the case of Jaro, 130 records were classified as matches, 8 record pairs were completely discarded with 0.0 as a summed weight as non matches, and 3 record pairs were classified as possible matches. 1 record pair obtained a sum weight of 0.70, there were 5 record pairs with a sum weight of 0.80, and 127 record pairs with a sum weight of .90. The 130 record pairs classified as matches were true positives. There were 8 false negatives and 3 possible matched record pairs. However, there were 8 record pairs classified as false negatives. Precision was 1, recall was of .9420 and f-measure was of .9701. The total time taken for the process was of 336 miliseconds, taking longer than Jaro and Monge-Elkan. Figure 7 shows the results.

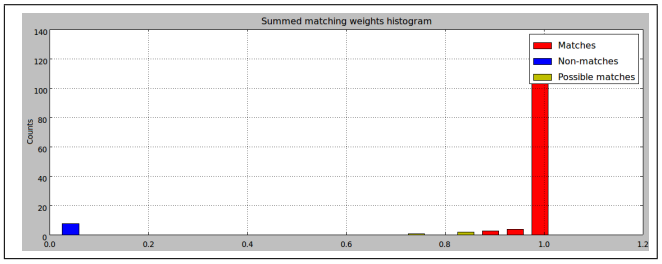


Figure 7. Classification of records with Jaro as comparison method

3) Monge-Elkan : In the case of Monge-Elkan method, 130 records were classified as matches, 8 record pairs were non matches, and 3 record pairs classified as possible matches. 4 record pairs were coompletely discarded with 0.0 as a summed weight, only 2 record pairs with a sum weight of 0.60, 53 record pairs with a sum weight of 0.90, and 82 record pairs with a sum weight of 1.80. The 130 record pairs classified as matches were true positives. However, there were 8 record pairs classified as false negatives. Precision was 1, recall was of .9420 and f-measure was of .9701. The total time taken for the comparison process was of 316 miliseconds, the Monge-Elkan method was the fastest.

As we can observe from the experiment results Jaro and Monge-Elkan obtained the same results and the same quality of data matching, both with better results than Hamming distance. Monge-Elkan distance was the fastest method, and Jaro distance was the slowest. We can observe that the Hamming distance method was more restrictive when making comparisons.

G. Fourth Scenario

The file called "data_comparacion4.csv" was generated with a total length of 800 records, 700 original records, 100 duplicated records. In the case of classification with Fellegi-Sunter, the lower threshold was 0.5 and the upper threshold was set to 0.85.

H. Analysis of Results of fourth scenario

As the number of duplicated records were 100, the number of records pairs evaluated for each method was 93.

1) Hamming Distance: In the case of Hamming distance, 68 record pairs were classified as matches, 20 record pairs classified as non matches, and 5 record pairs classified as possible matches. The 68 record pairs were true positives, 19 record pairs were true negatives, 1 record pair was false negative and no false positives. Regarding the data matching quality metrics, the scores corresponding to precision, recall, and f-measure were 1, 98.55, and 99.27 respectively. The total time taken for the process was of 305 miliseconds. Figure 8 shows the results.

2) Jaro: In the case of Jaro distance, 73 record pairs were classified as matches, 9 record pairs classified as non matches, and 11 record pairs classified as possible matches. The 73 record pairs were true positives, 8 record pairs were true negatives, 1 record paris was false negative and no false positives. Regarding the data matching quality metrics, the

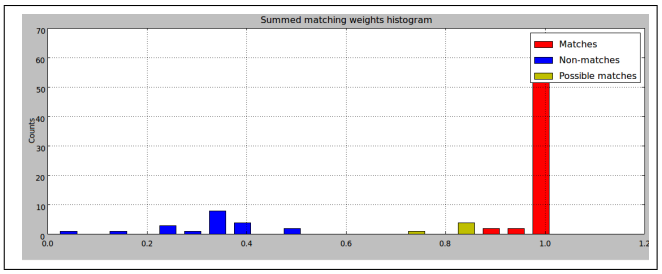


Figure 8. Classification of records with Hamming Distance

scores corresponding to precision, recall, and f-measure are 1, 98.64, and 99.31 accordingly. The total time taken for the process was of 327 miliseconds. Figure 9 shows the results.

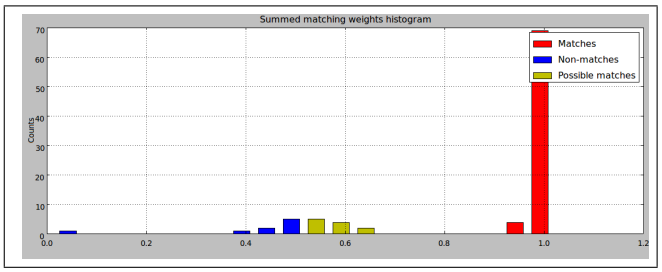


Figure 9. Classification of records with Jaro as comparison method

3) Monge-Elkan: In the case of Monge-Elkan method, 73 record pairs were classified as matches, 17 record pairs classified as non matches, and 3 record pairs classified as possible matches. The 73 record pairs were true positives, 16 record pairs were true negatives, 1 record paris was false negative and no false positives. Regarding the data matching quality metrics, the scores corresponding to precision, recall, and f-measure were 1, 98.64, and 99.31 respectively, presenting the same behaviour as Jaro. The total time taken for the comparison process was of 297 miliseconds, being the fastest method. Figure 10 shows the results.

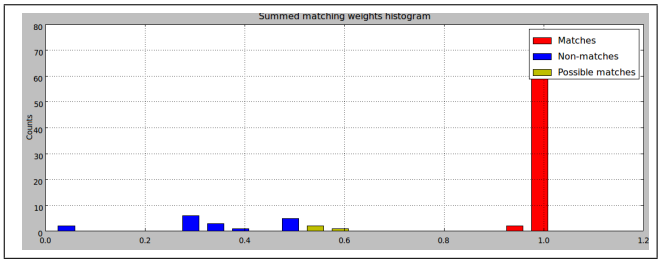


Figure 10. Classification of records with Hamming-Distance as comparison method

V. Performance of Hamming-distance, Jaro and Monge-Elkan

Table I shows data matching quality metrics obtained from the four scenarios. The metrics are: true positives, false positives, true negatives, false negatives, the comparison time, and the number of weights obtained per string function. We can observe that the Monge-Elkan distance function was the fastest method three times out of four scenarios. Hamming distance obtained more true negatives than Monge-Elkan and Jaro because is more strict, but obtained less matched pairs of records for the same reason. Figure 11 shows the corresponding precision, recall and f-measure graph for Hamming Distance. Almost the same performances of data matching were obtained for the three string metrics.

TABLE I. Data Matching Quality Metrics

Scenario	Function	TP	FP	TN	FN	(ms)	no. weigths
1	Hamming	406	0	0	8	337	5
1	Jaro	410	0	0	8	357	3
1	Monge-Elkan	410	0	0	8	336	3
2	Hamming	114	0	0	0	289	4
2	Jaro	114	0	0	0	286	2
2	Monge-Elkan	114	0	0	0	294	2
3	Hamming	122	0	0	8	328	5
3	Jaro	130	0	0	8	336	4
3	Monge-Elkan	130	0	0	8	316	4
4	Hamming	68	0	19	1	305	8
4	Jaro	73	0	8	1	327	6
4	Monge-Elkan	73	0	16	1	297	6

During the first scenario, the Hamming distance was the less effective in terms of f-measure and recall, classifying less pairs of records from the sample than Jaro and Monge-Elkan methods. Furthermore, Monge-Elkan method was the fastest.

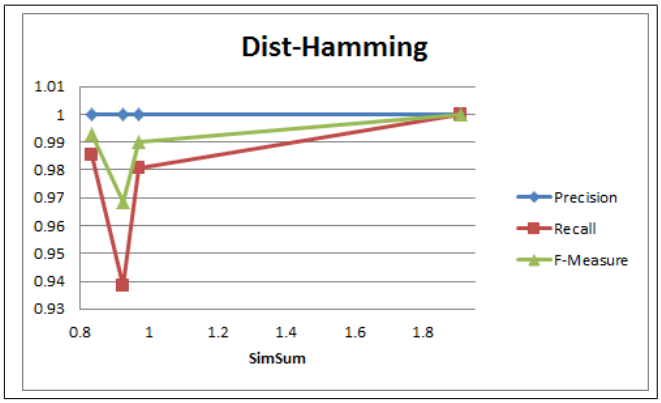


Figure 11. Quality of Data Matching of Hamming Distance

In the second scenario The three distance methods performed equal, but the Jaro method was the fastest.

In the third scenario, Jaro distance was slower than Hamming and Monge-Elkan. In the case of Hamming method, it obtained just one more weight, its comparison allows to classify less pairs of records as matches, presenting lower values of recall and f-measure than Jaro and Monge-Elkan methods.

Finally, in the fourth scenario, Hamming allow the classification of less pairs of records as matches. The three comparison methods obtained just one false negative.

VI. Conclusion and Future Work

We have compared in this research paper three string similarity metrics algorithms: the Hamming distance, the Jaro distance, and the Monge-Elkan distance through our prototype SEUCAD. [1].

We have enhanced the comparison algorithm Hamming distance, because it not only finds the order of the letters,

 ISBN: 978-1-61208-408-4

but also takes into consideration the content of the words regardless the order and size of both strings. We have improved our prototype in order to utilized any classification method, such as Fellegi Sunter, compute the quality metrics, and be able to assess the data matching with no consideration of exact classification methods such as String exact comparison method.

After a number of experiments we have been carried out we can conclude that as the Hamming-distance is stricter during comparison, it obtains a higher number of weights. Therefore, it is more sensible to the thresholds assigned. However, its performance was lower than Jaro and Monge-Elkan methods in terms of recall, and f-measure.

Since, there has not been a significant difference in comparing the string distance methods Jaro, Monge-Elkan and Hamming. There is still work to be done in publishing surveys that compare the various data matching and deduplication techniques that have been developed in different research fields. Furthermore, we will be focused on the enhancement of the already implemented methods and the test on large volume of data as part of our future work.

ACKNOWLEDGMENT

This work is being supported by a grant from Research Projects and Technology Innovation Support Program (Programa de Apoyo a Proyectos de Investigacin e Innovacin Tecnolgica, PAPIIT, UNAM Project IN114413 named Universal Evaluation System Data Quality (Sistema Evaluador Universal de Calidad de Datos).

REFERENCES

[1] P. Angeles, F. Garcia-Ugalde, C. Ortiz, R. Valencia, E. Reyes, A. Nava, and J. Pelcastre, "Universal evaluation system data quality," DBKDA 2014 : The Sixth International Conference on Advances in Databases, Knowledge, and Data Applicationst, vol. 32, 2014, pp. 13–19.

[2] P.Christen, "Febrl a freely available record linkage system with a graphical user interface," Second Australasian Workshop on Health Data and Knowledge Management (HDKM 2008), vol. 80, 2008, pp. 17–25.

[3] P. Christen and K. Goiser, "Quality and complexity measures for data linkage and deduplication," F. Guillet, H. Hamilton (eds.) Quality Measures in Data Mining, Studies in Computational Intelligence,Spriger, vol. 43, 2007, p. 127151.

[4] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," 2003, pp. 73–78.

[5] M. Cheatham and P. Hitzler, "String similarity metrics for ontology alignment," in The Semantic Web ISWC 2013, ser. Lecture Notes in Computer Science, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, Eds. Springer Berlin Heidelberg, 2013, vol. 8219, pp. 294–309. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41338-4_19

[6] K. Branting, "A comparative evaluation of name-matching algorithms." in ICAIL, 2003, pp. 224–232. [Online]. Available: http://dblp.uni-trier.de/db/conf/icail/icail2003.html#Branting03

[7] U. Pfeifer, T. Poersch, and N. Fuhr, "Retrieval effectiveness of proper name search methods," Information Processing and Management, vol. 32, no. 6, 1996, pp. 667–679.

[8] M. A. Jaro, "Advances in record-linkage methodology applied to matching the 1985 census of tampa, florida," Journal of the American Statistical Association, vol. 84, 1989, pp. 414–420.

[9] A. Monge and C. Elkan, "An efficient domain-independent algorithm for detecting approximately duplicate datadata records." 1997.

[10] D. Barone, A. Maurino, F. Stella, and C. Batini, "A privacy-preserving framework for accuracy and completeness quality assessment," Emerging Paradigms in Informatics, Systems and Communication, 2009, p. 83.

[11] A. M. I. H. Witten and T. C. Bell, Managing Gigabytes, 2nd ed. Morgan Kaufmann, 1999.

 ISBN: 978-1-61208-408-4

An Efficient Approach to Triple Search and Join of HDT Processing Using GPU

YoonKyung Kim, YoonJoon Lee
Computer Science
KAIST
Daejeon, South Korea
e-mail: {ykkim, yjlee}@dbserver.kaist.ac.kr

JaeHwan Lee
Computer Science
IUPUI
Indianapolis, United States
e-mail: johnlee@iupui.edu

***Abstract*— Resource Description Framework (RDF) is originally designed as a metadata data model. It has an advantage of efficient exchange between different metadata by supporting a set of common rules. To process RDF data efficiently, SPARQL (SPARQL Protocol and RDF Query Language) was introduced. In this era of emerging Web of Data, the amount and size of published RDF data have dramatically increased. Most studies so far have focused on the compression of RDF data and fast SPARQL query processing using single core CPUs. Thus, they do not utilize well current multicore environment. We in this study propose SPARQL query processing using a GPU multicore system. We focus on a search and join method using Header, Dictionary, Triples(HDT) data format and present its experimental results.**

Keywords-Resource Description Framework (RDF); HDT; SPARQL; GPGPU.

I. INTRODUCTION

The Resource Description Framework (RDF) is a standard model for data interchange on the Web [1]. It has been designed for flexible representation of the Semantic Web. It was proposed to efficiently represent connections between hyperlinks embedded along with a word or a string in web data/pages such as Wikipedia. RDF consists of three parts {*subject-predicate-object*} called triple. While *subject* represents resource, *predicate* represents relation between *subject* and *object*. For example, representing 'His name is Hoon' in RDF can be {*subject*: He, *predicate*: name, and *object*: Hoon}. As in this example, most of RDF data are string data. To efficiently process large size of RDF data, SPARQL Protocol and RDF Query Language (SPARQL) [9] was introduced, and it supports extensible value testing and constraining queries by source RDF graph. [2].

The size of published RDF datasets has dramatically increased in this era of emerging Web of Data [3]. Previous researches are mostly focused on the compression of RDF data [11] and fast SPARQL query processing. We suggest parallel processing using GPU multicore system.

This paper proposes a triple search method on HDT [3] data format using a GPU and presents its experimental result.

In Section 2, we introduce Header, Dictionary, Triples (HDT) which compresses RDF data format. RDF processing using GPU is in Section 3, and experimental results in Section 4.

II. BACKGROUND

A. Query patterns in RDF

If a query is consisted of following example, {?, hasName, ?}, it means that find every triple which has predicate 'hasName'. While this query has predicate only (subject and object are question mark), so it is called pattern ?P? or P. Likewise, {Bob, hasTitle, ?} query means that find triples that has subject 'Bob' and predicate 'hasTitle' This type of query is called pattern SP? or SP.

All queries should have at least one component that is not a question mark. So, there are 7 query patterns in RDF: S??, S?O, SP?, SPO, ?P?, ?PO and ??O. We focus on pattern ?P? in following content.

B. RDF data compression based on HDT

In the past, several studies have been proposed, in which their predominant method of RDF data compression is to use a dictionary. For the compression, entropy coding is often used to distribute short length words to more frequent bit-pattern.

The most representative studies of RDF compression are based on dictionary. RDF-3x [4] transforms triples using a dictionary, saves them with a B+ tree method, and divides queries to several partials to process the query.

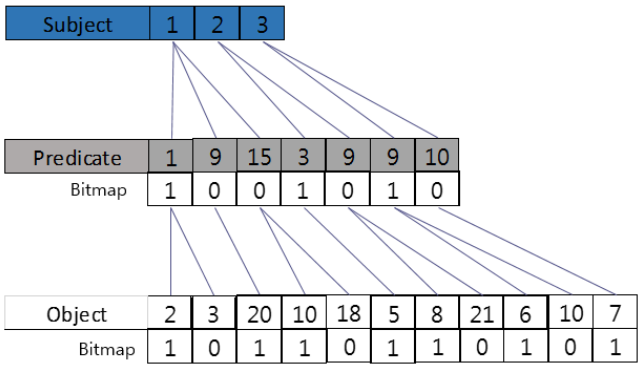


Figure 1. Triple compression of HDT.

 ISBN: 978-1-61208-408-4

Header, Dictionary, Triples (HDT) [3] transforms triple expressions (subject-predicate-object) to unique integer IDs using a lexicographically sorted dictionary. A triple is represented as a tree having subject as its root. Each subject has its own tree, so a whole set of triples are shaped like a forest. To compress triples in a forest, as shown in Figure 1, subjects, predicates, and objects are represented as bitmaps. This bitmap array is filled with zeros and ones. Zero means this element has same parent as the previous one, and one means it has a new parent. For example, the 4th predicate of Figure 1 is '3', which has subject '2', can be calculated by adding predicate bitmap values from 1 to 4. One is represented twice (at the 1st location and 4th location of the bitmap array), meaning that predicate '3' is connected with subject '2'. The bitmap array of objects is similarly encoded to grab its predicate.

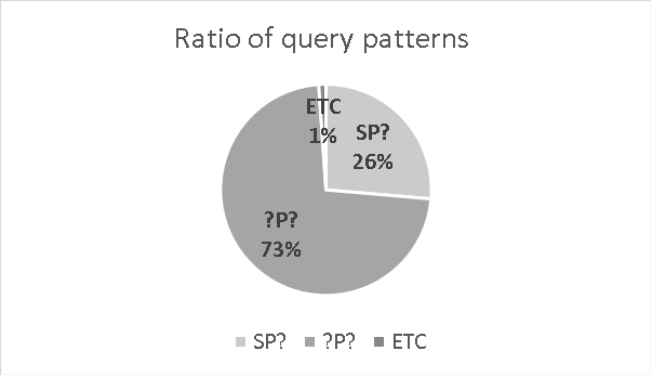


Figure 2. Query pattern ratio in SP2 benchmark.

This method of compression shows better performance when searching triples with subject (i.e., S??, SP?, S?O, SPO), and other search cases are accelerated by making index for predicate and object [4].

C. Problem Definition

Previous studies on RDF have mostly focused on a single core query processing. The disadvantage is that a triple search patterns, ?P? (P pattern) where only the predicate exists, is extremely slow. P pattern is a query that find subject and object which has a certain predicate. Unlike other components, predicate has a few variety of elements. It means that most of P pattern search results are too large to calculate using a single core.

The variance of predicates in the data is small, meaning that one predicate occurs very often. Also, the ratio of P pattern in SP2 benchmark [6] take 73% of total query triples as seen in Figure 2.

HDT makes an index for P pattern searching, but the number of the indexes is too large to process. We suggest an approach to improve this P pattern problem using GPGPU. General purpose graphic processing unit (GPGPU) is proposed in [5], focusing on its high level parallel computation power.

III. METHODOLOGY

A. Triple Pattern Searching

In the proposed approach, we modify bitmap arrays for parallel processing. The original HDT has used wavelet trees [3] to support faster search of predicate index. However, the method was optimized for single core processing. Instead, we use a GPU's scan algorithm to obtain the parent's index in a constant time. The scan algorithm constructs a position array that has the summation result of occurrences of ones from location 1 to each element's location in parallel as shown in Figure 3. In this way, predicate and object's parents (subject and predicate, respectively) can be found in a constant time using this array.

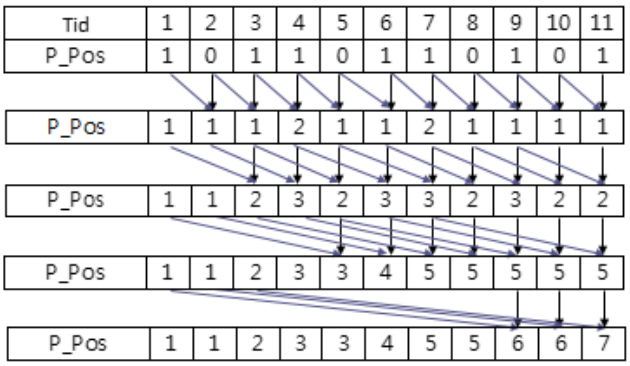


Figure 3. GPU scan algorithm for bitmap array.

After populating the position array, Figure 4 shows how this array represents predicates connected to each object. For example, object '20' (third element in object array) refers to position array (also third element in position array) which tells that '2' (second) element of predicate array is predicate of its triple. We can also find a subject using a predicate. As we allocate one object per GPU core, processing time to search a triple is constant.

Figure 4. Connection between predicate and object using object's position array.

We copy predicate, object, and object position arrays to GPU memory to search a pattern P as shown in Figure 5. After allocating all the triples to GPU cores, the search first

determines whether each object's predicate is equal to given pattern P or not. We use object array to determine triples, while others (predicate, subject) are compressed.

The object array points to the predicate array, and the predicate array points to subject array. These connections are directional, which means the predicate and subject array do not know which element of object is connected. So, the object array is needed to extract triples from the HDT data set which is compressed.

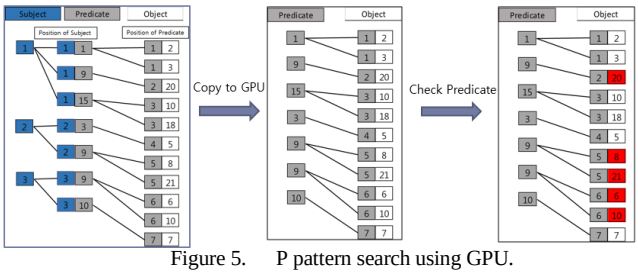


Figure 5. P pattern search using GPU.

B. Join Processing

Due to large size of RDF data, it restricts join algorithms from a result array point of view. First, if we save the result in one big array, an atomic operation of result array's pointer is needed for every new result. All other threads have to wait until the lock is released. Another method is allocating arrays to each triple, requiring N^2 memory where N is the size of input triples. It takes 1TB GPU memory for one million integer data. Considering these conditions, [7] use hash join algorithm for GPU.

With the result that GPU joins become faster with data larger than 5 million. [7] Even with data size larger than 5GB, a join hardly exceeds five million. Hence, we use a CPU sort-merge join method which original HDT uses in join processing.

IV. Experiment and Analysis

We process an experiment to compare the effect of RDF join process using GPU with previous researches (RDF-3x, original HDT).

TABLE I. Experimental Environment

CPU	AMD A10-5800K 3.80GHz
RAM	16GB
GPU	Geforce GTX 750Ti, Geforce GTX 660
Data	5GB, 1GB dataset, SP^2 benchmark
OS	Win7 in GPU, Ubuntu 12.04 in others

A. Experimental Approach

We use SP^2 benchmark [6] with several queries. Q1, Q2, Q3a, Q3b, Q3c, Q5a, and Q5b are join queries. Q10 is single O pattern query, and Q11 is single P pattern query. We exclude Q4, Q6, Q7, Q8, Q9, Q12, with following reasons. Q4 is a query for FILTER function in SPARQL, Q6, Q7 for OPTIONAL, Q8, Q9 for UNION, and Q12 for ASK. These functions are not supported in original HDT and don't have relationship with join processing. We also remove Q11's ORDERBY, LIMIT, and OFFSET to compare single P pattern search speed. All results are average of five executions while removing maximum and minimum outcomes.

We also try to use Apache Jena [10], and Bitmat [11] to compare results, but it did not work in our data set due to large size (1,5GB).

B. Experimental Results

Search speed values for each triple pattern are shown in Figure 6. The triple search using a GPU is eight times faster than other methods in P pattern, while the search speeds of other patterns are slower than those of existing methods.

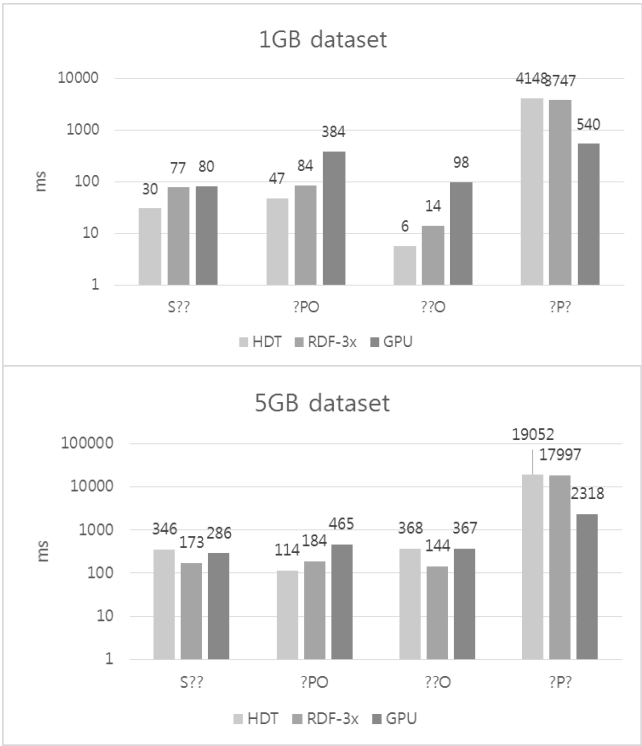


Figure 6. Speed of triple searching.

Next, we use two GPUs (shown in Table 1) to determine the number of GPU cores and memory size effect on processing speed. We use GTX 750 Ti and GTX 660, where the latter has 50% more cores than the former. Figure 7 shows the results for query processing speed. GTX660 is faster than GTX750 Ti with 5% ratio. Considering the difference in the number of cores between the two GPUS, it does not effect on GPU processing. Thus, memory transfer is critical for GPU processing. [8]

Figure 8 shows the results of SP^2 benchmark. GPU processing is slower in Q1, Q3b, Q3c, and Q10, which has result of small size. Other queries are faster. Q2's result for the original HDT is zero because it produced a wrong answer due to a program fault.

 ISBN: 978-1-61208-408-4

V. CONCLUSION

We propose an efficient RDF (HDT) query processing focusing on P pattern. Previous methods are slow based on the number of answer triples. Parallel processing using GPU increases the speed of P pattern searching. Our first approach was affected by size of memory, but not the number of GPU cores. Thus, we choose HDT to which compresses RDF data. This improved method is faster than original HDT and RDF-3x when the number of answers is large.

Our future work includes support for GPU SPARQL query processing (e.g. FILTER, UNION, and GROUPBY), and determine to use or not to use GPU based on expected speed of query processing. And a study for HDT join processing using GPU is also needed.

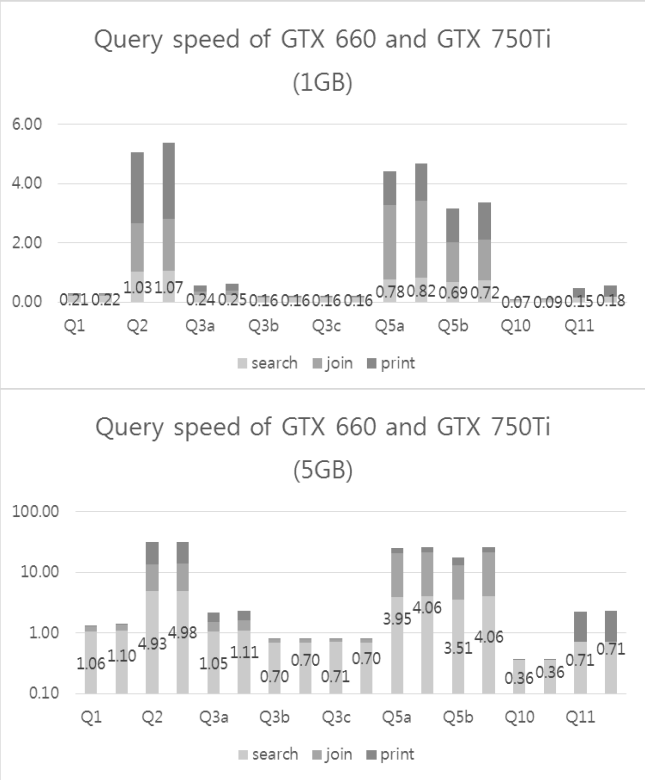


Figure 7. Query speed of GTX 660 (left), GTX 750Ti (right).

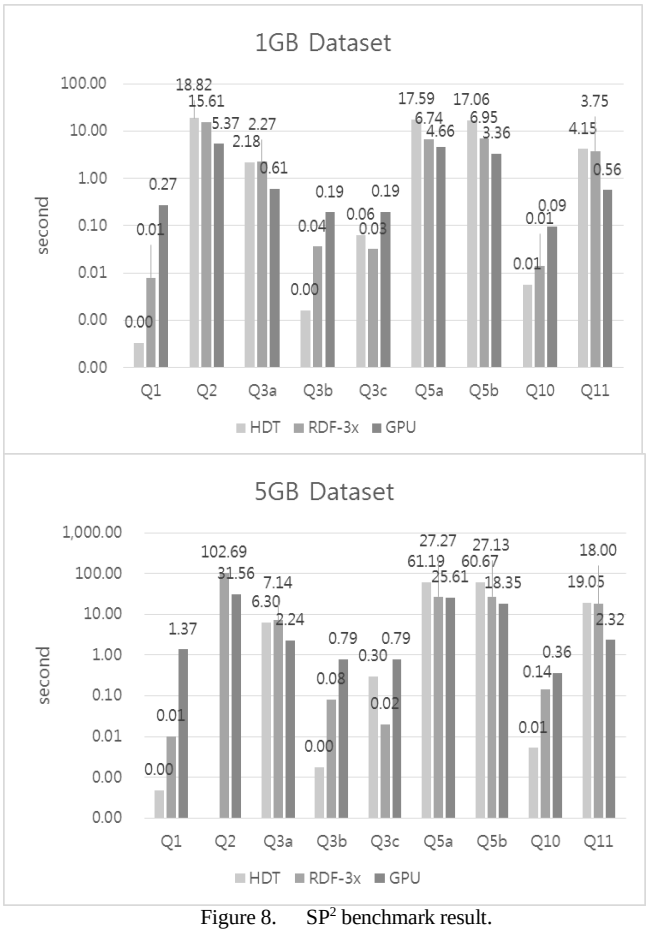


Figure 8. SP^2 benchmark result.

Acknowledgements This work was funded by Brain Pool Program, KOFST (The Korean Federation of Science and Technology Societies).

REFERENCES

[1] W3C. http://www.w3.org/RDF/, 2004 [retrieved: March, 2015].

[2] Prud'hommeaux, E., Seaborne, A. Sparql query language for rdf, http://www.w3.org/TRlrdf-sparqlquery, 2008 [retrieved: March, 2015].

[3] Martínez-Prieto, Miguel A., Mario Arias Gallego, and Javier D. Fernández. "Exchange and consumption of huge RDF data." The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 2012. pp. 437-452.

[4] Neumann, Thomas, Gerhard Weikum. "The RDF-3X engine for scalable management of RDF data." The VLDB Journal 19.1 (2010): pp. 91-113.

[5] NVIDIA CORPORATION. 2013. Nvidia GPU Programming Guide for Geforce 8 and later GPUs

[6] Schmidt, Michael, et al. "SP^ 2Bench: a SPARQL performance benchmark." Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009.

[7] Senn, Jürg. "Parallel Join Processing on Graphics Processors for the Resource Description Framework." Architecture of Computing Systems (ARCS), 2010 23rd International Conference on. VDE, 2010. pp. 1-8

[8] Dimitrov, Martin, Mike Mantor, and Huiyang Zhou. "Understanding software approaches for GPGPU reliability."

 ISBN: 978-1-61208-408-4

Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. ACM, 2009.

[9] Quilitz, Bastian, and Ulf Leser. Querying distributed RDF data sources with SPARQL. Springer Berlin Heidelberg, 2008.

[10] Jena, Apache. "Apache jena." jena. apache. org. Available: http://jena. apache. Org, 2013 [Accessed: Mar. 20, 2014].

[11] Atre, Medha, and James A. Hendler. "BitMat: a main memory bit-matrix of RDF triples." The 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009). 2009.

 ISBN: 978-1-61208-408-4

Towards a New Incremental Implementation Approach for Distributed Databases

Bouraoui Marwa
Université Tunis El Manar
SITI, ENIT
Tunisia
e-mail: bourawimarwa@gmail.com

Hassen Fadoua
Université Tunis El Manar
LIPAH, FST
Tunisia
e-mail: hassen.fadoua@gmail.com

Grissa touzi Amel
Université Tunis El Manar
ENIT, LIPAH, FST
Tunisia
e-mail: amel.touzi@enit.rnu.tn

Abstract—In this paper, we propose a new incremental implementation approach of a Distributed Database (DDB). On one hand, the frequent need to add and/or delete number of sites in the geographical distribution of a DDB has become a requirement of the user. On the other hand, we find that the already existing Distributed Database Management System (DDBMS) does not even offer an automatic implementation for a Distributed Database, which has been initially allocated to a predefined number of sites. In this approach, we propose a weak coupling with any existing DDBMS. This consists in garnishing all DDBMS with an intelligent layer that offers: 1) a convivial interface for an incremental definition of the different sites in the DDB, 2) an incremental design approach to DDB while knowing fragmentation attributes and 3) an automatic update of the scripts in the different sites. To validate our approach, we have used Oracle as an example of DDBMS.

Keywords-Distributed Database; Fragmentation; Allocation; Integrity Constraint; Distributed Updates.

I. INTRODUCTION

The design and implementation of a Distributed Database (DDB) has always been a challenge for the designers of this type of Database especially with: 1) the size of the original model, 2) the frequent need to add and/or suppress sites in the geographical distribution of DDB and 3) the limits of existing Distributed Database Management System (DDBMS).

Several studies have been conducted in this context. As examples, we can cite the work of Abdalla [1] and Moussa [2] who have proposed a support system for the design of DDB. We can also mention the work of Hassen and Grissa [3], who has proposed a new aid approach for the DDB implementation. This approach has been validated by the design and implementation of a tool that provides a graphical interface which guides the user through the DB fragmentation and validates his choices. The final product was a set of Structured Query Language (SQL) scripts automatically generated for each site from the initial configuration.

Unfortunately, these proposals remain static and do not take into account the evolution of the number of sites that occurs throughout the Database (DB) lifecycle.

In this paper, we propose a new incremental implementation approach of DDB taking into account the evolution of the number of the DB sites. This approach should allow 1) an incremental design of the DB taking into consideration the addition and/or deletion of a site, the fragmentation concepts and the duplication of data, 2) the updating of the various links between the sites, and 3) an automatic generation of DDL script for each site as well as Procedural Language (PL) / SQL procedures and the necessary triggers for the update and the verification of the DDB integrity constraints. This approach has been validated by an implementation of an intelligent layer under the Oracle DDBMS.

This paper is organized as follows: Section 2 presents some basic concepts of DDB. Section 3 presents an example of a DDB implementation, thus illustrating the problems and limitations of already existing DDBMS. Section 4 presents our motivation for this work. Section 5 presents a description of our proposed approach. Section 6 presents our implemented tool, Intelligent-Incremental-DDB. We end with a conclusion and some perspectives.

II. BASIC CONCEPTS

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network [4].

A Distributed Database Management System (DDBMS) is a software system that manages a set of databases which are physically distributed but logically connected and which provides the necessary means of access to ensure a transparent distribution [5].

In particular, a DDBMS must ensure a continuous functioning. Indeed, the need for a planned system shutdown should never be felt, even for some operations such as site adding or site deleting, or else the dynamic creation or deletion of fragments in one or more sites.

We cite as examples: Oracle 11g [6], Cassandra [7], Informix [8], INGRES [9], among the top DDBMS ranked in the market.

Distributed database design must take into account the number of sites on the distribution. It is based especially on the following concepts [10]:

- **Fragmentation:** Fragmentation is the process of the decomposition of a database into a set of sub databases. This decomposition should be with no loss of information [11]. We distinguish three

 ISBN: 978-1-61208-408-4

types of fragmentation: 1) *horizontal fragmentation*: It consists of dividing the relations into sub relations obtained through the selection of tuples in a table according to a specific criterion. The reconstruction of the relations is defined by the union of the fragments. 2) *Vertical Fragmentation*: Each fragment represents a subset of relationship attributes. The primary key must be maintained in each fragment. The reconstruction of the relations is defined by join. 3) *Mixed fragmentation*: It results from the successive application of horizontal and vertical fragmentation operations on a global relation.

- **Replication:** the replication of a database is the reproduction of a subset of the main database on remote sites.
- **Data allocation:** is the allocation of fragments to different sites depending on the origin of the queries that have been used during the fragmentation process.

III. EXAMPLE OF A DDB IMPLEMENTATION

In this section, our goal is to describe, through an example, the necessary process to implement a DDB on a number of initial sites, then show the necessary changes that have to be made by the DDB designer following the addition of a new site.

Consider the database described by the following global schema:

Client (ID-cl, Name, Address, City, Business_Sales, Rate_Reduction)
Command (Num-c, Date_c, # ID-cl, Delivery)

We propose first to distribute this DB on two sites: Tunis and Sousse. This distribution is shown in Figure 1.

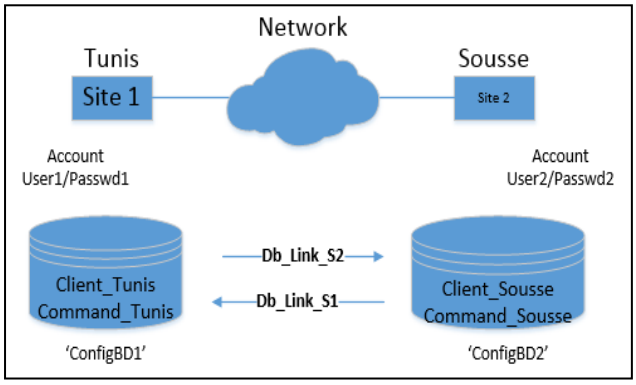


Figure 1. DB spread over two sites

Then, following the user needs, we propose to add a third site: Sfax, as it is illustrated in Figure 2.

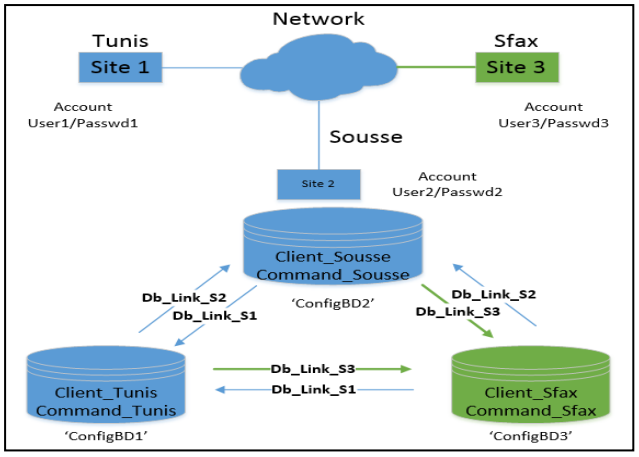


Figure 2. DB spread over three sites

In this section, we propose to use Oracle 11g as DDBMS to implement our DDB.

A. *Allocation mechanism in Oracle*

Like any commercial DDBMS, Oracle does not accept the distributed allocation mechanism, although the administrator can manually allocate DB data to produce similar results. This has the effect of shifting the responsibility under the auspices of the end user, who must know that a table has been fragmented and that he can convert this knowledge into the application. In other words, the Oracle DDBMS does not ensure transparency of the distribution, while it allows location transparency [8].

In order to ensure transparency, the designer must stick to the following steps:

- User accounts creation among sites.
- Bi-directional links creation between different sites (using oracle CREATE DBLINK command).
- Local schema implementation on each site
- Synonyms definition to ensure location transparency.
- View and/or materialized views creation and/or snapshots to ensure fragmentation independent schema. On each materialized view and snapshot definition, we have to specify update mode (asynchronous, synchronous) and refresh delay in accordance with the application need .
- Stored procedure definition, as PL/SQL script, on each update operation in a way to make data fragmentation and duplication automated and transparent.
- As a DBMS can ensure only local data integrity, the designer must define PL/SQL triggers that allow checking distributed data integrity among DDB.

B. *DB implementation at two sites*

The implementation of this database will be as follows (Figure 3):

<u>For site1 :Tunis</u>
DDL_Site1.sql

 ISBN: 978-1-61208-408-4

```
----------DDL FOR DATABASE LINK ----------
-------------------------------------------------------------------
create public database link DB_Link_S2
connect to user2 identified by passwd2
using 'ConfigBD2';
------------------ DDL FOR TABLES ------------------
-------------------------------------------------------------------
create table Client_Tunis ( ID-cl number (4), Name varchar2(10), First_Name varchar2(10), Adress varchar2(20), City varchar2(10), Business_Sales number (10,3), Rate_Reduction number (4,2), Constraint PK11 primary key (ID-cl) );
---------------------------
create table Command_Tunis ( Num_c number (4), Date_c Date, ID_cl number (4), Delivery char chek (Delivery in ('O','N') ,constraint PK12 primary key (Num_c), constraint FK1 foreign key (ID_cl) references Client_Tunis(ID-cl));
------------------- DDL FOR SYNONYMS --------------
-------------------------------------------------------------------
create public synonym Client_Sousse for Client_Sousse@DB_Link_S2;
---------------------------
create public synonym Command_Sousse for Command_Sousse@DB_Link_S2;
---------------- DDL FOR VIEW ---------------------
-------------------------------------------------------------------
create view Client
  as
 (select * from Client_Tunis)
 union
 (select * from Client_Sousse);
---------------------------
create view Command
  as
 (select * from Command_Tunis)
 union
 (select * from  Command_Sousse);
-------DDL FOR INSERT_CLIENT PROCEDURE ---
-------------------------------------------------------------------
Create or replace Procedure insert_client(idc number, namec varchar2, fnamec varchar2, addressc varchar2, cityc varchar2, bsc number, rrc number)
is
begin
   if (cityc='TUNIS') then
        insert into Client_Tunis values(idc, namec, fnamec , addressc, cityc, bsc, rrc);
    elsif (cityc='SOUSSE') then
        insert into Client_Sousse values(idc, namec, fnamec , addressc, cityc, bsc, rrc);
   else DBMS_OUTPUT.put_line('The client city must be either Tunis or Sousse') ;
end if ;
commit ;
end ;
```

Figure 3. Part of DDL_Site1.sql

For the site 2: Sousse: we follow the same principle of Site1

C. DB implementation at three sites

Now, suppose that we add a new site 'Sfax'. Such a change will result in various modifications in the generated scripts.

To involve the new site added, several changes will be applied in different sites. For example, one of these will be the updating of the site 1 Data Definition Language (DDL) script, which includes changes at three levels (Figure 4) :

- Views
- Synonyms
- The PL / SQL procedures

```
create view Client
as
    (select * from Client_Tunis)
  union
 (select * from Client_Sousse)
 union
  (select * from Client_sfax);
-------------------------------------------------------------------
create public synonym Client_Sousse for Client_Sousse@DB_Link_S2;
--------------------
create public synonym Client_Sfax for Client_Sfax@DB_Link_S3;
--------------------
create public synonym Command_Sousse for Command_Sousse@DB_Link_S2;
--------------------
create public synonym Command_Sfax for Command_Sfax@DB_Link_S3;
 -------------------------------------------------------------------
Create or replace Procedure insert_client(idc number, namec varchar2, fnamec varchar2, addressc varchar2, cityc varchar2, bsc number, rrc number)
is
begin
   if (cityc='TUNIS') then
      insert into Client_Tunis values(idc, namec, fnamec , addressc, cityc, bsc, rrc);
    elsif (cityc='SOUSSE') then
      insert into Client_Sousse values(idc, namec, fnamec , addressc, cityc, bsc, rrc);
   elsif (cityc='SFAX') then
      insert into Client_Sfax values(idc, namec, fnamec , addressc, cityc, bsc, rrc);
   else DBMS_OUTPUT.put_line('The client city must be either Tunis or Sousse or Sfax') ;
end if ;
commit ;
end ;
```

Figure 4. Part of DDL_Site1.sql automatically updated after adding a new site

IV. MOTIVATION

As shown previously, designers are still facing issues when dealing with Distributed DBs:

- The process of implementing a DDB is still a quite tedious task and time consuming even for a

fixed number of sites. The variety of scripts to generate, the size of the database, and the number of sites are factors that can rise the complexity of the exercise.

- Incremental implementation of a DDB with a variable number of sites (addition or deletion) is very delicate and error-prone mission. In fact, multiple updates must be applied in order to ensure data coherence, fragment validation and other critical constraints. In addition, such a modification will affect the initial design every time a site is added or removed.

Several studies have been conducted in this context. As examples, we can cite the work of Abdalla [1] and Moussa [2] who have proposed a support system for the design of DDB. We can also mention the work of Hassen and grissa [3], who has proposed a new aid approach for the DDB implementation. This approach has been validated by the design and implementation of a tool that provides a graphical interface which guides the user through the DB fragmentation and validates his choices.

Unfortunately, these contributions still static; they are designed for a fixed number of sites, so they do not offer a solution that supports dynamic design while adding, updating or deleting sites. They are also limited to the design and implementation of a DDB but do not offer a solution for updating this DDB through adding or deleting fragments.

We can mention here the work of Hsu [12], who explain the concepts of the Rensselaer Polytechnic Institute Metadatabase System while discussing a single approach to the integration problem. This contribution answers one part of the matter but it is limited for the Metadatabases. In addition, it is an integration approach of heterogeneous data application while we are interested in distribution approaches for DDBs.

What characterizes our work also is the support for fragmentation aspect, where each fragment is hosted in a remote DB. Such a feature introduces more complexity to the procedure. Not only it supports variable number of sites, but also, sychronizes scattered fragments in remote sites which are bound by some integrity rules.

In the following, we propose a new architecture of the DDBMS that supports an incremental implementation. This consists in garnishing all DDBMS with an intelligent layer that offers: 1) a graphical interface for an incremental definition of the different sites in the DDB, 2) an incremental design approach to DDB while knowing fragmentation attributes and 3) an automatic update of the scripts in the different sites. To validate our approach, we have used as an example of DDBMS, the Oracle DDBMS.

V. NEW APPROACH PROPOSALS FOR ORACLE DDBMS

A. The Approach Specification

To validate our approach, the new layer must ensure the followings:

- Verification of distributed data integrity: for the centralized DB, the existing DBMS validate the integrity constraint verification, but the problem occurs in case they check them for the distributed DB.
- Dynamic design of the database: adding / removing a site entails the integration / suppression of a local schema, which leads to the modification of the global schema.
- Updating scripts of:
  - Views
  - Synonyms
  - Stored procedures
  - Triggers
  - Allocation of data in remote databases

B. Suggested layer architecture

The architecture of our application is illustrated in Figure 5. The graphic interface provides an easy method to interact with the users who, in turn, may interact easily with different modules:

- A validation module, which informs the user if his operation is true or false, is implemented to help the less experienced user not to make mistakes when handling complex schema. In reality, this layer is implemented on two levels:
  - **Validate fragmentation:** check if the fragmentation process respects the three criteria: Reconstruction, Completeness and Disjointness.
  - **Validate data integrity:** check distributed data integrity across the remote databases.
- The site management module includes: update bases schemas, update scripts of (views, synonyms, triggers, data allocation), update database links.
- The script generation module allows the user to consult the SQL script of all transactions that took place during the database implementation process.

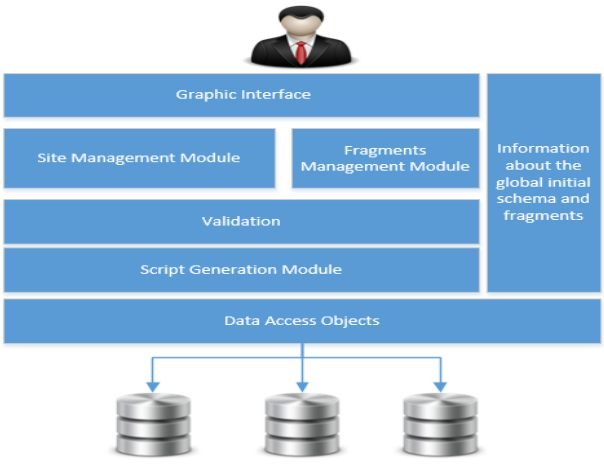


Figure 5. Layer architecture

C. Incremental implementation procedure

In the following (Figure 6), we present the procedure we have implemented for the addition of a new site:

Adding site principle

```
BEGIN
{
s = The site that we wish to add;
F = {f1, f2,.., fn} list of fragments //fi may be a horizontal, vertical, hybrid or dulicate fragment in a remote specific site;
1. Enter the list of configuration parameters of this site s (database link, site name, IP address ,remote database SID, login, password);
2. Generate scripts for creation of database links according to s;
3. Attribute fragments to the site s;
3.1 Retrieve the list of relationships based on initial schema ;
3.2 WHILE " Complete Fragmentation " is false
{
FOR each table from the centralized database
{
// Choose the type of fragmentation
IF horizontal fragmentation
THEN {
Select column fragmentation;
Affect the value of fragmentation ; }
ELSE IF vertical fragmentation
THEN Selected the columns of the fragment ;
ELSE IF hybrid Fragmentation
THEN Treat the hybrid fragment;
ELSE IF duplication
THEN Duplicate the table;
Validate the fragmentation;
Display the validation report;
IF validation is negative
THEN break;
ELSE
//Generate scripts fragments
generate scripts for creation of fragments;
generate scripts of data allocation;
}
}
4. Run the database link scripts according to s;
5. Generate the relational schema of this site s;
6. Regenerate the new global schema;
7. Add the new fragment to the list of fragments;
8. Add this site s to the list of sites;
9. Check distributed data integrity
9.1. Collect some necessary information: Site list, fragment list, etc;
9.2. Generate triggers scripts ;
9.3 Run the triggers scripts;
10. Generate CRUD procedures;
11. Generate views;
12. FOR each fragment fi according to s
{
  execute script of fragment creation;
// execute script of allocation
{ allocate data to the fragments of s;
  delete the allocated data from the centralized DB; }
}
}
END
```

Figure 6. Adding site principle

Considering the case of a site removing, it requires to delete its fragments, its database links and to update global conceptual schema and triggers. In the following (Figure 7), we present the procedure we have implemented for this operation.

Deleting site principle

```
BEGIN
{
s = The site that we wish to delete;
F = {f1, f2,.., fn} list of fragments; //fi may be an horizental, vertical, hybride or dulicate fragment in a remote specific site
1. FOR each fragment(fi) in the fragments lists(F)
{
IF fi belongs to the current site s
{
a. Delete fi's data
{
        IF fi does not have a primary Key
        THEN delete fi's data;
        IF fi has a primary key
        THEN activate a cascade suppression to delete fi's data(recursive procedures);
}
b. Delete fi from the fragments lists F;
}
}
2. Delete database links which refer to s;
3. Delete s from to the sites lists;
4. Generate the new global schema;
5. Check distributed data integrity
5.1. Collect information to generate scripts of distributed triggers(database link, site name, fragments lists,..);
5.2. Generate updating triggers scripts;
5.3. Execute triggers scripts;
6. Generate new scripts of views end synonyms ;
}
END
```

Figure 7. Deleting site principle

D. Performance analysis

Among the most important performance criterion that users seek today, is the response time. We are primarily interested in our application to the incremental aspect of to the number of sites, so we have analyzed the response time required for the generation of scripts when creating / deleting sites.

- First Scenario: Adding Site
- Second Scenario: Removing site

We varied the number of sites for each scenario from 1 to 60 sites. The results are described through the following figure (Figure 8):

 ISBN: 978-1-61208-408-4

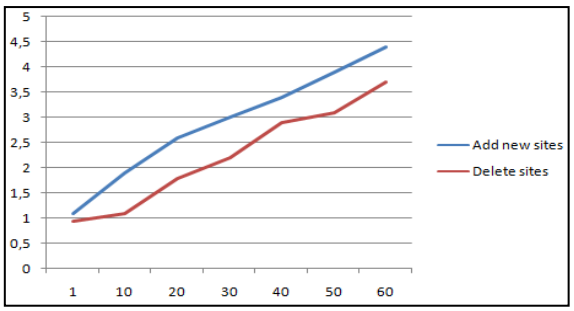


Figure 8. Performance analysis

The results presented in the Figure 8 shows that adding a site consumes more time than deleting a site. This is justified by, while adding a new site, we have more scripts generated for verifying data integrity of new integrated fragments. For each site, our application handles of updating scripts of: views, synonyms, stored procedures, triggers and allocation of data in remote databases. A cascade delete implemented by our solution can also speed up the removal process.

VI. INTELLIGENT-INCREMENTAL-DDB

In this section, we propose to validate our approach. For this we propose a weak coupling with Oracle as an example of DDBMS.

Thus, we used a Windows7 operating system. For the development environment, we have worked with DotNet framework 4.0 (CSharp). We installed virtual machines (Oracle Virtual Box) for the remote databases.

We detail our application operation with the most important interfaces:

Once authenticated, the user is asked to fill in the required coordinates to connect to the centralized DB. If the database is in a remote server, the user must switch to advanced mode to indicate the IP address and the port server (Figure 9).

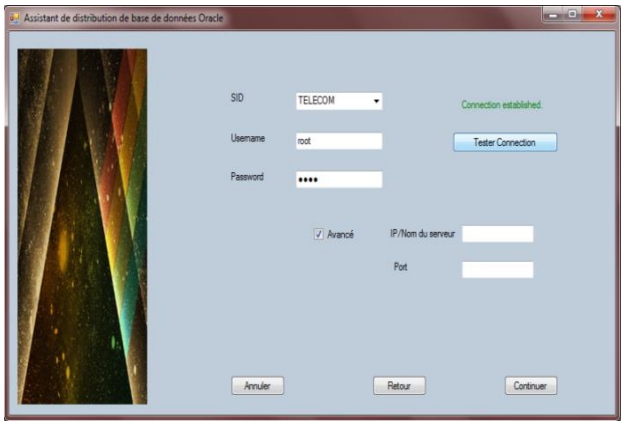


Figure 9. Connection to the centralized database

After a successful connection, the user can benefit from the functionality of the application. He can also refer to the overall relational schema of the centralized database as well as the local schemas of the remote databases (Figure 10).

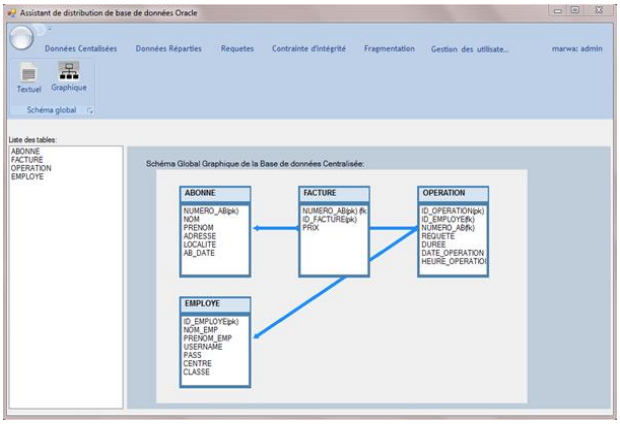


Figure 10. Consult relational schema

Figure 11 shows the site management interface. As an example, we present below the interface of adding a new site. Firstly, the user configures settings, which allow access to remote sites, by indicating (IP address, database link, login, password, SID).

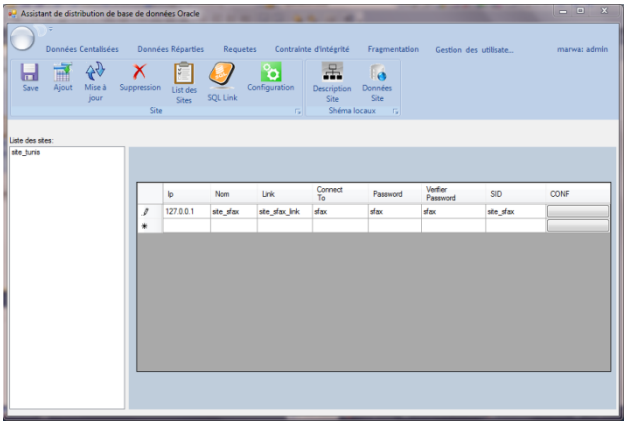


Figure 11. Configuration of parameters access to the remote site

Secondly, the user may add fragments from the centralized tables to the current site. He can also perform a horizontal fragmentation, a vertical fragmentation, or even duplicate an existing table. Figure 12 shows an example of such an operation.

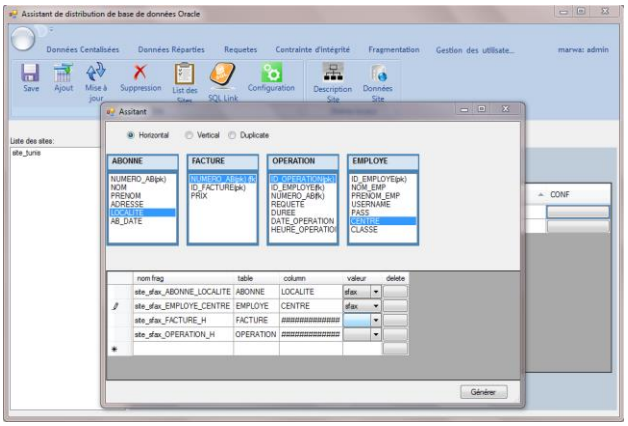


Figure 12. Fragment allocation to the new site

By clicking on the "Generate" button, the script of this operation is displayed; it consists of : a validation report of fragmentation, creation script fragments and the data allocation script. For a valid fragmentation, the user can run the script (Figure 13).

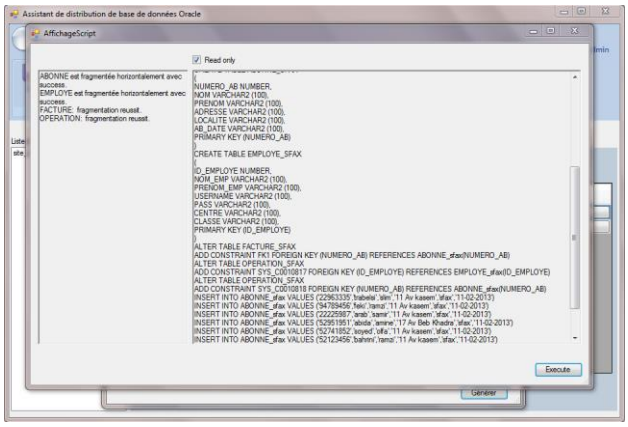


Figure 13. SQL script for the creation of the new site Generation

To check distributed data integrity, our application provides the user with the opportunity to run triggers that take into account the database distribution. He can also consult their SQL script. These triggers are updated automatically when adding, changing, or deleting a site.

In Figure 14, we can visualize an example of an automatically generating script for a deleting trigger (the user just mentions the centralized table and the system detects the tables' fragments and generate the distributed trigger script). The user can choose to activate a cascade delete.

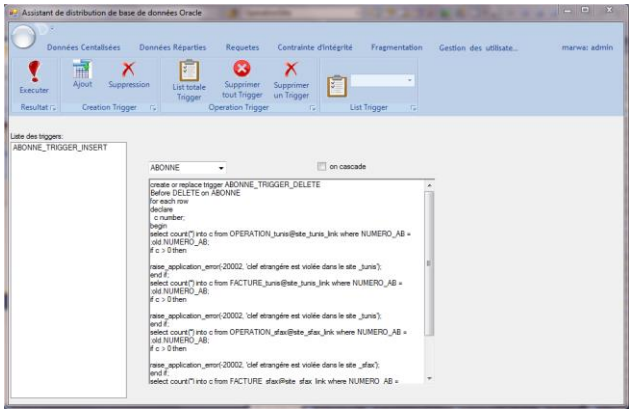


Figure 14. Triggers script generation

In this section, we have detailed our implemented tool with the most important interfaces.

VII. CONCLUSION

In this paper, we proposed an incremental solution for the implementation of DDB in the Oracle DBMS, that takes into account the addition or deletion of a site or the modification of a site by adding or removing fragments. Our approach has been validated by providing a graphical tool, which takes charge of the necessary updates that ensure distributed data integrity, remote access, transparency, allowance, etc.

However, our implementation still needs several improvements such as: 1) the refinement of the script generation algorithms 2) the expansion of the environment supporting our solution, which is currently limited to Oracle technology, by generalizing it to the other DBMS.

REFERENCES

[1] H.I. Abdalla, "A New Data Re-Allocation Model for Distributed Database Systems", International Journal of Database Theory and Application, 2012, Vol. 5, No. 2, pp. 45-60.

[2] R. Moussa, "DDB Expert: A Recommender for Distributed Databases Design", Database and Expert Systems Applications (DEXA), 2011, pp. 534-538.

[3] F. Hassen and A. Grissa Touzi, "Toward a new approach of distributed databases design and implementation assistance", DBKDA, 2014, pp. 104-110.

[4] S. Gupta, K. Saroha, and K. Bhawna, "Fundamental Research of Distributed Database", International Journal of Computer Science and Management Studies, 2011, Vol. 11, Issue 02, pp. 138-146.

[5] M. T. Özsu and P. Valduriez, "Principles of distributed database systems", New York, Springer, 2013.

[6] F. Bouzaiene, "Oracle Golden Gate", Oracle Technologie Day Tunis, March. 2013.

[7] E. Hewitt, "Cassandra: The definitive guide", O'Reilly, 2010.

[8] J. Shute, M. Oancea, S. Ellner, B. Handy, E. Rollins, B.Samwel, R. Vingralek, C. Whipkey, X. Chen, B. Jegerlehner,K. Littlefield, P. Tong, "F1: The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business", SIGMOD conference, 2012, pp.777-778.

[9] M. Stonebraker, "The INGRES Papers: natomy of a relational database system", Addison-Wesley Publishing Compay, 1986.

[10] H. Madi, "Design and implementation of a distributed database under Oracle: For accommodation of university residences", Faculty of Exact Sciences in Algeria, 2009.

[11] J. Guignard, "Design method for DDB", 2004.

[12] C. Hsu, A.Rubenstein , L. Yee, G. Babin, N. Lawson, W. Hofmann., "What Is Rensselaer's Metadatabase System? ", Proc. 3rd International Conference on computer integrated manufacturing, IEEE Computer Society, pp.424-433,1992.

 ISBN: 978-1-61208-408-4

OBDBSearch: A Keyword-based Semantic Search for Databases

Simone Miraglia*, Clarissa Molfino† and Costanza Romano‡
DIBRIS - Università degli Studi di Genova
Genoa, Italy
Email: *simone.miraglia@gmail.com, †clarissa.molfino@gmail.com, ‡costanza.romano.cr@gmail.com

***Abstract*—In this paper, we consider a semantic approach to information retrieval in structured data. We designed an ontology-based database search tool, namely OBDBSEARCH, that exploits ontologies to add semantics to traditional database-style searching. Our goal is to retrieve all database information somehow related to a set of keywords and to have them sorted by relevance. A heuristic method to calculate relevance has been devised. In order to apply and test our algorithm, we considered an application to Key Performance Indicators (KPIs), i.e., quantifiable and strategic measurements for an organization. We observed consistency between experimental results and those expected, even if testing on large amounts of data is still missing.**

Keywords–Keyword-based search; Semantic search; Information retrieval; Ontologies; Database access.

I. INTRODUCTION

Information retrieval concerns with all the activities related to the organization of, processing of, and access to, information of all forms and formats. The objective of an information retrieval system is to enable users to find relevant information from an organized collection of documents [1]. As pointed out in [2], in response to various challenges of providing information access, the field of information retrieval evolved to give principled approaches to search various forms of content. Information retrieval is fast becoming the dominant form of information access, overtaking traditional database-style searching. Since semantic web [3] is a growing field and information retrieval is still evolving - but database-style searching has not been overtaken yet - we sensed the need for a semantic approach to information retrieval in structured data.

This paper addresses the problem of retrieving information using ontologies to add semantics to traditional database-style searching. In particular, we are given a database, an ontology describing the underlying domain of the database and a set of query keywords provided by users. We are interested in retrieving and sorting by relevance all the database records which are related to all the keywords.

We developed an ontology-based database search algorithm, namely OBDBSEARCH, suitable for databases with textual information. It easy to integrate and provides an original heuristic ranking system to sort records. A case study has been devised to apply and test such algorithm, investigating the context of Key Performance Indicators (KPIs), i.e., quantifiable and strategic measurements that reflect an organizations critical success factors [4]. The main contributions of our work are:

- A novel keyword-based semantic search algorithm for databases.
- An easy to integrate and database schema-independent algorithm.
- An innovative heuristic ranking system to sort records by relevance.

This paper is structured as follows. Section II reviews related work, pointing out differences with our contribution. Section III of this paper gives a brief overview of ontologies. Section IV describes our case study, introducing what KPIs are and how our database and ontology have been designed. This should improve the understanding of our algorithm. In Section V, we introduce the problem we are tackling. Modeling and notation formalisation are given in Section VI. Section VII is devoted to the presentation of our algorithm. Section VIII is dedicated to the performance evaluation of our algorithm. Section IX summarizes the results of our work.

II. RELATED WORK

Semantic search, as defined in [5], is a search paradigm that makes use of explicit semantics to solve core search tasks, i.e., to use semantics for interpreting query and data, matching query against data and ranking results. As shown in [5], search is commonly seen as an user-oriented application, which uses user-friendly interfaces instead of complex structured queries. The success of commercial search engines shows that users are more comfortable with keyword-based interfaces.

We focus on approaches which express the information needs as keywords. Several approaches in semantic search are based on translating a given set of query keywords into a set of conjunctive queries, which are evaluated with respect to an underlying knowledge base. Examples of this approach can be found in engines such as Hermes [6], SemSearchPro [7], SPARK [8], AVATAR [9], CIRI [10] or in [11]. The semantic approach we propose differs from the above engines since we aim to retrieve data from databases instead of knowledge bases and no conjunctive queries are formulated. We sense that a semantic approach in database search is missing and our work goes in that direction.

A semantic approach closed to ours is the one proposed in [12]. The authors developed a semantic search engine based on query refinement powered by ontology navigation for traditional information retrieval purposes. The ontology is regarded purely as a taxonomy, where focalization and generalization are used to refine queries. Furthermore, results are ranked by relevance according to standard information retrieval techniques, such as tf/idf, while our approach uses a heuristic method suitable for textual records.

YACOB [13] is a tool that uses domain knowledge in the form of concepts and their relationships for formulating and processing queries. The idea behind this approach is similar to ours.

Substantial work has been done in the field of ontology-based database access [14], such as [15] or [16]. Our work differs from these approaches because databases and ontologies are used togheter to search, but they are clearly separated and database access is performed in a traditional fashion via SQL

 ISBN: 978-1-61208-408-4

queries. This should ease integration with existing databases.

If we left out semantics, our approach would be similar to the one of DBXplorer [17], a system that enables keyword-based search in relational databases. Given a set of keywords, DBXplorer returns all rows, either from single tables or joined tables, such that all the rows contain all the keywords. They introduced a symbol table, which is used at search time to determine the locations of query keywords in the database. Results are ranked by the number of joins involved.

III. ONTOLOGIES

According to [18], ontologies are means to formally model the structure of a certain system we are interested in. From its observation, relevant entities and relations could emerge. Entities are organized in *concepts* and *relationships*. The backbone of an ontology is the generalization/specification hierarchy of concepts, i.e., a taxonomy.

Ontologies could be defined as *formal, explicit specification of a shared conceptualization* [19].

A conceptualization is an *abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly* [20]. In other words, a conceptualization is a list of concepts relevant to the description of our domain. Such conceptualization should be expressed in a computer readable *formal* language, with proper definitions. *Shared* means that several parties should agree with the formal conceptualization. This helps large-scale interoperability.

One of the most popular languages for defining ontologies is OWL 2 (Web Ontology Language Ver. 2) described in [21] and supported by many tools. It points out that there are two main types of relationship, *is-a* and *object property*. The former implements the generalization/specification hierarchy, while the latter allows any other type of relationship between instances of two concepts, such as *has* or *is related to*. One of the many features of [21] is the usage of *Annotation properties*, useful to add literals, i.e., strings, to concepts.

IV. CASE STUDY

Now, we will briefly give an overview of KPIs and then we will explain the design process of both the database and ontology.

A. KPIs background

KPIs are defined as *quantifiable and strategic measurements that reflect an organization's critical success factors*. KPIs are very important for understanding and improving manufacturing performance; both from the lean manufacturing perspective of eliminating waste and from the corporate perspective of achieving strategic goals [4]. Several standards exist, but we focused on just two: ISO 22400-2 [4] and SCOR rev.11 [22]. Both describe KPIs for manufacturing operations management.

In more detail, ISO 22400-2 defines KPIs by a standard schema that contains: formula, time behaviour, unit/dimension, rating, the user group where the KPIs are used and to what production methodology they fit, as shown in [4].

SCOR is a reference model that links business process, metrics, best practices and technology into a unified structure to support communication among supply chain partners and to improve the effectiveness of supply chain [22]. Therefore, metrics are related to processes. Processes are organized in a hierarchical structure with three levels, each one more specialized than the previous. KPIs are also organized in a hierarchical structure with several levels, i.e., KPIs residing at a given level are calculated through KPIs from lower levels. These are the main features of SCOR, others can be found in [22].

B. Database design

In order to design a database to manage KPIs described by different standards, such as SCOR and ISO 22400-2, a common structure for KPIs has been designed. This common structure contains the most important fields of the two standards (KPI Code, Name, Description, Formula, Unit of Measure, Trend and Timing) and a note field for various other information.

Database design is oriented towards SCOR since it has additional features compared with ISO 22400-2. This means that the hierarchical structure of both processes and KPIs has been preserved, although in different ways. Process hierarchy has been directly maintained in our database design, i.e., three tables have been created, one for each hierarchical level. Conversely, since KPI hierarchy means that an upper level KPI is calculated through lower levels KPIs, we decided to have a single table with a recursive relationship to ensure links between KPIs.

Regardless of their level, KPIs can refer to processes of any level. Instead of linking the KPI table with all processes tables, for the sake of simplicity, we decided to link just KPIs and third level processes. In this way, we would preclude the possibility of linking first and second level processes to corresponding KPIs. To ensure that first and second level processes are linked to KPIs anyway, we introduced *default processes*. They are factitious processes which stand for corresponding processes of superior levels. This way, instead of linking KPIs with first or second level processes, we link KPIs to third level default processes.

Figure 1 shows the E/R diagram of our database, pointing out our design choices. We will now describe all our entities.

`standards` - this table allows us to consider several standards of KPIs, with the ID, name and description.

`performances` - this table contains performances, with their ID, name and description.

`kpi` - this table keeps track of KPIs, with their name, ID, formula, description, unit of measure, trend, code, timing, standard and performances.

`kpi_hierarchy` - this table shows the recursive relationship that allows us to define the KPI-hierarchy, listing the IDs of KPI-parents and KPI-sons

`processes_lv1` - this table keeps track of processes of the first level, with their name and description.

`processes_lv2` - this table keeps track of processes of the second level, with their name and description and the IDs of the processes of the first level. There are also first level default processes.

`processes_lv3` - this table keeps track of processes of the third level, with their name and description and first and second level default processes.

`kpi_processes` - this table realises the many-to-many

 ISBN: 978-1-61208-408-4

relationship between KPIs and processes of third level. It associates a KPI to a process, maintaining their IDs.

`input_processes` - Each process produces documents to be consumed by another one, i.e., a workflow exists. That means that every process becomes input (output) process for other processes. This table keeps track of input processes, with their IDs, documents and IDs of the processes they are input for.

`output_processes` - This table keeps track of output processes, with their IDs and IDs of the processes they are output for.

C. Ontology for KPIs

Afterwards, an ontology has been implemented in order to represent the domain of KPIs in a supply chain context, considering also the structure of the database. OWL 2 [21] has been used to define the ontology.

Figure 2 shows the ontology, pointing out the structure of KPIs domain. Obviously, it does not represent the whole KPIs domain, it is just a small example. Now, we will describe it in more detail.

KPI - this entity refers to the concept of KPI.

Process - this concept refers to supply chain processes. Processes are linked to KPIs and thus on the ontology we have a connection between concepts related to them.

Plan, *Source*, *Make*, *Deliver*, *Enable*, *Return* - these entities are related to the main processes individuated by the SCOR standard [22]. Therefore, they are linked to processes by an *is-a* relationship.

Sphere - this concept refers to the scope of KPIs application. It is linked to KPIs by a *has* relationship.

Time, *Cost*, *Quality*, *Energy* - these entities are related to four different scopes. They are useful because some indicators mainly refer to costs, whereas others point out timing aspects of supply chain and so on. They are linked to *Sphere* by an *is-a* relationship.

Industry - this concept refers to the industrial sector of KPIs.

Continuous, *Discrete* - these entities are related to the two types of industrial process. They are linked to *Industry* by an *is-a* relationship.

Textile - this concept refers to a certain kind of discrete industry and is linked to *Discrete* by an *is-a* relationship.

Pharma, *Alimentary* - these concepts refer to two types of continuous industrial process and are linked to *Continuous* by an *is-a* relationship.

Flour, *Food&Beverages* - these entities are related to two possible types of food industry and are linked to *Alimentary* by an *is-a* relationship.

As shown later in Section V, synonyms need to be managed. In the ontology, for the sake of simplicity, we keep track of synonyms by means of *Annotation properties*, not listed in Figure 2.

V. The problem

The problem we are addressing is the retrieval of information from a database given a set of keywords. In other words, we want to return a list of information related to *all* those keywords. Furthermore, it is desirable to have such information sorted by relevance. Our algorithm needs an existing database - preferably with textual information - and an ontology describing the underlying domain.

Our aim was to develop a schema-independent algorithm, both from database and ontology point of view. Furthermore, we expect the algorithm to navigate between tables and concepts of the ontology. Those issues reveal a need for abstraction, which is addressed in Section VI.

In order to devise a *flexible* search algorithm, synonyms need to be managed. In fact, a search keyword might not appear in the database as it is, but as one of its synonyms: if they were not tracked, these results would be lost. We can take the keyword *KPI* as an example: this word can be found in the database as it is or perhaps as *Performance Indicator*.

Since our results should be listed sorted by relevance, we should define what relevance means in order to be able to discriminate whether a certain result is more or less relevant than another one. In Section VII-D, we give a thorough description of the issue, devising a solution.

VI. Modeling and notation

In this section, we want to introduce some notation, useful to describe the algorithm.

We can easily think of a database as a graph, since it is none other than a set of tables linked by relationships. This is true assuming it is well normalized, i.e., all many-to-many relationships between a certain table A and a table B have been transformed with two one-to-many relationships from A to C and from B to C, where C is a newly created table storing at least both primary keys of A and B [23].

Database metadata contain information regarding the structure of the database, i.e., tables, fields, foreign keys constraints. Since they can be queried, we can get all information needed to build straightforwardly the graph. This solves the problem of having a schema-independent database because, once the graph is created, we will only reason in terms of tables and links, forgetting about the actual schema.

A similar approach is also applicable to ontologies, since they are basically a set of concepts linked by relationships. No particular assumptions have to be made to the ontology structure. We can also individuate two types of edges/relationships, i.e., is-a relationships and object properties. Building a graph is easy, since state-of-art programming libraries provide access to ontologies, such as Apache Jena.

A graph structure is useful for navigation between nodes, both database tables and ontology concepts, thanks to many accomplished algorithms, such as Depth First Search (DFS) and Breadth First Search (BFS)[24].

Therefore, we can define a database $\mathcal{DB}$ and an ontology $\mathcal{O}$ as

$$\mathcal{DB} := (\mathcal{T}, \mathcal{L}) \quad \mathcal{O} := (\mathcal{C}, \mathcal{E}) \tag{1}$$

where $\mathcal{T}$ is a set of tables and $\mathcal{L}$ a set of links between tables, while $\mathcal{C}$ is a set of concepts and $\mathcal{E}$ a set of relations among concepts. $\mathcal{E}$ can be subdivided in two disjoint sets $\mathcal{H}$ - is-a relationships - and $\mathcal{P}$ - object properties.

A link $l \in \mathcal{L}$ is defined just as pair of tables and a unique name. It is a matter of fact that multiple links between the same pair of tables could exist. As shown in Figure 1, `kpi_hierarchy` and `kpi` are connected twice. The unique name is useful to discriminate links between the same pair of

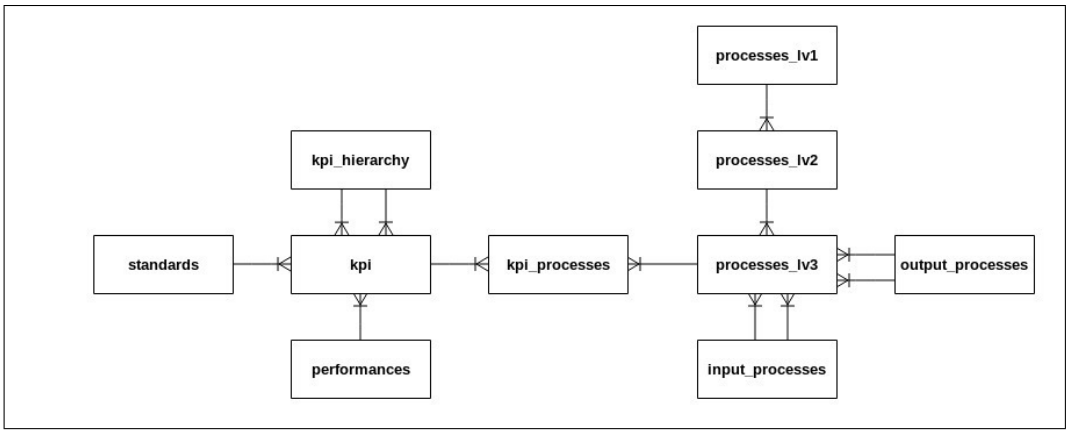


Figure 1. E/R diagram for KPIs. Boxes are entities, i.e., database tables, while lines between entities represent one-to-many relationships. Trebled end point indicates where the *many* is.

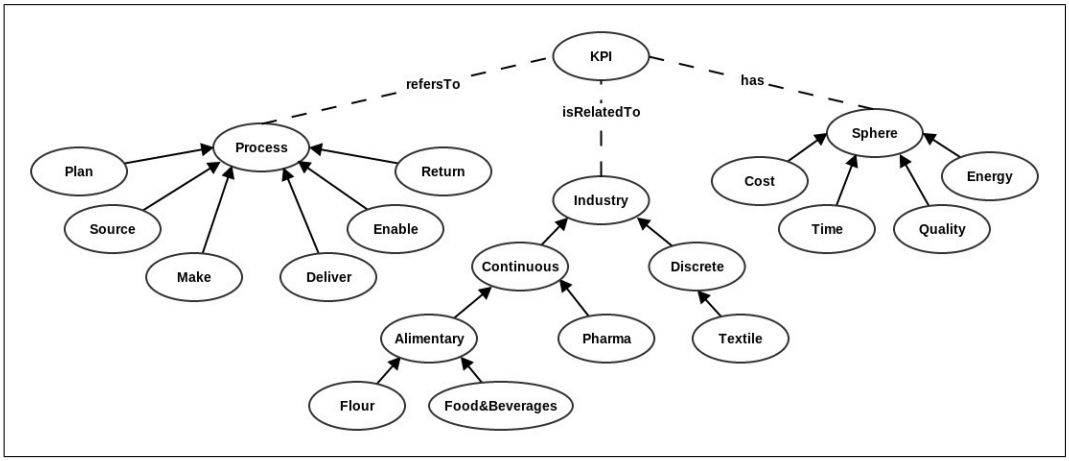


Figure 2. Designed ontology for the KPI context. Arrow lines refer to is-a relationships, e.g., *Plan* is-a *Process*, while dotted lines indicate object properties.

tables.

$$l := (\text{name}, t_1, t_2) \tag{2}$$

Let $t \in \mathcal{T}$ be a table, defined as follows

$$t := (\text{name}, R) \tag{3}$$

where name is table name and R a set of records.

We use a function GETLINKS (t): given a table, returns a set of links involving t. In addition, we define another function GETLINKEDTABLE (l, t): given a table and a link containing that table, returns the linked table.

A record $r \in \mathcal{R}$ is defined as follows

$$r := (\mathcal{F}) \tag{4}$$

where $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of fields.

A field f_i can be defined as

$$f_i := (\text{name}, \text{type}, \text{key}, \text{value}) \tag{5}$$

where name and type are trivial to understand, key indicates whether f_i is a primary key field or not and value is the actual value of the field.

We can further individuate a subset $\mathcal{LT} \subset \mathcal{T}$ containing all tables added to the database while transforming all many-to-many relationships into two one-to-many relationships.

Similarly, let $c \in \mathcal{C}$ be a concept, defined as follows

$$c := (\text{name}, \mathcal{S}) \tag{6}$$

where name is concept name and $\mathcal{S} = \{s_1, \ldots, s_k\}$ a set of synonyms.

We use a function GETNODE($\mathcal{O}$, name): given an ontology and a name, returns a concept c such that c.name is equal to name.

A relationship $e \in \mathcal{E}$ is defined just as a pair of concepts

$$e := (c_1, c_2) \tag{7}$$

Furthermore, we can use three functions GETPARENTS(c), GETCHILDREN(c) and GETHORIZLINKS(c): given a concept, they return a set of parent concepts, a set of children concepts and a set of concepts linked to c by object properties, respectively.

VII. THE ALGORITHM

Given a set of keywords $\mathcal{K}$, our goal is to retrieve a list of *results* relevant to all $k_i \in \mathcal{K}$ and have this list sorted by relevance. For the sake of simplicity, we assume that a *valid keyword* k must be a name of any concept of the ontology,

 ISBN: 978-1-61208-408-4

not one of the synonyms. In this section, we discuss how we achieved such goal.

First, we define a *result* as a record $r \in \mathcal{R}$ of any tables of $\mathcal{DB}$ such that is *relevant* to a certain keyword. We are looking for all those results relevant to all $k_i \in \mathcal{K}$. We further discuss in depth the relevance, but now say that a record is relevant to a certain keyword k if k or one of its synonyms is somehow contained in at least one of the fields of the record. This means that we should only consider fields $f_i : f.\text{type} \in \{\text{varchar}, \text{text}\}$.

More formally, given a table $t \in \mathcal{T}$, a record r and a concept c linked to k, we say that r is relevant to k iff $\exists\, f_i \in \mathcal{R}$ s.t.

$$k \in f_i.value \ \vee\ \exists\, s \in c.\mathcal{S} \ :\ s \in f_i.value \tag{8}$$

Let RS_i be the set of *heterogeneous* results related only to k_i. We say *heterogeneous* since two results can come from two different tables and thus their structure be different.

One of the cornerstones of our algorithm is the following lemma

Lemma 1. *A result r is relevant both to k_i and k_j iff $r \in RS_i \wedge r \in RS_j$*

This gives us a way to design our algorithm. Precisely, we can find a list of results relevant to all $k_i \in \mathcal{K}$ by finding all RS_i for each k_i separately and then intersect all those result sets. Let RS be the final set of results given by

$$RS = RS_1 \cap RS_2 \cap \cdots \cap RS_{|\mathcal{K}|} \tag{9}$$

Since it is a heterogeneous set, we assign an unique identifier to each result in order to perform intersection and union.

From now on we focus on the problem of finding all the results relevant to a single keyword k.

We describe the algorithm starting from an example. Suppose our keyword is *Textile*. As shown from the ontology of Figure 2, *Textile* is a *Discrete* type of *Industry* for *Kpi*. This means that we are looking for all KPIs related to textile industry, and linked information. The keyword gives us hints where to look, i.e., in which table (or tables) to start our search. In fact, we are primarily looking for KPIs with certain features, not for processes or anything else.

Once where to start searching has been found, we have to actually search. Given the keyword, we fetch all the relevant records. It is true that we are looking for *Textile*, but we need to look also for its synonyms. In fact, there could be relevant records that will not be added just because the description uses the word *Clothing* or *Fabric* instead of *Textile*. This shows how synonyms have to be taken into account when fetching records.

Suppose we fetched all records relevant to the keyword and its synonyms. We might be interested in fetching records relevant to generalizations or specifications of the concept linked to the keyword. For instance, we are also interested in records related to *Discrete* and its synonyms, since it is a more general concept of *Textile*. Loosely speaking, the records directly related to the keyword are more relevant than those related to generalizations or specifications.

It is also valuable to find all information connected to fetched records. For instance, if we had a certain KPI relevant to *Textile*, we would like to have all processes related. This could be achieved by moving from one table to another exploiting the database graph.

We will describe the general idea of algorithm and its main issues. Principal loop of the algorithm is shown in Figure 3. We will now describe in detail the main functions of our algorithm.

Figure 3. Ontology-based database search algorithm. $\mathcal{K}$ is a set of keywords.

```
procedure OBDBSEARCH(K)
    for all k_i ∈ K do
        table-list ← SEARCHSTARTINGTABLES(k_i)
        for all t ∈ table-list do
            RS_i ← RS_i ∪ SEARCH(k_i, t)
    return ⋂_i RS_i
```

A. Search starting tables

As we said in Section VII, the first step is to understand from which table (or tables) we should start our search. This is a key concept because it shows us one of the benefits of using an ontology. Naively, we could use a *brute-force* approach by looking for the keyword in every table of the database. Besides efficiency issues, many unrelated records could be found. In fact, suppose we are looking for *Cost*. Since *Cost* is a feature of *KPI*, we should initially search the word *Cost* into the `kpi` table. If we also looked for the word *Cost* in other tables, such as `processes_lv3` or `standards`, we would find records containing *Cost*, but they would be semantically unrelated. Indeed, a *brute-force* approach would go against a semantic direction.

Ontologies should be exploited in order to understand if a keyword can be a feature or a specification of another concept - meaning there is a sort of container - and thus we expect a database table related with such concept to exists. This way we add semantics to the search.

A modified BFS on the ontology graph is executed in order to find a set of starting tables. We start from the concept directly related to the given keyword. Then we check if there is correspondence between a database table and such concept. If no table is found, we should expand from the main concept to reach its neighbours and, for each neighbour, check the existence of a table with a similar name. Intuitively, if we are looking for a sort of container, there will be no need to expand downwards, i.e., towards specifications of the main concept. We stop this BFS once a table matching with a concept has been found. From the set of candidate tables we should keep out all tables $t \in \mathcal{LT}$, i.e., tables which perform many-to-many relationships. In fact, they are likely to be just a set of primary keys pair, thus quite few textual information could be inferred.

There could be more than one matching table. Suppose we are looking for *Plan*. Since there is no corresponding table, we will reach *Process*. There are at least three matching tables: `processes_lv3`, `processes_lv2` and `processes_lv1` (since `kpi_processes` has been kept out). Each one has to be considered.

We should also deal with the special case in which a matching table is found at the first round. Suppose we are looking for *KPI*. We will find a match with `kpi` table without any other expansions. In that case, fetching all the records connected to the keyword *KPI* from `kpi` table would be quite wrong. Hence, we would fetch all records. Since actual fetching is

 ISBN: 978-1-61208-408-4

performed by a FETCH function in Figure 6, we should be able to discriminate whether to get all the records or not. So, we suppose to have two functions - SETFETCHALL(table) and ISFETCHALL(table) - to indicate whether to fetch all the records from a certain table. The last function will be used inside FETCH.

The algorithm in Figure 4 recaps above features.

Figure 4. Starting tables search algorithm. k_i is the current keyword. $\mathcal{O}$ and $\mathcal{DB}$ are assumed to be global variables.

```
procedure SEARCHSTARTINGTABLES(k_i)
    tables ← ∅
    queue Q ← ∅
    tableFound ← false
    firstTime ← true

    ENQUEUE(Q, GETNODE(O, k_i))
    while Q ≠ ∅ ∧ ¬ tableFound do
        c ← DEQUEUE(Q)
        SETVISITED(c)
        KS ← c.name ∪ c.S
        for all t ∈ DB.T ∧ t ∉ LT do
            for all s ∈ KS do
                if s ∈ t.name then
                    tableFound ← true
                    if firstTime then
                        SETFETCHALL(t)
                    tables ← tables ∪{t}

        if ¬ tableFound then
            for all v ∈ GETPARENTS(c) ∪ GETHORIZLINKS(c) do
                if ¬ ISVISITED(v) then
                    ENQUEUE(Q, v)
        firstTime ← false
    return tables
```

B. Search concepts

Once the starting tables have been determined, we must search the keyword and its synonyms inside those tables. This is accomplished by the algorithm in Figure 5. As we said before in Section VII, the ontology can also be exploited to search generalizations/specifications of the main concept. Those results should be marked as less relevant than those directly connected to the main concept.

Since we are talking about concepts that are more or less pertinent, the problem of results relevance starts to take shape. This is completely clarified in Section VII-D, but we will start to outline the matter in order to explain this step of the algorithm.

Given a keyword, we can get its related concept and thus actually get results by searching that keyword in the current table - one of the starting tables. See algorithm in Figure 6. With the purpose of getting information related to generalizations or specifications of the principal concept, we perform a modified BFS on the ontology, similar to the one in Figure 4.

Concepts exploration is only upwards, towards generalizations. Downwards exploration, i.e., towards specifications, is useful only when starting from the main concept. In fact, suppose k = *Alimentary*. We can explore upwards and search *Continuous*. Since *Alimentary* is the main concept, we can also go downwards and search both *Flour* and *Food & Beverages*. This would be correct, because we would be looking for specifications of the main concept. On the contrary, if we explored downwards from *Continuous* towards *Pharma*, we would be wrong because *Pharma* is quite not linked with *Alimentary*. We assume we have two functions SETVISITCHILDREN(node) and VISITCHILDREN(node) that allow us to discriminate whether children of node must be visited or not.

In order to measure relevance, and thus differentiate concepts closer from those farther, we assign to each concept a starting ranking. Intuitively, concepts closer will have a starting ranking greater than those farther. How this works is shown in detail in Section VII-D.

Figure 5. Keyword search algorithm. k_i is current the keyword and t a database table in which search k_i. $\mathcal{O}$ is assumed to be a global variable.

```
procedure SEARCH(k_i, t)
    RS'_i ← ∅
    queue Q ← ∅
    firstTime ← true

    ENQUEUE (Q, GETNODE(O, k_i), baseRanking)
    while Q ≠ ∅ do
        (c, w) ← DEQUEUE(Q)
        SETVISITED(c)
        KS ← c.name ∪ c.S
        RS'_i ← RS'_i ∪ SEARCHTABLE(KS, t, w)

        if firstTime then
            SETVISITCHILDREN(c)
        if VISITCHILDREN(c) then
            for all u ∈ GETCHILDREN(c) do
                SETVISITCHILDREN(u)
            nodesToVisit ← GETCHILDREN(c) ∪ GETPARENTS(c)
        else
            nodesToVisit ← GETPARENTS(c)

        for all v ∈ nodesToVisit do
            if ¬ ISVISITED(v) then
                ENQUEUE(Q, v, w/10)
        firstTime ← false
    return RS'_i
```

C. Search table and linked records

Suppose we now have found a keyword to search - regardless of whether it is a principal keyword or another concept - and we want to actually fetch records from a certain database table. This is accomplished by the FETCH function from the algorithm in Figure 6, which is essentially a SQL Select statement with a Like condition. In more detail, we select all the records from the given table, checking if at least one varchar or text type field contains the keyword or one of its synonyms.

We should determine and attribute relevance to all records returned from the FETCH function. The ADDRESULTS function takes care of this issue. Record relevance in connection with a certain keyword is explained in Section VII-D, but, roughly speaking, it depends on the number of occurences of a certain word and in which type of field it occurs.

 ISBN: 978-1-61208-408-4

Figure 6. Search table algorithm. KS is a set of string composed by the keyword and its synonyms, t is the table in which search and w is the starting ranking.

procedure SEARCHTABLE(KS, t, w)
  $R \leftarrow$ FETCH(t, KS)
  $RS'_i \leftarrow$ ADDRESULTS(t, KS, R, w)
  for all $r \in R$ **do**
    $RS'_i \leftarrow RS'_i \cup$ LIMDFS($0, w, KS, t, l, r$)
  return RS'_i

So, we have a set of record R relevant to the keyword, but related just to the starting table. As we said, it is also valuable to get all records linked with those in R. In fact, we might be interested in the processes related to a given set of KPIs. The database graph could be exploited in order to get linked results. As a matter of fact, we should be able to move from the current table to those directly connected and, for each $r \in R$, execute a join between those tables. This process should be iterated to reach more tables. Hence, we launch a DFS from each record $r \in R$, in order to get connected records from linked tables.

This DFS should not visit tables, but relationships instead. It could seem slightly like a nuance, but in truth it is not. In fact, we are interested in getting connected records, i.e., to perform joins between tables. So, starting from a given table t_1, we must perform all possible joins between t_1 and all linked tables. Since join conditions are individuated by foreign key constraints and thus by relationships, we have to visit them instead of tables. Suppose we perform a join between t_1 and t_2, then we must iterate starting from t_2.

If we ran a complete DFS, we would explore all links and get a sort of big join among all database tables. This is not advisable, because we cannot lose sight of our goal. Since we are looking for records somehow *semantically* connected to those in R, we should limit the depth of the expansions. A heuristic maximum depth of two/three tables seems reasonable. Figure 7 shows the limited DFS algorithm. FETCHLINKEDTABLE processes the foreign key constraint and returns all records linked to the one passed.

D. Results relevance

Relevance is measured by observing how much and in which type of attribute the keyword appears in each record found through the FETCH operation. Evidently, records related to principal keyword with a relevant number of occurrences should be listed before records related to a generalization/specification of the keyword or, in the same way, related to the principal keyword but with few occurrences. Therefore, we associate a relevance level - starting ranking - to each record found. This relevance level depends on the level of the searched keyword. Later, this level will be increased or decreased using a bonus/penalty system.

In more detail, let R be the record set returned after the FETCH operation for a certain table $t \in \mathcal{T}$. If $r \in R$, it contains at least in one field a certain keyword k, as shown by Equation 8. A starting ranking sr is assigned to each $r \in R$. Sr will be different whether k is the principal keyword or a linked concept. For example, say $k =$ *Textile* has $sr = 10.000$, while $k =$ *Discrete* will have $sr = 1.000$. Each jump on the ontology graph entails a division by 10.

Every field $f_i \in \mathcal{F}$ gets a bonus/penalty w_i weighted on sr (e.g., $+20\%$ of sr), because we want the ranking to oscillate about the starting value. If f_i is a description field w_i depends on the number of occurrences of k, if instead f_i is a string field, e.g., a name or a formula, w_i is bonus if k is contained, no penalties otherwise. More precisely, we recap bonus/penalties values that we heuristically decided to use in Table I.

Figure 7. Limited Depth First search algorithm. d is current depth, w is the starting ranking, KS is a set of string composed by the keyword and its synonyms, t is the incoming table, l is the link we are visiting, r is the current record to link.

procedure LIMDFS(d, w, KS, t, l, r)
  $RS'_i \leftarrow \emptyset$
  for all $l \in$ GETLINKS(t) **do**
    if $\neg$ ISVISITED(l) **then**
      $RS'_i \leftarrow RS'_i \cup$ VISITDFS($0, w, KS, t, l, r$)
  return RS'_i
procedure VISITDFS(d, w, KS, t, l, r)
  $RS'_i \leftarrow \emptyset$
  if $d \leq$ maxDepth **then**
    $t_2 \leftarrow$ GETLINKEDTABLE(l, t)
    SETVISITED(l)
    $R \leftarrow$ FETCHLINKEDTABLE(r, l)
    $RS'_i \leftarrow$ ADDRESULTS(t_2, KS, R, w)
    for all $r_l \in R$ **do**
      for all $l_2 \in$ GETLINKS(t_2) **do**
        if $\neg$ ISVISITED(l_2) **then**
          $d \leftarrow d + 1$
          return $RS'_i \cup$ VISITDFS(d, w, KS, t_2, l_2, r_l)
  return RS'_i

TABLE I. BONUS/PENALTIES OVERVIEW.

Field type	*Occurrences*	*Bonus/penalty*
String	at least one	$+25\%$ of sr
Description	none	-20% of sr
Description	one	-15% of sr
Description	from 2 to 5	0
Description	more than 5	$+15\%$ of sr

So, the record relevance is given by the sum of the starting ranking and bonus or penalties associated with it. Negative w_i are not added if a string field containing k exists. This avoids penalizing results in which k is contained in the name (so r is very relevant) but description field has few occurrences. More formally, let $\mathcal{G} \subseteq \mathcal{F}$ be the set of string or description fields.

$$\text{Relevance}(r) := sr + \sum_{i:f_i \in \mathcal{G}} \delta(w_i) \tag{10}$$

where $\delta(w_i)$ is a function that returns 0 if $w_i < 0$ and a string field containing k exists, otherwise it returns w_i.

If a record is relevant to two or more keywords, the overall relevance is evaluated as the sum of the single relevances.

VIII. PERFORMANCE EVALUATION

Our algorithm was evaluated considering a set of query, where different combination of keywords have been considered, as shown in Table II. Standard precision, recall and balanced f-measure were computed on the returned results. We recall that

$$\text{Precision} = \frac{\text{Correct results}}{\text{Retrieved results}} \quad \text{Recall} = \frac{\text{Correct results}}{\text{Relevant results}} \tag{11}$$

 ISBN: 978-1-61208-408-4

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (12)$$

In our case, results whose rank is greater than 1.000 are considered as *relevant*. We tested our algorithm on a relatively small size database. Although testing on a larger amount of data is still missing, results are promising, as Table II clearly shows.

TABLE II. PERFORMANCE EVALUATION.

Keywords	*Precision*	*Recall*	*F-measure*
Plan	53.28%	100.00%	69.52%
Cost	80.00%	100.00%	88.89%
Make	31.30%	100.00%	47.68%
Plan, Cost	88.24%	100.00%	93.75%
Plan, Make	60.00%	100.00%	75.00%
Cost, Make	75.76%	100.00%	86.21%
Plan, Cost, Make	93.94%	100.00%	96.88%
Deliver	52.94%	100.00%	69.23%
Source	77.05%	100.00%	87.04%
Time	85.33%	100.00%	92.09%
Deliver, Source	83.48%	100.00%	91.00%
Deliver, Time	90.38%	100.00%	94.95%
Source, Time	96.43%	100.00%	98.18%
Deliver, Source, Time	96.08%	100.00%	98.00%
AVERAGE	76.01%	100.00%	84.89%

IX. CONCLUSION AND FUTURE WORK

In this work, we have devised a novel ontology-based database search algorithm. So, we achieve the objective of retrieving information from a database using a semantic approach. Specifically, given a set of keywords, our goal was to retrieve a list of results relevant to all the keywords and have this list sorted by relevance.

A case study has been examined to apply and test such algorithm. We focused on the context of KPIs. In order to design a database to manage KPIs described by several standards, we have individuated a common structure for KPIs. We also designed an ontology to describe the KPIs domain. Obviously both database and ontology could be extended and designed in many other ways.

We modeled both database and ontology as a graph to solve flexibility and navigation issues. Semantic has been added understanding which tables are relevant and searching not only the principal keyword, but also those linked. Synonyms have been taken into account while searching. In order to measure the relevance, we introduced a heuristic ranking system, that evaluates how much and in which type of attribute the keyword appears in each record found.

Our algorithm requires just an existing database and an ontology describing the underlying domain. From a practical point of view, OBDBSearch is an external tool that could be easily and immediately used with real-world systems to perform semantic searches.

We observed promising experimental results, but testing on a larger amount of data would be fundamental to try out our algorithm performances.

REFERENCES

[1] G. Chowdhury, Introduction to Modern Information Retrieval, 3rd ed., 2010.

[2] C. D. Manning, P. Raghavan, and H. Schütze, An Introduction to Information Retrieval, online ed. Cambridge University Press, Cambridge, England, 2009.

[3] P. Szeredi, G. Lukácsy, and T. Benkő, The Semantic Web Explained: The Technology and Mathematics behind Web 3.0. Cambridge University Press, Cambridge, England, Oct 2014.

[4] ISO-22400-2, "Manufacturing operations management — key performance indicators — part 2: Definitions and descriptions of kpis", Standard ISO 22400-2, December 2012.

[5] T. Tran and P. Mika, "A survey of semantic search approaches", 2012.

[6] T. Tran, H. Wang, and P. Haase, "Hermes: Data web search on a pay-as-you-go integration infrastructure", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 3, 2009, pp. 189 – 203.

[7] T. Tran, D. M. Herzig, and G. Ladwig, "Semsearchpro – using semantics throughout the search process", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, no. 4, 2011, pp. 349 – 364.

[8] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu, "Spark: Adapting keyword query to semantic search", in The Semantic Web. Springer Berlin Heidelberg, 2007, vol. 4825, pp. 694–707.

[9] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu, "Avatar semantic search: A database approach to information retrieval", in Proc. of the 2006 ACM SIGMOD International Conference on Management of Data. ACM, 2006, pp. 790–792.

[10] E. Airio, K. Järvelin, P. Saatsi, J. Kekäläinen, and S. Suomela, "Ciri - an ontology based query interface for text retrieval", in Web Intelligence: Proc. of the 11th Finnish Artificial Intelligence Conference, 2004.

[11] T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-based interpretation of keywords for semantic search", in Proc. of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference. Springer-Verlag, 2007, pp. 523–536.

[12] D. Bonino, F. Corno, L. Farinetti, and A. Bosca, "Ontology driven semantic search", dec 2004.

[13] K.U. Sattler, I. Geist, and E. Schallehn, "Concept-based querying in mediator systems", The VLDB Journal, vol. 14, 2005, pp. 97–111.

[14] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati, "Ontology-based database access", in Proc. of the Fifteenth Italian Symposium on Database Systems, 2007, pp. 324–331.

[15] A. Poggi, M. Rodriguez, and M. Ruzzi, "Ontology-based database access with dig-mastro and the obda plugin for protégé", in Proc. of the 4th Int. Workshop on OWL: Experiences and Directions, 2008.

[16] H. Dehainsala, G. Pierra, and L. Bellatreche, "Ontodb: An ontology-based database for data intensive applications", in Advances in Databases: Concepts, Systems and Applications. Springer Berlin Heidelberg, 2007, vol. 4443, pp. 497–508.

[17] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: a system for keyword-based search over relational databases", in Proc. of the 18th International Conference on Data Engineering, 2002, pp. 5–16.

[18] N. Guarino, D. Oberle, and S. Staab, International Handbooks on Information Systems. Springer Berlin Heidelberg, 2009, ch. What Is an Ontology?, pp. 1–17.

[19] R. Studer, R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods", Data and Knowledge Engineering, vol. 25 (1-2), 1998, pp. 161–198.

[20] R. T. Gruber, "A translation approach to portable ontologies", Knowledge Acquisition, vol. 5 (2), 1993, pp. 199–220.

[21] P. Patel-Schneider, B. Parsia, and B. Motik, "{OWL} 2 web ontology language structural specification and functional-style syntax (second edition)", W3C, {W3C} Recommendation, dec 2012.

[22] SCOR11, "Scor supply chain operations reference model", Standard SCOR rev. 11, December 2012.

[23] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 5th ed., 2005.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. The MIT Press Cambridge, Massachusetts, 2009.

 ISBN: 978-1-61208-408-4

Automatic Knowledge Transfer-based Architecture towards Self-Service Business Intelligence

Safwan Sulaiman, Tariq Mahmoud, Jorge Marx Gómez
Department of Computing Science
Carl von Ossietzky University of Oldenburg
Oldenburg, Germany
{safwan.sulaiman, tariq.mahmoud, jorge.marx.gomez}
@uni-oldenburg.de

Joachim Kurzhöfer
Lufthansa Industry Solutions
Norderstedt, Germany
joachim.kurzhoefer@lhind.dlh.de

***Abstract*—Enterprises use business intelligence systems to support their decision making process and give them the superiority against their competitors. However, business intelligence systems integrate data from internal and external resources and this in turn makes these systems more complex. The consequence behind such complexity is that business users cannot get the required information at the right time unless they get help from IT or professional (power) users. This is also conditional just in case that power users will have enough time to answer the questions of business users. In this paper, we present a new architecture for business intelligence systems to support the self-service functionalities that enable business users to get answers about their business questions at the right time. The main idea of the proposed architecture is to extract the knowledge of the power users and transfer this knowledge to the business users while they use business intelligence systems.**

Keywords-Knowledge Transfer; Self-Services; Business Intelligence; Power User; Business User.

I. INTRODUCTION AND PROBLEM DEFINITION

In the last decade, business environments have changed to be more complex and dynamic. Enterprises should operate their business in a continuously changing environment, which is influenced by globalization, legal changes, volatile markets and technical progress [1]. Because of the rapid change in the internal and external conditions of current economic life, the demand for information as an important production factor is increased [1][2]. Therefore, enterprises adopt Business Intelligence (BI) systems that offer them solutions for these challenges. BI is defined as "an integrated, company-specific, IT-Based total-approach for managerial decision support" [1]. Therefore, the main goal of BI is the improvement of the decision making process by enabling business users to get the required information at the right time [3]. Based on Gartner research [4], the market share of BI systems and analytical applications is constantly growing. Gartner CIO's Survey 2013 showed that analytics and BI came on top of CIO's technology Priorities [5].

However, BI as an integrated system tries to integrate data from internal and external sources of the enterprise to one central point, which is the data warehouse. The goal of integrating data is to eliminate data redundancy and to have one single point of truth [6]. Consequently, this led to complexities in BI systems [7]. Moreover, even if a BI is a flexible and powerful system, it is still very complex for business users in the sense that they still face substantial difficulties while carrying out their ad-hoc analysis [3]. Furthermore, because of the high complexity and irrelevance of the provided information, less than 30% of target users could benefit from or use BI systems [8].

To avoid the confusing of the terms' usage, this work distinguishes between two types of BI users, business users and power users. Examples of business users include executives, managers and operation staffs. This kind of users tends more to be information consumers of easy-to-use BI tools like predefined reports or dashboards. Moreover, they lack the needed knowledge to use complex and more advanced BI tools like Online Analytical Processing (OLAP) and data mining. On the other hand, examples of power users include business analysts and IT professionals. They can use all kinds of BI tools with the different usage's complexities by generating the information based on their needs, as well as answering business questions of business users [9][10].

Therefore, in case that business user requires information for her/his decision process, and because of the complexity of BI tools and lack of knowledge to use such tools, business users must send a request to one of the power users to answer their business questions. However, these latter cannot answer in most cases at the right time due to their time limitations, the large amount of such requests and the few number of employed power users. Consequently, this will make the decision making process slower.

To conclude, the main goals of BI systems are eliminating guesswork and enabling enterprises to respond quickly to the market changes as well as customers' preferences. The aforementioned problems hinder BI systems from reaching these goals. Consequently, there is a need to develop a new BI solution, which empowers

 ISBN: 978-1-61208-408-4

business users with the required knowledge to get the right information at the right time without the need to ask power users. This new approach is called Self-Service Business Intelligence (SSBI).

The Data Warehousing Institute (TDWI) defined SSBI as: "the facilities within the BI environment that enable BI users to become more self-reliant and less dependent on the IT organization. These facilities focus on four main objectives: easier access to source data for reporting and analysis, easier and improved support for data analysis features, faster deployment options such as appliances and cloud computing, and simpler, customizable, and collaborative end-user interfaces" [11]. The approach presented in this paper focuses on achieving the objective "easier and improved support for data analysis features" to enable SSBI functionalities. Unlike other material assets of enterprises, which are decreased by using them, the knowledge will always be increased while it is recalled. The sharing of knowledge will enrich its receiver [12][13]. Therefore, most businesses recognize their knowledge as a sustainable source of competitive advantage [13][14]. The main idea of the resulted architecture is to transfer the knowledge from power users to business users.

In the next section of this paper, the knowledge classification, conversion, and transfer will be explained. After that, the basic idea behind this work and the knowledge flow are illustrated. Section IV explains the resulted architecture and its component. It is then followed by the knowledge transfer model's workflows. Section VI lists related work and finally this paper concludes with a short conclusion.

II. KNOWLEDGE MANAGEMENT: CLASSIFICATION, CONVERSION AND TRANSFER

In the literature, there are many definitions of the term Knowledge Management (KM), most of them are focusing on the KM processes. Davenport and Ponzi defined KM as the process of capturing, distributing, and effectively using knowledge [14][15]. Bhatia and Mittal defined KM as an approach to discover, capture, and reuse both tacit (in people's heads) and explicit (digital or paper based) knowledge as well as the cultural and technological means of enabling the KM process to be successful [16]. Dalkir claimed that KM is a collaborative and integrated approach to create, capture, organize, access and use an enterprise's intellectual assets [17]. Therefore, KM is considered as a disciplined, holistic approach to effectively use the expertise for competitive advantage [18].

A. Tacit Knowledge vs Explicit Knowledge

In the literature, there are different classifications of the knowledge. This work is focusing on the following two types of knowledge namely: the tacit and explicit knowledge [19][20]. Tacit knowledge is known as personal know-how and it resides in the head of knower. It is difficult to articulate and to put into words or text. Tacit knowledge that represents the expertise and know-how is the most valuable knowledge [17][21]. In contrast to tacit knowledge, explicit knowledge represents the knowledge that has been captured in a tangible form like words, audio files and recording images [17].

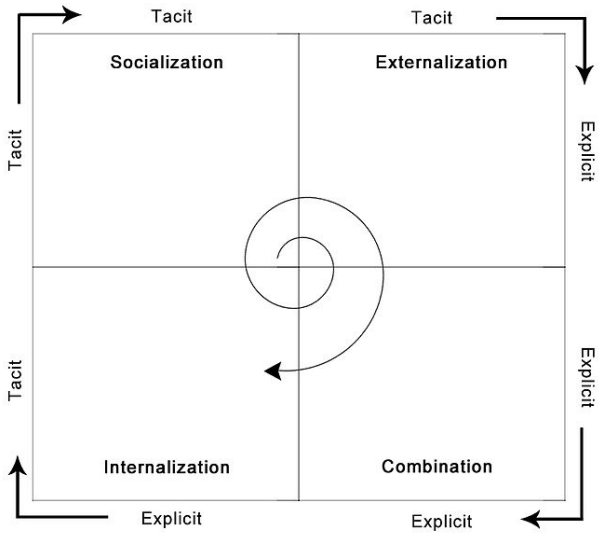


Figure 1. The SECI Process

Therefore, the processes of identifying, storing and retrieving explicit knowledge can be easily done by KM-system [22].

B. The Modes of Knowledge Conversion

Most enterprises try to externalize or convert the knowledge from tacit to explicit knowledge, after that they store this knowledge in their intranet or portal. The efforts in this sense must be set to improve the sharing of the stored knowledge [23][17][19].

Nonaka et al. explained in their Socialization, Externalization, Combination and Internalization (SECI) model how to transform processes between tacit and explicit knowledge [23]. As seen in Figure 1, there are four modes of knowledge transformation:

1) Socialization - Tacit to Tacit

In this case, the knowledge is converted through shared experience such as spending time together or living in the same environment, or face-to-face meeting.

2) Externalization - Tacit to Explicit

This is a process of articulating tacit knowledge into explicit knowledge by transforming the knowledge of people's minds into electronic forms like storing it in wikis, forums and collaborating systems.

3) Combination - Explicit to Explicit

The knowledge here is converted based on the desire of the user.

4) Internalization - Explicit to Tacit

The intranet of the enterprise allows the end users to access the information, which is stored in the knowledge repository. In this mode, explicit knowledge is used and learned from the user to extend her/his tacit knowledge and to become part of it. Internalization is related to the concept "Learning by doing".

 ISBN: 978-1-61208-408-4

C. Knowledge Transfer

Alavi and Leidner considered the knowledge transfer as an act of communication between source (the sender of the knowledge) and receiver (where the knowledge is transferred to). Both sides of the communication channel can be represented by a single person, as well as a team of people [18][24]. Knowledge transfer is the conveyance of knowledge from one place, person or ownership to another, and then this process can be considered as a successful process when it has a successful creation and application of the knowledge in the enterprise [25][26].

In the literature, the knowledge transfer process was described using models. Most of these models were focusing on the idea of collaboration and communication between the source and receiver of the knowledge [25]. Therefore, the basic knowledge transfer model consists of two main components namely: the source who share the knowledge and the receiver who acquire the knowledge. In addition, there are farther knowledge transfer modes, which describe different conversions between tacit and explicit knowledge (see the previous paragraph) [23][25].

III. BASIC IDEA AND KNOWLEDGE FLOW

"A little knowledge that acts is worth infinitely more than much knowledge that is idle" - Khalil Gibran [13]. The incentive from this quotation is that we argue that the knowledge should not stay idle in a database or a portal of any enterprise. One of the advantages of the proposed solution is that the knowledge must be automatically transferred to the receiver (business user).

In this work, the focus is on two modes of SECI knowledge conversion, which are the externalization and internalization. The transfer of knowledge between the expert and novice or specifically in this work between power and business users can be represented as socialization. However, in our case, it is not possible to use the tacit to tacit knowledge conversion. It is difficult from time and organization perspectives to put the power and business users in the same environment. Therefore, our approach uses two conversion modes. Firstly, in the phase of capturing or extracting the power user's knowledge, it is tacit to explicit conversion - externalization. Secondly, in the phase of knowledge application or sharing it, it is internalization-explicit to tacit conversion.

Figure 2 depicts an abstraction process model of the knowledge transfer that is independent from the technical components, which will be illustrated later in section IV. As shown in Figure 2, the process is divided into two steps. In the first step, the power user's knowledge is captured and extracted. This step represents the externalization conversion mode based on Nonaka SECI model (tacit to explicit conversion). In the second step, the captured knowledge will be shared or applied to the business user. This step represents the internalization conversion mode (explicit to tacit conversion).

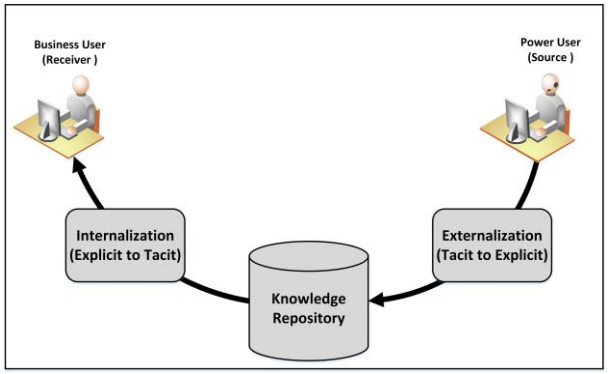


Figure 2. Knowledge Transfer Model

IV. KNOWLEDGE TRANSFER-BASED ARCHITECTURE FOR SELF-SERVICE BUSINESS INTELLIGENCE

The proposed architecture is based on top of the traditional business intelligence architecture to enable self-service functionalities. It is illustrated in Figure 3. It depicts the components of the proposed architecture and the relationships between them. Therefore, the overall architecture consists of two sets of components.

The first set represents typical components of BI-System architecture including data sources, ETL process, data warehouse and frontend – applications (in gray color).

As for the second set of components, it includes tracking module, analysis module, recommendation engine and knowledge repository components (in blue color).

Data sources represent all kind of storage resources like Enterprise Resource planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM) and any other external resources that include all data coming from outside the enterprise.

ETL process is responsible of extracting, transforming and loading data into a data warehouse (DWH).

As for the frontend - applications component, it offers BI users the access to the information in different formats and flexibility in the analyzing this information. BI users use the Single Sign-On (SSO) service to access the frontend applications (portal).

In the following sections, the new components of the architecture and their functionalities will be explained in details.

A. Tracking Module

This set includes three subcomponents:

1) User Interactions Catalogue

The catalogue includes all possible interactions between BI users and frontend - application components. The content of this catalogue comes from analyzing different frontend applications to consider all users' interactions that are filtered for the sake of knowledge extraction.

2) Tracer

This component traces the power user interactions that are previously defined in the user interactions catalogue. Then, it stores the trace log into the interactions database.

 ISBN: 978-1-61208-408-4

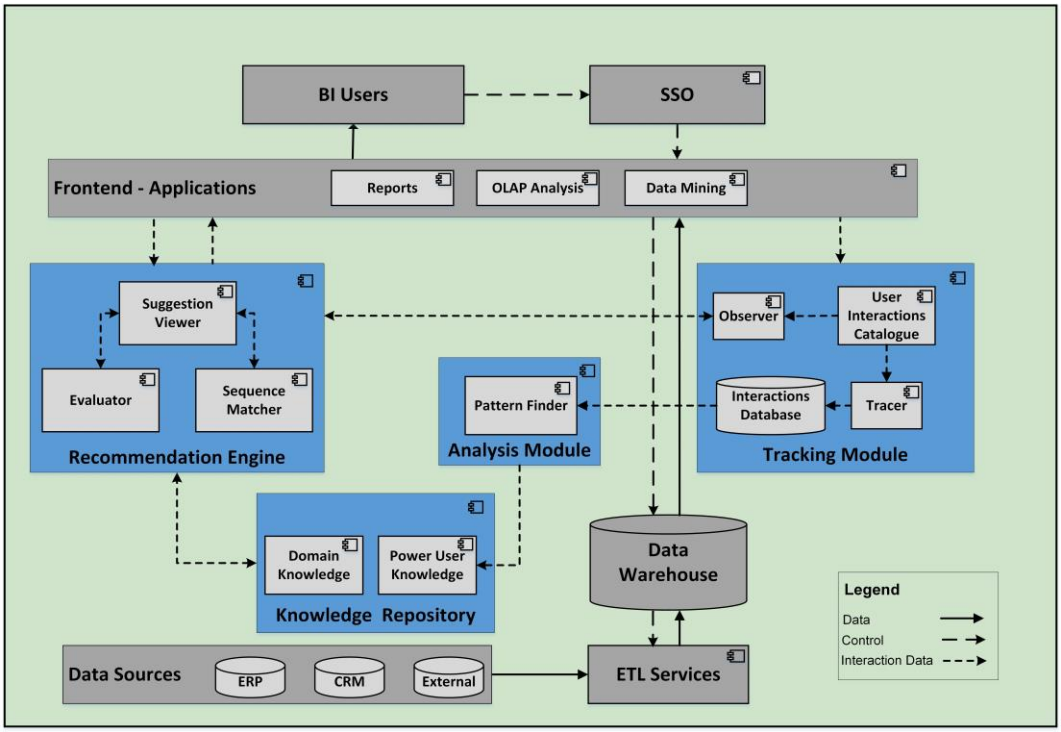


Figure 3. Automatic Knowledge Transfer Architecture for SSBI

3) Observer

This component is responsible of observing the interactions of business users while they use the BI frontend – application component in order to provide such log later to the recommendation engine.

4) Interactions Database

This database is merely responsible of storing the logs of power users' interactions. These logs can be stored either directly in the database or based on the power user's session, they will be stored in binary files.

B. Analysis Module

This set includes just the pattern finder component.

1) Pattern Finder

This component gets the log files from the tracer component as an input and processes them to extract patterns from them to be provided to the recommendation engine via the knowledge repository. The extraction process is done based on sequential data mining algorithm. These patterns are called analysis paths and they represent the procedural knowledge of the power user.

C. Knowledge Repository

This set has two components, the power user knowledge and the domain knowledge.

1) Power User Knowledge

This database stores the power user' analysis paths. It has a connection to the recommendation engine that uses these patterns to recommend similar paths to the business users.

2) Domain Knowledge

This database stores the typical knowledge harvested from the enterprise's data warehouse. Such knowledge includes several repetitive analyses of each enterprise's department.

D. Recommendation Engine

This set is responsible of offering business users with appropriate suggestions extracted from power users' knowledge to ease the analysis process for them. This set includes three components namely sequence matcher, suggestion viewer and evaluator.

1) Sequence Matcher

The main functionality of this component resides in displaying or showing business users the paths of extracted suggestions.

2) Suggestion Viewer

The main functionality of this component resides in displaying or showing business users the paths of extracted suggestions.

3) Evaluator

In this component, business users are given the possibility to evaluate the displayed suggestions based on the relevance to their analyses. There are two ways to get the evaluation of business users. The first way is implicit and similar to the ranking system of the google search engine. The evaluator gives high weights to the analysis paths that are selected by business users. The second one is an explicit way that allows business users to evaluate the suggestions themselves.

 ISBN: 978-1-61208-408-4

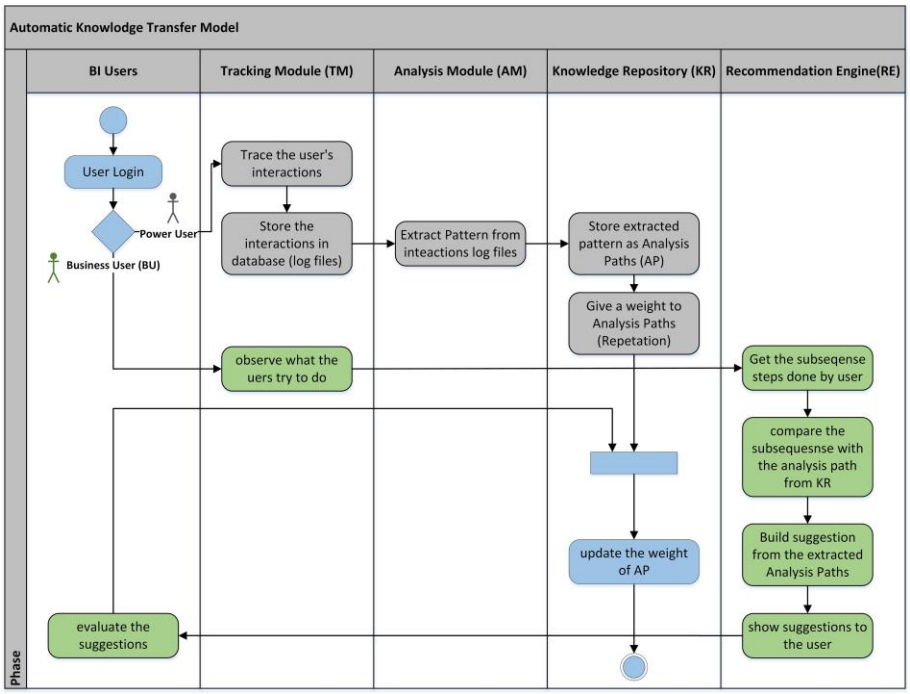


Figure 4. Knowledge Transfer Model Workflows

V. KNOWLEDGE TRANSFER MODEL WORKFLOWS

In this section, the interactions between the architecture components are illustrated as activity diagram in Figure 4. The interactions are illustrated for both business and power users. As can be seen in that figure, the first step represents the login of a BI user to the BI-Application. Based on the login information, the system recognizes the user type, because the users are already predefined as power or business users. Based on the user type, the system's functions are executed either for power or business users.

A. Power User Knowledge Extraction

This section explains the activities that are done to capture or extract power users' knowledge. After the system verifies that a user is a power user, the tracer component from the tracking module will trace all the interactions of this power user in a chronological manner. Then it stores its interactions into the interaction database or a log file. This log file will be then sent to the analysis module. The pattern finder applies a sequential pattern-mining algorithm to the log file or the interaction's database to extract a pattern from them. This pattern represents the analysis path of the power user. After that, the extracted analysis path will be stored in the knowledge repository. In this latter, every analysis path has a weight. This weight is a combination of its repetition in the interactions database and the evaluation of business users based on its relevance with business users' needs. Before storing an analysis path in the knowledge repository, there is a verification mechanism that checks whether an instance of this path is already stored in the repository or not. If an analysis path's instance exists in the knowledge repository, the repetition value of this path will be increased by one. Otherwise, a new instance will be stored in the knowledge repository.

B. Path Recommendation for Business Users

The previous section explained how power user's knowledge could be extracted and stored. This section illustrates how such extracted knowledge can be transferred to business users.

After the system verifies that the logged-in user is a business user, the observer component of the tracking module will verify what the business user wants to do, i.e., to trace the steps when the user tries to perform an analysis. After that, the tracking module will communicate with the recommendation engine by sending it the subsequence of steps, which are performed by this business user. Next, the recommendation engine will compare such subsequence with the existing analysis paths in the knowledge repository. The result of the comparison can be more than one analysis path with different weights. Therefore, the sequence matcher sends the result to the suggestion viewer, which is responsible of showing the business user the recommendation result as a list of suggestions. The suggestions should be sorted based on their weights. The analysis path with the highest weight will be appeared on the top of the suggestions list.

As soon as the business user sees the suggestions list, she/he can evaluate them to choose an analysis path. The repetition of this process will enhance the functionality of the whole system. This evaluator component will then evaluate the business user's selection to increase the repetition of the selected analysis path to update its weight in the power user knowledge's database.

 ISBN: 978-1-61208-408-4

VI. RELATED WORK

Two related works that have conceptual similarities to this work are to be explained here in this section. Mertens and Krahn had provided an approach of "Knowledge based business intelligence for business user information self-service" [3]. This approach is based on a semantic metadata layer, which is capable of modeling a semantic, machine readable and reasonable knowledge. This knowledge is imported and managed in the semantic metadata layer in form of domain ontology. It can be questions, analytics visualization or analysis results. However, the limitation lies in the issue that experts' knowledge should be explicitly derived and modeled, and then imported to the analytical information system. In comparison with these two approaches, the proposed research in this paper will automatically extract the knowledge of power user.

Another approach was provided by Baars about how to distribute BI knowledge [27][28]. This approach focused on the idea that the analysis results and templates should be accessed from other users in the enterprise via the knowledge management system. Analysis results can be interesting to the users of the same segment with the same needed information. Moreover, analysis templates can be used from users who belong to other segments or departments. This approach has several challenges. It requires combining different interfaces and formats. Moreover, it lacks motivating users to explain and distribute their knowledge to the knowledge management system. The proposed architecture tries to address the limitations of this approach by providing suggestions to the business users based on their functional usage of BI tools.

VII. CONCLUSION

This paper focused on presenting an extended BI architecture to enable self-service functionalities for business users. This architecture is based on a new knowledge transfer model that consists of two main processes. The capturing of the power user knowledge and sharing this extracted knowledge with the business users. Moreover, the functional components of this architecture have been explicated. Finally, two related works had been identified with their limitations to evaluate the strengths of the proposed approach. Future work will investigate the feasibility of this approach in the BI market.

ACKNOWLEDGMENT

We gratefully acknowledge that this work is supported by fund coming from Lufthansa Industry Solutions.

REFERENCES

[1] H.-G. Kemper, P. Rausch, and H. Baars, "Business Intelligence and Performance Management: Introduction," in Advanced Information and Knowledge Processing, Business Intelligence and Performance Management, P. Rausch, A. F. Sheta, and A. Ayesh, Eds.: Springer London, 2013, pp. 3–10.

[2] P. Chamoni and P. Gluchowski, Eds, Analytical Information Systems: Business intelligence technologies and applications, 4th ed. Berlin, Heidelberg: Springer, 2010.

[3] M. Mertens and T. Krahn, "Knowledge Based Business Intelligence for Business User Information Self-Service," in Collaboration and the Semantic Web: Social Networks, Knowledge Networks, and Knowledge Resources, S. Brüggemann and C. d'Amato, Eds.: IGI Global, 2012, pp. 271–296.

[4] R. Van der Meulen and J. Rivera, Gartner Says Worldwide Business Intelligence and Analytics Software Market Grew 8 Percent in 2013. Available: http://www.gartner.com/newsroom/id/2723717 [retrieved: Apr, 2015].

[5] Gartner, Top 10 CIO Business and Technology Priorities in 2013: Gartner, Inc, 2013.

[6] R. Winter and B. Strauch, "Information Requirements Engineering for Data Warehouse Systems," in Proceedings of the 2004 ACM Symposium on Applied Computing, New York, NY, USA: ACM, 2004, pp. 1359–1365.

[7] N. Kulkarni, Information Management: Embrace the Future of BI: Self Service. Available: http://www.information-management.com/newsletters/self-service-business-intelligence-bi-tdwi-kulkarni-10022855-1.html?zkPrintable=1&nopagination=1[retrieved: Apr, 2015].

[8] W. Kurzlechner, 3 levels of consumerization: Gartner: BI before radical change. Available: www.cio.de/knowledgecenter/bi/2283277 [retrieved: Apr, 2015].

[9] P. Gluchowski, R. Gabriel, and C. Dittmar, Management Support Systeme und Business Intelligence, 2nd ed.: Springer.

[10] S. Sulaiman, J. M. Gómez, and J. Kurzhöfer, "Business Intelligence Systems Optimization to Enable Better Self-Service Business Users," in WSBI, 2013, pp. 35–46.

[11] C. Imhoff and C. White, "Self-Service Business Intelligence- Empowering Users to Generate Insights," TDWI best practices report. On the TDWI site: www. tdwi. org, 2011.

[12] A. Brooking, Intellectual Capital: Core ASset for the Third Millenniurn Enterprise: London: Internah0nal Thornson Business Press, 1996.

[13] A. Tiwana, The knowledge management toolkit: Practical techniques for building a knowledge management system. Upper Saddle River, NJ: Prentice Hall PTR, 2000.

[14] T. H. Davenport and L. Prusak, Working knowledge: How organizations manage what they know: Harvard Business Press, 1998.

[15] L. Ponzi and M. Koenig, "Knowledge management: another management fad," Information Research, vol. 8, no. 1, p. 8, 2002.

[16] K. Bhatia and S. Mittal, Manpower Development for Technological Change: Excel Books India, 2009.

[17] K. Dalkir, "Knowledge management in theory and practice," McGill University, 2005.

[18] S. Gasik, "A model of project knowledge management," Proj Mgmt Jrnl, vol. 42, no. 3, pp. 23–44, 2011.

[19] I. Nonaka and N. Konno, "The Concept of "Ba": Building a Foundation For Knowledge Creation," California Management Review, vol. 40, no. 3, 1998, pp. 40–54.

[20] A. C. Botha, D. Kourie, and R. Snyman, Coping with continuous change in the business environment: Knowledge management and knowledge management technology. Oxford, UK: Chandos Pub, 2008.

[21] R. J. Sternberg and J. A. Horvath, Tacit knowledge in professional practice: Researcher and practitioner perspectives: Psychology Press, 1999.

[22] J. L. Wellman, Organizational Learning: How Companies and Institutions Manage and Apply Knowledge: Palgrave Macmillan, 2009.

 ISBN: 978-1-61208-408-4

[23] I. Nonaka, R. Toyama, and N. Konno, “SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation: nona,” Long Range Planning, vol. 33, no. 1, 2000, pp. 5–34.

[24] M. Alavi and D. E. Leidner, “Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues,” Mis Quarterly, 2001, pp. 107–136.

[25] C. Liyanage, T. Elhag, T. Ballal, and Q. Li, “Knowledge communication and translation- a knowledge transfer model,” Journal of Knowledge management, vol. 13, no. 3, 2009, pp. 118–131.

[26] E. Major and M. Cordey-Hayes, “Knowledge translation: a new perspective on knowledge transfer and foresight,” Foresight, vol. 2, no. 4, 2000, pp. 411–423.

[27] H. Baars, “Distribution von Business-Intelligence-Wissen,” in Analytische Informationssysteme: Springer, 2006, pp. 409–424.

[28] H. Baars and H.-G. Kemper, “Management support with structured and unstructured data—an integrated business intelligence framework,” Information Systems Management, vol. 25, no. 2, 2008, pp. 132–148.

 ISBN: 978-1-61208-408-4

IES's UIEs for Generating LACs for Arguing that a CI Satisfies SPL

Sigram Schindler
TELES Patent Rights International, TU Berlin
s.schindler@teles.de

***Abstract*—The paper outlines the User Interface Entities of an Innovation Expert System which enables its users to generate the Legal Argument Chains determined by the FSTP-Test, as required for testing a claimed - classical as well as emerging – technology invention under the Substantive Patent Law of any National Patent System. In particular, under the 4 §§ 101/102/ 103/112 of 35 USC and its interpretations provided by the Supreme Court's *KSR/Bilski/Mayo* decisions. Such Legal Argument Chains, in real-time-mode, are helpful in arguing (e.g., in a patent court's hearing) about a claimed invention's inventive concepts making it satisfying Substantive Patent Law, i.e. patent-eligible and patentable). The Innovation Expert System and its functionalities presented here are worldwide unique. They are being implemented as a prototype in a major R&D project.**

Keywords-User Interface Entities; FSTP-Test; SPL; Inventive Concept.

I. INTRODUCTION

The US Supreme Court's *Mayo* decision [1][18][19] requires describing a claimed invention (CI) by its "inventive concepts, inCs" if it deals with emerging technology subject matter and hence is "model-based" – and thus stimulated Advanced IT [2] research on decision making in testing such CIs under Substantive Patent Law (SPL), also holding if describing the CI needs no model [11][18][19][25][36][45][72][79].

Some examples of such models are: The "ISO/OSI" model of telecommunications, "molecular bonding forces" models of nano-technology, "RNA/DNA" models of genetics, "Natural Language" models of Advanced IT – some standardized, all implicitly used by SPL precedents without being aware of this. The philosophical synonym of the term model is "paradigm", the scientific one "reference system", e.g., "coordinate system". Using a model often enables de-scribing a CI precisely on top of it, though it itself is not understood precisely – as practiced with mathematics' "axioms/theorems/proofs", with physics' "laws of nature", and here with SPL's "claimed inventions". The here claimed invention is applicable to all model-based CIs.

Previous studies [10][18][19][25][46][47] proved that "A CI satisfies SPL iff it passes the "FSTP-Test" (Facts Screening/ Transforming/ Presenting)". Thus, the FSTP-Test may (semi-) automatically deliver all different "Legal Argument Chains, LACs" showing a CI satisfies SPL. This greatly facilitates every patent practitioner's decision making as to testing a CI under SPL, in particular if it is model-based. SPL reasoning is always of finite first order logic (FFOL).

While Section I gave the background of the paper's topic, Section II introduces IES, UIEs and LACs, Section III talks about the LACs generated during the testing of a given CI, Section IV draws some conclusions about the scope of the CI of [59] and discusses the need for scientizing the reasoning about model-based CIs, and Section V will close by showing that this way of generating LACs (α) - ζ)) is patentable/patent-eligible, i.e., specifically something completely new.

II. THE IES, ITS UIES AND LACS

A system based on a CI's alias TT.0's (Technical Teaching) PTR^{SPL}-DS (**P**air of technical **T**eaching over a prior art **R**eference Set - **D**ata **S**tructure) [6][7] – which stores all SPL-relevant functional and non-functional properties of this CI – is called an "Innovation Expert System, IES", iff it has a "User Interface Entity, UIE" enabling its user(s) to access of this CI all (legally non-redundant) Legal Argument Chains (LACs) showing its satisfying SPL. An IES leverages on its PTR-DS embodying, by all results of its CI's FSTP-Test, all "Arguable Subtests, ASTs" – these being the blueprints of all LACs of this CI. Automatic LACs generation according to this invention is not limited to CIs' tests under SPL.

The UIE of an IES is made-up from UIE.Ys, Y=1,2,..., any one comprising a knowledge representation "KR-UIE.Y", a human interaction "HI-UIE.Y", and an interaction control "IC-UIE.Y" entity, in config-/real-time-mode used separately resp. synchronously. An IES or a user of it invokes between them an "interaction". In config-mode, an interaction serves for generating or modifying of a UIE.Y by a user at least 1 of its just quoted 3 components. In real-time mode, an interaction serves for invoking, controlled by its IC-UIE.Y, the presentation of a HI-UIE.Y. In both modes, this interaction uses its KR-UIE.Y, which in turn uses the knowledge stored by PTR-DS [11][25]. A UIE.Y may be subdivided into (potentially nested) "UIE.Y Steps"; invoking a UIE.Y causes at least executing one of them partially.

A LAC.Z, Z=1, 2, …, is presented by executing at least 1 partial UIE.Y in real-time-mode. An AST.X, X=1, 2, …, is accessed by at least 1 KR-UIE.Y, each translated into at least 1 LAC.Z. An AST.X may be used in at least 1 "logics presentation", tied to at least 1 HI-UIE.Y by its own IC-UIE.Y, as customized by an IES user in config-mode – between which a user may toggle by invoking these IC-UIE.Ys, i.e., in config-mode of the IES, any AST is (semi-) automatically transformable into its 1 or more LAC.Zs, being AST's in various logics presentations translated into multimedia presentations by UIE.Ys – as needed by a judge, examiner, lawyer or an inventor. In real-time-mode, a user may toggle between these UIE.Ys of an AST.X for highlighting its aspects by the LAC.Zs into which AST.X is translated.

 ISBN: 978-1-61208-408-4

III. On Generating All LACs for a CI's Test Under Any FFOLLIN [A][B]

This paper leverages on scientific insights achieved in the FSTP project and the reference list. They showed how all ASTs – all being non-isomorphic – of a CI/TT.0 tested under SPL, may semi-automatically be transformed, using an IES in config-mode, into their peer LACs, which in the real-time-mode of the IES then may be automatically invoked. The role of the FFOLLIN is explained after **α)-ζ)** below.

For conveying the working of the IES in config-mode, the below bullet points specify technical features of an IES enabling a user of it to configure alias calibrate alias customize it according to the needs of its user(s) in its real-time-mode. Thereby one or several users may use the IES simultaneously in config- and/or real-time-mode, thus directly or indirectly communicating with each other. The understanding of the working of the IES in real-time-mode immediately follows from its config-mode understanding. They add such features sometimes redundantly, as explained already above and/or by these publications.

These bullet points thus also disclose the scope of the CI of the patent application this paper is based on.

- According to Schindler et al. [25], a CI satisfies SPL iff it passes the FSTP-Test. And: A CI passing FSTP test.m, $2 \leq m \leq 10$ (on top of a subset S resp. S' of TT.0's finite set of all its BED-inCs (binary elementary disclosed inventive concept) passes all FSTP test.n, $1 \leq n < m$, on top of this set. The inverse of this implication evidently needs not to hold.
- The complete FSTP-Test is a program evaluating, for a CI under SPL test, the whole FFOL expression modeling the logics (see below) of and between the 11 concerns embodied by the 35 USC SPL over the mirror predicates of BED-inCs of this CI, the conjunctions of these BED-inCs' mirror predicates modeling the properties of the elements of the CI. Their peers in prior art TTs may or may not exist – as decided by an FSTP-Test user (and confirmed by the person of ordinary skill and creativity (posc) – forming the AN (anticipated non-anticipated) matrix [6][7][11][43][59].
- Any AST is a lexically and syntactically correct "**sentence**" alias FFOL term from within this whole FFOL expression. Hence, for any CI, there are only finitely many ASTs, and for any AST its semantic is evident (except the semantics of the above properties and the relations between them that the user/posc has input into the PTR-DS when generating it – here assumed to be correct).
- PTR dependent, only finitely many (usually few hundred) ASTs exist. All these ASTs are executable on top of these finitely many and PTR-dependent BED-inC subsets. All these ASTs, resp. their basic AST arguments (BASTAs) are the blueprints for all LACs. Other (legally non-redundant) LACs don't exist – though different presentations to IES users of any AST as different LACs may.
- Any UIE.Y for any AST.X (to be translated into a LAC.Z) may be generated in config-mode by an IES user by its invoking the "**UIE-stub**" provided by any IES implementation and delivering to it this UIE.Y, depending on the parameters of this invocation being a fresh UIE.Y or an existing and defined UIE.Y for checking or changing the result of preceding input, or the interworking between presenting several UIE.Y invocations of LAC.Z, or its interworking with other LAC.Z' presentations. Thereby any UIE.Y may be composed by the user of one or several sequential "UIE steps, UIESes", whereby any UIES again may be composed by the user of one or several sequential such steps (nested UIE.Ys). Any UIE.Y and UIES.Y must be specified by the user – except automatic ones, depending on the particular IES implementation and/or configuration – as to the functionalities of their 3 resp. KR-/HI-/IC-UIE.Ys or KR-/HI-/IC-UIES.Ys.
- The just mentioned 3 components of any UIEs may vastly be generated automatically by the IES or interactively generated by a user guided by the IES – not elaborated here – and would basically be the same or similar, i.e. are principally stereotypical.
- Thereby the objective of the claimed invention presented here, is not limited to providing for a given CI only all LAC.Zs for justifying solely its classical claim construction – such LAC.Zs would only show that the CI has a chance to satisfy SPL – but to provide all LAC.Zs showing CI satisfies SPL.
- After automatically or semi-automatically/interactively having decomposed in config-mode, as deemed reasonable by an IES user, all the PTR-DS into all ASTs, any one potentially in a multitude of ASTs' logics, into peer LACs' multimedia representations and user interaction capabilities (as shown by the below steps **α)-ζ)**), in real-time-mode these ASTs or LACs may be invoked automatically (e.g., by an acoustic word spotter of the IES), and/or (semi-) automatically by an IES user (see the below steps **α)-ζ)**). Thereby its execution may comprise specific items for communicating with a user, e.g., about any kind of management issues. Pertinent ordinary skill knows, e.g., from IVR systems and their audio pattern spotting and matching functionalities how in principle to (semi-) automatically identify in real-time LACs to be instantly invoked, as the dialog just taking place generates an appropriate pattern. Here such LAC identification and invocation processes in real-time-mode may be substantially supported by the IES calibration providing resp. hints to these processes, e.g., leveraging on graphical and/or acoustic patterns embodied by a related multimedia thesaurus construction based on "AST patterns".
- The complete **FSTP-Test** of a CI for its satisfying 35 USC SPL comprises the 10 FSTP test.o, $1 \leq o \leq 10$. It is executed for the "set ∀ claim interpretations, SoI" of the CI, selected in **(b)** therein, i.e. all TT.0s

of this CI – a CI may enable several interpretations, if disclosed by its patent's (application's) specification [72][79]. The term/notion "technical teaching 0, TT.0" [6][7][11] then stands for one of them [72][79], i.e., the TT.0s are the elements of the CI's "set of interpretations, SoI".

- Note that there is a variety of execution sequences of the FSTP-Test for any one of these TT.0s: While the initialization sequence of the 10 FSTP test.o's must be that of their natural number indexes, they may be executed exhaustively or overlapping – i.e., for the latter case holds: ∀ FSTP test.n check of this CI only those of its inCs already confirmed by the FSTP test.m ∀ m<n.
- Advanced IT knows that the input and commands provided by an IES user to the IES just as the latter's output to an IES user must have, for being understandable by both, some before given – here a priori defined by the IES – alphabet (vocabulary) and syntax and semantics and pragmatics or these must be determined during the execution of the claimed invention's FSTP-Test by the IES under rules given a priori by the IES and under the control by an IES user.
- The term/notion "legal argument chain, LAC" stands for what is commonly understood by any posc with knowledge of the SPL. Its broad meaning is not limited in any other way. The index "Z" identifies a particular LAC.Z, more precisely: an instantiation Z of the "type LAC" (in terms of programming languages). The same applies for the types/instantiations "AST"/"AST.X", "UIE"/ "UIE.Y", … .
- The above UIE-stub provided by an IES on top of a PTR-DS – representing a CI's TT.0 to be tested for satisfying SPL - is available to an IES user all the time (unless locked by a user). As said above already: By means of the UIE-stub an IES user may define a broad range of UIE instantiations for configuring the UIE between an IES user and the IES for customizing the CI's SPL test for the IES user: Such as to facilitate for it using the functionality provided by the IES for this test.
- Whether a UIE.Y is to be generated/integrated/modified or executed is determined by the mode the IES is in at UIE.Y invocation time – whereby this mode may be set by an IES user (e.g., the one performing this invocation or another one) or by the IES and/or at whatsoever time of the existence of this UIE.Y and of the function execution being invoked. Thereby conflicts may occur and must be resolved by the implementation of the IES, either automatically or interactively with an IES user.
- Any invocation may refer to only a step within a UIE instantiation.
- The content of a human interaction, i.e. its semantics, is currently transparent to the IES unless it is automatically derived by the IES from the AST at issue, potentially occurring for very simple ASTs.

The usefulness of the CI disclosed in the patent application this paper is based on – i.e. of the IES resp. of the method controlling it – is to be seen in the HI-UIEs' capability of (semi-)automatic instant information presentations by one or several different LACs about any AST of a CI's TT.0 under, e.g., SPL test to an IES user, in response to the latter's invocation of some detail of the PTR-DS or its FSTP-Test representing this TT.0 resp. this detail.

The invention of the patent application this paper is based on has been invented, in particular, for thus enabling the IES to present automatically or interactively a LAC in response to a question being asked, as if this response were provided by a human being of total knowledge about the TT.0 being SPL tested.

To this end, this response must be represented by the IES – by having the person speaking and showing what it graphically uses for support of its presentation, both in reality or on a screen, anyway all media used in synchrony, what would be the normal cases in real-time mode use of the IES – as it were presented without the support by the IES. For achieving this, the IES enables a user first to acoustically and/or graphically input fragments of the arguments it later intends to present in its personalized fashion, then to combine these fragments into what it considers to be a complete legal argument chain, and finally to invoke the automatic reproduction of this argument. Responding this way to a listener/viewer of this LAC – to a question it or somebody else had input to the claimed invention before as a query – then would appear to the listener/viewer as a personal and potentially multimedia announcement/information of a smart IMR system (interactive multimedia response). This "user personalization" of the behavior of the above claimed invention's IMR subsystem would comprise that an IES user and the IES may cooperate in jointly presenting a complex LAC by alternatively speaking or reacting on interposed questions by answering them immediately – whereby such prompt reactions may be configured, also by IC-UIE.Ys, to be interventions and/or accompanying illustrations, always under an IES user control.

For achieving this result, the IES would execute many steps of such a whole process automatically or interactively, as outlined in **α)-ζ)** below, e.g., when directly or indirectly (i.e., on IES request) invoked by an IES user, the IES may basically:

- **α)** recognize by/for which "high level user interaction" – due to the FFOL nature of the problem only finitely many such user interactions are required by an IES – this invocation occurred, then
- **β)** derive, for this interaction, which technical items and/or legal items from the FSTP-DS it needs,
- **γ)** determine, by which "basic AST arguments, BASTAs" (see below) they are covered – due to the FFOL nature of the problem, i.e. of the FSTP-Test, there is only a finite number of BASTs (basic ASTSs) for any TT.0, the respective TT.0 independent BASTAs would be provided by the IES, and the TT.0 dependent BASTAs would be input by an IES user into the IES under the latter's guidance being controlled by the PTR-DS prior to using the IES as outlined by **α)-ζ)** – then

 ISBN: 978-1-61208-408-4

δ) compile from these BASTAs some "sequence of BASTA, SoBASTA" – due to the FFOL nature of the problem any sequence is correct, yet second thoughts being useful – a single complete sequence of "low level answers" to these questions, and have a KR-UIE instantiation represent this SoBASTA,

ε) translate this low level SoBASTA into one or several specific – but logically equivalent to each other and to the SoBASTA – sequences of future (if working in config-mode) or actual (if working in real-time-mode) multimedia outputs on what I/O devices, and have the same number of HI-UIE instantiations represent these future/actual outputs, whereby each such instantiation provides, potentially supported by the KR-UIE alias SoBASTA, a specific basis for one or several sequences of high-level user interactions invoked above but executed under the control of **ζ)**, and finally,

ζ) determine, for any HI-UIE instantiation of **ε)**, when in the future (if working in config-mode) or actually (if working in real-time-mode) on what event how to output on what I/O devices which part of this or another one of these HI-UIE instantiations of **ε)**, and have for this HI-UIE instantiation its specific IC-UIE instantiation represent these future/actual interaction controls – thus linking, to commands of IES users, not only parts of HI-UIE instantiations of **ε)** but also what any latter part needs for its execution from a KR-UIE.

Some comments on the steps **α)-ζ)** and, in particular, on this CI's philosophy may be helpful:

- Any step requires some interactive input from or control by an IES user or executes fully automatic.
- These steps differ when invoked in different modes, e.g., **i)** in explorative/calibrating/config-mode, **ii)** in reply-testing-mode, **iii)** in "one-way"-reply-mode, **iv)** in "two-way"- alias "interactive"-reply-mode, **v)** in some "consolidation"-reply mode, etc.
- The BASTAs (= basic AST arguments) in step **γ)** represent a complete (usually, neither not unique, nor non-redundant) finite set of basic building blocks into which the whole FSTP-Test may be decomposed. In any BASTA, the term "basic" has the meaning that it deals with only a single factual alias "technical" and/or legal question as to one of the 10 FSTP test.o (which enables dealing, e.g., with the finitely many such details or evaluations or relations of some kinds of inCs or the FSTP test.o at issue), and the term "argument" indicates that the BASTAs are translated into the basic building blocks also of the LACs.
- While an embodiment of the CI of the patent application this paper is based on working with the steps **α)-ζ)** uses the functionality specified for the CI in a pretty sophisticated manner, for the person of posc its implementation would nevertheless be straightforward realizable. This holds even more for the CI's simpler embodiments, always achievable by appropriately limiting the I/O flexibility of such embodiments.
- In addition to the steps **α)-ζ)**, an embodiment of the above claimed invention may provide "prototypes" of all user interactions and modes it provides, as well as macros for the stereotypically recurring parts when invoking them, such as repeating some passage in other words or particularly slowly, or skipping momentarily boring details, or prompting a user to continue, or asking for confirmation the understanding of the just said, or … .
- LACs may also be presented by their default configurations coming with user interactions specific for models of application areas. These prototype interactions are fine for inputting/defining/configuring specific UIE instantiations by a user for its personalization of the IES and/or its LACs for adapting them to the specificities of the actual PTR-DS under test; but, normally, these prototypes' functioning is not yet what an IES user ideally would like to use.
- This paper nowhere uses peculiarities of an SPL [A] or its FSTP-Test, i.e., SPLs are too narrow for specifying it. The next paragraphs shall clarify this and thus determine the scope of the CI of [59].

Speaking in terms of programming languages: SPL, "**Substantive Copyright Law**, **SCL**", …, may be seen as a range of "directive" type declarations, the defining commonality of which is their being a "**finite FOL legal norm**, **FFOLLN**". Hence, any such directive type declaration may be called FFOLLN and is defined by a finite set of conjunctively to be met requirements by any instantiation of this directive type, i.e., by any subject matter satisfying it.

Here, any instantiation of a FFOLLN would occur by means of a subject matter being a CI of FFOL, thus by means of a finite set of BED^{SCL}-inCs generative for this CI [72]. Hence, this instantiation – being a subject matter defined by this CI of this FFOLLN – is called "**finite FOL legal invention norm**, **FFOLLIN**".

IV. CONCLUSION/DISCUSSION

Based on this understanding, one sees that the scope of the above CI indeed comprises any $IES^{FFOLLIN}$ – which is confirmed by a careful analysis of the claims claiming this CI. Thus, from the above programming language considerations and definitions follows (in generalization of the considerations in, e.g., [10][18][19][25] mathematically reconsidered by [76] and putting it in terms independent of programming language and legal jargon): The scope of the patent application this paper is based on comprises any equally powerful "**t**est of a **c**reation **n**ecessary **a**nd **s**ufficient for its **m**eeting a given **r**equirement, TC.NaS.MR".

Being "equally powerful" means: This CI (based on the patent application this paper is based on) enables building for any FFOLLIN an $IES^{FFOLLIN}$, which by customization/configuration becomes that knowledgeable that, if asked a question about this $TT.0^{FFOLLIN}$'s satisfying a requirement its FFOLLIN instantiation states, it may instantly respond by one or several correct and complete LACs, their presentations being controllable by an IES user (as detailed above).

 ISBN: 978-1-61208-408-4

This generalization evidently impacts also on the $FSTP^{SPL}$-Test determining the PTR^{SPL}-DS, implying that an $FSTP^{TCNaSMR}$-Test determines a $PTR^{TCNaSMR}$-DS. Writing just "$FSTP^{FFOLLN}$-Test" and "PTR^{FFOLLN}-DS" is less specific in notation, but implies the same. This generalization even may be expanded to the FFOLLN's dependency on non-finite parameters, e.g., time. I.e., the above discussed CI has a much broader application area – i.e. all FFOLxNs areas, "x" standing not only for "law" but also for any private "directive" – than the one repeatedly explicitly addressed above for exemplary purposes, namely 35 USC SPL.

- An $IES^{FFOLLIN}$ defined by some FFOLLIN creation alias "technical teaching.0, $TT.0^{FFOLLIN}$," – defined to be a CI the properties of the elements of which are precisely describable by conjunctions of the mirror-predicates of this TT.0's $BED^{FFOLLIN}$-inCs – is all-knowing (in the above described sense) as to $TT.0^{FFOLLIN}$'s satisfying this FFOLLN, and is comprised in [59]. E.g.: An IES^{SPL} defined by a CI^{SPL}'s BED^{SPL}-inCs and the $FSTP^{SPL}$-Test is all knowing about CI^{SPL}'s satisfying this SPL.
- This enables several very interesting conclusions showing the total unreasonableness of trying to reason about model-based CIs without scientizing this reasoning. Namely, that
  - for implementing an IES^{FFOLLN} (as claimed in [59]) – the 35 USC SPL is just a specific FFOLLN – neither a concrete FFOLLN nor the $FSTP^{FFOLLN}$-Test is needed (i.e. it is sufficient to know that it is FFOL) nor a CI^{FFOLLN}. By calibrating a so implemented "abstract" IES^{FFOLLN} by a $CI^{FFOLLIN}$'s $PTR^{FFOLLIN}$-DS (based on a concrete FFOLLN, concrete $CI^{FFOLLIN}$, and concrete $FSTP^{FFOLLIN}$-Test here needed for construing the $PTR^{FFOLLIN}$-DS) it becomes an $IES^{FFOLLIN}$ all-knowing about $CI^{FFOLLIN}$'s satisfying FFOLLN.
  - for none of the application areas of the CI disclosed by the patent application this paper is based on (one of them being the "35 USC SPL area") – all being "FFOLLN areas" – the $FSTP^{FFOLLN}$-Test can be defined without basing it on a FFOL CI, i.e., any FFOL CI from a FFOLLN area creates, by its $FSTP^{FFOLLIN}$-Test, its specific compound metric for any prior just as posteriori art over the posc underlying this FFOLLN area.
  - recognizing any CI creates its own metric was not really necessary with classical technology CIs – there intuition insinuates it always is the same (though not understood by anybody prior to FSTP technology) – for model-based emerging technology CIs no intuition exists, thus making indispensable the scientification of their tests for satisfying their FFOLLNs, whatsoever [79].
- The below quoted and underlined phrases used by [59] have the following meanings:
  a) "On direct or indirect request by an IES user" says that this IES user may by itself invoke a function (= request its execution directly) or else it may be prompted by the IES to invoke a function (= request its execution indirectly).
  b) "Several different LACs about any AST of a CI/TT.0" shall indicate that the existence of several LACs need not only be due to an IES user having defined several different multimedia presentations for a LAC, but may also be caused by the AST itself comprising different ways of reasoning, e.g., having different disclosures for an inC the AST deals with and/or having for a disclosure more than one legal justification.
  c) An answer provided by the IES to a query put by an IES user (as to at least one aspect of at least one inC of the CI at issue) is called "complete and concise" iff it addresses and comprises all relevant legal and technical information and presents this information such that it shows the CI meets all respective requirements stated by SPL – unlike information provided by the classical claim construction, as missing both these objectives.
  d) "A question raised by an IES user intentionally or not" says that the user may raise this question quite purposefully, i.e., targeted, or incidentally, i.e., by chance, e.g., in presenting an argument.
  e) The different "logics" of an AST denote the various kinds this AST may present some issue, e.g. justify why an inC is disclosed by the specification or why the inCs in a set are independent.
  f) "All ASTs for a given CI and its FFOLLIN" says that any part of this CI's $FSTP^{FFOLLIN}$-Test is covered by an AST, i.e. the CI's complete $FSTP^{FFOLLIN}$-Test understood as a logical conjunction of basic logic statements is decomposed into sets of BASTs (see **γ)** above).
  g) Two LACs are "non-redundant", if the ASTs they represent share no BAST.

V. THE IES'S UIE PATENT APPLICATION'S CI SATISFIES THE 35 USC'S SPL

Considering [11][25], the claimed invention in [59] satisfies the 35 USC §§ 101/102/103/112 as it passes all 10 FSTP tests.o. It namely passes [C] the following tests:

1) The FSTP-Test prompts the user to input
<no "**multi-interpretable CI**" until [137]>
(a) $\forall TT.i \wedge 0 \leq i \leq I = |RS| \wedge 1 \leq n \leq N = N(TT.0)$:
the pair (Xin, BAD-crCin):
(b) $\forall 1 \leq n \leq N$ justof: (X0n,BAD-crC0n) is **definite**;
(c) $S::=\{BED\text{-}crC0nk | 1 \leq k \leq K^{n}, 1 \leq n \leq N\}$:
$BAD\text{-}crC0n = \wedge^{1 \leq k \leq K_n} BED\text{-}crC0nk \wedge K::=\sum^{1 \leq n \leq N} K^{n}$;
(d) $\forall 1 \leq k \leq K^{n} \wedge 1 \leq n \leq N$ justof: BED-crC0nk is **definite**;
2) $\wedge \forall \epsilon$ S for justof: their **lawful disclosure**;
3) $\wedge \forall \epsilon$ S for justof: their **definiteness** under § 112.6;
4) $\wedge \forall \epsilon$ S for justof: their **joint enablement of TT.0**;
5) $\wedge \forall \epsilon$ S for justof: their **joint independence**;

6) ∧ ∀ ϵ S for justof: their **joint *KSR/posc*-non-equivalence**
∧ define the BED*-AN matrix by
BED*-inCik ::= N ∀ 1≤k≤K^{n} ∧ 0≤i≤I;
BED*-inC0k::= A if BED-inC0k ϵ^{KSR} posc;
BED*-inCik::= A BED-inCik =^{KSR} BED-inC0k, 1≤i≤I;

7) ∧ for justof: by NAIO [*] S is **not an abstract idea only**;
<see **iii)** of [136] >

8) ∧ for justof: S **contains a patent-eligible BED-crC0nk**
<see **iii)** of [136] >

9) ∧ for justof: S **is a patent-eligible combination**;
<see **iii)** of [136] >

10) ∧ for justof: by NANO [**] S is **patentable** on S$^{pat\text{-}el}$ ⊆ S.
<see **iii)** of [136] >

Hence, as stated in [59], the there claimed invention satisfies 35 USC's SPL.

Finally, it is worthwhile noticing that this CI passes, by passing all 10 FSTP tests, even 16 tests – of which the classical claim construction only performs 6 ones, as explained by [25]. To put this insight into the *Mayo* context: If the classical claim construction were allegedly seen as an invention being that useful as to determine whether a claimed invention satisfies the US SPL or not, it would be – as seen by *Bilski/Mayo* – just an "abstract idea only" of a claim construction. Though, strangely enough, the classical claim construction never has been set out to be that useful. Indeed, it is more misleading than guiding to the complete and 35 USC conforming and by *Mayo* required claim construction [B][C].

REFERENCES

[A] While today differences still exist between the "Substantive Patent Laws, SPLs" of the US and other regions/nations, e.g., the EU with the SPL of its EPC, these should disappear soon, as internationally harmonizing so understood SPLs is politically not too controversial and economically highly beneficial for all parties as then being "Highest Courts" proof – in the US totally, in the EU and many Industrial nations vastly. Similar processes occurred in the past, e.g., with the national accounting procedures of public companies, today harmonized by the worldwide IFRS (International Financial Reporting Standard). Here, the PatentHighwayProgram of several large PTOs may play a decisive role.

[B] Due to their novelty, many details – also evident ones – were briefly explained in Section III of this paper. Such trivialities ought to be superfluous in a patent application. If a future patent application were supported by its PTR-DS – or even by an IES as discussed here – all such explanations, also trivial ones, would be presented to a user on its request in real-time, potentially in utmost controllable multimedia presentation.

[C] Performing the NANO test on the above CI determines its creative height to be ≥5 over posc, and there is no prior art or pragmatics which could reduce it. By [5][6] this CI's creative height is

- larger than 1, thus warranting its novelty (as by the posc there is no prior art document anticipating one of the 5 BID-inCs [59]), and as it is especially at least
- 5 or more, thus warranting its non-obviousness, due to the same reason.

[*] The **"Not Abstract Idea Only, NAIO"** test prompts the user
1) for input&justof: the CI specification discloses a problem, P, to be solved by TT.0 of CI;
2) for input&justof: S alias TT.0 solves P;
3) for input&justof: P is not solved, if in S a BED-inC0k is relaxed (i.e. the truth set of a BED-inC0k is enlarged);

If 1)-3) apply, then <CI,S> is "not an abstract idea only".

[**] The **"Not Anticipated And Not Obvious, NANO"** test checks of RS all its "anticipation combinations, ACs" as to S [5,6]:
1) It starts from the "anticipation/non-anticipation, AN" matrix of FSTP test.6, any one of the I+1 lines of which shows, by its K column entries, for i = 1,2,...,I, which of the peer TT.0 entries is anticipated/non-anticipated by a former one, and for i=0 is anticipated/non-anticipated by posc.
2) It automatically derives from the AN matrix the set of all {AC} with the minimal number, Q^{plcs}, of "N" entries.

[1] S. Schindler: "US Highest Courts' Patent Precedents in Mayo/Myriad/CLS/Ultramercial/LBC: 'Inventive Concepts' Accepted – 'Abstract Ideas' Next? Patenting Emerging Tech. Inventions Now without Intricacies".

[2] AIT, "Advanced Information Technology", denotes topical IT research areas, e.g., AI, KR, DL, NL, Semantics, …

[5] S. Schindler: "Math. Model. Substantive Patent Law (SPL) Top-Down vs. Bottom-Up", Yokohama, JURISIN 2013.

[6] S. Schindler: "FSTP" pat. appl.: "THE FSTP Expert System", 2012.

[7] S. Schindler: "DS" pat. appl.: "An Innovation Expert System, IES, & Its Data Structure, PTR-DS", 2013.

[10] SSBG's AB to CAFC in LBC, 2013.

[11] S. Schindler: "inC" pat. appl.: "Inventive Concepts Enabled Semi-Automatic Tests of Patents", 2013.

[18] SSBG AB to the Supreme Court in CLS, 07.10.2013.

[19] SSBG AB to the Supreme Court in WildTangent, 23.09.2013.

[25] S. Schindler, A. Paschke, and S. Ramakrishna: "Formal Legal Reasoning that an Invention Satisfies SPL", Bologna, JURIX-2013.

[36] S. Schindler: "Boon and Bane of Inventive Concepts and Refined Claim Construction in the Supreme Court's New Patent Precedents", Berkeley, IPSC, 08.08.2014.

[43] S. Schindler: "LAC" pat. appl.: „Semi-Automatic Generation/Customization of (All) Confirmative Legal Argument Chains (LACs) in a Claimed Invention`s SPL Test, as Enabled by Its Inventive Concepts", 2014.

[45] SSBG's AB to the Supreme Court as to the CII Question, 28.01. 2014.

[46] S. Schindler: "Autom. Deriv. of Leg. Arg. Chains (LACs) from Arguable Subtests (ASTs) of a Claimed Invention's Test for Satisfying. SPL", University of Warsaw, 24.05.2014.

[47] S. Schindler: "Auto. Generation of All ASTs for an Invention's SPL Test", submitted for publication.

[59] S. Schindler: "UI" pat. appl.: "An IES Capable of Semi-Auto. Generating/Invoking All Legal Argument Chains (LACs) in the SPL Test of a Claimed Invention (CI), as Enabled by Its Inventive Concepts (inCs)", 2014.

[72] .a) S. Schindler: "The Supreme Court's 'SPL Initiative': Scientizing Its SPL Interpretation Removes 3 Evergreen SPL Obscurities", Press Release, 08.04.2014.

.b) S. Schindler: "The Supreme Court's 'SPL Initiative': Scientizing Its SPL Interpretation Removes 3 Evergreen SPL Obscurities – and Enables Automation in a CI's SPL Tests and Argument Chains", Honolulu, IAM2014S, 18.07.14.

[76] D. Crouch: "En Banc Federal Circuit Panel Changes the Law of Claim Construction", 13.07.2005.

[79] S. Schindler: "On the BRI-Schism in the US National Patent System (NPS) – A Challenge for the US Highest Courts", 22.05.2014, submitted for publication

[136] S. Schindler: "High-Level Tutorial I to Claimed Inventions' post-*Mayo* SPL Testing", submitted for publication.

 ISBN: 978-1-61208-408-4

[137] S. Schindler: "The Rationality of a Claimed Invention's (CI's) post-*Mayo* SPL Test – It Increases CI's Legal Quality and Professional Efficiency in CI's Use, and Stimulates/Inspires the Inventivity to/in CI's Further Development", in preparation.

 ISBN: 978-1-61208-408-4

Applying Fuzzy weights to Triple Inner Dependence AHP

Shin-ichi Ohnishi Takahiro Yamanoi

Faculty of Engineering
Hokkai-Gakuen University
Sapporo, Japan
email: {ohnishi, yamanoi}@hgu.jp

***Abstract* - Analytic Hierarchy Process (AHP) is major method for decision making, and inner dependence AHP is used for cases in which criteria or/and alternatives are not independent enough. Using the original AHP or inner dependence AHP may cause results that cannot have enough reliability because of the inconsistency of the comparison matrix. In such cases, fuzzy representation for weighting criteria or/and alternatives using results from sensitivity analysis is useful. In the previous papers, we defined local weights of criteria and alternatives and overall weights for double inner dependence AHP (among criteria and among alternatives, respectively) via fuzzy sets. In this paper, we extend these weighs to those of triple inner dependence structure AHP (among 2 levels of criteria and among alternatives, respectively).**

Keywords - AHP; fuzzy sets; sensitivity analysis.

I. INTRODUCTION

The Analytic Hierarchy Process (AHP) proposed by T.L. Saaty in 1977 [1] is widely used in decision making, because it reflects humans feelings naturally. A normal AHP assumes independence among criteria and alternatives, although it is difficult to choose enough independent elements. The inner dependence method AHP [3] is used to solve this problem even for criteria or alternatives having dependence.

A comparison matrix may not have enough consistency when AHP is used because, for instance, a problem may contain too many criteria or alternatives for decision making, meaning that answers from decision-makers, i.e., comparison matrix components, do not have enough reliability, they are too ambiguous or too fuzzy [3]. To avoid this problem, we usually have to revise again or abandon the data, but it takes a lot of time and it is expensive [1][2].

Then, we consider that weights should also have ambiguity or fuzziness. Therefore, it is necessary to represent these weights using fuzzy sets. Our research first applied sensitivity analysis to inner dependence AHP to analyze how much the components of a pairwise comparison matrix influence the weights and consistency of a matrix [4]. This may enable us to show the magnitude of fuzziness in weights. We previously proposed new representation for criteria and alternatives weights for inner dependence, as L-R fuzzy numbers [5]. In the next step, we address the double inner dependence structure [7]. Then, we consider composition of weights to obtain over all alternative weights for double inner dependence structure, using results from sensitivity analysis and fuzzy operations. At last, we apply these fuzzy weighs to a result of triple inner dependence (among 2 levels of criteria and among alternatives respectively) when comparison matrices in all levels do not have enough consistency.

In Sections 2 and 3, we introduce the inner dependence AHP, consistency index, and sensitivity analyses for AHP. Then, in Section 4, we define fuzzy weights, and Section 5 is a summary.

II. CONSISTENCY AND INNER DEPENDENCE

A. Process of Normal AHP

(Process 1) Representation of structure by a hierarchy. The problem under consideration can be represented in a hierarchical structure. At the middle levels, there are multiple criteria. Alternative elements are put at the lowest level of the hierarchy.

(Process 2) Paired comparison between elements at each level. A pairwise comparison matrix A is created from a decision maker's answers. Let n be the number of elements at a certain level, the upper triangular components of the matrix a_{ij} ($i< j = 1,\ldots,n$) are 9, 8, .. , 2, 1, 1/2, …, or 1/9. These denote intensities of importance from element i to j. The lower triangular components a_{ji} are described with reciprocal numbers, for diagonal elements, let $a_{ii} = 1$.

(Process 3) Calculations of weight at each level. The weights of the elements, which represent grades of importance among each element, are calculated from the pairwise comparison matrix. The eigenvector that corresponds to a positive eigenvalue of the matrix is used in calculations throughout in the paper.

(Process 4) Priority of an alternative by a composition of weights. With repetition of composition of weights, the overall weights of the alternative, which are the priorities of the alternatives with respect to the overall objective, are finally found**.**

B. Consistency

Since components of the comparison matrix are obtained by comparisons between two elements, coherent consistency is not guaranteed. In AHP, the consistency of the

 ISBN: 978-1-61208-408-4

comparison matrix A is measured by the following consistency index (C.I.)

$$\text{C.I.} = \frac{\lambda_A - n}{n-1}, \tag{1}$$

where n is the order of comparison matrix A, and λ_A is its maximum eigenvalue (Frobenius root).

If the value of C.I. becomes smaller, then the degree of consistency becomes higher, and vice versa. The comparison matrix is consistent if the following holds.

$$\text{C.I.} \leq 0.1 \tag{2}$$

C. Inner Dependence Structure

The normal AHP ordinarily assumes independence among criteria and alternatives, although it is difficult to choose enough independent elements. The dependency means some kind of interaction among the elements. Inner dependence AHP [2] is used to solve this type of problem even for criteria or alternatives having dependence.

In the method, using a dependency matrix $F=\{ f_{ij} \}$, we can calculate modified weights $\boldsymbol{w}^{(n)}$ as follows,

$$\boldsymbol{w}^{(n)}=F\boldsymbol{w} \tag{3}$$

where $\boldsymbol{w}$ represents weights from independent criteria or alternatives, i.e., normal weights of normal AHP and dependency matrix F is consist of eigenvectors of influence matrices showing dependency among criteria or alternatives.

If there is dependence in both lower levels, i.e., not only among criteria but also among alternatives, we call such kind of structure "double inner dependence". In the double inner dependence structure, we have to calculate modified weights of criteria and alternatives, $\boldsymbol{w}^{(n)}$ and $\boldsymbol{u}_i^{(n)}$. Then we composite these 2 modified weights to obtain overall weights of alternative k, $v_k^{(n)}$ as follow:

$$v_k^{(n)} = \sum_i^m w_i^{(n)} u_{ik}^{(n)} \tag{4}$$

where m is the number of criteria.

Also, using the same steps again, we can composite weights of "triple inner dependence" structure, in the case when there is dependency in the 3 lower levels, i.e., not only among alternatives and 1 level criteria but also 2 levels of criteria.

III. Sensitivity Analyses

When we actually use AHP, it often occurs that a comparison matrix is not consistent or that there is not great difference among the overall weights of the alternatives. In these cases, it is very important to investigate how components of the pairwise comparison matrix influence its consistency or the weights. In this study, we use a method that some of the present authors have proposed before. It evaluates a fluctuation of the consistency index and the weights when the comparison matrix is perturbed. It is useful because it does not change the structure of the data.

Since the pairwise comparison matrix is a positive square matrix, Perron-Frobenius theorem holds. From Perron-Frobenius theorem, the following theorem about a perturbed comparison matrix holds.

Theorem 1 *Let $A = (a_{ij})$, $(i, j = 1,..., n)$ denote a comparison matrix and let $A(\varepsilon) = A + \varepsilon D_A$, $D_A=(a_{ij}d_{ij})$ denote a matrix that has been perturbed. Let λ_A be the Frobenius root of A, $\boldsymbol{w}$ be the eigenvector corresponding to λ_A, and $\boldsymbol{v}$ be the eigenvector corresponding to the Frobenius root of A'. Then, a Frobenius root $\lambda(\varepsilon)$ of $A(\varepsilon)$ and a corresponding eigenvector $\boldsymbol{w}(\varepsilon)$ can be expressed as follows*

$$\lambda(\varepsilon) = \lambda_A + \varepsilon\lambda^{(1)} + o(\varepsilon), \tag{5}$$

$$\boldsymbol{w}(\varepsilon) = \boldsymbol{w} + \varepsilon\boldsymbol{w}^{(1)} + \boldsymbol{o}(\varepsilon), \tag{6}$$

where

$$\lambda^{(1)} = \frac{\boldsymbol{v}' D_A \boldsymbol{w}}{\boldsymbol{v}' \boldsymbol{w}}, \tag{7}$$

$\boldsymbol{w}^{(1)}$ is an n-dimension vector that satisfies

$$(A - \lambda_A I)\boldsymbol{w}^{(1)} = -(D_A - \lambda^{(1)} I)\boldsymbol{w}\ , \tag{8}$$

where $\boldsymbol{o}(\varepsilon)$ denotes an n-dimension vector in which all components are $o(\varepsilon)$.

About a fluctuation of the consistency index, the following corollaries hold.

Corollary 1 *Using appropriate g_{ij}, we can represent the consistency index* C.I.(ε) *of the perturbed comparison matrix $A(\varepsilon)$ as follows*

$$\text{C.I.}(\varepsilon) = \text{C.I.} + \varepsilon \sum_i^n \sum_j^n g_{ij} d_{ij} + o(\varepsilon). \tag{9}$$

To see g_{ij} in the equation (9) in Corollary 1, we can know how the components of a comparison matrix impart influence on its consistency.

Corollary 2 *Using appropriate $h_{ij}^{(k)}$, we can represent the fluctuation $\boldsymbol{w}^{(1)}=(w_k^{(1)})$ of the weight* (*i.e., the eigenvector corresponding to the Frobenius root*) *as follows*

$$w_k^{(1)} = \sum_i^n \sum_j^n h_{ij}^{(k)} d_{ij}. \quad (10)$$

Then, we can evaluate how the components of a comparison matrix impart influence on the weights, to see $h_{ij}^{(k)}$ in the equation (10).

Proofs of these corollaries are shown in [4].

IV. Fuzzy Weights Representations

When a comparison matrix has poor consistency (i.e., 0.1<C.I.<0.2), comparison matrix components are considered to be fuzzy because they are results from human fuzzy judgment. Weights should therefore be treated as fuzzy numbers [5][6].

Definition 1 (fuzzy weight) Let $w_k^{(n)}$ be a crisp weight of criterion or alternative k of inner dependence model, and $g_{ij} \mid h_{ij}^{(k)}\mid$ denote the coefficients found in Corollary 1 and 2. If 0.1<C.I.<0.2, then a fuzzy weight $\tilde{w}_k$ is defined by

$$\tilde{w}_k = (w_k, \alpha_k, \beta_k)_{LR} \quad (11)$$

$$\alpha_k = \text{C.I.} \sum_i^n \sum_j^n s(-, h_{kij}) g_{ij} \mid h_{kij} \mid, \quad (12)$$

$$\beta_k = \text{C.I.} \sum_i^n \sum_j^n s(+, h_{kij}) g_{ij} \mid h_{kij} \mid, \quad (13)$$

For double inner dependence structure, we can define and calculate modified fuzzy local weights of a criteria $\tilde{\boldsymbol{w}}^{(n)} = (\tilde{w}_i^{(n)})$, $i = 1, ..., n$ and also weights of alternatives $\tilde{\boldsymbol{u}}_i^{(n)} = (\tilde{u}_{ik}^{(n)})$, $k = 1, ..., m$ with only respect to criterion i using an dependence matrix F_C, F_A, as follows

$$\boldsymbol{w}^{(n)} = (w_i^{(n)}) = F_C \boldsymbol{w} \quad (14)$$

$$\boldsymbol{u}_i^{(n)} = (u_{ik}^{(n)}) = F_A \boldsymbol{u}_i \quad (15)$$

$\boldsymbol{w}$ is crisp weights of criteria and $\boldsymbol{u}_i$ is crisp local alternative weights with only respect to criterion i. $\alpha_i, \beta_i, \alpha_{ik}, \beta_{ik}$ are calculated by fuzzy multiple operations, using (3) and Definition 1.

For triple inner dependence structure, we can also define overall weights of alternatives $\tilde{y}_k^{(n)}$, as follows:

$$\tilde{y}_k^{(n)} = \sum_j^l x_i^{(n)} \tilde{v}_{jk}^{(n)} \quad (16)$$

V. Conclusions

There are many cases in which data of AHP does not have enough reliability. We show the possibility to apply fuzzy weight representation to triple inner dependence. Our approach can show how to represent weights and will be efficient to investigate how the result of AHP has fuzziness when data is not sufficiently consistent.

References

[1] T. L. Saaty, The Analytic Hierarchy Process. McGraw-Hill, New York, 1980.

[2] T. L. Saaty, Inner and Outer Dependence in AHP, University of Pittsburgh, 1991

[3] S. Ohnishi, D. Dubois, H. Prade, and T. Yamanoi, "A Fuzzy Constraint-based Approach to the Analytic Hierarchy Process," Uncertainty and Intelligent Information Systems, June 2008, pp.217-228.

[4] S. Ohnishi, H. Imai, and M. Kawaguchi, "Evaluation of a Stability on Weights of Fuzzy Analytic Hierarchy Process using a sensitivity analysis," J. Japan Soc. for Fuzzy Theory and Sys., 9(1), Jan. 1997, pp.140-147.

[5] S. Ohnishi, T. Yamanoi, and H. Imai, "A Fuzzy Weight Representation for Inner Dependence AHP," Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.15, No.3, June 2011, pp. 329-335.

[6] S. Ohnishi, T. Furukawa, and T. Yamanoi, " Compositions of Fuzzy Weights for Double Inner Dependence AHP," COGNITIVE2013, May 2013, pp. 83-86.

 ISBN: 978-1-61208-408-4

Near Real-time Synchronization Approach for Heterogeneous Distributed Databases

Hassen Fadoua, Grissa Touzi Amel
LIPAH
FST, University of Tunis El Manar
Tunis, Tunisia
e-mail: hassen.fadoua@gmail.com, amel.touzi@enit.rnu.tn

Abstract—**The decentralization of organizational units led to database distribution in order to solve high availability and performance issues regarding the highly exhausting consumers nowadays. The distributed database designers rely on data replication to make data as near as possible from the requesting systems. The data replication involves a sensitive operation to keep data integrity among the distributed architecture: data synchronization. While this operation is necessary to almost every distributed system, it needs an in depth study before deciding which entity to duplicate and which site will hold the copy. Moreover, the distributed environment may hold different types of databases adding another level of complexity to the synchronization process. In the near real-time duplication process, the synchronization delay is a crucial criterion that may change depending on querying trends. This work is intended to establish a standard synchronization process between different sites of a distributed database architecture including database heterogeneity, variable synchronization delays, network capability restrictions and fault management ability.**

Keywords-replication; synchronization; distributed databases; heterogeneous databases

I. INTRODUCTION

A distributed database system is defined as a collection of interconnected sites geographically stretched, but logically related. The design of a distributed database (DDB) may start from scratch to install a new environment of work. It may also start from an existing environment of isolated "data islands". The bottom-up approach on a distributed databases design starts with an existing environment logically separated that must be unified behind a distributed database system (DDBMS). Throughout this unification process, many technical and foreign constraints may prevent the use of the same database system on each site (database license restriction, available operating systems, etc.). The resulting distributed architecture is a heterogeneous environment. A vital feature in such systems is the synchronization process among the different copies of an entity. In relational databases, it is always a matter of data tables updates. However, a table definition may differ from a DBMS to another. This is the case for data types, for example. A simple string is not stored and defined in PostgreSql [18] for example as in Firebird [14]. The blob data type usually used to store binary files inside a table column is not available in the list of the existing relational database management systems (RDBMS). This storage unit's variety holds the first problem discussed in this paper: data exchange format between the heterogeneous nodes in the same Distributed Database Management System (DDBMS). The process of transforming raw data from source system to an exchange format is called serialization. The reverse operation, executed by the receiver side, is called deserialization. In the heterogeneous context, the receiver side is different from one site to another, and thus the deserialization implementation may vary depending on the local DBMS. This data serialization is a very sensitive task that may speed down the synchronization process and flood the network. Moreover, the data integrity may be altered if the relation between serialization and deserialization is not symmetric.

The exhaustive nature of nowadays users also inserts a new level of complexity toward building a standard protocol. The user must have one-copy view of the database and hence the correctness criterion known as 1-Copy Serializability (1SR). In order to afford the same result given by a centralized system to the end-users, the duplicate copies of the physical data must be synchronized in the near real-time scale between all the nodes. The real-time copy updates is so far impossible in large scale systems due to write lock concurrency problems and network latency. A Delayed update process [8] can solve the issue if the exchange protocol considers a proper synchronization interval and reduces the local operation delays. The Brewer's theorem [11] [17] states that a database cannot simultaneously guarantee consistency, availability, and partition tolerance [9]. Thus, to achieve partition tolerance and availability strong consistency must be sacrificed. The eventual-consistency mechanism can improve the availability by providing a weakened consistency for example.

In this work, we establish an exchange protocol for duplicate entities in a distributed environment based on a rotating pivot. This pivot must not be a bottleneck so it does not block the full process on a single node failure. In such cases, another pivot or leader will be elected by subscribed nodes on the synchronization process. The frequency of exchange operation and the different intervals are studied in both most pessimistic and most optimistic scenario in order to suggest the correct value for this protocol.

 ISBN: 978-1-61208-408-4

Besides this introduction, this paper includes five sections. Section 2 describes the different approaches and their limitations. Section 3 presents the suggested synchronization protocol. Section 4 details the data serialization and the deserialization mechanism regarding network bandwidth consumption and packet preparation time impact. Section 4 studies the protocol performance on the most optimistic and pessimistic scenarios. Section 5 reports the experimental tests of the implemented example and discusses different aspects of the result. Section 6 summarizes the result of this work and opens the future perspectives for it.

II. RELATED WORKS

A. Distributed databases synchronization

Along with the spread of distributed database systems, many strategies were adopted by designers in order to keep the distributed copies of an entity up-to-date. We are not going to deal with the plain synchronization protocols such as SyncML [15] and DRBDB because they better fit files synchronization [19]. In any database, synchronizing data files does not necessary ensure that the user will have the same view of data. This fact is due to the programmatic nature of a DBMS: Any update must be traced either in the transaction manager history or the archive log and maintained in a memory registry. As a consequence, a flat copy of data files may corrupt the database.

The implemented approaches may be divided into two classes: optimistic and pessimistic. The pessimistic family operates as Read-One-Write-All (ROWA) [12]. If any site from the topology is not available, the update will not be written to any site. This is the main reason behind considering this family as pessimistic. The processing method replicates eagerly on all sites. The read operation can be done from any replicated copy but the write operation must be executed at all replicas [21].

In contrast, the optimistic class holds the Read-One-Write-All Available (ROWA-A) [1] family. This class of algorithms offers a flexible strategy regarding fault tolerance, providing more flexibility in presence of failures. Any site can read any replicated copy, but writes to all available replicas (if any site is not available ROWA cannot proceed but ROWA-A will still continue with write).

The exhausting consumers and the huge amounts of data are still the main challenges. Dispatching the effort between different nodes in a distributed architecture has shown a good performance but data consistency does always matter as the synchronization process implies huge efforts in active mode (all the nodes access data in read/write mode). Hence, we introduce a new protocol to solve the replica synchronization problems in a distributed environment.

B. Serialization process

The serialization process in the context of our approach may be defined as the metamorphosis of raw data from the local database format to a universal format that preserves the original information and can be processed by the symmetric reverse operation. Much work has been invested in this task. The old exchange protocols were inefficient especially from a performance point of view [9]. Google invested in Protocol Buffer and made it open source since 2007(under Berkeley Software Distribution license) [6]. The apache community is also maintaining the great THRIFT project [10] initially started in the Facebook labs. Both technologies offer a good performance regarding serialization and deserialization time [3], quite better than plain Extensible Markup Language (XML) exchange protocols [7]. Moreover, the binary format of Thrift and ProtoBuf packets reduce drastically the bandwidth consumption in a network without altering processing time.

III. THE DSYNC APPROACH

In this work, we introduce a new approach acting on the applicative layer. Each successful operation executed to a local database on a domain object is kept inside a queue. Each node has its own queue (Figure 1). A dynamically chosen node (the master) leads the objects' exchange protocol in a way to propagate all the queues items to all the subscribed servers. This particular node also acts as a man-in-the-middle [22] in network hacking context. The exchange protocol is a set of scheduled events as in [4] and [5], triggered either in the leader side or in the secondary nodes (clients).

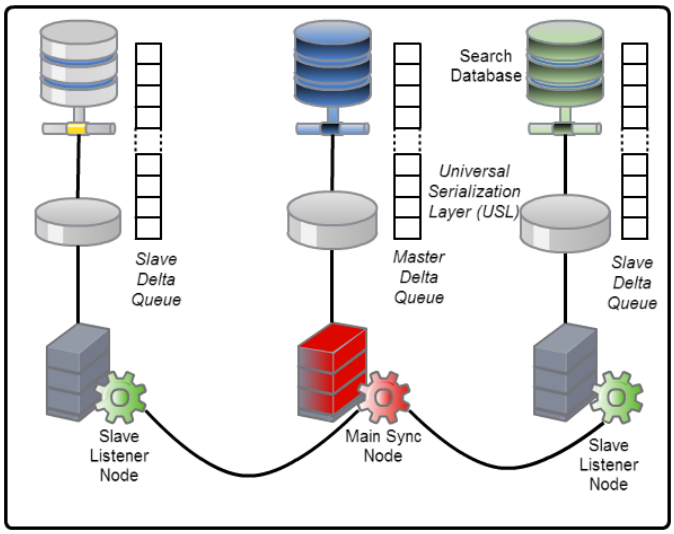


Figure 1. Dsync based approach

A. Dynamic Leader Node election

The leader node is a single network entity chosen by all other entities to coordinate, organize, initiate and sequence different tasks among the distributed architecture. This node simplifies updates propagation over the distributed topology. However, it creates a single point of failure as a

leader hang will limit the synchronization service [16]. Thus, electing this specific node dynamically provides a solution for the single point of failure.

Several algorithms (LeLann-Chang-Roberts, Hirschberg and Sinclair) were elaborated to establish the most efficient method for election [2]. In this work, we can consider a new facility to accelerate the election process. In a distributed database context, adding nodes is not executed as often as in a mobile network for example. To add a server in a distributed context, many settings must be prepared before adding a backup or a rescue server to the architecture. In the configuration process, each server is given, manually, a unique identifier over the network (may be its IP address) and a sequence number (SN). The server with the lowest SN is the most eligible server to be the leader (Figure 2). The "Leader" election procedure is executed during the system initialization and on the current leader failure. The failure condition is relative to the distributed application context. In this work, we consider the first communication problem for all the nodes as a leader failure. When a node joins the network with an active synchronization service, the leader node information is given arbitrarily targeting any reachable node on the distributed architecture. On the first inquiry, if the described node is the active leader node, the new server saves the information. The leader adds the new server to the topology and propagates the information to the rest of the nodes. If the target server is not the active leader, the answer is rerouting the new node to the correct leader.

The complete topology is held by every node and is synchronized on real time. Thus, on leader node failure, the successor is automatically elected based on the SN criteria. A minimal node object is described by its SN, an access URL (protocol, IP, port, root context), the synchronization objects contract, items to generate and the last synchronization items treated on initialization. The synchronization objects' contract is the list of items to which the node is configured to listen for synchronization. A node can subscribe to all the items updates or limit its subscription to some items only. In the other side of the contract, the node is asked to generate a list of changes for all the nodes of the topology.

Algorithm 1 : Dynamic Leader Node Election on plug

Require: *Predefined Leader List LL, ordered by weight*

1. **L** ← *send_join_query*(**L**.get_first(), current_node)
2. If(L is NULL)
   a. While ((LL.has_next()) and (L is NULL))
      i. Li ← LL.get_next()
      ii. **L** ← *send_join_query*(Li, current_node)
3. If (L is NULL)
   a. **L** ← current_node

Figure 2. Dynamic Leader Node Election Algorithm on plug

B. Secondary nodes subscription

After adding a server to the network, the new node can ask for registration on the synchronization service. This query may be achieved via a public web service exposed by every node [20]. The subscription query is sent to the active leader including the server unique identifier and the desired items for synchronization. The leader acknowledges the new node by the result of its subscription and if the synchronization contract contains a new entity to synchronize, it is added to the list of synchronization data for generation (Figure 3).

Algorithm 2 : Secondary Node Subscription

Require: *Leader Node L, a subscription contract ctr, a node definition* N_i

1. *contract* ← handle_subscription_query(L, N_i, ctr)
2. *new_items* ← diff(contract.sync_entities, L.get_sync_entities())
3. if(new_items is not NULL)
   a. add(new_items,L.get_sync_entities)
   b. propagate_changes(new_items)

Figure 3. Secondary Node Subscription Algorithm

C. Synchronization data generation

In this work, an exchanged message containing a record update is called "Dsync". A Dsync holds the serialized object in its latest state, the source server of the update and a timestamp. The detailed specification of the Dsync is summarized after the synchronization process specification. Each entity on the database must be represented in the application level by a Dsync custom implementation. Only the serialization process will differ from an entity to another. Dsyncs are generated on a record creation, update or delete. The created Dsyncs are persisted in a database queue to ensure an acceptable level of traceability and allow the reconstruction process after a critical failure. By making the queue persistent, the system administrator is able to restore the database state based on a checkpoint and then execute all the "Dsyncs" correctly after fixing the problem that caused the hang or the failure.

D. Dsync execution

Periodically (every D_{EX} milliseconds), an integration process will collect the non-processed hits from the "Dsync" table (the queue) where the source server is different from the requesting server itself. "Dsync" processing is always in timestamp order. The same applicative procedure used on each server will be used to perform insert, update and delete. It is essential to perform all those actions using the same applicative procedure and not directly using the database triggers.

 ISBN: 978-1-61208-408-4

The deserialization process impact is observed mainly in this step. Serializing data may be efficient and very fast, but transforming the binary data to an adapted copy of original information may add a considerable duration to each exchange sequence. This is especially true when lots of format conversion is needed to transform a field from the source node format to the target database format.

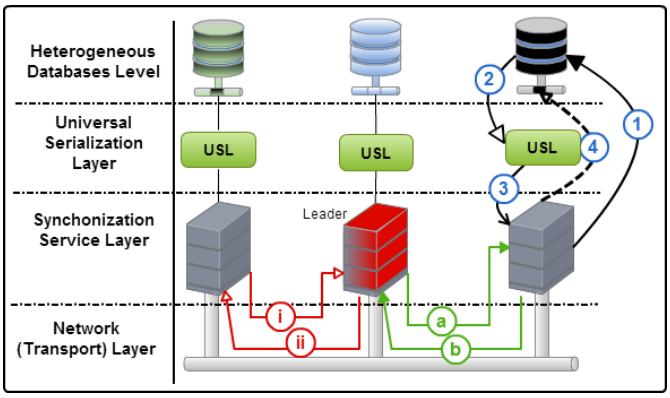


Figure 4. Sample synchronization workflow in the Dsync approach

E. Dsync exchange protocol

The running queries on the network in the established communication protocol can be divided into two groups: Leader initiated and Slave initiated. In Figure 4, the leader initiated calls are represented in green color (a, b). The slave's initiated ones appear in red color (i, ii). A local synchronization call is traced in black (1,2,3) and described in Figure 7 (Algorithm 5). Finally, the call number (4) is a Dsync execution command.

F. Leader initiated workflow

Periodically (each Leader Node Delay D_{LN}), the leader node sends queries to all the subscribed nodes asking for any updates (Figure 5). When a slave node receives a query from the leader, it fetches eligible "Dsyncs" from its database. To avoid flooding the network, the maximum number of "Dsyncs" to send in one response is a custom configuration value (Maximum Packet Size *MPS*). An eligible "Dsync" to send to the leader is any locally generated "Dsync" (source server is the current node). On leader response handling, the "Dsyncs" are saved into the main queue ordered by timestamp. The leader then acknowledges the source server by accepted items. Rejected line processing is discussed in the subsection (III.H). The accepted items are flagged as transferred to the leader on the source server queue (Figure 6).

Algorithm 3 : Leader RFN – CRON- triggered

Require: Subscribed Nodes List LL

1. *for (Li in LL)*
   a. serialized_packet ← request_for_updates(Li)
   b. updated_rows_ Ii← deserialize(serialized_packet)
   c. updated_rows.append(updated_rows_ i)
   d. ACK(Li,updated_rows_i.size())
2. end For
3. arrange_Items_by_timestamp(updated_rows)
4. save(updated_rows)

Figure 5. Leader RFN – CRON- triggered Algorithm

Algorithm 4 : Slave Request Processing (*SRPi)*

Require:

1. updated_rows ←*Fetch at most MPS rows from local Dsync where "target server ids" contains L.getId() and " source node" equals S.getId()*
2. packet_to_send ←serialize (updated_rows)
3. ack←answer_leader(packet_to_send)
4. if(ack.count() equals updated_rows.count())
   a. mark_as_delivered(updated_rows)

Figure 6. Sample synchronization workflow in the Dsync approach

G. Slaves initiated workflow

Periodically (each Secondary Node Delay D_{SN}), a slave asks the leader server for related updates (Figure 7). When the leader receives this type of queries, it fetches from database the Dsyncs that were not generated by the querying slave, not yet communicated to it and already mentioned in its subscription contract (Figure 8). The subscription contract holds the entity names that must be synchronized with a server. The leader responses must not hold more than MPS Dsyncs. When the agent receives the response from the leader, it persists them into its queue and acknowledges the server by the successfully received items. The non-leader node flags the successful Dsyncs as transferred to the leader with a timestamp. The obvious scenarios would be removing them from the queue in order to keep it in a workable size. In this scenario, it is advised to keep this queue as long as possible in order to restore the whole system in case of functional failure.

Algorithm 5 : Slave RFN – CRON- triggered

Require: Slave Node S

1. serialized_packet ← request_for_updates(S)
2. updated_rows← deserialize(serialized_packet)
3. ACK(L,updated_rows.size())
4. arrange_Items_by_timestamp(updated_rows)
5. mark_as_processed(updated_rows, false)
6. save(updated_rows)

Figure 7. Slave RFN – CRON- triggered Algorithm

Algorithm 6 : Leader Request Processing

Require: Slave Node S

5. updated_rows ←*Fetch at most MPS rows from local*

```
    Dsync where "target server ids" contains S.getId()
6.  packet_to_send ←serialize (updated_rows)
7.  ack←answer_agent(packet_to_send)
8.  if(ack.count() equals updated_rows.count())
    a. mark_as_delivered(updated_rows)
```

Figure 8. Leader Request Processing Algorithm

The mark_as_delivered function updates the target servers IDs field to remove the node that acknowledged the reception of this item. We denote the processing duration for an update query "$Pers_{i''}$. The Dsync object can be described by this minimal list of attributes:

- Source Node ID: The id of the server where the Dsync was generated.
- Serialized Data: The content of the object to propagate (last state).
- Generation timestamp: The moment when the Dsync was generated.
- Delivery timestamp: The timestamp when the Dsync reached a node.
- Processing timestamp: The timestamp just after Dsync processing
- Target server Ids: Comma separated list of separated servers IDs.
- Processed: indicates whether the Dsync was integrated onto database or not
- Operation Type: INSERT, UPDATE, DELETE
- Target Domain Object: Name of the target object

H. Fault Management

The application has two functional modes: permissive and strict mode. In the permissive mode, the principle is to ignore and continue. In the strict mode, the synchronization is suspended on the first failure. The leader is informed by the problem. The protocol recommends a data destructive behavior and all the updates after the rollback checkpoint (failure item) are erased [11].

IV. SYNC INTERVALS EVALUATION

Several terms are considered in order to calculate the propagation delay over the whole topology in the most pessimistic and the most optimistic scenario. In the first scenario, in a topology made of *N* nodes, the propagation delay is :

$$T0 + D_{LN} + D_{SN} + (N * SRP_i) + ND + DEX + Pers_i$$

where ND is the network delay introduced.

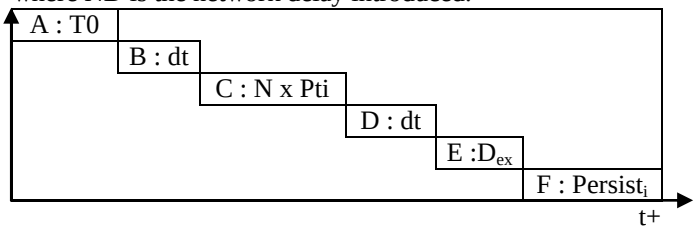


Figure 9. Dsync lifecycle steps and time evaluation

The steps of the exchange operation (Figure 9) can be described as:

A: Sending update requests to the leader from all the subscribed servers.
B: Network latency.
C: Packet making duration required by the server.
D: Network Latency.
E: Dsync processing trigger interval.
F: Dsync processing time.

This delay equation may be reduced to:

$$D_{max} = C + E + F$$

In the most pessimistic scenario, the maximum duration to propagate a change to all the topology is D_{max}. This result is considerable because it does not depend on network size.

V. IMPLEMENTATION AND EVALUATION

In this section, we describe the implementation of the Dsync protocol and discuss the experimental results in a test environment.

A. Implementation

We implemented the prototype running the Dsync protocol as a web application using Java 7 backend (Spring Framework, JSF, Quartz API). The transport protocol is HTTP1.1. The benchmark is a telecom operator directory database. The simulated use case describes the synchronization of all clients' information between the different systems of the operator. For example, if the technical team installed a new phone land line for a client, this telephone number must be available to the different contributors as soon as possible (directory service, billing, marketing, etc). The definition of "as soon as possible" varies between the different systems depending on their criticality. The more a system is critical, the smaller a server D_{SN} must be. We consider the billing system as the most critical system, and thus D_{SN} was defined to 2 seconds. The leader node packet size is set up to 500 Dsyncs. The current topology holds four (4) nodes with different RDBMS behind: Oracle 11g, MySQL 6, Firebird 2 and PostgreSQL 9. The four nodes are virtual machines (VM Player) distributed between two physical servers (laptops with i7 CPU and 8GB of memory). To simulate a realistic load, we proceeded in 3 steps. The first step starts with a pre-filled database with 1 million rows in the leader node, and empty databases for the three other nodes. To synchronize the rest of the nodes, a thread generates every 5 seconds 10000 dsyncs with "INSERT" in the operation type field. The three nodes are subscribed for the synchronization of the generated object in the leader. Once the four databases are filled with all the records, we built a SQL script to update addresses of 100000 rows and generated the "UPDATE" Dsyncs on each slave node. Once we reached a distributed synchronization state, we shutdown the initial leader and we generated 1000 delete dsyncs on each node. The purpose of

shutting down the initial leader node is to test the leader node election process. In this small topology, the result was not significant as the operation took only 12 seconds.
The watched parameters are the Dsync propagation duration (from Generation timestamp to Processing timestamp), the bandwidth consumption, CPU consuming percentage.

B. *Experimental result evaluation*

The raw results were tuned on the smallest D_{SN} (2s) for presentation purposes only. The synchronization delay was almost invisible in Figure 10 scale. Axis value unit is seconds. Figure 10 shows the arrival time of the Dsyncs to the three subscribed nodes. "Dsync gen" curve is the generation time for each Dsync in the source node (leader). The combination of D_{SN} and the packet size for the "most critical node" results into an acceptable synchronization delay (< 50s). For non critical systems, we set the D_{SN} to 5 seconds with a packet size (MPS) of 500 Dsyncs. The propagation delay kept growing in time as the awaiting Dsyncs count in the stack grows on each Dsync generation interval. However, a distinct analysis of the running threads on the leader shows minimal CPU time. The consumed bandwidth is the lowest targeting the less critical system.

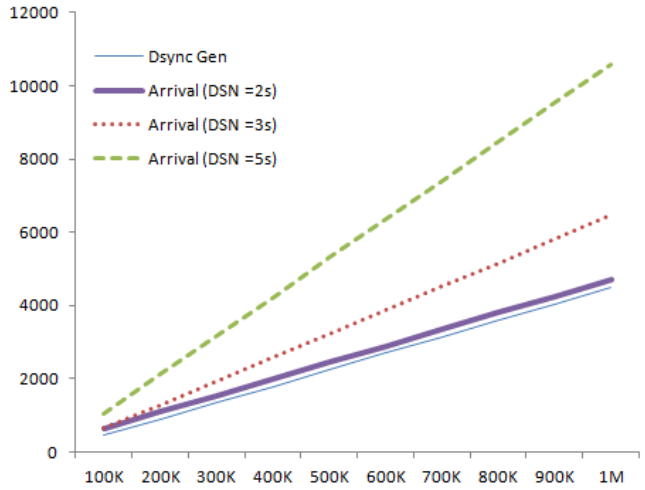


Figure 10. Experimental Result (tuned)

Figure 11 shows the different evaluation parameters and the experimental relation between the memory usage, the CPU time, the consumed bandwidth and the synchronization delay. The applied D_{SN} may be obsolete in a powerful production environment, but we are limited with material constraints. Real server may go further with smaller D_{SN} values and bigger packet size (MPS). The resulting protocol is an easy to deploy solution that offers a distributed database synchronization service where the user 1SR condition is ensured even in heterogeneous context. The solution is highly scalable and can be adopted even in heterogeneous environments. In addition to basic process, the protocol offers many custom services depending on the nodes' subscription contracts: A node can be subscribed to a subset of items only and not all domain objects (very helpful in distributed databases).

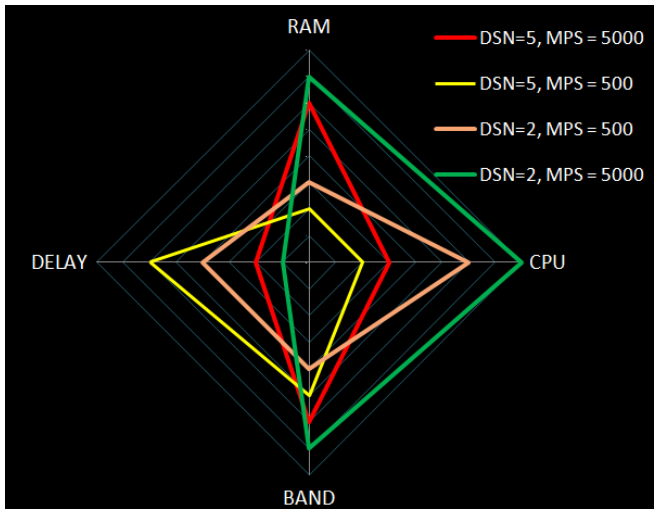


Figure 11. Watched parameters for different configurations

VI. CONCLUSION AND FUTURE WORK

The processing delays are fully customizable to be adapted to client network performances and capabilities. The bandwidth can be controlled too by setting the maximum size of items' package (queue items). The two key features in this system are 1) the completeness of the solution as no extra product licenses are needed (database synchronization tools are very expensive) and 2) the bottleneck free topology as main node can be quickly ignored on fault detection. The system is also fault tolerant and the recovery process does not need extra calculation (rollback back to the first error detected).

Further work will cover the serialization process in order to reduce packet sizes and lower consumed bandwidth. The serialization process also must be associated with a universal database access in order to overcome the diversity constraint in heterogeneous environment.

REFERENCES

[1] A. Ghaffari, N. Chechina, P. Trinder, and J. Meredith, "Scalable persistent storage for Erlang," in Proceedings of the twelfth ACM SIGPLAN workshop on Erlang - Erlang '13, 2013, pp. 73–74.

[2] A. Iványi, "Leader election in synchronous networks," Acta Univ. Sapientiae, Math., vol. 5, no. 1, pp. 54–1, Jan. 2013.

[3] A. Sumaray and S. K. Makki, "A comparison of data serialization formats for optimal efficiency on a mobile platform," in Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication - ICUIMC '12, 2012, pp. 48:1–48:6.

[4] C. M. Krishna, "Fault-tolerant scheduling in homogeneous real-time systems," ACM Comput. Surv., vol. 46, no. 4, pp. 1–34, Mar. 2014.

[5] F. Meyer, B. Etzlinger, F. Hlawatsch, and A. Springer, "A distributed particle-based belief propagation algorithm for

cooperative simultaneous localization and synchronization," in 2013 Asilomar Conference on Signals, Systems and Computers, 2013, pp. 527–531.

[6] G. Kaur and M. M. Fuad, "An evaluation of Protocol Buffer," in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, 2010, pp. 459–462.

[7] J. Delgado, "Service interoperability in the Internet of Things," Stud. Comput. Intell., vol. 460, pp. 51–87, 2013..

[8] M. M. Elbushra and J. Lindström, "Eventual Consistent Databases: State of the Art," Open J. Databases, vol. 1, no. 1, pp. 26–41, 2014.

[9] M. Raynal, Distributed Algorithms for Message-Passing Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[10] M. Slee, A. Agarwal, and M. Kwiatkowski, "Thrift: Scalable cross-language services implementation," *Facebook White Paper*, 2007. [Online]. Available: http://trillian42.homelinux.org/dir/pdfs/thrift-20070401.pdf. [Retrieved: April, 2015].

[11] M. Stonebraker, "Errors in database systems, eventual consistency, and the cap theorem," *Communications of the ACM, BLOG@ ACM*, 2010. [Online]. Available: http://db.cs.berkeley.edu/cs286/papers/errors-cacmblog2010.pdf. [Retrieved: April, 2015].

[12] N. Ahmad, R. M. Sidek, M. F. J. Klaib, and T. L. Jayan, "A Novel Algorithm of Managing Replication and Transaction through Read-one-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS)," in 2010 Second International Conference on Network Applications, Protocols and Services, 2010, pp. 20–25.

[13] O. Kononenko, O. Baysal, R. Holmes, and M. W. Godfrey, "Mining modern repositories with elasticsearch," Proc. 11th Work. Conf. Min. Softw. Repos. - MSR 2014, pp. 328–331, 2014.

[14] P. Vinkenoog, "Firebird SQL." [Online]. Available: http://www.firebirdsql.org/manual/qsg10-firebird-sql.html. [Accessed: 04-May-2015].

[15] S. Chun, S. Lee, and D. Oh, "Formal verification of SyncML protocol for ubiquitous data coherence," in Lecture Notes in Electrical Engineering, 2013, vol. 214 LNEE, pp. 415–422..

[16] S. Eken, F. Kaya, Z. Ilhan, A. Sayar, A. Kavak, U. Kocasarac, and S. Sahin, "Analyzing distributed file synchronization techniques for educational data," in 2013 International Conference on Electronics, Computer and Computation (ICECCO), 2013, pp. 318–321..

[17] S. Gilbert and N. Lynch, "Perspectives on the CAP Theorem," Computer, vol. 45, no. 2. pp. 30–36, 2012.

[18] T. G. Lockhart, "PostgreSQL: Documentation: 8.1: Character Types." [Online]. Available: http://www.postgresql.org/docs/8.1/static/datatype-character.html. [Retrieved: 04-May-2015].

[19] W. Guo, F. Liu, Z. Q. Zhao, and C. Wu, "Method and apparatus for efficient memory replication for high availability (HA) protection of a virtual machine (VM)." 02-Apr-2013.

[20] W. Zhao and Z. Xiaohong, "The Transaction Processing of Heterogeneous Databases Application in Web Service," 2012, vol. 289, pp. 140–147.

[21] Y. Tang, H. Gao, W. Zou, and J. Kurths, "Distributed synchronization in networks of agent systems with nonlinearities and random switchings," IEEE Trans. Cybern., vol. 43, no. 1, pp. 358–370, 2013.

[22] Y. Wang and Y. Li, "Heterogeneous Data Sources Synchronization Based on Man-in-the-Middle Attack," Proc. 4th Int. Conf. Comput. Eng. Networks, pp. 467–476, 2015.

 ISBN: 978-1-61208-408-4

Skyline Objectset: Efficient Selection of Complete Non-dominate Sets from Database

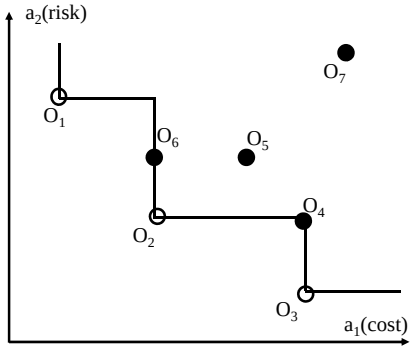
Md. Anisuzzaman Siddique, Asif Zaman, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima, Japan
Email: {siddique@, d140094@, morimoto@mis.}hiroshima-u.ac.jp

***Abstract*—A skyline query retrieves a set of non dominate objects. In this paper, we consider a skyline query for sets of objects (objectsets) in a database. Since a skyline query of objectsets is important in portfolio analysis, privacy aware data analysis, outlier-resistant data analysis, etc., we have previously considered "convex skyline objectsets query", in which we did not select some of skyline objectsets that is not on convex hull. To solve the shortcoming, we propose an efficient algorithm to select complete skyline objectsets in this paper. We investigated the properties of objectset skyline computation and develop two major pruning conditions to avoid unnecessary objectset enumerations as well as comparisons among them. We conduct a set of experiments to show the meaningfulness and scalability of the proposed skyline objectset algorithm.**

Keywords–Dataset; Skyline queries; Objectsets; Dominance relationship.

ID	a_1(cost)	a_2(risk)
O_1	2	8
O_2	4	4
O_3	8	2
O_4	8	4
O_5	6	6
O_6	4	6
O_7	10	10

a) Dataset



b) Skyline

Figure 1. A skyline problem

I. INTRODUCTION

Skyline query [1] and its variants are functions to find representative objects from a numerical database. Given a m-dimensional dataset D, an object O is said to dominate another object O' if O is not worse than O' in any of the m dimensions and O is better than O' in at least one of the m dimensions. A skyline query retrieves a set of non dominate objects. Consider an example in the field of financial investment. In general, an investor tends to buy the stocks that can minimize cost and risk. Based on this general assumption, the target can be formalized as finding the skyline stocks with smaller costs and smaller risks. Figure 1 (a) shows seven stocks records with their costs (a_1) and risks (a_2). In the list, the best choice for a client comes from the skyline, i.e., one of $\{O_1, O_2, O_3\}$ in general (see Figure 1 (b)).

A key advantage of the skyline query is that it does not require a specific ranking function; its results only depend on the intrinsic characteristics of the data. Furthermore, the skyline is not affected by potentially different scales at different dimensions (risk unit or cost unit in the example of Figure 1); only the order of the dimensional projections of the objects is important. Skyline query has broad applications including product or restaurant recommendations [2], review evaluations with user ratings [3], querying wireless sensor networks [4], and graph analysis [5]. Algorithms for computing skyline objects have been discussed in the literature [6] [7] [8] [9].

One of the known weaknesses of the skyline query is that it can not answer various queries that require us to analyze not just individual object of a dataset but also their combinations. It is very likely that an investor has to invest more than one stock. For example, investment in O_1 will render the lowest cost. However, this investment is also very risky. Are there any other stocks or sets of stocks which allow us to have a lower investment and/or a lower risk? These answers are often referred to as the investment portfolio. How to efficiently find such an investment portfolio is the main issue studied in this paper.

We consider a skyline query for sets of objects (objectsets) in a database. Let k be the number of objects in each set and n be the number of objects in the dataset. The number of sets in the dataset amounts to ${}_nC_k$. We propose an efficient algorithm to compute skyline of the ${}_nC_k$ sets.

Assume an investor has to buy two stocks. In Figure 1, conventional skyline query outputs $\{O_1, O_2, O_3\}$, which doesn't provide sufficient information for the set selection problem. Users may want to choose the portfolios which are not dominated by any other sets in order to minimize the total costs and risks. Figure 2 shows sets consisting of two stocks, in which attribute values of each set are the sums of two component stocks. Objectsets $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ cannot be dominated by any other objectsets and thus they are the answers for the objectset skyline query. Again, if the investor wants to buy three stocks then the objectset skyline query will retrieve objectsets $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ as the query result.

Though a skyline query of objectsets is important in portfolio analysis, privacy aware data analysis, outlier-resistant data analysis, etc., there have been few studies on the objectsets skyline problem because of the difficulty of the problem. The objectsets skyline operator was introduced by Siddique and Morimoto in 2010 [10]. They tried to find the skyline objectsets that are on the convex hull enclosing all the objectsets, yet it misses some skyline objectsets, which are not on the convex hull. Su et al. proposed a solution to find the top-k optimal objectsets according to a user defined preference order of attributes [11]. However, it is difficult to define a user preference

 ISBN: 978-1-61208-408-4

beforehand for some complicated decision making tasks. Guo et al. proposed a pattern based pruning (PBP) algorithm to solve the objectsets skyline problem by indexing individuals objects [12]. The key problem of the PBP algorithm is that it needs object selecting pattern in advance and the pruning capability depends on this pattern. Moreover, this algorithm is for fixed size objectset k and failed to retrieve result for all k. In this paper, we present an efficient solution that can select skyline objectsets, which include not only convex skyline objectsets but also non-convex skyline objectsets. The objectset size k can be varied from 1 to n and within which a user may select a smaller subset of his/her interest.

We propose an algorithm to resolve the objectsets skyline query problem. It progressively prunes the objectsets that are impossible to be the objectsets skyline result, and uses a filtering mechanism to retrieve the skyline objectsets without enumerating all objectsets. We develop two pruning strategies to avoid generating a large number of unpromising objectsets. The efficiency of the algorithm is then examined with experiments on a variety of synthetic and real datasets.

The rest of this paper is organized as follows. Section II reviews the related work. Section III presents the notions and properties for objectsets as well as the problem of objectsets skyline. In Section IV, we provide details of our proposed algorithm with appropriate examples and analysis. We experimentally evaluate the proposed algorithm in Section V under a variety of settings. Finally, Section VI concludes the paper and introduces our future works.

II. Related Work

Our work is motivated by previous studies of skyline query processing as well as objectsets skyline query processing.

A. Skyline Query Processing

Borzsonyi et al. first introduced the skyline operator over large databases and proposed three algorithms: $Block\text{-}Nested\text{-}Loops(BNL)$, $Divide\text{-}and\text{-}Conquer$ $(D\&C)$, and B-tree-based schemes [1]. BNL compares each object of the database with every other object, and reports it as a result only if any other object does not dominate it. A window W is allocated in main memory, and the input relation is sequentially scanned. In this way, a block of skyline objects is produced in every iteration. In case the window saturates, a temporary file is used to store objects that cannot be placed in W. This file is used as the input to the next pass. $D\&C$ divides the dataset into several partitions such that each partition can fit into memory. Skyline objects for each individual partition are then computed by a main-memory skyline algorithm. The final skyline is obtained by merging the skyline objects for each partition. Chomicki et al. improved BNL by presorting, they proposed $Sort\text{-}Filter\text{-}Skyline(SFS)$ as a variant of BNL [6]. Among index-based methods, Tan et al. proposed two progressive skyline computing methods Bitmap and Index [13]. In the Bitmap approach, every dimension value of a point is represented by a few bits. By applying bit-wise AND operation on these vectors, a given point can be checked if it is in the skyline without referring to other points. The index method organizes a set of m-dimensional objects into m lists such that an object O is assigned to list i if and only if its value at attribute i is the best among all attributes of O. Each list is indexed by a B-tree, and the skyline is computed by scanning the B-tree until an object that dominates the remaining entries in the B-trees is found. The current most efficient method is $Branch\text{-}and\text{-}Bound$ $Skyline(BBS)$, proposed by Papadias et al., which is a progressive algorithm based on the *best-first nearest neighbor (BF-NN)* algorithm [8]. Instead of searching for nearest neighbor repeatedly, it directly prunes using the R*-tree structure.

Recently, more aspects of skyline computation have been explored. Chan et al. proposed k-dominant skyline and developed efficient ways to compute it in high-dimensional space [14]. Lin et al. proposed n-of-N skyline query to support online query on data streams, i.e., to find the skyline of the set composed of the most recent n elements. In the cases where the datasets are very large and stored distributedly, it is impossible to handle them in a centralized fashion [15]. Balke et al. first mined skyline in a distributed environment by partitioning the data vertically [16]. Vlachou et al. introduce the concept of extended skyline set, which contains all data elements that are necessary to answer a skyline query in any arbitrary subspace [17]. Tao et al. discuss skyline queries in arbitrary subspaces [18]. More skyline variants such as dynamic skyline [19] and reverse skyline [20] operators also have recently attracted considerable attention.

B. Objectsets Skyline Query Processing

There are two closely related works, which are "top-k combinatorial skyline queries" [11] and "convex skyline objectsets" [10]. Su et al. studied how to find top-k optimal combinations according to a given preference order in the attributes. Their solution is to retrieve non-dominate combinations incrementally with respect to the preference until the best k results have been found. This approach relies on the preference order of attributes and the limited number (top-k) of combinations queried. Both the preference order and the top-k limitation may largely reduce the exponential search space for combinations. However, in our problem there is no preference order nor the top-k limitation. Consequently, their approach cannot solve our problem easily and efficiently. Additionally, in practice it is difficult for the system or a user to decide a reasonable preference order. This fact will narrow down the applications of [11].

Siddique and Morimoto studied the "convex skyline objectset" problem. It is known that the objects on the lower (upper) convex hull, denoted as CH, is a subset of the objects on the skyline, denoted as SKY. Every object in CH can minimize (maximize) a corresponding linear scoring function on attributes, while every object in SKY can minimize (maximize) a corresponding monotonic scoring function [1]. They aims at retrieving the objectsets in CH, however, we focuses on retrieving the objectsets in $CH \subseteq SKY$. Since their approach relies on the properties of the convex hull, it cannot extend easily to solve complete skyline problem.

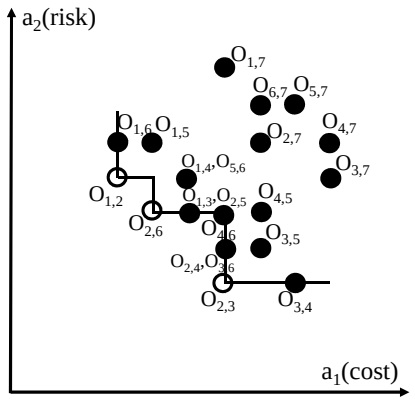
The similar related work is "Combination Skyline Querues" proposed in [12]. Guo et al. proposed a pattern based pruning (PBP) algorithm to solve the objectsets skyline problem by indexing individuals objects. The key problem of PBP algorithm is that it needs object selecting pattern in advance and the pruning capability depends on this pattern. For any initial wrong pattern this may increase the exponential search space. Moreover, it fails to vary the cardinality of objectset size k. Our

 ISBN: 978-1-61208-408-4

ID	a_1(cost)	a_2(risk)
$O_{1,2}$	6	12
$O_{1,3}$	10	10
$O_{1,4}$	10	12
$O_{1,5}$	8	14
$O_{1,6}$	6	14
$O_{1,7}$	12	18
$O_{2,3}$	12	6
$O_{2,4}$	12	8
$O_{2,5}$	10	10
$O_{2,6}$	8	10
$O_{2,7}$	14	14

ID	a_1(cost)	a_2(risk)
$O_{3,4}$	16	6
$O_{3,5}$	14	8
$O_{3,6}$	12	8
$O_{3,7}$	18	12
$O_{4,5}$	14	10
$O_{4,6}$	12	10
$O_{4,7}$	18	14
$O_{5,6}$	10	12
$O_{5,7}$	16	16
$O_{6,7}$	14	16

a) Sets of 2 Stocks



b) Skyline of 2 Stocks

Figure 2. Objectset skyline problem

solution does not require to construct any pattern previously and also vary the objectset size k from 1 to n.

There are some other works focusing on the combination selection problem but related to our work weakly [21] [22]. Roy et al. studied how to select "maximal combinations". A combination is "maximal" if it exceeds the specified constraint by adding any new object. Finally, the k most representative maximal combinations, which contain objects with high diversities, are presented to the user. Wan et al. study the problem to construct k profitable products from a set of new products that are not dominated by the products in the existing market [22]. They construct non-dominate products by assigning prices to the new products that are not given beforehand like the existing products.

III. PRELIMINARIES

Given a dataset D with m-attributes $\{a_1, a_2, \cdots, a_m\}$ and n objects $\{O_1, O_2, \cdots, O_n\}$. We use $O_i.a_j$ to denote the j-th dimension value of object O_i. Without loss of generality, we assume that smaller value in each attribute is better

Dominance

An object $O_i \in D$ is said to dominate another object $O_j \in D$, denoted as $O_i \leq O_j$, if $O_i.a_r \leq O_j.a_r$ $(1 \leq r \leq m)$ for all m attributes and $O_i.a_t < O_j.a_t$ $(1 \leq t \leq m)$ for at least one attribute. We call such O_i as *dominant object* and such O_j as *dominated object* between O_i and O_j.

Skyline

An object $O_i \in D$ is said to be a *skyline object* of D, if and only if there does not exist any object $O_j \in D$ $(j \neq i)$ that dominates O_i, i.e., $O_j \leq O_i$ is not true. The skyline of D, denoted by $Sky(D)$, is the set of skyline objects in D. For dataset shown in Figure 1(a), object O_2 dominates $\{O_4, O_5, O_6, O_7\}$ and objects $\{O_1, O_3\}$ are not dominated by any other objects in D. Thus, skyline query will retrieve $Sky(D) = \{O_1, O_2, O_3\}$ (see Figure 1(b)).

In the following, we first introduce the concept of objectset, and then use it to define objectsets skyline. A k-objectset s is made up of k objects selected from D, i.e., $s = \{O_1, \cdots, O_k\}$ and for simplicity denoted as $s = O_{1,\cdots,k}$. Each attribute value of s is given by the formula below:

$$s.a_j = f_j(O_1.a_j, \cdots, O_k.a_j), (1 \leq j \leq m) \quad (1)$$

where f_j is a monotonic aggregate function that takes k parameters and returns a single value. For the sake of simplicity, in this paper we consider that the monotonic scoring function returns the sum of these values, i.e.,

$$s.a_j = \sum_{i=1}^{k} O_i.a_j, (1 \leq j \leq m) \quad (2)$$

though our algorithm can be applied on any monotonic aggregate function. Recall that the number of k-objectsets in D is ${}_nC_k = \frac{n!}{(n-k)!k!}$, we denote the number by $|S|$.

Dominance Relationship

A k-objectset $s \in D$ is said to dominate another k-objectset $s' \in D$, denoted as $s \leq s'$, if $s.a_r \leq s'.a_r$ $(1 \leq r \leq m)$ for all m attributes and $s.a_t < s'.a_t$ $(1 \leq t \leq m)$ for at least one attribute. We call such s as dominant k-objectset and s' as dominated k-objectset between s and s'.

Objectsets Skyline

A k-objectset $s \in D$ is said to be a skyline k-objectset if s is not dominated by any other k-objectsets in D. The skyline of k-objectsets in D, denoted by $Sky_k(D)$, is the set of skyline k-objectsets in D. Assume $k = 2$, then for the dataset shown in Figure 2(a), 2-objectset $O_{1,2}, O_{2,3}$, and $O_{2,6}$ are not dominated by any other 2-objectsets in D. Thus, 2-objectset skyline query will retrieve $Sky_2(D) = \{O_{1,2}, O_{2,3}, O_{2,6}\}$ (see Figure 2(b)).

Domination Objectsets

Domination objectsets of k-objectsets, denoted by $DS_k(D)$ is said to be a set of all dominated k-objectsets in D. Since the 1-objcetsets skyline result is $Sky_1(D) = \{O_1, O_2, O_3\}$, then the domination objectsets of 1-objectsets is $DS_1(D) = \{O_4, O_5, O_6, O_7\}$, i.e., $D - Sky_1(D)$.

TABLE I. domRelationTable for 1-objectsets

Object	*Dominant Object*
O_1	$\varnothing$
O_2	$\varnothing$
O_3	$\varnothing$
O_4	O_2, O_3
O_5	O_2, O_6
O_6	O_2
O_7	$O_{1,\cdots,6}$

IV. Complete Skyline Objectsets Algorithm

In this section, we present our proposed method called Complete Skyline objectSets (CSS). It is a level-wise iterative algorithm. Initially, CSS computes conventional skyline, i.e., 1-objectsets skyline then 2-objectsets skyline, and so on, until k-objectsets skyline.

Initially, $k = 1$ and we can compute 1-objectsets skyline using any conventional algorithms. In this paper, we use SFS method proposed in [6] to compute 1-objectsets skyline and receive the following domination relation table called $domRelationTable$.

For the objectsets skyline query problem, the number of objectsets is $|S| = {}_nC_k$ for a dataset D containing n objects when we select objectsets of size k. This poses serious algorithmic challenges compared with the traditional skyline problem. As Figure 2(a) shows, $|S| = 21$ $({}_7C_2)$ possible combinations are generated from only seven objects when $k = 2$. Even for a small dataset with thousands of entries, the number of objectsets is prohibitively large. Thanks to the Theorem 1, which gives us opportunity to eliminate many non-promising objectsets without composing them.

Theorem 1. *If all k member of an objectset s are in $DS_k(D)$, where $DS_k(D) = DS_1(D) \cup \cdots \cup DS_k(D)$, then objecsets $s \notin Sky_k(D)$.*

Proof: Assume $s = \{O_1, \cdots, O_k\}$ and $s \in Sky_k(D)$. Since all members of s are in $DS_k(D)$, then there must be k distinct dominant objectsets for each member of s. Suppose they are $\{O'_1, \cdots, O'_k\}$ and construct an objectset s'. Now, according to dominance relationship $s' \leq s$, which contradict initial assumption $s \in Sky_k(D)$. Hence, a k-objectsets contains at least one skyline objectset. ■

Dominance relation given in Table I retrieves $DS_1(D) = \{O_4, O_5, O_6, O_7\}$. By using Theorem 1 and $k = 2$ we can safely prune ${}_4C_2 = 6$ objectsets such as $\{O_{4,5}, O_{4,6}, O_{4,7}, O_{5,6}, O_{5,7}, O_{6,7}\}$ for $Sky_2(D)$ query without composing them. The remaining objecsetsets number 15 (21-6) is still too large for our running example. However, CSS applies the second pruning strategy as follows:

Theorem 2. *Suppose S_1, S_2, and S_3 be the three objectsets in D. If objectset $S_1 \leq S_2$, then their super objectset with S_3 also dominates, i.e., $S_1S_3 \leq S_2S_3$ is true.*

Proof: Suppose $S_1S_3 \leq S_2S_3$ is not true. After eradicate S_3 from both objectsets we get $S_1 \leq S_2$, which contradict our assumption. Thus, if $S_1 \leq S_2$ and S_3 is another objectset then $S_1S_3 \leq S_2S_3$ is always true. ■

Theorem 2 gives us another opportunity to eliminate huge number objectsets without computing them. Table I shows that object O_4 is dominated by O_2 and O_3. By considering $\{O_1, O_2, O_3\}$ as common objects and using Theorem 2 without any computation as well as any comparisons we get $O_{1,2} \leq O_{1,4}, O_{2,3} \leq O_{2,4}, O_{2,3} \leq O_{3,4}$ dominance relation for 2-objectsets. Similarly, object O_5 is dominated by $\{O_2, O_6\}$ and using $\{O_1, O_2, O_3\}$ as common objects gives us $O_{1,2} \leq O_{1,5}, O_{2,3} \leq O_{3,5}, O_{2,6} \leq O_{2,5}$. For $O_2 \leq O_6$ and $\{O_1, O_3\}$ as common objects produces $O_{1,2} \leq O_{1,6}, O_{2,3} \leq O_{3,6}$. Finally, for $\{O_{1,\cdots,6} \leq O_7\}$ gives guarantee of the following dominance relationship $\{O_{1,2} \leq O_{1,7}, O_{1,2} \leq O_{2,7}, O_{1,3} \leq O_{3,7}\}$. Thus according to Theorem 2 we can safely prune 11 objectsets such as $\{O_{1,4}, O_{2,4}, O_{3,4}, O_{1,5}, O_{2,5}, O_{3,5}, O_{1,6}, O_{3,6}, O_{1,7}, O_{2,7}, O_{3,7}\}$ for $Sky_2(D)$ query without composing them. Actually, CSS algorithm will compose remaining (15 - 11) four objecsets such as $\{O_{1,2}, O_{1,3}, O_{2,3}, O_{2,6}\}$ and perform domination checks among them. After performing domination check it retrieves $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ as $Sky_2(D)$. However, during this procedure CSS also updates the dominance relation table for 2-objectsets as shown in II.

Dominance relation table II retrieves $DS_2(D) = \{O_{1,3}, O_{1,4}, O_{1,5}, O_{1,6}, O_{1,7}, O_{2,4}, O_{2,5}, O_{2,7}, O_{3,4}, O_{3,5}, O_{3,6}, O_{3,7}\}$. When $k = 3$, then for any conventional skyline algorithm needs to dominance relation check among $|S| = 35$ $({}_7C_3)$ objectsets. However, according to Theorem 1 we are enough lucky and need not to compose any 3-objectsets if the distinct 3 objects in $DS_1(D) \cup DS_2(D)$. For $DS_1(D)$, CSS does not compute ${}_4C_3 = 4$ objectsets such as $\{O_{4,5,6}, O_{4,5,7}, O_{4,6,7}, O_{5,6,7}\}$. Whereas for $DS_1(D) \cup DS_2(D)$ we have checked that CSS pruned another 22 objectsets. After implementing Theorem 1 successfully the remaining objectset number is 9 (35-26). Applying Theorem 2 CSS does not need to compute another 5 objectets. Finally, proposed algorithm will compose only five objecsets such as $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ and perform domination check among them to obtain $Sky_3(D)$. After domination check CSS retrieves $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ as $Sky_3(D)$. CSS will continue similar iterative procedure for the rest of the k values to compute $Sky_k(D)$.

V. Performance Evaluation

We conducted a set of experiments with different dimensionalities (m), data cardinalities (n), and objectset size (k) to evaluate the effectiveness and efficiency of our proposed method. All experiments are run on a computer with Intel Core i7 CPU 3.4GHz and 4 GB main memory. We compiled the source codes under Java V8 in Windows 8 32-bit operating system. We also compared the performance with Brute Force method. To make the comparison fair, we have exclude all the pre-processing cost,i.e., cost of objectset generation. Each experiment is repeated five times and the average result is considered for performance evaluation.

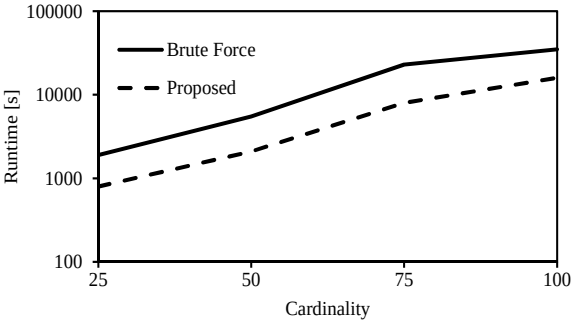
A. Performance on Synthetic Datasets

Three data distributions such as correlated, anti-correlated, and independent data distribution are considered to do all of the experiments. The results are shown in Figure 3, 4, and 5. All figures are shown in a logarithmic scale. From the experimental results we observe a pattern that the speedup of proposed method over Brute Force is 2 to 3 times faster.

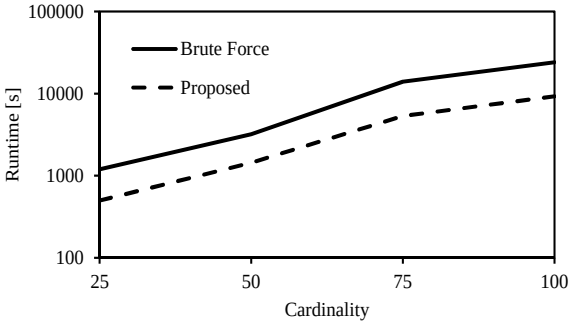
 ISBN: 978-1-61208-408-4

TABLE II. domRelationTable for 2-objectsets

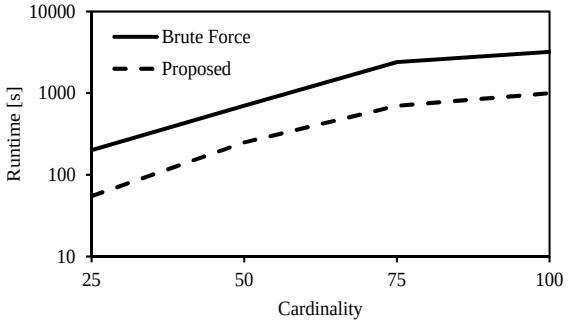
$Objectset$	$Dom.\ Objectset$	$Objectset$	$Dom.\ Objectset$	$Objectset$	$Dom.\ Objectset$
$O_{1,2}$	$\varnothing$	$O_{1,7}$	$O_{1,2}$	$O_{2,7}$	$O_{2,3}$
$O_{1,3}$	$O_{2,6}$	$O_{2,3}$	$\varnothing$	$O_{3,4}$	$O_{2,3}$
$O_{1,4}$	$O_{1,2}$	$O_{2,4}$	$O_{2,3}$	$O_{3,5}$	$O_{2,3}$
$O_{1,5}$	$O_{1,2}$	$O_{2,5}$	$O_{2,6}$	$O_{3,6}$	$O_{2,3}$
$O_{1,6}$	$O_{1,2}$	$O_{2,6}$	$\varnothing$	$O_{3,7}$	$O_{2,3}$



a) Anti-correlated



b) Independent



c) Correlated

Figure 3. Performance for different cardinality

Effect of Cardinality

For this experiment, we fix the data dimensionality m to 4, objectset size k to 3, and vary dataset cardinality n. n takes the values of 25, 50, 75, and 100 that means total objectset size respectively become 2.3k, 19.6k, 67.5k, and 161.7k. Figure 3(a), (b), and (c) reports the performance on correlated, independent, and anti-correlated datasets. Where both of the methods are affected by data cardinality. If the data cardinality increases then their performances decreases. The result shows that proposed method significantly outperforms the Brute Force method. However, the performance of Brute Force method degrades rapidly as the the dataset size increases, especially for anti-correlated data distribution. This represents that proposed method can successfully avoid objectset composing as well as many unnecessary comparisons.

Effect of Dimensionality

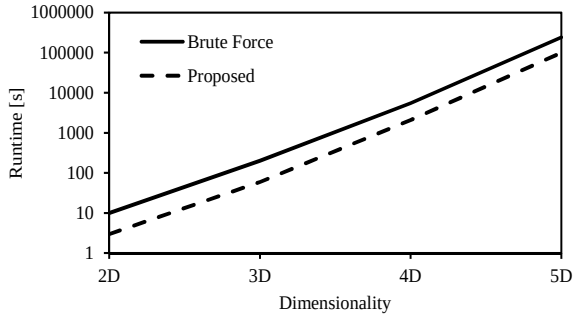
We study the effect of dimensionality on our technique. We fix the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m ranges from 2 to 5. The elapsed time results for this experiment are shown in Figure 4(a), (b), and (c). The result exhibits that as the dimension increases the performance of the both methods becomes slower. This is because for high dimension the number of non dominant objectset increases as a result the performance of both methods degraded. The proposed algorithm achieves a satisfactory running time even when the dimension size is large. However, the result on correlated data dataset is 9 times and 16 times faster than independent and anti-correlated data dataset respectively.

Effect of Objectset Size

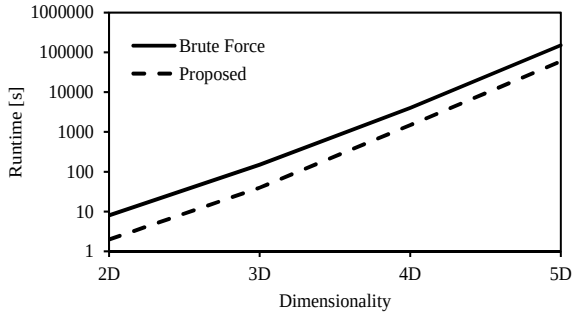
In another experiment, we study the performance of proposed method under various objectset size k. We fix the data cardinality n to 50 and dataset dimensionality m to 4. The results are reported in Figure 5(a), (b), and (c). The result indicate that as the objectset size k increases the performance of the both methods becomes slower. However, the result of Brute Force method is much worse than that of proposed method when the value of objectset size k is greater than 1. This is because for $k = 1$ the proposed method use SFS algorithm to construct $domRelationTable$, and performing domination check. After that for higher s it does not required to compose all objectset as well as succeeded to avoid huge number of unnecessary comparisons.

B. Performance on Real Dataset

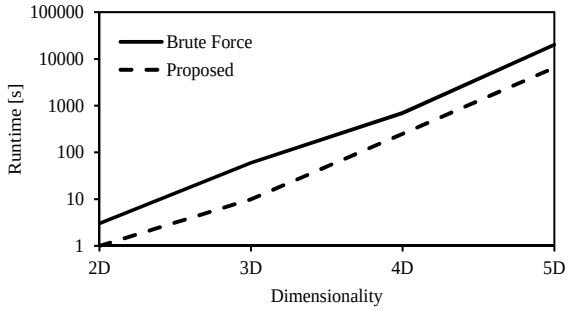
To evaluate the performance for real dataset, we use the FUEL dataset which is extracted from "`www.fueleconomy.gov`". FUEL dataset is 24k 6-dimensional objects, in which each object stands for the performance of a vehicle (such as mileage per gallon of gasoline in city and highway, etc). For this dataset attribute



a) Anti-correlated

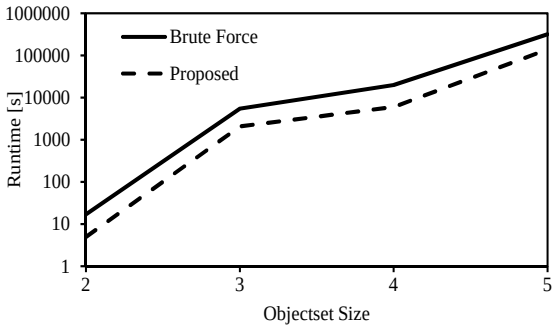


b) Independent

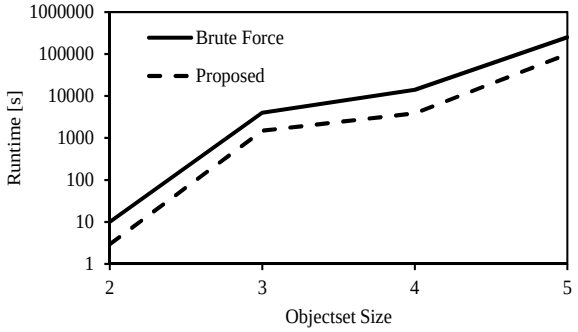


c) Correlated

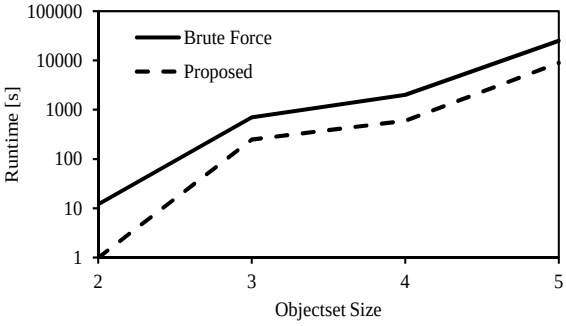
Figure 4. Performance for different data dimension



a) Anti-correlated



b) Independent



c) Correlated

Figure 5. Performance for different objectset size

domain range is [8, 89] and we conduct the following experiments.

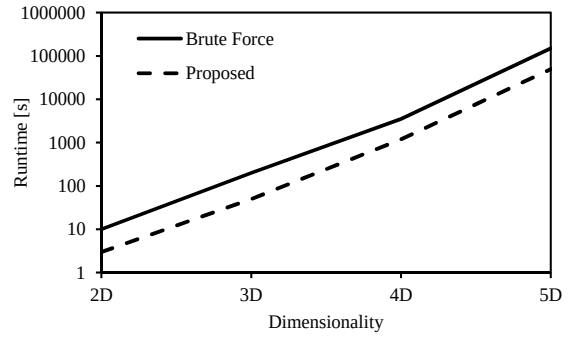
For FUEL dataset, we performed similar experiments like synthetic datasets. For cardinality experiment, we set the dimensionality m to 4 , objectset size k to 3, and vary dataset cardinality n from 25 to 100. Result is shown in Figure 6(a). To study dimensionality, we fix the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m ranges from 2 to 5. Figure 6(b) shows the result. In the final experiment, we study the performance under various objectset size k. We fix the data cardinality n to 50 and dimensionality m to 4. The result is reported in Figure 6(c). For all experiments with FUEL dataset, we obtain similar result like independent dataset that represents the scalability of the proposed method on real dataset. However, in all experiments with real FUEL dataset proposed method outperform than Brute Force method.

VI. CONCLUSION

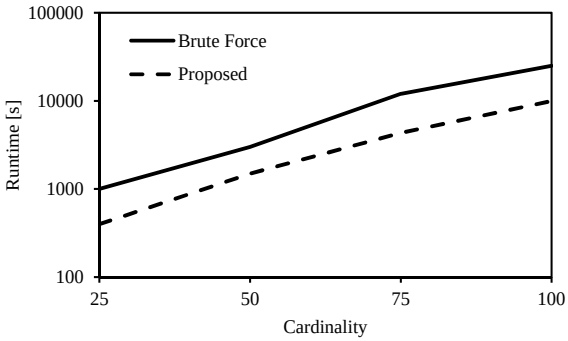
This paper addresses a skyline query for set of objects in a dataset. We propose an efficient and general algorithm called CSS to compute objectsets skyline. In order to prune the search space and improve the efficiency, we have developed two major pruning strategies. Using synthetic and real datasets, we demonstrate the scalability of proposed method. Intensive experiments confirm the effectiveness and superiority of our CSS algorithm.

It is worthy of being mentioned that this work can be expanded in a number of directions. First, how to solve the problem when the aggregation function is not monotonic. Secondly, to design more efficient objectsets computation on distributed MapReduce architectures. Finally, to find small number of representative objectsets is another promising future research work.

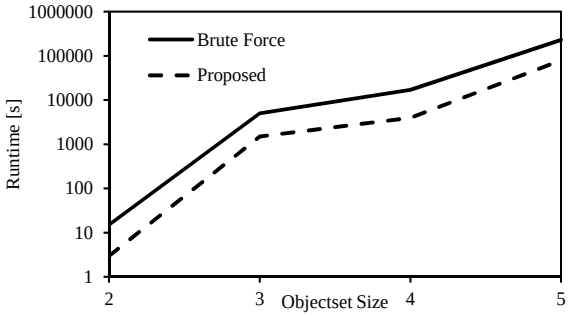
 ISBN: 978-1-61208-408-4



a) Varying dimensionality



b) Varying cardinality



c) Varying objectset size

Figure 6. Experiments on FUEL dataset

ACKNOWLEDGMENTS

This work is supported by KAKENHI (23500180, 25.03040) Japan.

REFERENCES

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in Proceedings of the 17th International Conference on Data Engineering (ICDE) April 2–6, 2001, Heidelberg, Germany, 2001, pp. 421–430.

[2] J. Lee, S. Hwang, Z. Nie, and J.-R. Wen, "Navigation system for product search," in Proceedings of the 26th International Conference on Data Engineering (ICDE) March 1–6, 2010, California, USA, 2010, pp. 1113–1116.

[3] T. Lappas and D. Gunopulos, "CREST:Efficient confident search in large review corpora," in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) September 20–24, 2010, Barcelona, Spain, 2010, pp. 467–478.

[4] G. Wang, J. Xin, L. Chen, and Y. Liu, "Energy efficient reverse skyline query processing over wireless sensor networks," IEEE Transactions on Knowledge Data Engineering (TKDE), vol. 24, no. 7, 2012, pp. 1259–1275.

[5] L. Zou, L. Chen, M. T. Ozsu, and D. Zhao, "Dynamic skyline queries in large graphs," in Proceedings of the Database Systems for Advanced Applications (DASFAA) April 1–4, 2010, Tsukuba, Japan, 2010, pp. 62–78.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in Proceedings of the 19th International Conference on Data Engineering (ICDE) March 5–8, 2003, Bangalore, India, 2003, pp. 717–719.

[7] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB) August 20–23, 2001, Hong Kong, China, 2002, pp. 275–286.

[8] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," ACM Transactions on Database Systems, vol. 30, no. 1, 2005, pp. 41–82.

[9] T. Xia, D. Zhang, and Y. Tao, "On Skylining with Flexible Dominance Relation," in Proceedings of the 24th International Conference on Data Engineering (ICDE) April 7–12, 2008, Cancun, Mexico, 2008, pp. 1397–1399.

[10] M. A. Siddique and Y. Morimoto, "Algorithm for computing convex skyline objectsets on numerical databases," IEICE TRANSACTIONS on Information and Systems, vol. E93-D, no. 10, 2010, pp. 0916–8532.

[11] I.-F. Su, Y.-C. Chung, and C. Lee, "Top-k combinatorial skyline queries," in Proceedings of the Database Systems for Advanced Applications (DASFAA) April 1–4, 2010, Tsukuba, Japan, 2010, pp. 79–93.

[12] X. Guo, C. Xiao, and Y. Ishikawa, "Combination Skyline Queries," Transactions on Large-Scale Data- and Knowledge-Centered Systems VI, vol. 7600, 2012, pp. 1–30.

[13] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in Proceedings of the 27th International Conference on Very Large Data Bases (VLDB) September 11–14, 2001, Rome, Italy, 2001, pp. 301–310.

[14] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-Dominant Skyline in High Dimensional Space," in Proceedings of the ACM SIGMOD June 26–29, 2006, Chicago, USA, 2006, pp. 503–514.

[15] X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the sky: Efficient Skyline computation over sliding windows," in Proceedings of the 21th International Conference on Data Engineering (ICDE) April 5–8, 2005, Tokyo, Japan, 2005, pp. 502–513.

[16] W.-T. Balke, U. Gntzer, and J.-X. Zheng, "Efficient distributed skylining for web information systems," in Proceedings of the 9th International Conference on Extending Database Technology (EDBT) March 14–18, 2004, Crete, Greece, 2004, pp. 256–273.

[17] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data," in Proceedings of the 23th International Conference on Data Engineering (ICDE) April 15–20, 2007, Istanbul, Turkey, 2007, pp. 416–425.

[18] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient Computation of Skylines in Subspaces," in Proceedings of the 22th International Conference on Data Engineering (ICDE) April 3–7, 2006, Georgia, USA, 2006, pp. 65–65.

[19] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in Proceedings of the ACM SIGMOD June 9–12, 2003, California, USA, 2003, pp. 467–478.

[20] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," in Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB) September 23–27, 2007, Vienna, Austria, 2007, pp. 291–302.

[21] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu, "Constructing and exploring composite items," in Proceedings of the ACM SIGMOD June 6–11, 2010, Indiana, USA, 2010, pp. 843–854.

[22] Q. Wan, R. C.-W. Wong, and Y. Peng, "Finding top-k profitable products," in Proceedings of the 27th International Conference on Data Engineering (ICDE) April 11–16, 2011, Hannover, Germany, 2011, pp. 1055–1066.

 ISBN: 978-1-61208-408-4

On the Number of Rules and Conditions in Mining Incomplete Data with Lost Values and Attribute-Concept Values

Patrick G. Clark
Department of Electrical Eng. and Computer Sci.
University of Kansas
Lawrence, KS, USA
e-mail: patrick.g.clark@gmail.com

Jerzy W. Grzymala-Busse
Department of Electrical Eng. and Computer Sci.
University of Kansas, Lawrence, KS, USA
Department of Expert Systems and Artificial Intelligence
University of Information Technology and Management
Rzeszow, Poland
e-mail: jerzy@ku.edu

***Abstract*—In mining incomplete data, we have a choice for interpretation of missing attribute values. In this paper, we consider two such interpretations: lost values and attribute-concept values. To measure the number of conditions and rules for each interpretation, we conducted experiments on eight incomplete data sets using three kinds of probabilistic approximations: singleton, subset and concept, with eleven values of probability. Using a 5% significance level, the results show that the number of rules is always smaller for attribute-concept values than for lost values. Additionally, the total number of conditions is smaller for attribute-concept values than for lost values for seven out of eight data sets.**

***Index Terms*—Data mining; rough set theory; probabilistic approximations; MLEM2 rule induction algorithm; lost values; attribute-concept values.**

I. Introduction

In this paper, we use two interpretations of a missing attribute value: lost values and attribute-concept values. Lost values indicate that the original values were erased, and as a result we should use only existing, specified attribute values for rule induction. Attribute-concept values mean that the missing attribute value may be replaced by any specified attribute value, typically for a given concept.

The idea of complexity of rule sets induced from incomplete data sets with lost values and attribute-concept values was introduced in [1]. In [1], experiments were conducted using only one type of probabilistic approximation (concept) and only three probabilities used for probabilistic approximations (0.001, 0.5 and 1.0). In this paper we use three kinds of probabilistic approximations (singleton, subset and concept) and eleven values of probability, starting from 0.001 and then from 0.1 to 1.0 with an increment of 0.1.

Our previous research [2][3] shows that the quality of rule sets, evaluated by an error rate computed by ten-fold cross validated, does not differ significantly with different combinations of missing attribute and probabilistic approximation type. As a result the main objective of this paper is research on complexity of rule sets, in terms of the number of rules and total number of rule conditions, induced from data sets with lost values and attribute-concept values. In previous work and in this work, the Modified Learning from Examples Module, version 2 (MLEM2) was used for rule induction.

The main results of this paper show that the size of rule set was always smaller for attribute-concept values than for lost values. The total number of conditions in rule sets was smaller for attribute-concept values for 22 of 24 combinations of the type of data set and probabilistic approximation. In the remaining two combinations, the total number of conditions in rule sets did not differ significantly. Thus, we may claim that attribute-concept values are better than lost values in terms of rule complexity. While our previous work discussed a subset of these findings, the primary contribution of this paper is that the results are more extensive and decisive than those presented in [1].

Our secondary objective was to check which probabilistic approximation (singleton, subset or concept) is the best from the point of view of rule complexity. The difference between all three approximations was insignificant.

This paper starts with a discussion on incomplete data in Section II where we define approximations, attribute-value blocks and characteristic sets. In Section III, we present singleton, subset and concept probabilistic approximations for incomplete data. Section IV contains the details of our experiments. Finally, conclusions are presented in Section V.

II. Incomplete Data

Fundamental concepts of rough set theory are standard lower and upper approximations. A probabilistic approximation, associated with a probability α, is an extension of the standard approximation. For $\alpha = 1$, the probabilistic approximation is reduced to the lower approximation; for very small α, it is reduced to the upper approximation. Research on theoretical properties of probabilistic approximations was initiated in [4] and then was continued in, e.g., [5]–[9].

Incomplete data sets are analyzed using special approximations such as singleton, subset and concept [10][11]. Probabilistic approximations, for incomplete data sets and based on an arbitrary binary relation, were introduced in [12], while first experimental results using probabilistic approximations were published in [13]. In experiments reported in this paper, we used three kinds of probabilistic approximations: singleton, subset and concept.

We assume that the input data sets are presented in the form of a decision table. An example of a decision table

 ISBN: 978-1-61208-408-4

TABLE I
A DECISION TABLE

	Attributes			Decision
Case	Wind	Humidity	Temperature	Trip
1	high	low	high	yes
2	–	low	?	yes
3	low	–	high	yes
4	–	low	low	yes
5	high	high	?	no
6	low	?	–	no
7	high	high	low	no
8	–	low	low	no

is shown in Table I. Rows of the decision table represent cases, while columns are labeled by variables. The set of all cases will be denoted by U. In Table I, U = {1, 2, 3, 4, 5, 6, 7, 8}. Independent variables are called attributes and a dependent variable is called a decision and is denoted by d. The set of all attributes will be denoted by A. In Table I, A = {*Wind*, *Humidity*, *Temperature*}. The value for a case x and an attribute a will be denoted by $a(x)$.

In this paper, we distinguish between two interpretations of missing attribute values: lost values, denoted by "?" and attribute-concept values, denoted by "–" [14][15]. Table I presents an incomplete data set affected by both lost values and attribute-concept values.

One of the most important ideas of rough set theory [16] is an indiscernibility relation, defined for complete data sets. Let B be a nonempty subset of A. The indiscernibility relation $R(B)$ is a relation on U defined for $x, y \in U$ as defined in equation 1.

$$(x, y) \in R(B) \text{ if and only if } \forall a \in B \ (a(x) = a(y)) \quad (1)$$

The indiscernibility relation $R(B)$ is an equivalence relation. Equivalence classes of $R(B)$ are called *elementary sets* of B and are denoted by $[x]_B$. A subset of U is called *B-definable* if it is a union of elementary sets of B.

The set X of all cases defined by the same value of the decision d is called a *concept*. For example, a concept associated with the value *yes* of the decision *Trip* is the set {1, 2, 3, 4}. The largest B-definable set contained in X is called the *B-lower approximation* of X, denoted by $\underline{appr}_B(X)$, and defined in equation 2.

$$\cup\{[x]_B \mid [x]_B \subseteq X\} \quad (2)$$

The smallest B-definable set containing X, denoted by $\overline{appr}_B(X)$ is called the *B-upper approximation* of X, and is defined in equation 3.

$$\cup\{[x]_B \mid [x]_B \cap X \neq \emptyset\} \quad (3)$$

For a variable a and its value v, (a, v) is called a variable-value pair. A *block* of (a, v), denoted by $[(a, v)]$, is the set $\{x \in U \mid a(x) = v\}$ [17]. For incomplete decision tables the definition of a block of an attribute-value pair is modified in the following way.

- If for an attribute a there exists a case x such that $a(x) =$?, i.e., the corresponding value is lost, then the case x should not be included in any blocks $[(a, v)]$ for all values v of attribute a,
- If for an attribute a there exists a case x such that the corresponding value is an attribute-concept value, i.e., $a(x) = -$, then the corresponding case x should be included in blocks $[(a, v)]$ for all specified values $v \in V(x, a)$ of attribute a, where $V(x, a)$ is defined by equation 4.

$$V(x, a) = \{a(y) \mid a(y) \text{ is specified}, y \in U, \ d(y) = d(x)\} \quad (4)$$

For the data set from Table I, we have $V(2, Wind) = \{low, high\}$, $V(3, Humidity) = \{low\}$, $V(4, Wind) = \{low, high\}$, $V(6, Temperature) = \{low\}$ and $V(8, Wind) = \{low, high\}$.

For the data set from Table I the blocks of attribute-value pairs are:
[(Wind, low)] = {2, 3, 4, 6, 8},
[(Wind, high)] = {1, 2, 4, 5, 7, 8},
[(Humidity, low)] = {1, 2, 3, 4, 8},
[(Humidity, high)] = {5, 7},
[(Temperature, low)] = {4, 6, 7, 8}, and
[(Temperature, high)] = {1, 3}.

For a case $x \in U$ and $B \subseteq A$, the *characteristic set* $K_B(x)$ is defined as the intersection of the sets $K(x, a)$, for all $a \in B$, where the set $K(x, a)$ is defined in the following way:

- If $a(x)$ is specified, then $K(x, a)$ is the block $[(a, a(x))]$ of attribute a and its value $a(x)$,
- If $a(x) =$? then the set $K(x, a) = U$, where U is the set of all cases,
- If $a(x) = -$, then the corresponding set $K(x, a)$ is equal to the union of all blocks of attribute-value pairs (a, v), where $v \in V(x, a)$ if $V(x, a)$ is nonempty. If $V(x, a)$ is empty, $K(x, a) = U$.

For Table I and $B = A$,
$K_A(1)$ = {1},
$K_A(2)$ = {1, 2, 3, 4, 8},
$K_A(3)$ = {3},
$K_A(4)$ = {4, 8},
$K_A(5)$ = {5, 7},
$K_A(6)$ = {4, 6, 8},
$K_A(7)$ = {7}, and
$K_A(8)$ = {4, 8}.

First we will quote some definitions from [18]. Let X be a subset of U. The B-*singleton lower approximation* of X, denoted by $\underline{appr}_B^{singleton}(X)$, is defined by equation 5

$$\{x \mid x \in U, K_B(x) \subseteq X\} \quad (5)$$

The B-*singleton upper approximation* of X, denoted by $\overline{appr}_B^{singleton}(X)$, is defined by equation 6.

$$\{x \mid x \in U, K_B(x) \cap X \neq \emptyset\} \quad (6)$$

 ISBN: 978-1-61208-408-4

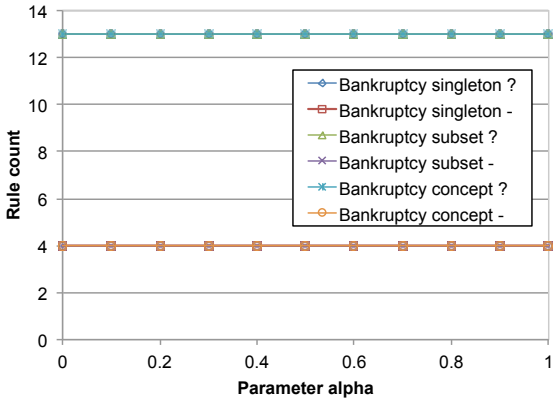


Fig. 1. Size of the rule set for the *Bankruptcy* data set

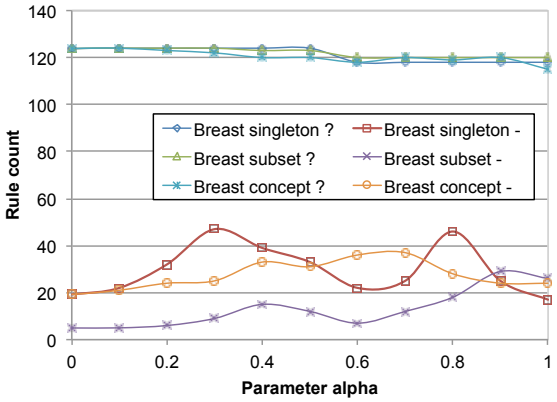


Fig. 2. Size of the rule set for the *Breast cancer* data set

The B-*subset lower approximation* of X, denoted by $\underline{appr}_B^{subset}(X)$, is defined by equation 7.

$$\cup \{K_B(x) \mid x \in U, K_B(x) \subseteq X\} \quad (7)$$

The B-*subset upper approximation* of X, denoted by $\overline{appr}_B^{subset}(X)$, is defined by equation 8.

$$\cup \{K_B(x) \mid x \in U, K_B(x) \cap X \neq \emptyset\} \quad (8)$$

The B-*concept lower approximation* of X, denoted by $\underline{appr}_B^{concept}(X)$, is defined by equation 9.

$$\cup \{K_B(x) \mid x \in X, K_B(x) \subseteq X\} \quad (9)$$

The B-*concept upper approximation* of X, denoted by $\overline{appr}_B^{concept}(X)$, is defined by equation 10.

$$\cup \{K_B(x) \mid x \in X, K_B(x) \cap X \neq \emptyset\} = \cup \{K_B(x) \mid x \in X\} \quad (10)$$

For Table I and X = {5, 6, 7, 8}, all A-singleton, A-subset and A-concept lower and upper approximations are:
$\underline{appr}_A^{singleton}(X) = \{5, 7\}$,
$\overline{appr}_A^{singleton}(X) = \{2, 4, 5, 6, 7, 8\}$,
$\underline{appr}_A^{subset}(X) = \{5, 7\}$,
$\overline{appr}_A^{subset}(X) = U$,
$\underline{appr}_A^{concept}(X) = \{5, 7\}$,
$\overline{appr}_A^{concept}(X) = \{4, 5, 6, 7, 8\}$.

III. Probabilistic Approximations

In this section we will extend definitions of singleton, subset and concept approximations to corresponding probabilistic approximations. A B-singleton probabilistic approximation of X with the threshold α, $0 < \alpha \leq 1$, denoted by $appr_{\alpha,B}^{singleton}(X)$, is defined by equation 11.

$$\{x \mid x \in U,\ Pr(X \mid K_B(x)) \geq \alpha\} \quad (11)$$

Here $Pr(X \mid K_B(x)) = \frac{|X \cap K_B(x)|}{|K_B(x)|}$ is the conditional probability of X given $K_B(x)$ and $|Y|$ denotes the cardinality of set Y. A B-subset probabilistic approximation of the set X with the threshold α, $0 < \alpha \leq 1$, denoted by $appr_{\alpha,B}^{subset}(X)$, is defined by equation 12.

$$\cup\{K_B(x) \mid x \in U,\ Pr(X \mid K_B(x)) \geq \alpha\} \quad (12)$$

A B-concept probabilistic approximation of the set X with the threshold α, $0 < \alpha \leq 1$, denoted by $appr_{\alpha,B}^{concept}(X)$, is

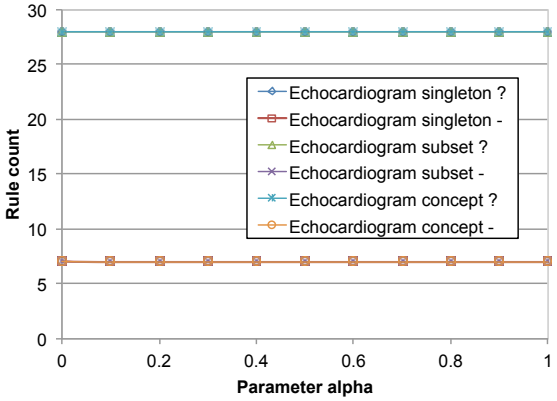


Fig. 3. Size of the rule set for the *Echocardiogram* data set

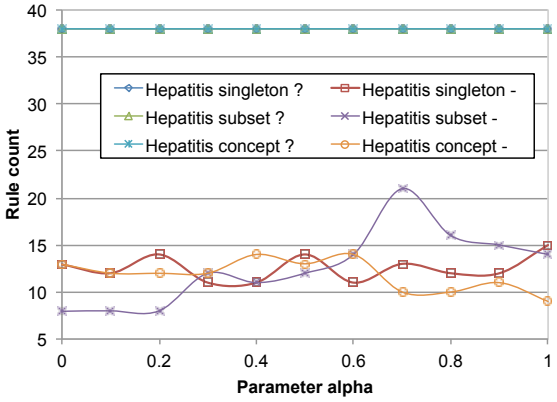


Fig. 4. Size of the rule set for the *Hepatitis* data set

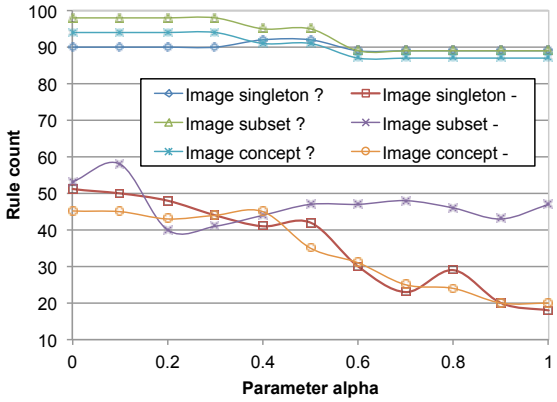


Fig. 5. Size of the rule set for the *Image segmentation* data set

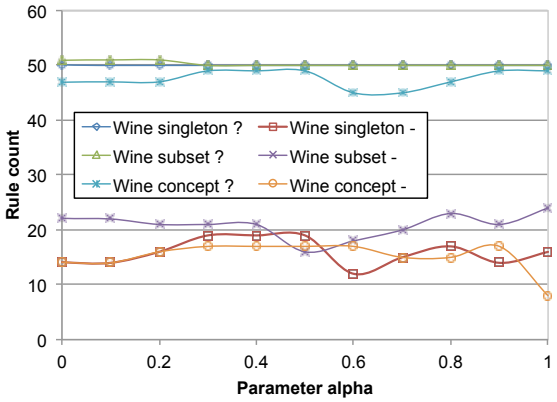


Fig. 8. Size of the rule set for the *Wine recognition* data set

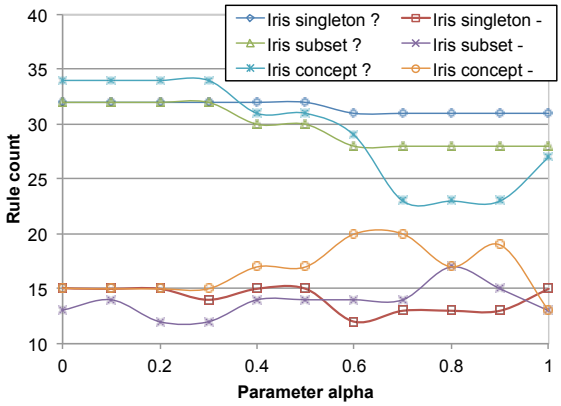


Fig. 6. Size of the rule set for the *Iris* data set

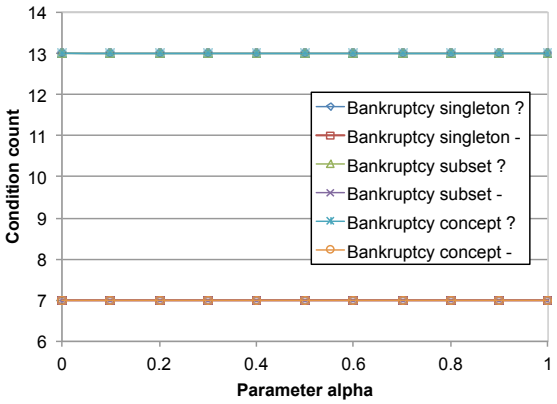


Fig. 9. Number of conditions for the *Bankruptcy* data set

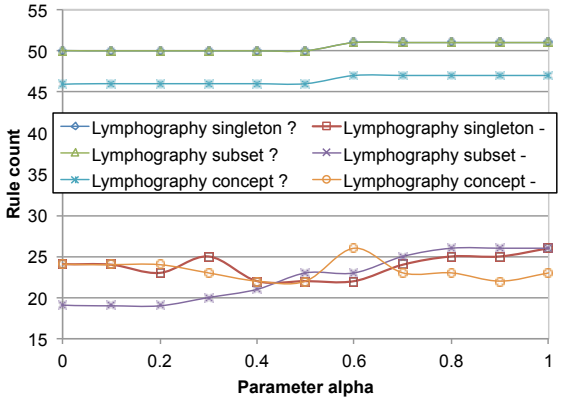


Fig. 7. Size of the rule set for the *Lymphography* data set

defined by equation 13.

$$\cup\{K_B(x) \mid x \in X,\ Pr(X \mid K_B(x)) \geq \alpha\} \qquad (13)$$

Note that if $\alpha = 1$, the probabilistic approximation becomes the standard lower approximation and if α is small, close to 0, in our experiments it was 0.001, the same definition describes the standard upper approximation.

For Table I and the concept X = [(*Trip*, *yes*)] = $\{1, 2, 3, 4\}$, there exist the following distinct probabilistic approximations:

$appr^{singleton}_{1.0,A}(\{1,2,3,4\}) = \{1,3\}$,
$appr^{singleton}_{0.8,A}(\{1,2,3,4\}) = \{1,2,3\}$,
$appr^{singleton}_{0.5,A}(\{1,2,3,4\}) = \{1,2,3,4,8\}$,
$appr^{singleton}_{0.333,A}(\{1,2,3,4\}) = \{1,2,3,4,6,8\}$,
$appr^{subset}_{1.0,A}(\{1,2,3,4\}) = \{1,3\}$,
$appr^{subset}_{0.5,A}(\{1,2,3,4\}) = \{1,2,3,4,8\}$,
$appr^{subset}_{0.333,A}(\{1,2,3,4\}) = \{1,2,3,4,6,8\}$,
$appr^{concept}_{1.0,A}(\{1,2,3,4\}) = \{1,3\}$,
$appr^{concept}_{0.333,A}(\{1,2,3,4\}) = \{1,2,3,4,8\}$,

IV. Experiments

Our experiments are based on eight data sets that are available on the University of California at Irvine *Machine Learning Repository*.

For every data set, a template was created. Such a template was formed by replacing randomly 35% of existing specified attribute values by *lost values*. The same template was used for constructing a corresponding data set with *attribute-concept values*, by replacing "?"s with "–"s.

 ISBN: 978-1-61208-408-4

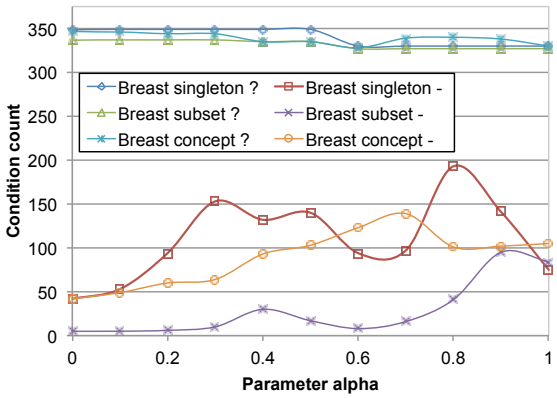


Fig. 10. Number of conditions for the *Breast cancer* data set

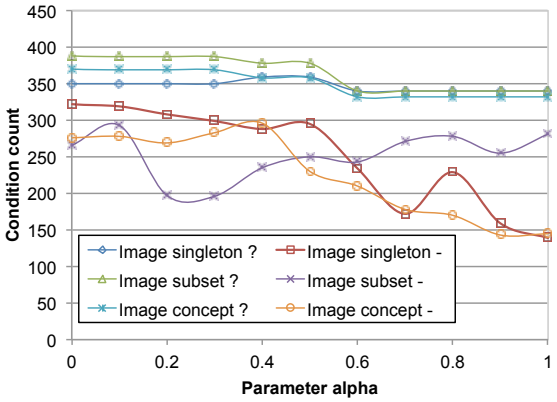


Fig. 13. Number of conditions for the *Image segmentation* data set

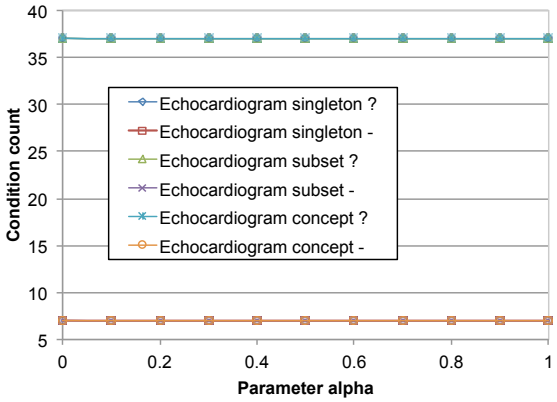


Fig. 11. Number of conditions for the *Echocardiogram* data set

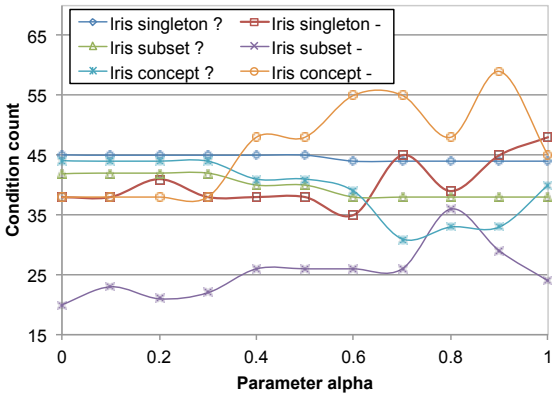


Fig. 14. Number of conditions for the *Iris* data set

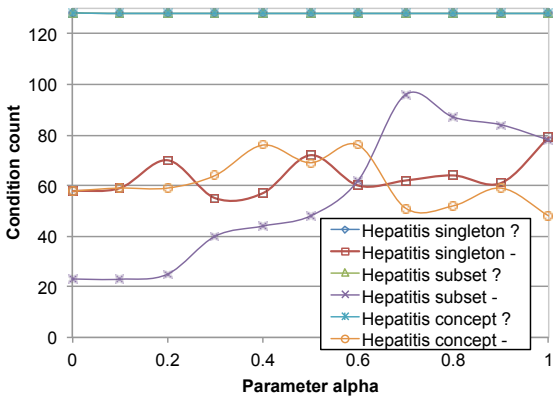


Fig. 12. Number of conditions for the *Hepatitis* data set

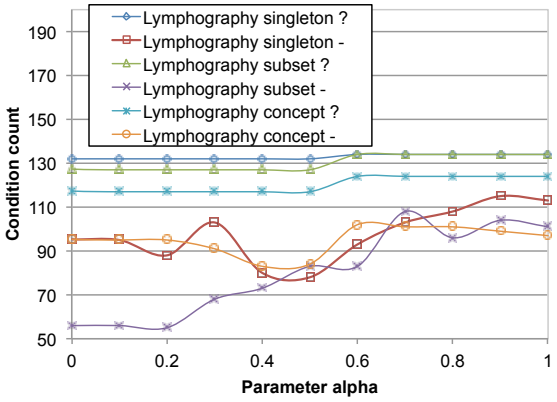


Fig. 15. Number of conditions for the *Lymphography* data set

For any data set we compared the size of rule set and the total number of conditions in the rule set for two interpretations of missing attribute values assuming the same type of probabilistic approximation. We used the Wilcoxon matched-pairs signed-ranked test with 5% significance level and with Bonferroni correction for multiple comparisons. In our experiments, we used the MLEM2 rule induction algorithm of the Learning from Examples using Rough Sets (LERS) data mining system [13][19][20]. Results of our experiments are presented in Figures 1–16.

For rule sets, in all 24 combinations of the type of probabilistic approximation and data set, the size of the rule set

 ISBN: 978-1-61208-408-4

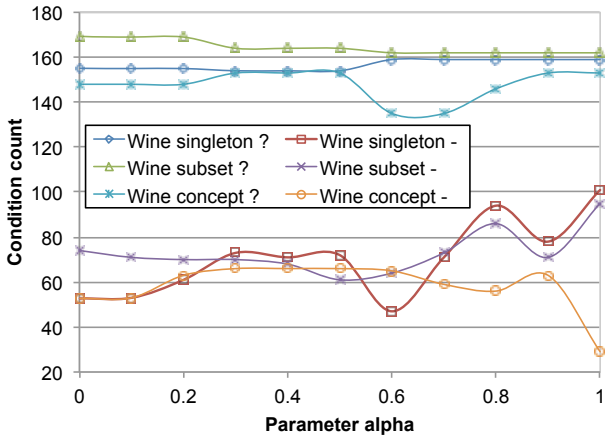


Fig. 16. Number of conditions for the *Wine recognition* data set

was always smaller for attribute-concept values than for lost values. For the total number of conditions, for the *iris* data set and for singleton and concept probabilistic approximations (Figure 6), the results were statistically inconclusive. For remaining 22 combinations, the total number of conditions was always smaller for attribute-concept values than for lost values.

In six out of the eight data sets, results of our experiments show some level of variability in the number of rules and conditions as the parameter alpha changes. However, for the *bankruptcy* and *echocardiogram* data sets, within given interpretation of missing attribute values, these numbers are constant for all values of the parameter α. The results for the *bankruptcy* and *echocardiogram* are different from the results for the remaining six data sets since for the former all attributes have numeric values while for the latter attributes are symbolic. For numeric attributes, during rule induction, the same numeric intervals are created for all possible values of the parameter α.

V. Conclusions

As follows from our experiments, the size of rule set was always smaller for attribute-concept values than for lost values. The total number of conditions in rule sets was smaller for attribute-concept values for 22 combinations of the type of data set and probabilistic approximation (out of 24 combinations total). In remaining two combinations, the total number of conditions in rule sets did not differ significantly. Thus, we claim that attribute-concept values are better than lost values in terms of rule complexity.

Additionally, results of our experiments show that in induction of least complex rule sets the difference between all three probabilistic approximations (singleton, subset and concept) was not statistically significant.

References

[1] P. G. Clark and J. W. Grzymala-Busse, "Complexity of rule sets induced from incomplete data with lost values and attribute-concept values," in *Proceedings of the Third International Conference on Intelligent Systems and Applications*, 2014, pp. 91–96.

[2] P. G. Clark, J. W. Grzymala-Busse, and W. Rzasa, "Mining incomplete data with singleton, subset and concept approximations," *Information Sciences*, vol. 280, pp. 368–384, 2014.

[3] P. G. Clark and J. W. Grzymala-Busse, "Mining incomplete data with lost values and attribute-concept values," in *Proceedings of the 2014 IEEE International Conference on Granular Computing*, 2014, pp. 49–54.

[4] Z. Pawlak, S. K. M. Wong, and W. Ziarko, "Rough sets: probabilistic versus deterministic approach," *International Journal of Man-Machine Studies*, vol. 29, pp. 81–95, 1988.

[5] Z. Pawlak and A. Skowron, "Rough sets: Some extensions," *Information Sciences*, vol. 177, pp. 28–40, 2007.

[6] D. Ślęzak and W. Ziarko, "The investigation of the bayesian rough set model," *International Journal of Approximate Reasoning*, vol. 40, pp. 81–91, 2005.

[7] Y. Y. Yao, "Probabilistic rough set approximations," *International Journal of Approximate Reasoning*, vol. 49, pp. 255–271, 2008.

[8] Y. Y. Yao and S. K. M. Wong, "A decision theoretic framework for approximate concepts," *International Journal of Man-Machine Studies*, vol. 37, pp. 793–809, 1992.

[9] W. Ziarko, "Probabilistic approach to rough sets," *International Journal of Approximate Reasoning*, vol. 49, pp. 272–284, 2008.

[10] J. W. Grzymala-Busse, "Rough set strategies to data with missing attribute values," in *Notes of the Workshop on Foundations and New Directions of Data Mining, in conjunction with the Third International Conference on Data Mining*, 2003, pp. 56–63.

[11] ——, "Data with missing attribute values: Generalization of indiscernibility relation and rule induction," *Transactions on Rough Sets*, vol. 1, pp. 78–95, 2004.

[12] ——, "Generalized parameterized approximations," in *Proceedings of the 6-th International Conference on Rough Sets and Knowledge Technology*, 2011, pp. 136–145.

[13] P. G. Clark and J. W. Grzymala-Busse, "Experiments on probabilistic approximations," in *Proceedings of the 2011 IEEE International Conference on Granular Computing*, 2011, pp. 144–149.

[14] J. W. Grzymala-Busse and A. Y. Wang, "Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values," in *Proceedings of the 5-th International Workshop on Rough Sets and Soft Computing in conjunction with the Third Joint Conference on Information Sciences*, 1997, pp. 69–72.

[15] J. Stefanowski and A. Tsoukias, "Incomplete information tables and rough classification," *Computational Intelligence*, vol. 17, no. 3, pp. 545–566, 2001.

[16] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, pp. 341–356, 1982.

[17] J. W. Grzymala-Busse, "LERS—a system for learning from examples based on rough sets," in *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory*, R. Slowinski, Ed. Dordrecht, Boston, London: Kluwer Academic Publishers, 1992, pp. 3–18.

[18] J. W. Grzymala-Busse and W. Rzasa, "Definability and other properties of approximations for generalized indiscernibility relations," *Transactions on Rough Sets*, vol. 11, pp. 14–39, 2010.

[19] J. W. Grzymala-Busse, "A new version of the rule induction system LERS," *Fundamenta Informaticae*, vol. 31, pp. 27–39, 1997.

[20] ——, "MLEM2: A new algorithm for rule induction from imperfect data," in *Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2002, pp. 243–250.

 ISBN: 978-1-61208-408-4

Patent Threat Analysis Search Engine

Yung Chang Chi
Department of Industrial and Information Management and
Institute of Information Management,
National Cheng Kung University,
Tainan City, Taiwan ROC
e-mail:charles.y.c.chi@gmail.com

Hei Chia Wang
Department of Industrial and Information Management and
Institute of Information Management,
National Cheng Kung University,
Tainan City, Taiwan ROC
e-mail:hcwang@mail.ncku.edu.tw

***Abstract*-This paper proposes a framework for a patent threat analysis search engine. The framework employs text mining based on the patent map approach to identify which particular patent is similar to the one in dispute. The patent map is a visual representation that uses technological proximities among patents. The patent threat analysis method will analyze patent infringement issues from judgements in the USA and Europe. Having examined and compared the patent map and patent infringement analysis, we can identify what kind of product and technology are subject to the threat of patent infringement with the help of solution integration and analysis of the two different databases.**

Keywords-patent;patent threat;patent map;content analysis.

I. INTRODUCTION

Patents are important knowledge sources for industrial research and product development because of their innovation and practicability. In recent years, patent analysis increased in importance for high-technology management as the process of innovation became more complex, the cycle of innovation became shorter and the market demand more volatile [22].

Patent analysis technologies include patent bibliometric data analysis [21], patent citation analysis [5], patent statistical analysis [26] and patent classification. Bibliometric analysis of patents provides information on the growth of the inventive activity and technological trends [22].

Patent mining is an emerging research topic that grew in recent years. So far, only few researches have been done on the topic. Patent mining consists of patent retrieval, patent categorization and patent clustering [25].

Retrieving patent documents can be done through the cluster-based approach [6]. Distributed information retrieval for patent can be done by generating ranking lists for the query by CORI (The collection retrieval inference network) or KL (Kernighan–Lin) algorithms [14]. Categorizing patent documents can be done automatically using the k-Nearest Neighbor classifiers and Bayesian classifiers [12][13], or by using a variety of machine learning algorithms [1], the k-Nearest Neighbor on the basis of patent's semantic structure [7], the classifier built through back-propagation network [19]. Patent documents can be clustered through the k-Means algorithm and represented in a visualized patent map [8], and the structured SOM (Self-Organizing Map) clustering algorithm [3]. Clustering algorithms can also be adopted to form a topic map for presenting patent analysis and summarization results [19], to create a system interface for retrieving patent documents [3].

Content analysis is indigenous to communication research and is potentially one of the important research techniques in social sciences. It seeks to analyze data within a specific context in view of the meaning someone –a group or a culture-attributes to them. Communications, messages, and symbols differ from observable events, things, properties, or people in that they inform about something other than themselves; they reveal some properties of their distant producers or carriers, and they have cognitive consequences for their senders, their receivers, and the institutions in which their exchange is embedded [9].

Content analysis is a research technique for making replicable and valid inferences from texts to the contexts of their use. As a technique, content analysis involves specialized procedure. It provides new insights, increases a researcher's understanding of particular phenomena, or informs practical actions. Content analysis is a scientific tool [10].

The judgements of patent infringement, unlike the patent documents, can be mined using text mining techniques, since the judgements are legal documents. The judgements can be transformed into patterns by content analysis, and readers can easily access them the same way as reading newspapers to understand the key points and issues in dispute.

The rest of the paper is structured as follows. Section II presents the research background. Section III states our objective. In Section IV, we describe our proposed research method. The paper concludes with expected results and future work considerations.

II. RESEARCH BACKGROUND

So far, patent analysis technologies include patent bibliometric data analysis [21], patent citation analysis [5], patent statistical analysis [26], and patent classification. Patent mining consists of patent retrieval, patent categorization and patent clustering [25] that focuses just on the patent documents analysis and patent mining. However, patent infringement constitutes the biggest threat in patents use. Through patent analysis and mining, one can just discover newly developed products and their similarity with

 ISBN: 978-1-61208-408-4

the claims of other patents, but no one can foresee where potential patent threats are and the likelihood of patent infringement.

This framework of patent database is based on the United States Patent and Trademark Office (USPTO) patent database and the European Patent Office (EPO) patent database. The patent infringement judgements are based on the case judgements in United States and European Union.

III. RESEARCH OBJECTIVE

The purposes of this study is to provide the patent threat analysis and reference regarding patent infringement as well as technology trends for new product designers and technology research engineers at the stage before and after developing a new product or technology. This will also provide the information needed so that management can make strategic decisions.

Because of a lack in legal background, it is difficult for ordinary readers to fully grasp the judgements rendered by professional judges. With the implementation of content analysis, ordinary people will be able to use content analysis technology to analyze the patent infringement verdict contents, and try to use big data concepts across different databases to discover any relation.

IV. RESEARCH METHOD

The patent documents can be collected from United States Patent and Trademark Office (USPTO) patent database and the European Patent Office (EPO) patent database.

A. Patent documents analysis

Base on the collected patent documents and the subject-action-object (SAO) structures extracted by using Natural Language Processing (NLP), the study uses a content analysis approach to generate the patent map.

NLP is a text mining technique that can conduct syntactic analysis of natural language; NLP tools include Stanford parser (Stanford2013)[27], Minipar (Lin2003)[28] and KnowledgistTM2.5[29].

NLP tools will be used for build a set of SAO structures from the collected patents.

Multidimensional scaling (MDS) is a statistical technique used to visualize similarities in data [11][16]. Patent documents in different fields have different key issues that trigger different Multidimensional scaling, so the paper will design a new algorithm to identify which particular patent field shall correspond to what extent of scaling.

B. Patent infringement verdict content analysis

The most obvious source of data appropriate for content analysis is text to which meanings are conventionally attributed: verbal discourse, written documents, and visual representations. The text in the patent infringement judgements is important because that is where the meanings are. For this reason, it is essential for the content analysis technology to analyze the patent infringement text in order to develop strategies and preventive measures in patent litigation.

Content analyses commonly contain six steps that define the technique procedurally, as follows:

Design. Design is a conceptual phase during which analysts define their context, what they wish to know and are unable to observe directly; explore the source of relevant data that either are or may become available; and adopt an analytical construct that formalizes the knowledge available about the data-context relationship thereby justifying the inferential step involved in going from one to the other.

Unitizing. Unitizing is the phase of defining and ultimately identifying units of analysis in the volume of available data. Sampling units makes possible the drawing of a statistically representative sample from a population of potentially available data, such as issues of a newspaper, whole books, television episodes, fictional characters, essays, advertisements.

Sampling. While the process of drawing representative samples is not indigenous to content analysis, there is the need to (1) undo the statistical biases inherent in much of the symbolic material analyzed and (2) ensure that the often conditional hierarchy of chosen sampling units become representative of the organization of the symbolic phenomena under investigation.

Coding. Coding is the step of describing the recording units or classifying them in terms of the categories of the analytical constructs chosen. This step replicates an elementary notion of meaning and can be accomplished either by explicit instructions to trained human coders or by computer coding. The two evaluative criteria, reliability as measured by inter coder agreement and relevance or meaningfulness, are often at odds.

Drawing inferences. Drawing inferences is the most important phase in a content analysis. It applies the stable knowledge about how the variable accounts of coded data are related to the phenomena the researcher wants to know about.

Validation. Validation is the desideratum of any research effort. However, validation of content analysis results is limited by the intention of the technique to infer what cannot be observed directly and for which validation evidence is not readily available.

C. Search engine

Our proposed search engine is a program that has three parts: (1) The first part searches patent documents for specified keywords and returns a list of the documents where the keywords were found. Then, the engine will use data and text mining technology to design a specified algorithm (first algorithm) in order to analyze the legal documents and try to find out the most similar patents or patent group. (2) Next, the engine searches the patent infringement judgements for specific keywords and returns a list of the documents as above patent documents by introducing the content analysis technology into specified design algorithm (second algorithm) in order to analyze the infringement cases/precedents. It also finds the nearest infringement judgements/precedents. (3) Finally, the engine uses different analysis technologies in two different

 ISBN: 978-1-61208-408-4

databases to render a cross comparison to generate a possible result algorithm (third algorithm) with the introduction of big data concepts.

A search engine is really a general class of programs. However, the term is often used to specifically describe systems. Our proposed search engine core technologies are used to analyze patent infringement content and to use algorithms and the comparative analysis between two databases in order to generate accurate result.

D. The framework of patent threat analysis search engine

The framework of patent threat analysis search engine is depicted in Figure 1. The top part of Figure 1 represents the content analysis research process [9]. We implement the patent infringement judgement is this process. The process framework needs to be designed as algorithm.

The middle of Figure 1 is the framework of patent documents analysis process. The process includes SAO structure extraction (NLP) and patent characteristic measurement and visualization (MDS). Here, we attempt to generate the patent map. In this phase, the study has generated some results based on past research.

The lower part in Figure 1 represents the proposed search engine core technology. The study will construct the knowledge and technology database in order to support the findings of particular products that are likely to be sued, technology trends, and the threat of patent infringement. The cross patent comparison and analysis will also utilize big data concepts to construct the algorithm.

V. EXPECTED RESULT AND FUTURE WORK

This study aims to develop a search engine similar to Google for patent analysis. When the user enters a keyword, the engine does an analysis and will inform on the related patents as well as potential patent threats. It can also provide the technology trend analysis.

The study aims to employ different analysis methods to analyze different databases and further use the analysis results by cross-comparison. An accurate algorithm in different fields can be constructed and achieved in patent threat analysis.

The next step will be to employ the image recognition functions to identify drawings and pictures. If the search engine has the capability to analyze drawings and pictures, the accuracy of the results will be increased in the future.

REFERENCES

[1] C. J. Fall, A. Torcsrari, K. Benzineb, and G. Karetka, Automated categorization in the international patent classification. “SIGIR Forum”. 2003, pp.10-25. 37(1).

[2] S.H. Huang, H.R. Ke, and W.P. Yang, Structure clustering for Chinese patent documents. “Expert system with application”. 2008, pp.2290-2297.34.

[3] S.H. Huang, C.C. Liu, C.W. Wang, H.R. Ke, and W.P. Yang, Knowledge annotation and discovery for patent analysis. “International Computer Symposium”. 2004, pp.15-20.

[4] H. Park, J. Yoon, and K. Kim, “Identification and evaluation of corporations for merger and acquisition strategies using patent information and text mining” Scientometrics. April 2013, pp.883–909.

[5] J. Michel, and B. Bettels, “Patent citation analysis: a closer look at the basic input data from patent search reports”, Scientometrics. 2001, pp.185-201. Vol.51. no. 1.

[6] I. S. Kang, S.H. Na, J. Kim, and J.H. Lee, Cluster-based patent retrieval. “Information Processing & Management”. 2007, pp.1173-1182.43(5).

[7] J.H. Kim, and K.S. Choi, Patent document categorization based on semantic structural information. “Information processing & Management”. 2007, pp.1200-1215.43(5).

[8] Y.G. Kim, J.H. Suh, and S.C. Park, Visualization of patent analysis for emerging technology. “Expert System with Applications”. 2008, pp.1804-1812.34(3).

[9] K. Krippendorff, Content analysis In E. Barnouw, G. Gerbner, W. Schramm, T. L. Worth, and L. Gross (Eds.), International encyclopedia of communication New York, NY: Oxford University Press. 1989, pp.403-407.Vol. 1.

[10] K. Krippendorff, “Content Analysis An Introduction to Its Methodology” second Edition, Sage Publications, Inc. 2004.

[11] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika. 1964, pp.1-27.29(1).

[12] L.S. Larkey, Some issues in the automatic classification of U.S. patents. In: Working notes for the AAAI-98 workshop on learning for text categorization. 1998, pp.87-90.

[13] Larkey L.S. A patent search and classification system. In: Proceedings of the fourth ACM conference on digital libraries. 1999, pp.79-87.

[14] Larkey L.S., Connell, M.E., and Callan, J. Collection selection and results merging with topically organized US patents and TREC data. In Proceedings of ninth international conference on informaiton knowledge and management. 2000, pp.282-289.

[15] D. Lin, Dependency-based evaluation of MINIPAR. In A. Abeille(Ed.), Treebanks: Building and using parsed corpora. Dordrecht: Kluwer, 2003, pp.317-332.

[16] U. Schmoch, Evaluation of technological strategies of companies by means of MDS maps. “International Journal of Technology Management”, 1995, pp.4-5.10(4-5).

[17] Stanford. The Stanford parser: A statistical parser. from http://nlp.standford.edu/software/lex-parser.shtml. Retrieved March 2013.

[18] T. Joachims “Text Categorization with Support Vector Machines: Learning with Many Relevant Features” University Dortmund Informatik LS8, Baroper Str. 301 44221 Dortmund, Germany.

[19] A.J.C. Trappey, F.C. Hsu, C.V. Trappy, C.I. Lin, Development of a patent document classification and search platform using a back-propagation network. “Expert Systems with Applications”. 2006, pp.755-765.31(4).

[20] Y.H Tseng, Y.M. Wang, Y.I. Lin, C.J. Lin, and D.W. Juang, Patent surrogate extraction and evaluation in the context of patent mapping. “Journal of Information Science”. 2007, pp.718-736.33(6).

[21] V. K. Gupta, and N. B. Pangannaya, Carbon nanotubes; bibliometric analysis of patents, “World Patent Information”. Sep. 2000, pp.185-189.Vol.22,issue 3.

[22] Y. Liang, R. Tan, and J. Ma, “Patent Analysis with Text Mining for TRIZ” IEEE ICMIT. 2008, pp.1147-1151.

[23] Y.L. Chen, and Y.C. Chang, A three-phase method for patent classification ” Information Processing and Management”. 2012, pp.1017-1030.48.

 ISBN: 978-1-61208-408-4

[24] Y.L. Chen, and Y.T. Chiu, Vector space model for patent documents with hierarchical class labels ”Journal of Information Science”. 2012, pp.222-233.38(3)

[25] Y.L. Chen, and Y.T. Chiu, An IPC-based vector space model for patent retrieval “Information Processing and Management”. 2011, pp.309-322.47.

[25] Y. H. Tseng, C. J. Lin, and Y. I. Lin, “Text mining for patent mapanalysis”. Information Processing & Mangement”. Sep. 2007, pp.1216-1247. vol.43, issue 5.

[26] Y.H. Tseng, C.J. Lin, and Y.I. Lin, “Text mining techniques for patent analysis ” Information Processing and Managemnet”. 2007, pp.1216-1247.43.

[27] The Stanford Natural Language Processing Group, The Stanford Parser: A statistical parser, http://nlp.stanford.edu/software/lex-parser.shtml

[28] MINIPAR is a broad-coverage parser for the English language. http://webdocs.cs.ualberta.ca/~lindek/minipar.htm

[29] Knowledgist retrieves, analyzes, and organizes information into a meaningful, robust, personal knowledge base. https://invention-machine.com/

 ISBN: 978-1-61208-408-4

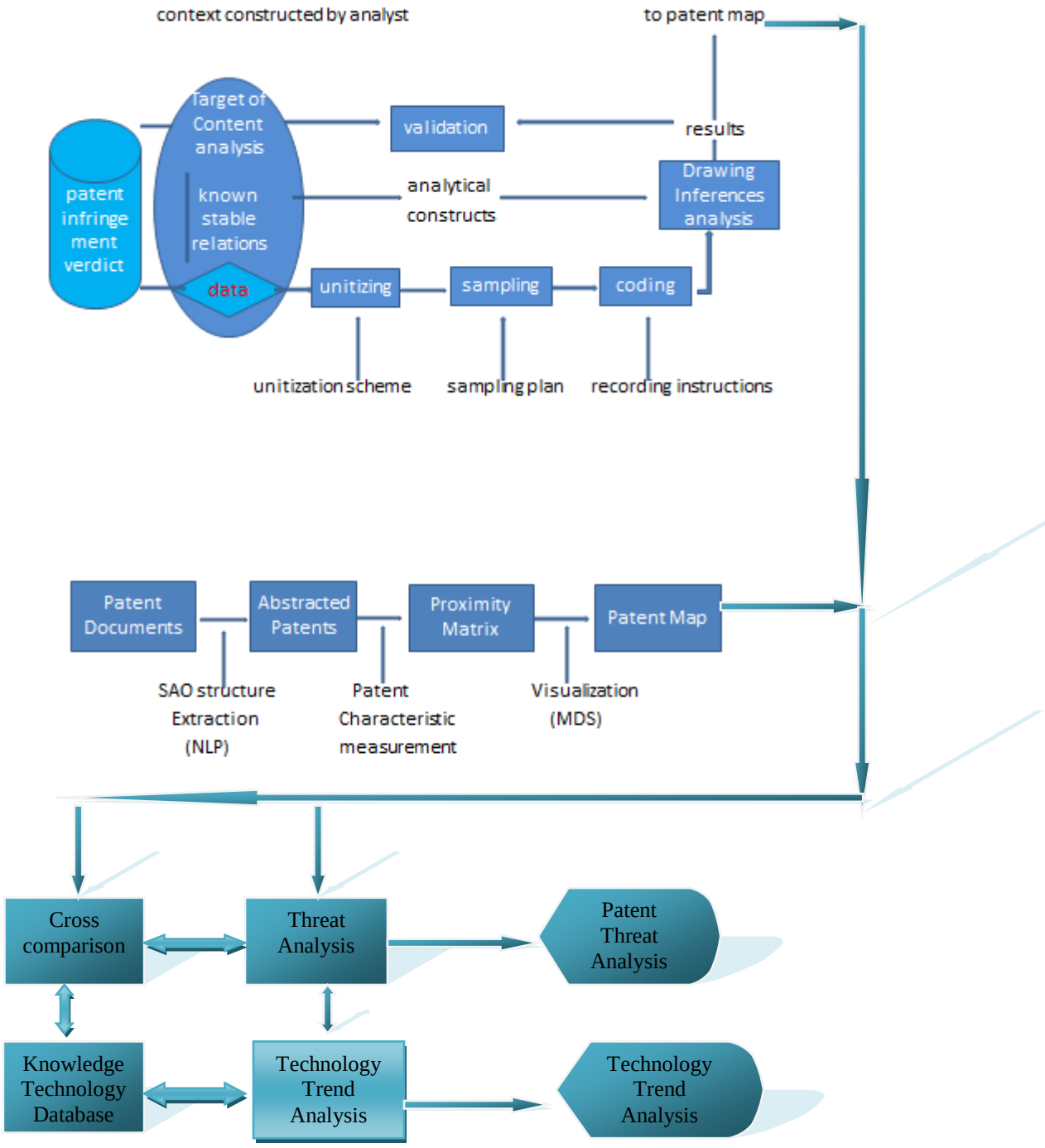


Figure 1. The framework of patent threat analysis search engine

 ISBN: 978-1-61208-408-4

Dataconda
A software Framework for Mining Relational Databases

Michele Samorani

Alberta School of Business
University of Alberta
Edmonton, AB, Canada
Email: samorani@ualberta.ca

***Abstract*—Dataconda is a software program, freely available to academics on www.dataconda.net, which solves classification and regression problems in a relational database, as opposed to a single table. The user selects a class attribute contained in a table of a relational database, and the software builds and selects predictors by exploring the whole database and aggregating information, without any user intervention. For example, Dataconda may find that the best predictor for "customer value" is the amount of money spent by the customer in cheap electronics, even if the user has not built any such attribute. This demo will introduce a brief theoretical background, illustrate how to use Dataconda in sample databases, and show how to easily extend it.**

Keywords–Relational Data Mining; Data Preparation; Propositionalization; Classification; Regression

I. Theoretical Background on Attribute Generation

Preparing a flat mining table from a relational database is a critical task of the data mining process, because it determines both the predictive performance of the statistical model and the discovery of new knowledge. This task is typically performed manually by an analyst, whereas Dataconda aims at performing it automatically. The main advantage of an automatic approach over a manual one is the ability to find unexpected patterns more easily.

As an example, let us consider the entity relationship diagram of Figure 1, which shows a database of *Purchases*, *Products*, and *Clients*. A client makes 0-to-n purchases through time. Each purchase involves only one product, but a product may be sold in 0-to-n purchases. The goal is to find predictors of the binary attribute *Return* (target attribute) of the table *Purchases* (target table), which indicates whether a purchase was later returned to the store.

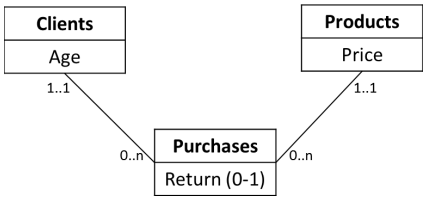


Figure 1. Entity-relationship diagram of a database of purchases

Dataconda adds to the table *Purchases* new attributes that summarize information from other tables, in the hope that they are good predictors of *Return*. The attribute generation procedure is typically referred to as *Propositionalization* [1] and works in two steps: in the first step, the procedure generates a path that starts from the target table; in the second step, a "Roll-up" procedure iteratively joins the tables of the path in order to add a new attribute to the target table, which summarizes the information along the path. The summarization is performed by using refinements, i.e., *where* conditions in the Structured Query Langauge (SQL) or aggregation operators like average (*AVG*), sum (*SUM*), etc. This procedure ultimately results in the addition of a large number of attributes to the target table.

In the example of Figure 1, the first step could generate the path *Purchases* $\rightarrow$ *Clients* $\rightarrow$ *Purchases*. In the second step, the Roll-up procedure will iteratively join the tables of the path in order to generate a new attribute for the table *Purchases*. For example, one attribute that can be generated along this path is the average value of *Return* among the client's past purchases, which represents the client's past return rate.

The initial Propositionalization work in [1] has been improved in [2] with the addition of more aggregation operators; however, the method in [2] does not allow the same table to be traversed twice during step 1 of the procedure, which drastically limits the space of the possible predictors. The method in [3] increases the number of possible refinements; however, their methodology is unsuitable to handle temporal data, because the automatic generation would generate predictors that use future information. These problems are overcome by Dataconda [4].

II. Using Dataconda

This section gives a brief overview of how to use Dataconda. The software and the complete tutorial [5] can be found online.

A. Declare Tables and Relationships

The first step is to load the tables of an existing relational database and to declare the relationships among them (*0-to-n* or *0-to-1*). Then, the user needs to define which "aggregating functions" and "refinements" may be used on each attribute.

Figure 2 shows the Dataconda configuration for a database similar to that of Figure 1. Aggregating functions (e.g., *AVG*, *SUM*) are used to aggregate a set of values into one. For example, applying the operator *AVG* on the attribute *Return* enables the generation of an attribute representing the proportion of a client's purchases that were returned.

Refinements, which are the same as "where" conditions in SQL, allow the aggregations to be performed on a subset of

 ISBN: 978-1-61208-408-4

rows, as in the attribute *AVG(Return) where Online = 1*, which represents a client's return rate among online purchases.

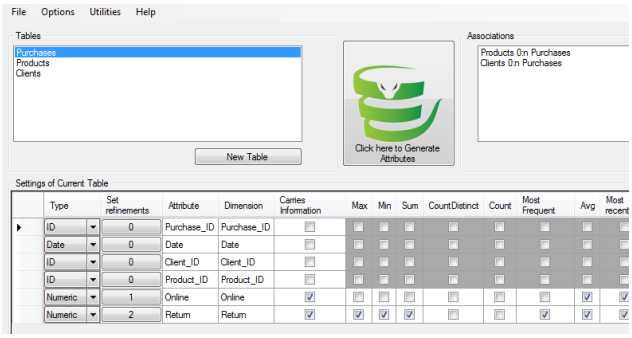


Figure 2. Snapshot of Dataconda

B. Generate Attributes

The user can then select a *target table* and a *target attribute* in it, and Dataconda will generate all possible predictors for the target attribute that can be obtained using the aggregating functions and refinements defined in the previous step. Practically, the set of columns of the target table is "expanded" with the generated predictors. This process may take seconds or hours, depending on the size of the database and on the number of total predictors to build.

C. Output of Dataconda

The output of the attribute generation procedure consists of several files, which will be located in the same folder as the data:

- The files *data.csv* and *data.arff*, which contain the flat table (in two different formats);
- A file *attributes.txt* with the English description of each generated attribute;
- A file *Analysis Output.txt*, which contains the detailed result of the attribute selection procedure.

D. Select Attributes

After generating attributes, Dataconda will find the best predictors through the default attribute selection procedure, which is based on a Lasso regression [6]. Depending on whether the dependent variable is a numeric or a categorical attribute, a linear or logistic Lasso regression is executed with decreasing values of the shrinkage coefficient λ, so as to retrieve the first 20 attributes that enter the set of selected predictors. Then, these 20 attributes are regressed by themselves against the dependent variable, and only the significant ones (with p-value ≤ 0.05) are finally returned. Figure 3 shows how the selected predictors are displayed in Dataconda.

The choice of implementing this Lasso-based attribute selection procedure is justified by its computational speed: it takes a time between 0.1 and 1.9 seconds to analyze a data set of about 500 records and 222 attributes. However, this demo will compare the attribute selection and classification performance of several other data mining techniques.

E. Extending Dataconda

Dataconda can be extended in two ways: by modifying the default attribute selection procedure or by introducing new aggregating operators.

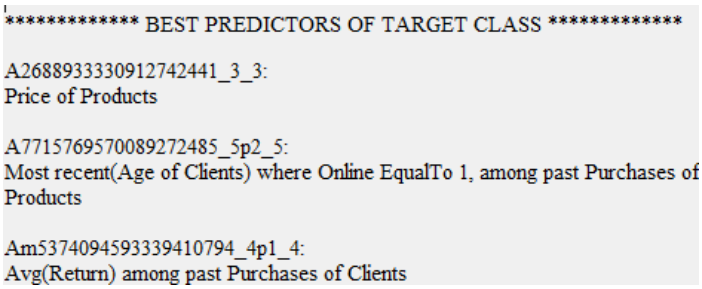


Figure 3. Selected predictors

The attribute selection procedure can be modified by changing the file *RTemplate.R*, which contains the attribute selection procedure executed by Dataconda at the end of the attribute generation procedure.

To add new aggregating operators, the user needs to extend the interface *IAggregatingFunction* by specifying the name, description, and logic of the new aggregating operator. Then, the new code needs to be compiled into a *dll*, which then needs to be placed in the same folder as the executable file *Dataconda.exe*. At start-up, Dataconda will load the new operator and will display it in the graphical interface together with the default operators (see Figure 2) .

III. Conclusion

Preparing a flat mining table from a relational database is a critical task of the data mining process, because it determines the predictive performance and the discovery of new knowledge. This task is typically performed manually by an analyst who, guided by domain knowledge, constructs a set of attributes that will hopefully result in a high predictive performance. Aside from being very time consuming, this process is also unlikely to discover unexpected knowledge, as the attributes in the mining table are those that the analyst "suspects" being related to the target attribute.

By automatically generating new attributes, Dataconda can discover new knowledge more easily, and it could also lead to a higher predictive performance than the manual process. To assess the validity of this claim, more experimentation is needed both on simulated and on real-world data sets.

References

[1] A. J. Knobbe, M. De Haas, and A. Siebes, "Propositionalisation and aggregates," in *Principles of Data Mining and Knowledge Discovery*, pp. 277–288, Springer, 2001.

[2] C. Perlich and F. Provost, "Distribution-based aggregation for relational learning with identifier attributes," *Machine Learning*, vol. 62, no. 1-2, pp. 65–105, 2006.

[3] M. Samorani, M. Laguna, R. K. DeLisle, and D. C. Weaver, "A randomized exhaustive propositionalization approach for molecule classification," *INFORMS Journal on Computing*, vol. 23, no. 3, pp. 331–345, 2011.

[4] M. Samorani, "Automatic generation of relational independent variables," in *Proceedings of 2014 INFORMS Workshop on Data Mining and Analytics (DMA 2014) D. Sundaramoorthi, H. Yang, eds.*, 2014.

[5] M. Samorani, *Dataconda User Guide*. http://www.dataconda.net/tutorial.html, 2014. [accessed: 2015-03-03].

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

 ISBN: 978-1-61208-408-4

Efficient Media Digital Library of Summarized Video based on Scalable Video Coding for H.264 (MDLSS): Design and Implementation

Hesham Farouk, Mayada Khairy
Dept. of Computers and Systems
Electronics Research Institute, ERI
Cairo, Egypt
{hesham, mayada}@eri.sci.eg

Kamal A. ElDahshan, Amr Abozeid
Dept. of Math., CS Division
Faculty of Science, Al-Azhar University
Cairo, Egypt
{dahshan, amrapozaid}@gmail.com

Alaa Hamdy, Amr Elsayed
Dept. Telecomm. And Electronics.
Faculty of Engineering, Helwan Univ.
Cairo, Egypt
{alaa.hamdy, amr18188}@gmail.com

***Abstract*—With the fast progress of wireless networks bandwidth and mobile devices, large scale digital video library systems are growing rapidly. However, the huge increasing of content and the data intensive nature of video make the management and browsing of video collections, as well as their search and retrieval, increasingly difficult.**

The need of having a media digital library is essential these days with intelligent tools for indexing the video with allocating the suitable metadata that describe the content of such videos and appropriate tools for retrieving the archived video with fast techniques. These will be achieved in 3 steps, working on the stream coding with multi bit rates and methods of handling, representing the video with summarized stream carrying the same information of the full stream and deriving a media digital library for indexing and retrieval process. The first step, stream handling, will address implementing scalable video techniques which set the bit rate according to the required application and the delivery devices. Scalable video coding offers a solution for meeting such heterogeneous requirements. The second step, video summarization, plays an important role in this context; it makes navigation easier and provides the user with a quick idea about the content. Another issue is that the same video content can be accessed from a wide variety of terminal devices which differ with respect to bandwidth limitation, decoding complexity, power constraints and screen size. The third step is implementing a media digital library for storing the code and/ or the summarized video based on Media Asset management system.

The main contribution of this paper is to explore the use of scalable video coding and video summarization techniques to enhancing a digital video library and the integration between these 3 modules.

Keywords-Video processing; Scalable video coding; Video summarization; Digital library.

I. INTRODUCTION

Nowadays, multimedia communications have significantly facilitated and enriched people's daily life. People have witnessed the fast development of various wireless multimedia applications, such as video content distribution (e.g., YouTube) and live video communications (e.g., Skype, Microsoft Network (MSN) messenger, etc.). As a result, the volume of video data is rapidly increasing; over 6 billion hours of video are watched each month on YouTube and more than 100 hours of video are uploaded to YouTube every minute [1]. Moreover, the increased popularity of mobile devices and wireless networks and their ubiquitous use for video recording and streaming leads to a dramatic increase traffic of videos traffic on such devices. Cisco stated that "Mobile Video will generate over 69 Percent of Mobile Data Traffic by 2018". Mobile makes up almost 40% of YouTube's global watch time [2].

A. Problem identification

Video delivery, especially, via mobile wireless networks, faces diverse challenges, including limited bandwidth, dynamic network conditions with low stability, a variety of relay equipment, different terminal decoding speeds, various display screen resolutions, limited battery capacity, etc. [3]. Therefore, the video coding system must encode the video sequence in different frame sizes, frame rates, and bit rates to meet such heterogeneous demands [4]. Another problem is that, the increasing amount of content and the intensive nature of video data make very difficult the management and browsing of stored video collections, as well as their search and retrieval [5].

B. Need for the system

Video is increasingly becoming one of the most pervasive technologies in terms of everyday usage, both for entertainment and in the enterprise environments. Mobile video is responsible for a majority of the growth seen in mobile broadband data volume. The demand for better video services (streaming, storing, retrieving, browsing, etc.) for mobile devices is a key challenge. The proposed system aims to solve some of these challenges.

The Middle East region, especially Egypt, has a huge amount of audio and visual heritage. There is a real need for developing a system to help in archiving digitized content and operate these rich video libraries with up to date video technologies, such as scalable video and video summarization.

This paper is organized as follows: Section II introduces the related work. Section III explains the research approach and methodology. Section IV presents the proposed system, Media Digital Library of Summarized Video based on Scalable Video Coding, MDLSS, design architecture and discusses its modules. Finally, in Section V, we conclude the paper and hint to future work.

 ISBN: 978-1-61208-408-4

II. REALTED WORK

This section will look at the state of the art of Module 1 and 2 from the proposed system (Module 3 will be the host of the deliverables from Module 1 and 2).

A. Scalable video coding (SVC)

A number of fast algorithms of fast mode decision schemes have been proposed for SVC [4][6][7][8]. For spatial scalability, a fast mode decision algorithm based on distribution relationship between the base layer and enhancement layers is used [4][9].

The scheme in Li et al.[8] represents the mode distribution relationship between base layer and enhancement layer. It is employed to reduce the candidate mode set at enhancement layers. The experimental results show that the proposed scheme provides significant reduction in computational complexity without any noticeable coding loss. Kim et al. [7] proposed a fast mode spatial, temporal, quality, and combined scalability. This algorithm is based on Coded Block Pattern (CBP) of 16×16 mode in current frame and CBP of coded best mode in selected reference frame.

B. Video summarization

Video summarization is the process of extracting the most important information and reduces the amount of redundant information from the video. The input video must be well processed in order to extract only the most useful contents [10]. But, to generate a good video summary, a full understanding of the video is required, which is still a research challenge.

In literature, many video summarization approaches have been introduced [11]. Farouk et al. [12] the authors have analyzed and compared between various techniques of mobile video summarization according to criteria. Some examples of these criteria include: content structure, final summary representation, features based, targeted devices, summarization speed, summarization purposes, adaptability and complexity. These criteria have been extracted from the reading and the analysis of literature and works in the video summarization field. Here is a summary of the observations derived from the current literature:

- The final representation of video summarization techniques is a static summary.
- Color feature was widely used in literature to summarize the video contents
- In recent years (from 2010), there is a new research direction to summarize home videos originating from the mobile camera as in [13][14].
- The goal is to reduce the technique complexity and use it to generate an online video summary.
- The purpose of mobile video summarization approaches is usually for browsing or/and streaming to another device.

III. RESEARCH APPROACH AND METHODOLOGY

The proposed research in this paper will address 3 research tracks:

1. The scalable video coding will target to enhance the bit rate transmission for spatial video streams.
2. The video summarization will target to develop an effective summarization approach comparable to the state of the art approaches.
3. The Digital library technology will work on developing a smart indexing and retrieving technique based on Media Asset Management concepts already installed in Electronics Research Institute, ERI.
4. The Integration phase, which will work on implementing the work flow described in Figure 1.

A. The first motivation of this system

Today, there is a wide range of different devices available for viewing video content, including smartphones, tablets, laptops and televisions. Every client's requirement differs with respect to bandwidth limitation, decoding complexity, power constraints and screen size. Scalable Video Coding (SVC) offers a solution for meeting such heterogeneous requirements [15].

A video bit stream is called scalable if a part of the stream can be removed in such a way that the resulting bit stream is still decodable. The three types of scalability are [16]:

1. Temporal (frame rate) scalability: the motion compensation dependencies are structured so that complete pictures (i.e., their associated packets) can be dropped from the bit stream. Temporal scalability is already enabled by H.264/MPEG-4 Advanced Video Coding (AVC) [17]. The SVC has only provided supplemental enhancement information to improve its usage.
2. Spatial (picture size) scalability: video is coded at multiple spatial resolutions. The data and decoded samples of lower resolutions can be used to predict data or samples of higher resolutions in order to reduce the bit rate to code the higher resolutions.
3. Signal to Noise Ratio (SNR)/Quality/Fidelity scalability: video is coded at a single spatial resolution, but at different qualities. The data and decoded samples of lower qualities can be used to predict data or samples of higher qualities in order to reduce the bit rate to code the higher qualities.

This work is represented as Module 1 of the proposed work given in Figure 1.

B. The second motivation of this system

The content of video may be huge and crowded with much redundant information, so that it often takes a long time to browse the content from the beginning to the end. Also, the user may not have sufficient time to watch the entire video or the video content, as a whole, may not be of interest to the user. In such cases, the user may just want to view the

 ISBN: 978-1-61208-408-4

summary of the video instead of watching the whole video [18].

Video summarization is a mechanism for generating compact representation of a video sequence, which includes only the important parts in the original video [19]. Video summarization is useful when a system is operating under tight constraints (e.g., limited bandwidth, watching time, or memory size). For example, in surveillance applications the video may be recorded nearly for 24 hours per day, a summary version of the original video may be useful to watch the important events only in such case. Also, video summarization is useful when we need to transmit an important video segment to another device in real time [20]. Video summarization techniques target different domains of video data, such as sports, news, movies, documentaries, e-learning, surveillance, home videos, etc., and discuss various assumptions and viewpoints to produce an optimal or good video summary [18].

This work is represented as module 2 of the proposed work given in Figure 1.

There are two fundamental types of video summaries [21]: static video summary (also called representative frames, still-image abstracts or static storyboard) and dynamic video skimming (also called video skim, moving image abstract or moving storyboard). The static video summary is a collection of video frames extracted from the original video. The dynamic video summary is a set of short video clips, joined in a sequence, and played as a short video clip. Usually, from the users viewpoint, a dynamic video summary may provide a better choice since it contains both audio and motion information that makes the summarization more interesting and natural, while static video summary may provide a glance of video contents in a more concise way. In addition, once video frames are extracted, there are further possibilities of organizing them for browsing and retrieving purposes [22].

C. The third motivation of this system

The Digital library for media file under Media Asset Management (MAM) system will be the main storage system for the processed video by Module 1 and Module 2 and will be based on the MAM purchased by ERI through the 'EQUIPME' initiative issued by research academy 2 years ago [23].

IV. THE PROPOSED SYSTEM

A. The proposed system architecture

The architecture of MDLSS consists of three modules; scalable video coding, video summarization and digital library, as shown in Figure 1. MDLSS aims at providing better video services (streaming, storing, retrieving and browsing) for mobile devices which increase the interactions and activities between users and digital libraries. In other words, the main goal of MDLSS is to explore the use of SVC and video summarization techniques for enhancing digital video library. This goal can be further specified in the following.

- Design and develop SVC algorithm to meet the requirements of applications and devices heterogeneities.
- Design and develop an automatic video summarization algorithm, which engages in providing concise and informative video summaries to help in browsing and managing video files efficiently.

B. The system design methods and procedures

For Module 1:

The scalable video coding has three types, namely, spatial scalability, quality scalability and temporal scalability. This paper focuses on spatial scalability.

The output of SVC stage (Module 1 in the proposed system) is one bit stream of compressed video in H.264/SVC format. This bit stream has two levels of spatial scalability. The first one is the base-layer Quarter Common Intermediate Format, QCIF, and the second one is the enhancement-layer Common Intermediate Format, CIF. There are seven modes for inter prediction (SKIP, 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4) and there are two intra prediction modes (INTRA 4x4 and INTRA 16x16) based on the minimum rate cost equation.

General adapted methods for SVC:

1. Determine number of layers for scalable video.
2. Determine number of bitrates available.
3. Analyze the video stream.
4. Select the type of scalability according to 3 steps before.
5. Implement the scalable video types according to previous steps.

The proposed algorithm for Module 1 achieved saving in time of encoding with up to 50%, and saving in time of decoding with up to 30% compared to Joint Scalable Video Coding, JSVC. This is done by selecting best macroblock mode.

For Module 2:

A general adapted method for video summarization module is shown in Figure 2. Each step is described as follows:

1. Frames sampling

The first step towards automatic video summarization is splitting the video stream into a set of meaningful and manageable basic elements (e.g., shots, frames) that are used as basic elements for summarization. Most of existing methods for automatic video summarization have focused on splitting the video stream into frames. The video sequence is decoded and each frame is extracted and treated separately [13].

 ISBN: 978-1-61208-408-4

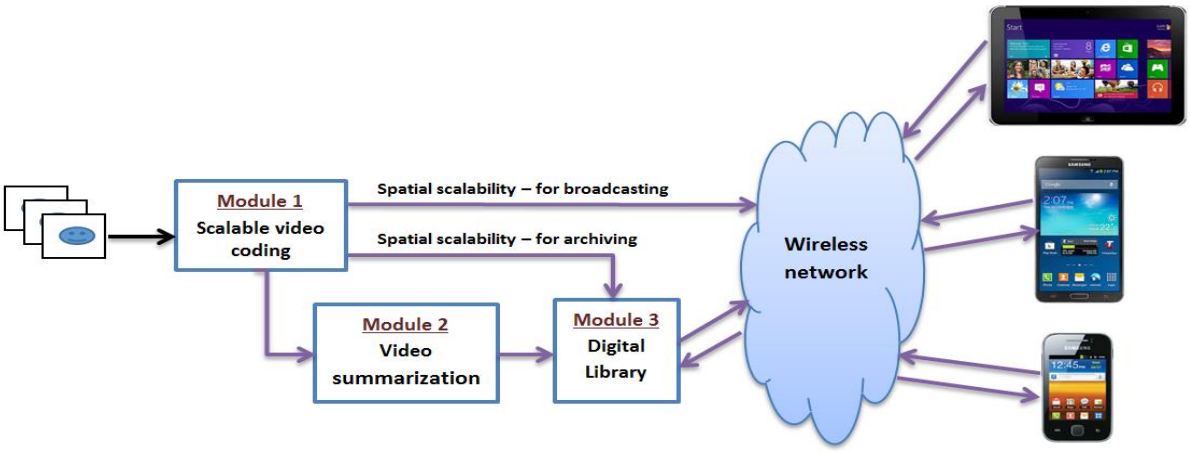


Figure 1. MDLSS architecture

2. Feature extraction

Digital video contains many features, like color, motion, voice, etc. Color feature is considered an important aspect of video. That is why it has been used quite often for video summarization. Color based summarization techniques are very simple and easy to use. However, color-based methods accuracy is not reliable, as color based techniques may consider noise as part of the summary [12].

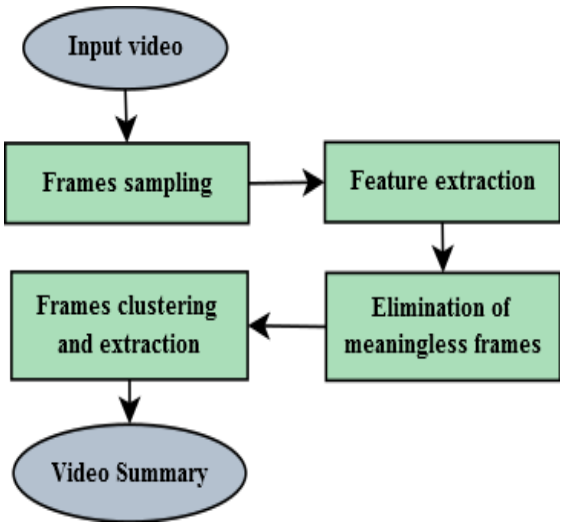


Figure 2. Flowchart of video summarization module

3. Elimination of meaningless frames

The goal of this step is to avoid possible meaningless frames in a video summary. It has been generally observed that a video, usually, has some meaningless frames, such as totally black frames, totally white frames (a monochromatic frame) and faded frames [22].

4. Frames clustering and extraction

The goal of this step is to group similar video frames together and to select a representative frame per each group, to produce the video summary. The effectiveness of grouping similar frames depends on the suitable choice of a similarity metric used for comparing two frames [24].

Video summarization is has been a very active research field in recent years due to its important role in many video services (e.g., browsing, indexing and streaming). The reader can find a comprehensive review of video summarization techniques in [11][25]. Moreover, Farouk [12] produced an analysis and comparative study between various techniques proposed in literature for the summarization of video content, which can be useful for mobile applications.

For Module 3:

In this step, we will manage video storage after editing and applying Meta data through the Media Asset Management we already have in our Digital Signal Processing Lab in Electronics Research Institute.

V. A CASE STUDY OF VIDEO SUMMARIZATION

In this case study, we show how can generate a static video summary from the SVC originated from Module 1.

A. Frames sampling

The input of video summarization (Module 2) is a QCIF video in H.264/SVC format. This video is partially decoded to select sample frames from it based on a predefined sampling rate. In this case study, the sampling rate is set to be two frames per second.

B. Feature extraction

In this case study, a color histogram is applied to describe the visual content of video. There are two key issues in applying the color histogram technique which are the selection of a suitable color space and the quantization of that color space [21]. Since the Human Visual System (HVS) is more sensitive to luminance than color [17], we choose the YUV 4:2:0 color space and compute histogram only for Y (luminance) component and discard the chrominance

components U and V. The quantization of the color histogram is set to 16 color bins that are normally used for hue component aiming at reducing the amount of data significantly without losing the important information.

C. Elimination of meaningless frames

In this step, for each candidates frame the standard deviation of pixels is computed, as in [21][26]. If the standard deviation is very low (close to zero) then this frame is considered as a meaningless frame and is discarded. We apply this step before the clustering step (as preprocessing step) like [21], which saves the computation cost.

D. Frames clustering and extraction

In this case study, we use the adopted Zero-mean Normalized Cross Correlation (ZNCC) [22] as the similarity metric between two frames. ZNCC is widely used in template matching, motion analysis, stereo vision, and industrial inspection. Let H_{t_1} and H_{t_2} be the color histograms extracted from the video frames F_{t_1} and F_{t_2} taken at the times t_1 and t_2, respectively. The ZNCC between H_{t_1} and H_{t_2} is defined by (1).

$$ZNCC(H_{t_1}, H_{t_2}) = \frac{\sum_i \left((H^i_{t_1} - \bar{H}_{t_1}) \times (H^i_{t_2} - \bar{H}_{t_2}) \right)}{\sqrt{\sum_i (H^i_{t_1} - \bar{H}_{t_1})^2 \times \sum_i (H^i_{t_2} - \bar{H}_{t_2})^2}}. \quad (1)$$

Where H^i be the ith bin of the color histogram H and $\bar{H}$ is the mean value of all entries of H. The ZNCC function returns a real value from -1 to 1. The value of -1 is returned for situations in which those histograms are not similar at all, and the value of +1 is returned for situations in which they are identical [22].

In order to group similar video frames together, we apply the cluster algorithm which is described as Algorithm 1.

Where ε is a threshold for the similarity between frames and through the experiment, we have found that the values of ε between 0.3 and 0.7 are best choices. Finally, the middle frame is selected from each cluster to form the video summary.

Algorithm 1: the cluster algorithm

Input: F_{t_k}, $k = 1,2,\dots,n$ // the set of candidates frames
Output: C_j, $j = 1,2,\dots,m$; $m <$ n // a set of
Start
1. Initialize j=1
2. Loop for each $k: 1 \rightarrow N$
3. If ($ZNCC(H_{t_k}, H_{t_{k+1}}) > \varepsilon$) then
4. Add $F_{t_k}, F_{t_{k+1}}$ to the cluster C_j // without duplicate
5. k=k+1
6. Else
7. Add F_{t_k} to the cluster C_j // without duplicate
8. k=k+1, j=j+1
9. End loop

End

.

E. Experiments and Results

The setup environment for testing the proposed modules is based on a network has bandwidth 100 Mb/s. We assume 70% of the bandwidth is dedicated for transferring videos.

1) Data set and testing device

This experiment is carried out on 4 videos from the standard data set available at the Video Summarization (VSUMM) web site [27]. Table I contains the descriptions of these videos. Each video format is MPEG-1 with resolution of 352×240 pixels, 30 frames per second and in color and with sound. Because of the input format to our approach is H.264/AVC, each video is firstly transcoded to match the input format.

TABLE I. THE TEST VIDEOS DESCRIPTION

#	Video name	Duration	#Frames	Genre
1	Exotic Terrane, segment 01 of 12	00:01:38	2,940	Documentary
2	A New Horizon, segment 05 of 13	00:01:59	3,561	Documentary
3	Senses And Sensitivity, Introduction to Lecture 3 presenter	00:02:32	4,566	Lecture
4	Digital Jewelry: Wearable Technology for Every Day Life	00:03:00	4,204	Educational

We implemented a prototype to test this case study based on Java platform. The JCodec library is used to partially decode the input video [28]. All the experiments were performed on a PC device equipped with an 8 GB of DDR3-memory and Intel Core i7 processor.

2) Evaluation method

The Mean Opinion Score (MOS) method proposed in [21][29] is used in this evaluation. In this method, the quality of the automatically generated summary is compared to the users (Mostly, five users) generated summary. Then, we compute the Accuracy Rate (AR) and Error Rate (ER) metrics as in (2) and (3) respectively.

$$AR = \frac{N_{TM}}{N_{US}}. \quad (2)$$

$$ER = \frac{N_{FM}}{N_{US}}. \quad (3)$$

Where N_{TM} denotes the total number of the frames that exists as key frame in both the user and the automatic

summary. The symbol N_{FM} denotes the total number of the frames that is exists in the automatic summary and not exists in the user summary. Finally, N_{US} denotes the total frames number in the user summary. Both accuracy rate and error rate are complementary metrics and the highest quality of summary was achieved when AR = 1 and ER = 0.

3) Comparison with other techniques

In order to evaluate the quality of the proposed approach, we compare it with other static video summary approaches found in the literature. The compared approaches include VSUMM [21], Delaunay Triangulation (DT) [30], STIll and MOving (STIMO) video storyboard for the web scenario [31] and Open Video Project (OV) [32].

The comparative results are provided in Table II. Also, an example is shown in Figure 3. The results demonstrate that the proposed approach achieved an average AR of 0.84 and an average ER of 0.24 with respect to the users' generated summary. These results indicated that, the proposed approach has a balance between a high accuracy rate and a low error rate.

TABLE II. THE COMPARISON OF DIFFERENT TECHNIQUES

	OV		***DT***		***STIMO***		***VSUMM***		***Proposed***	
NO.	***AR***	***ER***	***AR***	***ER***	***AR***	***ER***	***AR***	***ER***	***AR***	***ER***
1	0.98	1.39	0.48	0.32	0.66	0.53	0.82	0.53	0.85	0.23
2	0.63	0.12	0.24	0.08	0.43	0.15	0.82	0.19	0.81	0.12
3	0.6	0.3	0.38	0.34	0.79	0.57	0.84	0.43	0.82	0.32
4	1	0.7	0.6	0.1	0.94	0.46	0.9	0	0.89	0.27
Average	**0.8025**	**0.6275**	**0.425**	**0.21**	**0.705**	**0.4275**	**0.845**	**0.2875**	**0.8425**	**0.235**

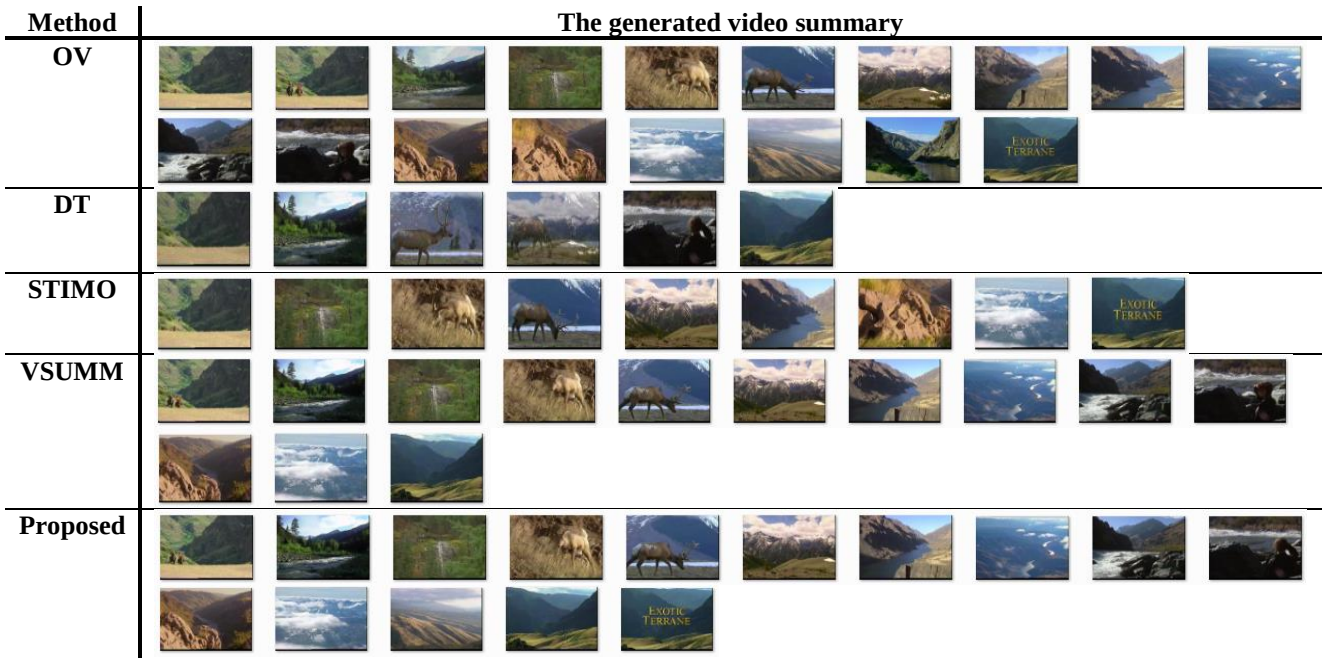


Figure 3. Comparison of video summary extraction for "Exotic Terrane, segment 01 of 12" video

VI. CONCLUSION AND FUTURE WORK

The SVC, as well as the video summarization, plays an important role in many video services. So, in this paper, we presented an efficient media digital library framework design of summarized video based on SVC for H.264 (MDLSS) with a test case study as a proof of concept. The proposed design will utilize the conjunction between SVC and video summarization techniques to enhance the digital video library.

In the future, we plan to do a full implementation to this proposed system (MDLSS). Our implementation activities will be organized as follows:

- Analyzing the system requirement for each module and for integration
- Developing a system prototype.
- Testing the system and updates.

REFERENCES

1. Youtube statistics. Available from: http://www.youtube.com/yt/press/statistics.html, [retrieved: 5, 2015].
2. V. N. I. (VNI). Cisco visual networking index: Global mobile data traffic forecast update, 2013–2018. Available from: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, [retrieved: 5, 2015].
3. N. V. Uti and R. Fox, The challenges of compressing and streaming real time video originating from mobile devices, in Multimedia services and streaming for mobile devices: Challenges and innovation. 2011, IGI Global. p. 1-24.
4. P.-C. Wang, G.-L. Li, S.-F. Huang, M.-J. Chen, and S.-C. Lin,"Efficient mode decision algorithm based on spatial, temporal, and inter-layer rate-distortion correlation coefficients for scalable video coding," ETRI journal, vol. 32, 2010, pp. 577-587, doi:org/10.4218/etrij.10.0109.0622.
5. L. Herranz and J. M. Martínez,"Combining mpeg tools to generate video summaries adapted to the terminal and network," The Computer Journal, vol. 56, 2013, pp. 529-553, doi:10.1093/comjnl/bxs104.
6. L. Shen, Z. Liu, P. An, R. Ma, and Z. Zhang,"Fast mode decision for scalable video coding utilizing spatial and interlayer correlation," Journal of Electronic Imaging, vol. 19, 2010, pp. 033010-033010-8.
7. T.-J. Kim, J.-J. Yoo, J.-W. Hong, and J.-W. Suh,"Fast mode decision algorithm for scalable video coding based on luminance coded block pattern," Optical Engineering, vol. 52, 2013, doi:10.1117/1.OE.52.1.017401.
8. H. Li, Z. Li, C. Wen, and L.-P. Chau,"Fast mode decision for spatial scalable video coding," Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2006). . IEEE, May 2006, pp. 4, doi:10.1109/ISCAS.2006.1693257.
9. H. Li, Z. Li, and C. Wen,"Fast mode decision algorithm for inter-frame coding in fully scalable video coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, 2006, pp. 889-895, doi:10.1109/TCSVT.2006.877404.
10. H. Karray, M. Ellouze, and A. Alimi, Indexing video summaries for quick video browsing, in Pervasive computing. 2010, Springer. p. 77-95.
11. M. Ajmal, M. H. Ashraf, M. Shakir, Y. Abbas, and F. A. Shah,"Video summarization: Techniques and classification," Computer Vision and Graphics, Springer, vol. 7594, 2012, pp. 1-13, doi:10.1007/978-3-642-33564-8_1.
12. H. Farouk, K. ElDahshan, and A. Abozeid,"The state of the art of video summarization for mobile devices: Review article," Graphics, Vision and Image Processing GVIP, vol. 14, 2014, pp. 37-50.
13. J. Niu, D. Huo, K. Wang, and C. Tong,"Real-time generation of personalized home video summaries on mobile devices," Neurocomputing, ScienceDirect, vol. 120, 2013, pp. 404-414, doi:10.1016/j.neucom.2012.06.056.
14. G. Abdollahian, C. M. Taskiran, Z. Pizlo, and E. J. Delp,"Camera motion-based analysis of user generated video," IEEE Transactions on Multimedia, vol. 12, 2010, pp. 28-41, doi:10.1109/TMM.2009.2036286.
15. M. Ransburg, et al.,"Scalable video coding impact on networks," Mobile Multimedia Communications, vol., 2012, pp. 571-581.
16. S. Ibrahim, A. H. Zahran, and M. H. Ismail,"Svc-dash-m: Scalable video coding dynamic adaptive streaming over http using multiple connections," Proc. 21st International Conference on Telecommunications (ICT). IEEE, May 2014, pp. 400-404, doi:10.1109/ICT.2014.6845147.
17. I. E. Richardson, The h. 264 advanced video compression standard. John Wiley & Sons, 2011,
18. B. T. Truong and S. Venkatesh,"Video abstraction: A systematic review and classification," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), vol. 3, 2007, pp. 37, doi:10.1145/1198302.1198305.
19. G. Guan, et al.,"A top-down approach for video summarization," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 11, 2014, doi:10.1145/2632267.
20. L. Zhu, Z. Fan, and K. Aggelos K,"Joint video summarization and transmission adaptation for energy-efficient wireless video streaming," EURASIP Journal on Advances in Signal Processing, vol., 2008, doi:10.1155/2008/657032.
21. S. E. F. de Avila and A. P. B. Lopes,"Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method," Pattern Recognition Letters, Elsevier vol. 32, 2011, pp. 56-68, doi:10.1016/j.patrec.2010.08.004.
22. J. Almeida, N. J. Leite, and R. d. S. Torres,"Online video summarization on compressed domain," Journal of Visual Communication and Image Representation, vol. 24, 2013, pp. 729-738, doi:10.1016/j.jvcir.2012.01.009.
23. Ncpower. Available from: http://www.norcom.de/en/features-ncpower, [retrieved: 5,2015].
24. S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien,"Low complexity on-line video summarization with gaussian mixture model based clustering," Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) IEEE, May 2014, pp. 1260-1264, doi:10.1109/ICASSP.2014.6853799.
25. R. Pal, A. Ghosh, and S. K. Pal, Video summarization and significance of content: A review, in Handbook on soft computing for video surveillance. 2012, CRC Press. p. 79-102.
26. N. Ejaz, T. B. Tariq, and S. W. Baik,"Adaptive key frame extraction for video summarization using an aggregation mechanism," Journal of Visual Communication and Image Representation, vol. 23, 2012, pp. 1031-1040.
27. Vsumm (video summarization). Available from: http://www.npdi.dcc.ufmg.br/VSUMM [retrieved: 4, 2015].
28. Jcodec. Available from: http://jcodec.org/, [retrieved: 5, 2015].
29. N. Ejaz, I. Mehmood, and S. W. Baik,"Feature aggregation based visual attention model for video summarization," Computers & Electrical Engineering, vol. 40, 2014, pp. 993-1005.
30. P. Mundur, Y. Rao, and Y. Yesha,"Keyframe-based video summarization using delaunay clustering," International Journal on Digital Libraries, vol. 6, 2006, pp. 219-232.
31. M. Furini, F. Geraci, M. Montangero, and M. Pellegrini,"Stimo: Still and moving video storyboard for the web scenario," Multimedia Tools and Applications, vol. 46, 2010, pp. 47-69.
32. The open video project. Available from: http://www.open-video.org/index.php, [retrieved: 5, 2015].

 ISBN: 978-1-61208-408-4

Behind the Skyline

Markus Endres

University of Augsburg
Augsburg, Germany
Email: markus.endres@acm.org

Timotheus Preisinger

DEVnet Holding GmbH
Grünwald, Germany
Email: t.preisinger@devnet.de

***Abstract*—A *Skyline* query selects those tuples from a dataset that are optimal with respect to a set of designated preference attributes. However, in some cases, not only the Pareto frontier is of interest, but also the stratum behind the Skyline. In this paper, we extend the definition of the Skyline to form *multi-level* Skyline sets. We propose an algorithm for multi-level Skyline computation and apply this concept for efficient *top-k Skyline* evaluation. Given a dataset, a *top-k Skyline* query returns the *k most interesting* elements of the Skyline query based on some kind of user-defined preference. We demonstrate through extensive experimentation on synthetic and real datasets that our algorithm can result in a significant performance advantage over existing techniques.**

Keywords–Skyline; Preferences; Multi-level; Top-k.

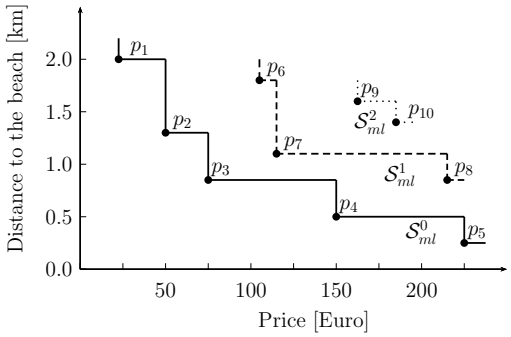


Figure 1. Skyline example.

I. INTRODUCTION

The Skyline operator [1] has emerged as an important and popular technique for searching the best objects in multi-dimensional datasets. A Skyline query selects those objects from a dataset D that are not dominated by any others. An object p having d attributes (dimensions) dominates an object q, if p is strictly better than q in at least one dimension and not worse than q in all other dimensions, for a defined comparison function. Without loss of generality, we consider subsets of $\mathbb{R}^d$ in which we search for Skylines w.r.t. the natural order $\leq$ in each dimension.

The most cited example on Skyline queries is the search for a hotel that is *cheap* and *close to the beach*. Unfortunately, these two goals are conflicting as the hotels near the beach tend to be more expensive. In Figure 1, each hotel is represented as a point in the two-dimensional space of *price* and *distance to the beach*. Interesting are all hotels that are not worse than any other hotel in both dimensions. The hotels p_6, p_7, p_9, p_{10} are dominated by hotel p_3. The hotel p_8 is dominated by p_4, while the hotels p_1, p_2, p_3, p_4, p_5 are not dominated by any other hotels and build the *Skyline* $\mathcal{S}$. From the Skyline, one can now make the final decision, thereby weighing the personal preferences for price and distance.

Unfortunately, the size of the Skyline $\mathcal{S}$ can be very small (e.g., in low-dimensional spaces). Hence, a user might want to see the *next best objects behind the Skyline*. In our example above maybe five hotels are not enough, so we have to present the next stratum called $\mathcal{S}^1_{ml}$ (*Skyline, multi-level 1*, dashed line in Figure 1): p_6, p_7, p_8. Also, the third best result set $\mathcal{S}^2_{ml}$ might be of interest: p_9, p_{10}.

Furthermore, in the presence of high-dimensional Skyline spaces, the size of the Skyline $\mathcal{S}$ can still be very large, making it unfeasible for users to process this set of objects [2]. Hence, a user might want to see the *top-k* objects. That means a maximum of k objects out of the complete Skyline set if $|\mathcal{S}| >= k$, or for $|\mathcal{S}| < k$ to use the Skyline set plus the next best objects such that there will be k results. In the previous example a *top-3* Skyline query would identify, e.g., p_1, p_2, and p_3, whereas in a *top-10* query it is necessary to consider the second and third stratum to identify p_6, p_7, p_8, p_9, and p_{10} as additional Skyline points.

In this work we generalize the well-known Skyline queries to *multi-level Skylines* $\mathcal{S}_{ml}$. We present an efficient algorithm to compute the *l-th stratum* of a Skyline query exploiting the lattice structure constructed over low-cardinality domains. Following [2] many Skyline applications involve domains with small cardinalities – these cardinalities are either inherently small (such as star ratings for hotels), or can naturally be mapped to low-cardinality domains (such as price ranges on hotels). In addition, we propose an evaluation strategy for *top-k Skyline* queries, which is based on the multi-level approach. To our best knowledge, until now there are no efficient algorithms that are specialized on finding *multi-level Skyline* sets or *top-k Skylines*. Motivated by this fact, this paper addresses this issue.

The remainder of this paper is organized as follows: In Section II we present the formal background. Based on this background we will discuss *multi-level Skyline* computation in Section III and *top-k Skyline* computation in Section IV. We conduct an extensive performance evaluation on synthetic and real datasets in Section V. Section VI contains some related work. Section VII contains our concluding remarks.

 ISBN: 978-1-61208-408-4

II. SKYLINE QUERIES REVISITED

In this section, we revisit the problem of Skyline computation and shortly describe the Lattice Skyline approach, since this is the basis of our algorithms.

A. Skyline Queries

The aim of a Skyline query is to find the *best objects* in a data set D, i.e., $\mathcal{S}(D)$. Note that Skylines are not restricted to numerical domains [3]. More formally:

Definition 1 (Skyline)**.** *Assume a set of vectors $D \in \mathbb{R}^d$. We define the so called Pareto ordering for all $x = (x_1, ..., x_d)$, $y = (y_1, ..., y_d) \in D$:*

$$x <_{\otimes} y \iff \begin{array}{lll} \forall j \in \{1, ..., d\} & : & x_j \leq y_j \\ \exists i \in \{1, ..., d\} & : & x_i < y_i \end{array} \quad \wedge \tag{1}$$

The Skyline $\mathcal{S}$ *of D is defined by the maxima in D according to the ordering $<_{\otimes}$, or explicitly by the set*

$$\mathcal{S}(D) = \{t \in D \mid \nexists u \in D : u <_{\otimes} t\} \tag{2}$$

In this sense we prefer the minimal values in each domain and write $x <_{\otimes} y$ if x is better than y.

In general, algorithms of the block-nested-loop class (BNL) [1] are probably the best known algorithms for computing Skylines. They are characterized by a tuple-to-tuple comparison-based approach, hence having a worst case complexity of $\mathcal{O}(n^2)$, and a best case complexity of the order $\mathcal{O}(n)$; n being the number of input tuples, cf. [4]. The major advantage of a BNL-style algorithm is its simplicity and suitability for computing the maxima of arbitrary partial orders. Furthermore, a multitude of optimization techniques [4][5] and parallel variants [6][7][8][9] have been developed in the last decade.

B. Lattice Skyline Revisited

Lattice-based algorithms depend on the lattice structure constructed by a Skyline query over low-cardinality domains. Examples for such algorithms are *Lattice Skyline* [10] and *Hexagon* [11], both having a worst case linear time complexity. Both algorithms follow the same idea: the partial order imposed by a Skyline query constitutes a *lattice*. This means if $a, b \in D$, the set $\{a, b\}$ has a least upper bound and a greatest lower bound in D. Visualization of such lattices is often done using *Better-Than-Graphs* (*BTG*) [12], graphs in which edges state dominance. The nodes in the BTG represent *equivalence classes*. Each equivalence class contains the objects mapped to the same feature vector of the Skyline query. All values in the same equivalence class are considered substitutable.

An example of a BTG over a 2-dimensional space is shown in Figure 2 where $[0..2] \times [0..4]$ describes a domain of integers where attribute $A_1 \in \{0, 1, 2\}$ and $A_2 \in \{0, 1, 2, 3, 4\}$ (abbr. $[2; 4]$). The arrows show the dominance relationship between elements of the lattice. The node $(0, 0)$ presents the *best node*, i.e., the least upper bound, whereas $(2, 4)$ is the *worst node*. The bold numbers next to each node are *unique identifiers* (ID) for each node in the lattice, cp. [11]. Nodes having the same level in the BTG are indifferent, i.e. for example, that neither the objects in the node $(0, 4)$ are better than the objects in $(1, 3)$ nor vice versa. A dataset D does not necessarily contain tuples for each lattice node. In Figure 2, the gray nodes are occupied (*non-empty*) with elements from the dataset whereas the white nodes have no element (*empty*).

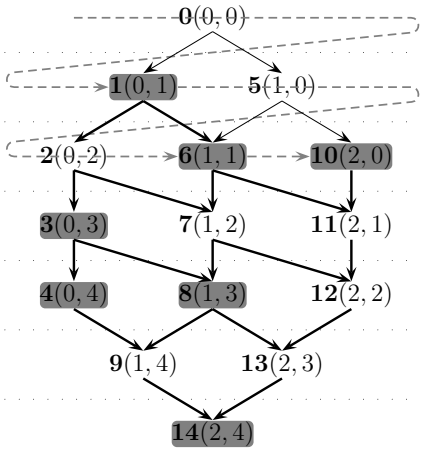


Figure 2. Lattice over $[0..2] \times [0..4]$.

The method to obtain the Skyline can be visualized using the BTG. The elements of the dataset D that compose the Skyline are those in the BTG that have *no path leading to them from another non-empty node in D*. In Figure 2, these are the nodes $(0, 1)$ and $(2, 0)$. All other nodes have direct or transitive edges from these both nodes, and therefore are *dominated*.

Lattice based algorithms exploit these observations to find the Skyline of a dataset over the space of vectors drawn from low-cardinality domains and in general consist of three phases:

1) **Phase 1:** The *Construction Phase* initializes the data structures. The lattice is represented by an *array* in main memory. Each position in the array stands for one node ID in the lattice. Initially, all nodes of the lattice are marked as *empty* and *not dominated*.
2) **Phase 2:** In the *Adding Phase* the algorithm determines for each element $t \in D$ the unique ID and therefore the node of the lattice that corresponds to t. This node will be marked as *non-empty*.
3) **Phase 3:** After all tuples have been processed, in the *Removal Phase* dominated nodes are identified. The nodes of the lattice that are marked as *non-empty* and which are not reachable by the transitive dominance relationship from any other *non-empty* node represent the Skyline values. Nodes that are *non-empty* but are reachable by the dominance relationship are marked *dominated* to distinguish them from present Skyline values.
   From an algorithmic point of view this is done by a combination of *breadth-first traversal* (BFT) and *depth-first traversal* (DFT). The nodes of the lattice are visited level-by-level in a breadth-first order (the dashed line in Figure 2). Each time a *non-empty* and *not dominated* node is found, a DFT will start. The DFT does not need to explore branches already marked as *dominated*. The BFT can stop after processing a whole level not containing *empty* nodes hence marking the end of Phase 3.
   For example, the node $(0, 1)$ in Figure 2 is not empty. The DFT recursively walks down and marks all dominated nodes as *dominated* (thick black arrows). After the BFT has finished, the *non-empty* and *not dominated* nodes (here $(0, 1)$ and $(2, 0)$) contain the Skyline objects.

 ISBN: 978-1-61208-408-4

III. Multi-Level Skyline Computation

In some cases it is necessary to return not only the best tuples as in common Skyline computation, but also to retrieve tuples directly dominated by those of the Skyline set (the second *stratum*), i.e., the tuples *behind the Skyline*. Following this method transitively, the input is partitioned into multiple levels (*strata*) in a way resembling the elements' quality w.r.t. the search preferences. In this section, we introduce the concept of *multi-level Skylines* and present an algorithm for efficient computation of iterated preferences in linear time.

A. Background

We extend Definition 1 of the Skyline by a level value to form *multi-level Skyline* ($\mathcal{S}_{ml}$) sets.

Definition 2 (Multi-Level Skyline)**.** *The multi-level Skyline set of level l (i.e., the l-th stratum) for a dataset D is defined as*

$$\mathcal{S}^l_{ml} := \mathcal{S}\left(D \setminus \bigcup_{i=0}^{l-1} \mathcal{S}^i_{ml}(D)\right) \tag{3}$$

Thereby $\mathcal{S}^0_{ml}(D)$ is identical to the standard Skyline $\mathcal{S}(D)$ from Definition 1, and $\mathcal{S}^{l_{max}}_{ml}$ denotes the non-empty set with the highest level.

Lemma 1. *For each tuple t in a finite dataset D, there is exactly one $\mathcal{S}^l_{ml}$ set it belongs to:*

$$\forall t \in D : \left(\exists! l : t \in \mathcal{S}^l_{ml}(D)\right) \tag{4}$$

Proof: A Skyline query on $D \neq \emptyset$ never yields an empty result, i.e., $\mathcal{S}(D) \neq \emptyset$. Starting at 0, for each $l = 0, 1, 2, \ldots$ the input dataset diminishes as all selection results for smaller values of l are removed from the input. Since $D \setminus \bigcup_{i=0}^{l} \mathcal{S}^i_{ml}(D) \subset D \setminus \bigcup_{i=0}^{l-1} \mathcal{S}^i_{ml}(D)$ and $|D|$ is finite, there has to be some l_{max} for which the following holds:

$$\bigcup_{i=0}^{l_{max}-1} \mathcal{S}^i_{ml}(D) \subset D \ \wedge \ \bigcup_{i=0}^{l_{max}} \mathcal{S}^i_{ml}(D) = D$$

So each tuple in D belongs to exactly one $\mathcal{S}^i_{ml}(D)$. ■

Lemma 1 shows that all tuples in a dataset belong to a $\mathcal{S}_{ml}$ set of some level. So a kind of order on D w.r.t. the Skyline query is induced.

Lemma 2. *All elements of $\mathcal{S}^l_{ml}(D)$ are dominated by elements of $\mathcal{S}^i_{ml}(D)$ for all $i < l$:*

$$\forall y \in \mathcal{S}^l_{ml}(D) : (\exists x \in \mathcal{S}^i_{ml}(D) : x <_{\otimes} y)) \ \ \textit{if} \ \ i < l \tag{5}$$

Proof: Consider a tuple $y \in \mathcal{S}^l_{ml}(D)$ that is not dominated by any element of $\mathcal{S}^i_{ml}(D)$ for $i < l$. Following Definition 2, $y \in \mathcal{S}^i_{ml}(D)$. This is a contradiction. ■

For every Skyline query on $D \neq \emptyset$ there is at least a $\mathcal{S}_{ml}$ set of level 0. If it is the only one, no tuple in D is worse than any other w.r.t. the preference. Just as well, it is possible that all tuples in D belong to $\mathcal{S}_{ml}$ sets of different levels. The Skyline query then defines a total order on the elements of D.

B. The Multi-Level Lattice Skyline Algorithm (MLLS)

We will now see how the lattice based Skyline algorithms described in Section II-B can be adjusted to support multi-level Skyline computation. We call this algorithm *Multi-Level Lattice Skyline* (MLLS). The first two phases of the standard lattice algorithms, *construction* and *adding*, remain unchanged. Modifications have to be done solely in the *removal phase*. Actually, as dominated nodes are not removed anymore, the *removal phase* is replaced by a *node classification phase*, cp. Algorithm 1.

The *classification phase* uses the same breadth-first and depth-first traversal as the original lattice Skyline algorithms. We need the node states *empty* and *non-empty*. In addition, we need to store a temporary value tmp_{ml} for the level of the $\mathcal{S}_{ml}$ set a node belongs to currently. When a node n is reached, we reset the tmp_{ml} values for the nodes $v_1, v_2, \ldots$, that are directly dominated by n. The value $\text{tmp}_{ml}(v_i)$ for a node v_i is computed as follows:

$$\text{tmp}_{ml}(v_i) = \begin{cases} \max\left(\text{tmp}_{ml}(v_i),\ \text{tmp}_{ml}(n)\right) & \Leftrightarrow \ n \text{ is empty} \\ \max\left(\text{tmp}_{ml}(v_i),\ \text{tmp}_{ml}(n)+1\right) & \Leftrightarrow \ n \text{ is not empty} \end{cases}$$

Algorithm 1 Multi-Level Skyline – Classification Phase

Global data structure: BTG
Output: list of $\mathcal{S}_{ml}$ sets

```
1:  function CLASSIFY
2:      S_ml ← list<list>() // initialize list to store S_ml sets
3:      tmp_ml[|BTG|] ← 0 // initialize tmp_ml array with 0's
4:      // iterate over all nodes n (BFT), start with node ID 0
5:      n ← node(ID = 0)
6:      repeat
7:          // use offset for tmp_ml computation
8:          offset ← n.isEmpty() ? 0 : 1
9:          // let domNodes be the list of direct dominated nodes
10:         domNodes ← getDirectDominatedNodesBy(n)
11:         for all v in domNodes do // compute tmp_ml
12:             tmp_ml(v) = max(tmp_ml(v), tmp_ml(n) + offset)
13:         end for
14:         // node not empty, add objects to S_ml sets
15:         if !n.isEmpty() then
16:             i ← tmp_ml[n]
17:             // add all elements in node n to the S^i_ml set
18:             S_ml.addAll(i, n.getElements())
19:         end if
20:         n ← nextNode() // next node in BFT
21:     until n == NIL // repeat until end of BTG is reached
22:     return S_ml
23: end function
```

In Algorithm 1, for a more convenient and efficient access to each of the $\mathcal{S}_{ml}$ sets after the classification phase, we generate a list of nodes belonging to each $\mathcal{S}_{ml}$ set while walking through the BTG. For this, we initialize a list of lists which will store the $\mathcal{S}^i_{ml}$ sets for each level i and an *array* of size of the BTG for the tmp_{ml} levels values (line 2–3). Then we start the BFT at node 0 (line 5). If the current node n is empty we set an offset to 0, otherwise to 1 (line 8). The function `getDirectDominatedNodesBy()` retrieves all nodes directly dominated by n (line 10, same functionality as used in the *removal phase* (DFT) of the lattice Skyline algorithms, cp. [11]). The complexity of this function is given

by the number of Skyline dimensions as for each of them not more than one node can be dominated and we only visit directly dominated nodes (so the DFT ends at depth 1). The actual complexity of finding each of the directly dominated nodes or a node's successor in the BFT is specific to the representation of the BTG in memory, but can be assumed as $\mathcal{O}(1)$ [10][11]. For all direct dominated nodes compute the tmp_{ml} value as the equation above (line 11–13). Afterward, if the node n contains elements from the input dataset, we retrieve for $\mathcal{S}^i_{ml}$ the level i the elements belongs to (line 16) and add all elements of the node n to the $\mathcal{S}^i_{ml}$ building up the multi-level Skyline sets (line 18). We continue with the next node in the BFT (line 20) until the end of the BTG is reached (line 21). The result is a list of $\mathcal{S}^i_{ml}$ sets.

Example 1. *Figure 3 shows an example of Algorithm 1.*

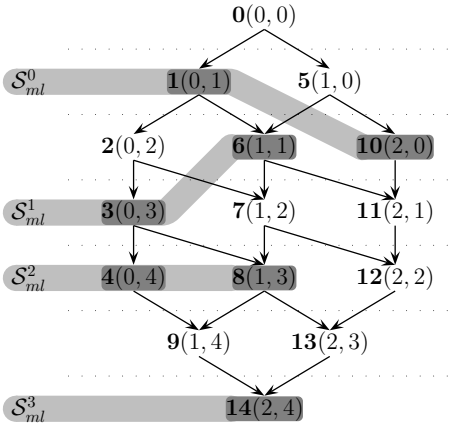


Figure 3. Multi-level Skylines $\mathcal{S}^l_{ml}$.

Since node 0 *is empty, the first relevant node is* 1*. Therefore we set* $\text{tmp}_{ml}[1] = 0$ *and add* 1 *to all direct dominated nodes, i.e.* $\text{tmp}_{ml}[2] = \text{tmp}_{ml}[6] = 1$*. We continue with node 5 which does not affect anything (the offset for the node is 0 and hence the* tmp_{ml} *values for the dominated nodes* 6 *and* 10 *remain unchanged). Since node* 2 *is empty, we set* $\text{tmp}_{ml}[3] = \text{tmp}_{ml}[7] = 1$*. Node 6 has already* $\text{tmp}_{ml}[6] = 1$*. The next node is 10, which still has* $\text{tmp}_{ml}[10] = 0$*. Node* 3 *sets* $\text{tmp}_{ml}[4] = \text{tmp}_{ml}[8] = 2$*, and so on. After the BFT has finished we have* 4 $\mathcal{S}_{ml}$ *sets.*

IV. Top-k Skyline Computation

The concept of *top-k* ranking is used to rank tuples according to some score function and to return a maximum of k objects [13]. On the other hand, Skyline retrieves tuples where all criteria are equally important concerning some user preference [1]. However, the number of Skyline answers may be smaller than required by the user, for whom k are needed. Therefore, *top-k Skyline* was defined as a unified language to integrate them [14][15].

A. Background

Top-k Skyline allows to get exactly the top k from a partially ordered set stratified into subsets of non-dominated tuples. The idea is to partition the set into subsets (strata, multi-level Skyline sets) consisting of non-dominated tuples and to produce the top-k of these partitions.

In general, existing solutions calculate the first stratum with some sort of post-processing [14][15][16]. That means, after identifying the first stratum $\mathcal{S}^0_{ml}(D)$, they remove the contained objects from the original input dataset D and continue Skyline computation on the reduced data. Hence, the second stratum is $\mathcal{S}^1_{ml} = \mathcal{S}(D \setminus \mathcal{S}(D))$. This workflow is continued until k objects are found. More formally:

Definition 3 (Top-k Skyline)**.** *A* top-k Skyline *query* $\mathcal{S}^k_{tk}(D)$ *on an input dataset* D *computes the top* k *elements with respect to the Skyline preferences. Formally:*

1) *If* $|\mathcal{S}(D)| > k$*, then return only* k *tuples from* $\mathcal{S}(D)$*, because not all elements can be returned due to result set size limitations. Any* k *tuples are a correct choice.*
2) *If* $|\mathcal{S}(D)| = k$*, then* $\mathcal{S}^k_{tk}(D) = \mathcal{S}(D)$*. That means return all tuples of* $\mathcal{S}^0_{ml}(D)$*. In this case there is no difference between the* Skyline *set and the top-k result set.*
3) *If* $|\mathcal{S}(D)| < k$*, then the elements of* $S(D)$ *are not enough for an adequate answer. We have to find a value* j *which meets the following criterion:*

$$\left|\bigcup_{i=0}^{j-1} \mathcal{S}^i_{ml}(D)\right| < k \leq \left|\bigcup_{i=0}^{j} \mathcal{S}^i_{ml}(D)\right| \tag{6}$$

That means, not only all elements of $\mathcal{S}(D) = \mathcal{S}^0_{ml}(D)$ *are returned, but also some of* $\mathcal{S}^1_{ml}(D)$*, and if the number of result tuples is still less than* k*, then* $\mathcal{S}^2_{ml}(D)$*, and so on. Note that from* $\mathcal{S}^j_{ml}(D)$ *exactly* $k - \left|\bigcup_{i=0}^{j-1} \mathcal{S}^i_{ml}(D)\right|$ *elements will be returned, which might not be all of it.*

B. The Top-k Lattice Skyline Algorithm (TkLS)

In this section, we adapt the concept of *multi-level Skyline* computation in Section III to the computation of *top-k Skylines*.

Algorithm 1 returns a set of all $\mathcal{S}^i_{ml}$ sets, hence the first k elements of these sets correspond to the *top-k* elements. However, in a top-k approach it is not necessary to compute all strata. It is enough to compute l multi-levels such that

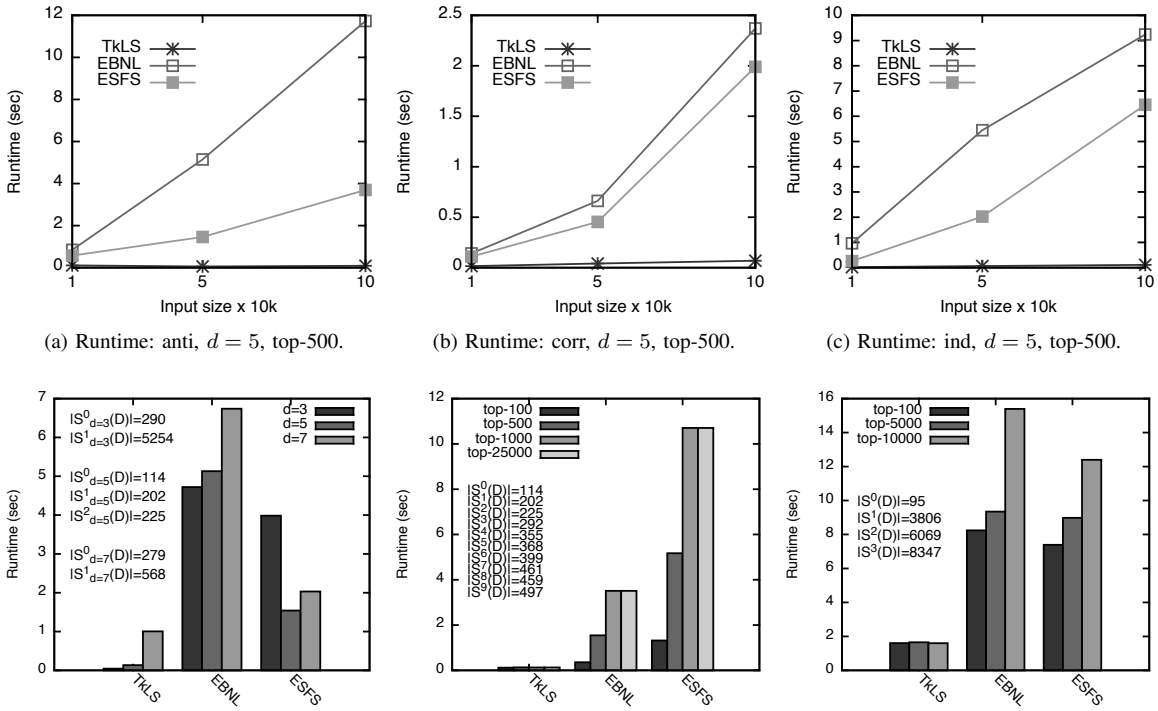
$$k \leq \left|\bigcup_{i=0}^{l} \mathcal{S}^i_{ml}(D)\right| \tag{7}$$

For this, we append a simple break condition after line 19 in Algorithm 1 which checks the above equation. Afterward, we can return the top-k elements.

One may criticize that picking the *top-k* results from the different equivalence classes (nodes of the BTG) is arbitrary in some manner, especially if a multi-level set $\mathcal{S}^j_{ml}$ only partially belongs to the top-k result set. In this case we have to pick some arbitrary elements out of this $\mathcal{S}^j_{ml}$ set to fill up the top-k elements. To handle this *"problem"* we can think about some kind of ordering or sorting before returning the top-k elements. Whichever additional conditions and characteristics are used, the top-k results can be taken then from the nodes coming first in the new order. We omit the discussion of the effects of different ordering strategies here as wrt. the original Skyline query, all candidates in $\mathcal{S}^j_{ml}$ are equally good results.

We have seen that in spite of being developed for Skyline queries, our *multi-level Skyline algorithm* can easily be applied to *top-k Sykline queries* as well. Its greatest advantage remains: the *linear* runtime complexity in the number of input tuples and size of the BTG.

 ISBN: 978-1-61208-408-4



(a) Runtime: anti, $d = 5$, top-500. (b) Runtime: corr, $d = 5$, top-500. (c) Runtime: ind, $d = 5$, top-500.

(d) Dimension: anti, $d = 3, 5, 7$, top-500. (e) Different k: anti, $d = 5$, $n = 50K$. (f) Zillow data. $d = 4$, $n = 2M$.

Figure 4. Experimental results.

V. EXPERIMENTS

This section provides our benchmarks on synthetic and real data to reveal the performance of the outlined algorithms. The concept of MLLS is the basis of TkLS, hence the performance of our top-k approach reflects the power of our multi-level Skyline algorithm. And since there is no competitor for MLLS, we only compared our approach *TkLS* to the state-of-the-art algorithms in generic top-k Skyline computation, *Extended Block-Nested-Loop* (EBNL) and *Extended Sort-Filter-Skyline* (ESFS) [15]. EBNL is a variant of the standard BNL algorithm [1] with the modification that each computed stratum is removed from the dataset and the Skyline is computed again. ESFS is an extension of SFS [5] exploiting some kind of data pre-sorting. In the worst-case EBNL and ESFS have a time complexity of $\mathcal{O}(n^2)$. The algorithms have been implemented in Java 7.0. TkLS follows the implementation details given in [9] and [11]. The experiments were performed on a machine running Debian Linux 7.1 equipped with an Intel Xeon 2.53 GHz processor.

For our synthetic datasets we used the data generator commonly used in Skyline research [1]. We generated *anti-correlated* (anti), *correlated* (corr), and *independent* (ind) distributions and varied (1) the data cardinality n, and (2) the data dimensionality d. For the experiments on real-world data, we used the well-known *Zillow* dataset from *www.zillow.com*. This dataset contains more than 2M entries about real estate in the United States. Each entry includes number of *bedrooms* and *bathrooms*, *living area* in sqm, and *age* of the building. The Zillow dataset also serves as a real-world application which requires finding the Skyline on data with a low-cardinality domain.

Figures 4a – 4c present the runtime of all algorithms on a 5 dimensional anti-correlated, correlated, and independent distributed dataset. We used a *top-500* query on different data cardinality. The low-cardinality domain was constructed by $[2; 3; 5; 10; 100]$. For all Skyline sets it holds that $|S(D)| < 500$ to get the effect of computing more than the 0-stratum.

Figure 4d shows the runtime of all algorithms on 3, 5, and 7 dimensions having anti-correlated data (up to $[2; 3; 5; 10; 10; 10; 100]$). The underlying data cardinality is $n = 50000$ and the target was to find the *top-500* elements. We also present the size of the different multi-level Skylines. For example, if $d = 3$ we have $|\mathcal{S}^0_{ml}(D)| = 290$ and the first stratum has $|\mathcal{S}^1_{ml}(D)| = 5254$ objects. This is also the reason why ESFS in this case is worse than for $d = 5$ or $d = 7$. ESFS has to compare all objects of the first stratum to all others, not yet dominated tuples.

Figure 4e visualizes the effect of different values of k. Therefore we computed top-k elements for $k \in \{100, 500, 1K, 25K\}$ using 5 dimensions (as in Figures 4a – 4c) and a data cardinality of $n = 50000$. The runtime for TkLS for all ks is very similar. This is due to the lattice based approach, where no tuple-to-tuple comparison is necessary, but only the construction of the BTG. Since the BTG for all ks is the same, the runtime for all top-k queries is quite similar. In the top-100 query only the Skyline $S(D)$ has to be computed. For $k = 500$ we have to compute stratum 0, 1, and 2. For top-1000 the first five strata are necessary, and for $k = 25K$ we need 41 strata to answer the query. We also see in this experiment that ESFS exploiting some pre-sorting is worse than EBNL. This is due the reordering of the elements.

Figure 4f presents our results on real data, i.e., the Zillow dataset (domain $[10; 10; 36; 46]$). Again, we compute the top-k elements for $k \in \{100, 5000, 10000\}$ to show the effect of computing different strata. TkLS clearly outperforms EBNL and ESFS. Again, since our algorithm is based on the lattice of a Skyline query the runtimes for the different ks are quite similar. EBNL and ESFS have to compute four strata to fulfill the $k = 10000$ query which results in a long runtime and hence bad performance.

VI. Related Work

The idea of *multi-level Skylines* was already mentioned by Chomicki [3] under the name of *iterated preferences*. However, Chomicki has never presented an algorithm for their computation. Apart from that there is no other work on computing the i-th stratum of a Skyline query.

Regarding top-k [13] and Skyline [2] queries, there are some approaches that combine these both paradigms to *top-k Skyline queries*. In [14] and [15] the authors calculate the first stratum of the *Skyline* with some sort of post-processing. Afterward, they define the k best objects or continue Skyline computation without the first stratum. The authors of [16] abstract Skyline ranking as a dynamic search over Skyline subspaces guided by user-specific preferences. In [17][18] and [19] an index based approach is used for top-k Skyline computation. However, index based algorithms in general cannot be used if there is a join or Cartesian product involved in the query. Su et al. [20] considers top-k combinatorial Skyline queries, and Zhang et al. [21] discuss a probabilistic top-k Skyline operator over uncertain data. Top-k queries are also of interest in the computation of spatial preferences [22], where the aim is to retrieve the k best objects in a spatial neighborhood of a feature object. Yu et al. [23] consider the problem of processing a large number of continuous top-k queries, each with its own preference. Although there is some related work, the problem of efficiently evaluating top-k Skylines is still an open issue.

VII. Conclusion and Future Work

We have found a promising algorithm for the iterated evaluation of a Skyline query. After a single run through a set of input tuples, we are able to return not only the Pareto frontier, but also the tuples that are directly dominated by them, and so on. Our approach supports multi-level and top-k Skyline computation without computing each stratum of the Skyline query individually. Although *multi-level Skyline sets* have been introduced some years ago, no algorithms specialized on this problem have been proposed. Furthermore, our TkLS algorithm also provides an efficient way to compute top-k Skylines. The only restriction we suffer from are low-cardinality domains. Since this is a huge restriction to Skyline computation, we want to adjust our algorithms as suggested in [24]. They use some kind of down-scaling of the domain. Furthermore, on the basis of [9] we want to develop a parallel algorithm for top-k Skyline computation. But this remains for future work.

References

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline Operator," in Proceedings of ICDE '01. Washington, DC, USA: IEEE, 2001, pp. 421–430.

[2] J. Chomicki, P. Ciaccia, and N. Meneghetti, "Skyline Queries, Front and Back," SIGMOD, vol. 42, no. 3, 2013, pp. 6–18.

[3] J. Chomicki, "Preference Formulas in Relational Queries," in TODS '03: ACM Transactions on Database Systems, vol. 28, no. 4. New York, NY, USA: ACM Press, 2003, pp. 427–466.

[4] P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and Analyses for Maximal Vector Computation," The VLDB Journal, vol. 16, no. 1, 2007, pp. 5–28.

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in Proceedings of ICDE '03, 2003, pp. 717–816.

[6] J. Selke, C. Lofi, and W.-T. Balke, "Highly Scalable Multiprocessing Algorithms for Preference-Based Database Retrieval," in Proceedings of DASFAA '10, ser. LNCS, vol. 5982. Springer, 2010, pp. 246–260.

[7] S. Park, T. Kim, J. Park, J. Kim, and H. Im, "Parallel Skyline Computation on Multicore Architectures," in Proceedings of ICDE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 760–771.

[8] S. Liknes, A. Vlachou, C. Doulkeridis, and K. Nørvåg, "APSkyline: Improved Skyline Computation for Multicore Architectures," in Proceedings of DASFAA '14, 2014, pp. 312–326.

[9] M. Endres and W. Kießling, "High Parallel Skyline Computation over Low-Cardinality Domains," in Proceedings of ADBIS '14. Springer, 2014, pp. 97–111.

[10] M. Morse, J. M. Patel, and H. V. Jagadish, "Efficient Skyline Computation over Low-Cardinality Domains," in Proceedings of VLDB '07, 2007, pp. 267–278.

[11] T. Preisinger and W. Kießling, "The Hexagon Algorithm for Evaluating Pareto Preference Queries," in Proceedings of MPref '07, 2007.

[12] T. Preisinger, W. Kießling, and M. Endres, "The BNL++ Algorithm for Evaluating Pareto Preference Queries," in Proceedings of MPref '06, pp. 114–121.

[13] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A Survey of Top-k Query Processing Techniques in Relational Database Systems," ACM Comput. Surv., vol. 40, no. 4, Oct. 2008, pp. 11:1–11:58.

[14] M. Goncalves and M.-E. Vidal, "Top-k Skyline: A Unified Approach," in OTM Workshops, 2005, pp. 790–799.

[15] C. Brando, M. Goncalves, and V. González, "Evaluating Top-k Skyline Queries over Relational Databases," in DEXA '07: Proceedings of the 18th International Conference on Database and Expert Systems Applications, ser. LNCS, vol. 4653. Springer, 2007, pp. 254–263.

[16] J. Lee, G. w. You, and S. w. Hwang, "Personalized Top-k Skyline Queries in High-Dimensional Space," Information Systems, vol. 34, no. 1, Mar. 2009, pp. 45–61.

[17] Y. Tao, X. Xiao, and J. Pei, "Efficient Skyline and Top-k Retrieval in Subspaces," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 8, 2007, pp. 1072–1088.

[18] M. Goncalves and M.-E. Vidal, "Reaching the Top of the Skyline: An Efficient Indexed Algorithm for Top-k Skyline Queries," in Database and Expert Systems Applications, ser. LNCS. Springer, 2009, vol. 5690, pp. 471–485.

[19] P. Pan, Y. Sun, Q. Li, Z. Chen, and J. Bian, "The Top-k Skyline Query in Pervasive Computing Environments," in Joint Conferences on Pervasive Computing (JCPC). IEEE, 2009, pp. 335–338.

[20] I.-F. Su, Y.-C. Chung, and C. Lee, "Top-k Combinatorial Skyline Queries," in Database Systems for Advanced Applications. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2010, pp. 79–93.

[21] Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei, "Ranking Uncertain Sky: The Probabilistic Top-k Skyline Operator," Information Systems, vol. 36, no. 5, Jul. 2011, pp. 898–915.

[22] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg, "Efficient Processing of Top-k Spatial Preference Queries," in Proceedings of the VLDB Endowment, vol. 4, no. 2, 2010, pp. 93–104.

[23] A. Yu, P. K. Agarwal, and J. Yang, "Processing a Large Number of Continuous Preference Top-k Queries," in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012, pp. 397–408.

[24] M. Endres, P. Roocks, and W. Kießling, "Scalagon: An Efficient Skyline Algorithm for all Seasons," in DASFAA '15: Proceedings of the 20th international conference on Database systems for advanced applications, 2015, in press.

 ISBN: 978-1-61208-408-4

O|R|P|E - A Data Semantics Driven Concurrency Control Mechanism

Tim Lessner*, Fritz Laux†, Thomas M Connolly‡
*freiheit.com technologies gmbh, Hamburg, Germany
Email: tim.lessner@freiheit.com
†Reutlingen University, Reutlingen,Germany
Email: fritz.laux@reutlingen-university.de
‡University of the West of Scotland, Paisley, UK
Email: thomas.connolly@uws.ac.uk

***Abstract*—This paper presents a concurrency control mechanism that does not follow a 'one concurrency control mechanism fits all needs' strategy. With the presented mechanism a transaction runs under several concurrency control mechanisms and the appropriate one is chosen based on the accessed data. For this purpose, the data is divided into four classes based on its access type and usage (semantics). Class O (the optimistic class) implements a first-committer-wins strategy, class R (the reconciliation class) implements a first-n-committers-win strategy, class P (the pessimistic class) implements a first-reader-wins strategy, and class E (the escrow class) implements a first-n-readers-win strategy. Accordingly, the model is called O|R|P|E. Under this model the TPC-C benchmark outperforms other CC mechanisms like optimistic Snapshot Isolation.**

Keywords–Multimodel concurrency control; transaction processing; optimistic concurrency control; snapshot isolation; performance analysis.

I. INTRODUCTION

The drawbacks of existing concurrency control (CC) mechanisms are that pessimistic concurrency control (PCC) is likely to block transactions and is prone to deadlocks, optimistic concurrency control (OCC) may experience a sudden decrease in the commit rate if contention increases. Snapshot Isolation (SI) better supports query processing since transactions generally operate on snapshots and also prevents read anomalies, but depending on the implementation of SI, either pessimistic or optimistic, it is also subject to the previously mentioned drawbacks of PCC or OCC. Semantics based CC (SCC) such as the mechanism proposed in [1] remedies some problems of PCC or OCC. It performs well under contention, reduces the blocking time, and better supports disconnected operations. However, its applicability is limited since data and transactions have to comply with specific properties such as the commutativity of operations. In addition to the previously mentioned drawbacks, neither PCC nor OCC nor SCC support long-lived and disconnected data processing. However, these properties are essential to achieve scalability.

This paper presents a mechanism originally introduced in [2] that combines OCC, PCC, and SCC and steps away from the **'one concurrency control mechanism fits all needs'** strategy. Instead, the CC mechanism is chosen depending on the data a transaction accesses. To address scalability, the mechanism was designed with a focus on long-lived and disconnected data processing.

Consider, for example, the whole sales scenario of the TPC-C [3]. With PCC using shared and exclusive locks, the likelihood of deadlocks increases for hotspot fields such as the stock's quantity or the account's debit or credit. If transactions are long-lived, PCC is even worse since deadlocks manifest during write time and a significant amount of work is likely to be lost [4], [2]. With OCC, deadlocks cannot occur. However, hot-spot fields like an account's debit or credit would experience many version validation failures under high load causing the restart of a transaction. Like PCC, validation failures manifest during the write-phase of a transaction and a significant amount of work is likely to be lost. Both PCC and OCC cannot ensure that modifications attempted during a transaction's read-phase will prevail during the write-phase. Wheras PCC is prone to deadlocks (in the case of shared locks), OCC is prone to its optimistic nature itself.

O|R|P|E resolves these drawbacks and data can be classified in CC classes. For example, customer data such as the address or password can be controlled by a PCC that uses exclusive locks only and performs lock pre-claiming [5]. Such a rigorous measure ensures ownership of data and should be used if data is modified that belongs to one transaction. For example, account data or master data should not be modified concurrently and given the importance of this data a rigorous isolation is justififed. The debit or credit of an account can be classified in CC class R, which guarantees no lost updates and no constraint violations. Such a guarantee is often sufficient for hot-spot fields. Class E can be used to access an item's stock, for example. Class E is able to handle use cases such as reservations. It should be used if during the read-phase a guarantee is required that changes will succeed during the write-phase. Class O is the default class. It avoids blocking and under normal load it represents a good trade-off between commit and abort-rate.

Section II defines these four CC classes with different data access strategies used by the mechanism. In case of a conflict, class O implements a **first-committer-wins** strategy, class R implements a **first-n-committers-win** strategy, class P implements a **first-reader-wins** strategy, and class E implements a **first-n-readers-win** strategy. The number n is determined by the semantics of the accessed data, e.g., by database constraints. According to the classes, the mechanism is called O|R|P|E. The "|" indicates the demarcation between data.

 ISBN: 978-1-61208-408-4

Section III proofs the correctnes of the model. Section IV briefly describes the prototype implementation. Section V highlights some advantages of O|R|P|E, because it provides an application flexibility in choosing the best suitable CC mechanism and thereby significantly increases the commit rate and outperforms optimistic SI. Finally, the paper summarizes related work (see Section VI) and provides an outlook (see VII).

II. MODEL

A. Transaction

To support long-lived and disconnected data processing, which both supports scalability, O|R|P|E models a transaction as a disconnected transaction τ, with separate read- and write-phase, i.e., no further read after the first write operation (see Definition II.1, taken from [2]). To disallow blind writes O|R|P|E guarantees that in addition to the value of a field, the version of a data field has to be read, too.

DEFINITION II.1: Disconnected Transaction:

1) Let ta be a flat transaction that is defined as a pair $ta = (OP, <)$ where OP is a finite set of steps of the form $r(x)$ or $w(x)$ and $<(\subseteq OP \times OP)$ is a partial order.
2) A disconnected transaction $\tau = (TA^R, TA^W)$ consists of two disjoint sets of transactions. $TA^R = \{ta_1^R, \ldots, ta_i^R\}$ to read and $TA^W = \{ta_1^W, \ldots, ta_j^W\}$ to write the proposed modifications back.
3) A transaction has to read any data item x before being allowed to modify x (no blind writes).
4) If a transaction only reads data it has to be labelled as read only.

B. CC Classes

Class O is the default class and is implemented by an optimistic SI mechanism, which is advantageous since reads do not block writes and non-repeatable or phantom phenomena do not happen. However, SI is not serializable [6], [7].

As stated, the drawback of optimistic mechanisms prevails if load increases, because many transactions may abort during their validation at commit time. An abort at commit time is expensive, because significant amount of work might be lost. A circumstance particularly crucial for long-lived transactions (see [2]).

Regarding the strategy, optimistic SI follows a "first-committer-wins" semantics revealing another drawback of O. It is the lack of an option allowing a transaction to explicitly run as an owner of some data. Consider, for example, the private data of a user such as its password or address. A validation failure should be prevented by all means, since it would mean that at least two transactions try to concurrently update private data. Although technically this is a reasonable state, for this kind of data a pessimistic approach that acquires all locks at read time is more appropriate. Such a mechanism follows a "first-reader-wins" (ownership) semantics and directly leads to class P. Lock acquisition at read time enables a strict sequential access and preclaiming (all reads and locks appear before the first write) prevents deadlocks during the write-phase if exclusive locks are used.

The decision if a data item is classified as O or P is based on the following properties [2]:

1) *Mostly read* (mr): Is the data item mostly read? If 'Yes', there is no need for restrictive measures and the data item should by classified for optimistic validation. A low conflict probability is assumed.
2) *Frequently written* (fw): fw is the opposite of mr.
3) *unknown* (un): It means neither mr nor fw apply, i.e., it is unknown whether an item is mostly read or written or approximately even.
4) *Ownership* (ow): if accessing a data item should explicitly cause the transaction to own this item for its lifetime?

EXAMPLE II.1: Classify data items in class O and P (taken from [2]).

This example is based on the TPC-C [3] benchmark and its "New-Order" transaction. Note that an additional table Account has been introduced to keep track about a customer's bookings (column debit and credit). It also defines an overdraft limit (column limit). The following tables are used in our example: Customer (id, name, surname), Stock (StockId, ItemId, quantity), Account (AcctNo, debit, credit, limit), and Item (ItemId, name, unit, price). Table I shows an initial classification.

Attributes *name*, *surname*, and *id* of a customer are expected to be mostly read, but if modified by a transaction it should definitively be the owner. The *id* of a customer, like all ids, is expected to become modified rarely. If the *id* becomes modified, ownership is required. In principal, all business keys should be classified in P, because they are owned by the application provider (see Rule II.1 (1)).

Stock.quantity is expected to become modified frequently (fw) and to prevent the situation where an item was marked as available during the read phase, but at commit time the item is no longer available due to concurrent transactions, it is also marked as ow. For the time being, however, *quantity* will be classified as an ambiguity (see also Rule II.1 (3)), which will be discussed below.

The *Account.credit* and *Account.debit* of a customer's account might be accessed frequently depending on a customer's activity and un is a good choice. However, since multiple transactions might concurrently update the balance, and an owner is hardly identifiable, $\neg ow$ is chosen. So, it is also an ambiguity (see Rule II.1 (3)).

The *Account.limit* is the overdraft limit of a customer and expected to be mostly read, hence, mr is a good choice. Since it is neither owned by the customer nor by others, $\neg ow$ is a good choice (see Rule II.1 (2)).

Assuming the application is a high frequency trading application, *Item.Price* might quickly become a bottleneck. An exact prediction is not possible though, hence, un is a good choice. Property ow would not be a good choice, because transactions of different components (dc) might simultaneously calculate the price (see Rule II.1 (3)).

The ambiguities A of Example II.1, see class A in Table I, highlight that classes O and P and their properties are not sufficient. Particularly, hot spot items such as *Stock.quantity* would benefit from a CC mechanism that allows many winners and resolves the drawbacks of OCC and PCC.

 ISBN: 978-1-61208-408-4

TABLE I. CLASSIFICATION OF EXAMPLE II.1

x	mr	fw	un	ow	CC class
Customer.name	1	0	0	1	P
Customer.surname	1	0	0	1	P
Customer.id	1	0	0	1	P
Stock.StockId	1	0	0	1	P
Stock.ItemId	1	0	0	1	P
Stock.quantity	0	1	0	1	**A**
Account.debit	0	0	1	0	**A**
Account.credit	0	0	1	0	**A**
Account.limit	0	0	1	0	**A**
Item.name	1	0	0	1	P
Item.unit	1	0	0	1	P
Item.price	0	0	1	0	**A**

Laux and Lessner [1] propose the usage of a mechanism that reconciles conflicts –class R–. Their approach is an optimistic variant of O'Neil's [8] Transactional Escrow Method (TEM). Both approaches exploit the commutativity of write operations. If operations commute, it is irrelevant which operation is applied first as long as the final state can be calculated (see [1], [2] for further details) and no constraint is violated.

Unlike TEM, the reconciliation mechanism requires a dependency function. Consider, for example, two transactions that update an account and both read an initial amount of 10€, one credits in 20€ and the other debits 10€. Once both have committed, it is relevant that no constraint was violated at any time and the final amount has to be 20€. Usually, a database would write the new state for each transaction causing a lost update. A dependency function would actually add or subtract the amount (the delta!) and would always take the latest state as input. In other words, reconciliation replays the operation in case of a conflict. However, this is only possible if no further user input is required. In the example above this means the user wants to credit 10€ (or debit 20 €) independent of the account's amount as long as no constraint is violated! Another requirement is that each dependency function has to be compensatable (see also [2]).

The reconciliation mechanism [1] follows a "first-n-committers-win" semantics and the number of winners n is solely determined by constraints. The correctness of the mechanism is proven in [1] which also introduces "Escrow Serializability", a notion for semantic correctness.

TEM grants guarantees to transactions during their read-phase. For example, a reservation system is able to grant guarantees to a transaction about the desired number of tickets as long as tickets are available. The consequence is that transactions need to know their desired update in advance (see [8] for further details).

Whereas TEM [8] is pessimistic (constraint validation during the read phase) and works for numerical data only, Reconciliation [1] is optimistic (constraint validation during the write phase) and works for any data as long as a dependency function is known. The proof that E, like R, is escrow serializable can be found in [2].

The decision if an item is member of R or E is based on the following properties:

1) con: Does a constraint exist for this data item?
2) num: Is the type of the data item numeric?
3) com: Are operations on this data item commutative?
4) dep: Is a dependency function known for an operation modifying the data item?
5) in: Is user input independence given for an operation modifying the data item?
6) gua: Is a guarantee needed that a proposed modification will succeed?

TABLE II. ILLUSTRATIVE CLASSIFICATION OF AMBIGUITIES OF EXAMPLE II.1.

x	con	com	num	dep	in	gua	CC class
Stock.quantity	1	1	1	1	0	1	E
Account.credit	1	1	1	1	1	0	R
Account.debit	1	1	1	1	1	0	R
Item.price	0	0	1	1	0	0	O

RULE II.1: Derivation of CC classes for data item x

1) $ow \rightarrow$ classify x in P (identify P).
2) $\neg ow \wedge mr \rightarrow$ classify x in O (identify O).
3) all other combinations of ow and mr: classify x in A (ambiguity).
4) $com \rightarrow$ classify x in $E \cup R$
   a) $(con \wedge num \wedge com \wedge gua) \rightarrow$ classify x in E (identify E).
   b) $(in \wedge dep \wedge com) \rightarrow$ classify $x \in R$ (identify R).
5) $x \in A \rightarrow$ item x will be eventually in O.

EXAMPLE II.2 (Classification of data items in R and E)**:** The ambiguities of Table I are the input for this example. Table II shows the result of the classification of these ambiguities.

`Stock.quantity` has a constraint $value > 0$ and is numeric. The dependency function dep is known too. As stated above, a dependency function performs a context dependent write. For example, dependency function d would be $d(x, xread, xnew) = x + (xnew - xread)$. User input independence in is not given. If placing the order fails at the end, a replay would also fail. So, class R is not an option. Since an order requires a guarantee that the requested amount of items remains available, Rule II.1 (4 a)) applies.

`Account.credit` and `Account.debit` are classified as R. Property dep is known, because operations are either additions or subtractions. Property in is given, because the account has to be updated if the order is placed and no constraint is violated. As the updates follow a dependency functions they can be reconciled and should not raise an exception. Again, only a constraint violation such as an overdraft can cause the abort. Rule II.1 (4 b)) applies.

`Item.price` depends on a variety of parameters including the last price itself. As a result, a price update might not be commutative. `Item.price` remains ambiguous and remains in O, because O is the default class. Rule II.1 (5) applies

III. CORRECTNESS

A transaction potentially runs under four different CC mechanisms. Due to the CC classes' individual semantics, each class has a different notion for a conflict, too.

Usually, a conflict is given if two operations access the same data item and the corresponding transaction overlap in their execution time, and at least one operation writes the data item [5]. Whereas for O this is a correct definition of a

 ISBN: 978-1-61208-408-4

conflict, for R and E it is not, because both can resolve certain write conflicts. The resolution of conflicts is a key aspect and advantage of SCC, and SCC questions the seriousness of a conflict. In other words the meaning of a read-write or write-write conflict is a interpreted. For R and E only a constraint violation is a conflict. Moreover, the state read by an operation is assumed to be irrelevant, otherwise commutativity is not given. It follows that any final serialization graph $SG-R$ and $SG-E$ for class R and E is non-cyclic because potential conflicts are reconciled (see [2] for a thorough discussion).

For P, the common definition of a conflict is correct, but the peculiarities of lock preclaiming during the read phase mean that if a transaction wants to modify item p (let $p \in P$), it has to acquire a lock on p during its read-phase to become the exclusive owner. If not, the transaction does a blind write, which is disallowed according to Definition II.1. Hence, every write in P cannot encounter a concurrent write or read, because if a transaction writes p it has to be the exclusive owner of P.

Lock preclaiming in P takes place at the begin of a read phase. Consider the following schedule, for example (let $disc_i$ denotes the disconnect phase of transaction i and let $o \in O$ and $p \in P$):

$$r_i(o), r_j(p), r_j(o), disc_j, w_j(p), c_j, r_i(p), disc_i, w_i(p), c_i$$

Transaction i precedes j in class O and j precedes i in P. Having contradicting orders, i.e., $i \rightarrow j$ in one, but $j \rightarrow i$ in another class violates serializability. If lock preclaiming is the first step carried out during a transaction's read phase, this unfavourable situation between classes O and P is avoided. Consider the following schedule, for example:

$$r_i(p), r_i(o), r_j(p), r_j(o), disc_i, w_i(p), c_i, disc_j, w_j(p), c_j$$

Now, transaction j has to wait until transaction i has committed, because j cannot be owner of p if i owns p.

Based on these findings it is possible to state Theorem III.1. The corresponding proof III.1 exploits that for R, P, and E the corresponding serialization graphs are non-cyclic.

THEOREM III.1: Let $SG-G$ be the global serialization graph, which is the union of $SG-O$, $SG-R$, $SG-P$, and $SG-E$. The global serialization graph $SG-G$ is non-cyclic if $SG-O$ is non-cyclic.

PROOF III.1 (By contradiction)**:** Given that ta_i is serialized before ta_j $(i \rightarrow j)$ in $SG-O$. In P, no other transaction can access an item in P if transaction ta_i has read the item. This includes ta_j and it is impossible to have a serialization order $j \rightarrow i$ in P. Since i and j can be arbitrarily changed there is a contradiction if $i \rightarrow j$ exists in one, and $j \rightarrow i$ in another class. $SG-R$ and $SG-E$ are negligible because any conflict is finally reconciled and both serialization graphs are non-cyclic.

COROLLARY III.1: $SG-O$ sets the global serialization order for P.

If a ta does not modify data in O, then P sets the order. If a ta does not modify data in P, then R sets the order, because it is prone to validation conflicts as opposed to E that already has a guarantee to succeed.

IV. PROTOTYPE REFERENCE IMPLEMENTATION OF O|R|P|E

The prototype of O|R|P|E is not a full database system. It was implemented using the JAVA programming language and Figure 1 illustrates its architecture. A client API provides access to the data and depending on the operation's type read or write, the operation is executed by a dedicated pool. Pools' "reads" and "writes" represent an read- and write-lane. In addition, a pool to handle the termination (commit and abort) has been implemented. Pools' reads and writes handle all incoming and outgoing operations and the classification has been placed directly into the index. Depending on an item's classification the corresponding CC mechanism is plugged in. Once an item has been read or written, the additional pools' "read-callback" and "write-callback" deliver the results back to the clients. Pool WFG (Wait-for-Graph) is used to handle access to the WFG. Deadlocks may occur during the read-phase of a transcation if the transcation accesses data items in class P. Deadlocks can occur in class P during the read-phase, because lock acquisition is not globally ordered.

Having separate pools to handle incoming and outgoing operations means that the prototype supports disconnected transactins, because the entire communication is asynchronous. There is no single thread that represents a specific client or transaction. Figure 2 illustrates the message flow within the prototype. A read operation is passed to the "reads" pool. Each read is executed asynchronously and the complete read set is sent back to the client via a dedicated callback pool. To support asynchronous writes, a write operation is passed to the "writes" pool and if all writes have been applied the write set is sent back to the client. Clients always sent their complete write-set.

Data is kept solely in memory and no data is written to disk unless the operating system needs to swap data to disk due to memory limitations. The only output to disk is to write logging events that are used for performance evaluation. Other functionality that has been implemented includes:

- CC mechanisms O, R, P and E as well as P;
- The prototype supports constraints.
- The prototype supports selects, range-selects, updates, and inserts. The deletion of an item is an update that invalidates a data item.
- A WFG implementation.

V. PERFORMANCE STUDY

The performance study has been carried out based on the prototype presented in the previous section (Section IV). As benchmark, the TPC-C++ benchmark [7] has been chosen, because we also conducted a study comparing O|R|P|E with Serializable SI, which is beyond the scope of this paper.

The performance study measures the response-time (resp. -time), the abort rate (ab-rate), the degree of concurrency (deg. conc.), and commits per second. The degree is the quotient of the serial estimated time over the elapsed time of the experiment. In addition, the arrival rate λ of new transactions has been varied to be set to the optimum (minimised abort rate and response time, maximised degree of concurrency). This optimum λ has been taken to conduct fair and calibrated comparisons. Each experiment has been repeated three times and the mean value is reported. Values refer to the execution

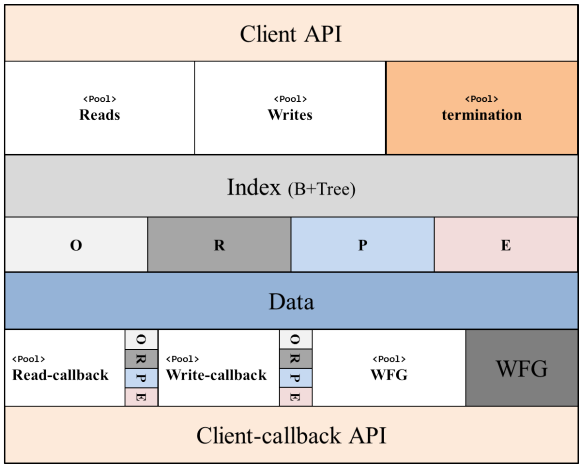


Figure 1. Architecture of the prototype.

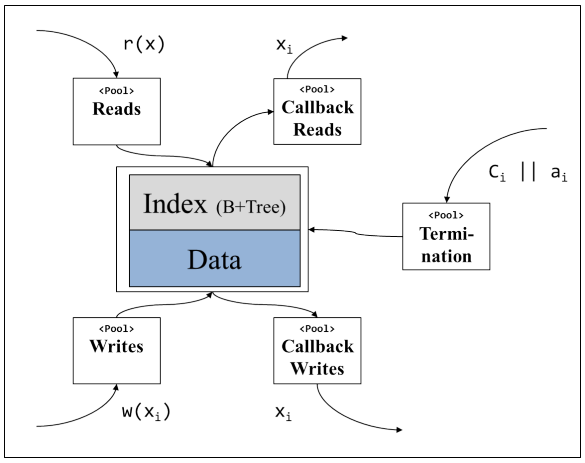


Figure 2. Message flow of the prototype.

of a transaction mix –deck– (42 New Order-, 42 Payment-, 4 Delivery-, 4 Credit check-, 4 Update Stock Level-, and 4 Read Stock Level - transactions see [7], [3], [2]). The classification of data shown in Table III is similar to the classification of Examples II.1 and II.2.

Figure 3 illustrates the abort rate and degree of concurrency for SI under full contention and shows the drawbacks of optimistic SI: the higher the number of concurrent transactions, the higher the abort rate. Also, the system starts thrashing if

TABLE III. TPC-C: CLASSIFICATION OF DATA ITEMS.

Item	CC Class	operation
Customer	P	read
CustomerCredit	P	update
CustomerBalance	R	read
Customer	P	read
CustomerBalance	R	update
Customer	P	read
CustomerCredit	P	read
StockQuantity	E	update
Customer	P	read
CustomerBalance	R	update
WarehouseYTD	R	update
DistrictYTD	R	update
StockQuantity	E	read only
StockQuantity	E	update

the degree of concurrency drops below one, which is the point where a serial execution outperforms a concurrent. Table IV #1-6 shows that the response-time increases with larger λ, which is expected and normal behaviour. A good degree of concurrency with a low abort rate is given by $\lambda = 133$ (see Table IV #3).

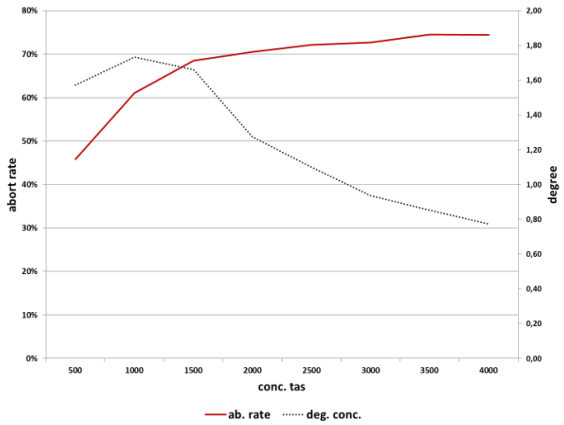


Figure 3. TPC-C++, optimistic SI (class O), abort rate and degree of concurrency.

Figure 4 shows the response-time and degree of concurrency for O|R|P|E for increasing λ. Unlike SI, O|R|P|E has no aborts caused by serialization or validation conflicts due to the classification of hot-spot data items in R or E, which prevents ww-conflicts. As shown by Figure 4, O|R|P|E has its best degree with $\lambda = 1000$ transactions per second achieving 227 commits per second (see Table IV, #15).

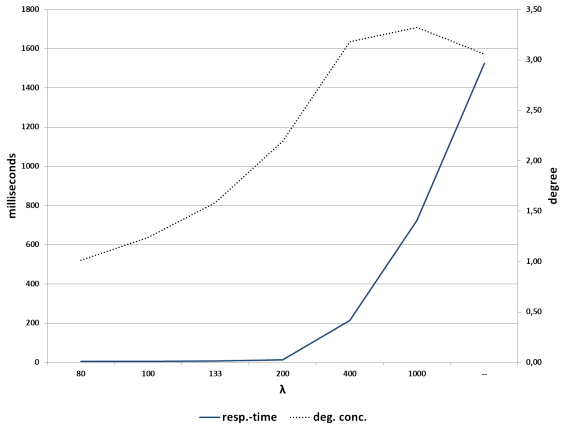


Figure 4. Response time and degree of concurrency for increasing λ for O|R|P|E .

The comparison of O|R|P|E and SI uses $\lambda = 133$ (Table IV #3, and #7-9) for SI and $\lambda = 1000$ (Table IV #15-18) for O|R|P|E . For SI, $\lambda = 133$ was considered as being the best trade-off with respect to the degree of concurrency, $\lambda = 1000$ was considered as being the best trade-off for O|R|P|E. Figure 5 illustrates the degree and the response-time for O with SI and O|R|P|E if both use the λ which reflect the best trade-off. As the figure shows, SI has a better response-time for 1000, 2000, and 3000 concurrent transactions, but then suddenly undergoes thrashing and the response-time grows exponentially. However,

 ISBN: 978-1-61208-408-4

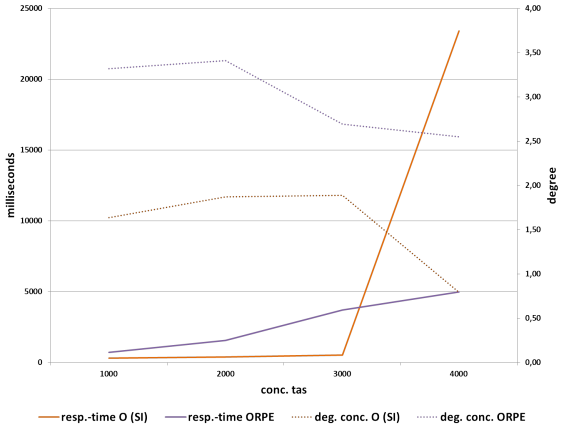


Figure 5. TPC-C++, SI and O|R|P|E : response-time and degree of concurrency for $\lambda = 133$ (SI) and $\lambda = 1000$ (O|R|P|E).

TABLE IV. MEASURED VALUES OF ALL EXPERIMENTS #1-18.

#	tas	λ	resp.-time	ab. rate	commits /second	deg. conc.
1	1000	80	43	2%	71	1,39
2	1000	100	84	3%	80	1,57
3	1000	133	309	5%	82	1,63
4	1000	200	1640	20%	62	1,50
5	1000	400	2091	26%	61	1,57
6	1000	1000	2464	27%	62	1,61
7	2000	133	388	9%	90	1,87
8	3000	133	522	8%	91	1,89
9	4000	133	23416	46%	22	0,79
10	1000	80	5	4%	69	1,01
11	1000	100	5	4%	85	1,24
12	1000	133	8	4%	108	1,58
13	1000	200	14	4%	150	2,19
14	1000	400	213	4%	217	3,18
15	1000	1000	724	4%	227	3,32
16	2000	1000	1551	4%	234	3,41
17	3000	1000	3704	4%	184	2,69
18	4000	1000	4968	5%	174	2,55

O|R|P|E shows a moderate and stable increase of the response-time even for 4000 concurrent transactions. It would be wrong to conclude that SI has a better performance than O|R|P|E for workloads below 4000 concurrent transactions, because λ has to be taken into account and for O|R|P|E $\lambda = 1000$ as opposed to $\lambda = 133$ for SI. Hence, under high contention O|R|P|E has the lowest abort rate and considering the trade-off, O|R|P|E has the shortest response-time. Furthermore, the abort rate is independent of the contention.

The drawback of O|R|P|E is that a wrong classification of data can quickly cause performance issues. For example, P as well as E are expensive, because P requires locking during the read-phase and E has to validate all constraints against each other. There is also overhead to classify the data. If a classification is not possible, class O is the default class –there is a fallback–.

VI. RELATED WORK

This paper is based on the findings of [2], which introduces O|R|P|E. A vast amount of work [5], [9] has been carried out in the field of transaction management and CC, but so far no attempt was undertaken to use a combination of mechanisms according to the data usage (semantics). Most authors use the semantics to divide a transaction into sub-transactions thus achieving a finer granularity that hopefully exhibit less conflicts. Some authors [10] use the semantics of the data to build a compatibility set while others try to reduce conflicts using multiversions [11], [12]. The reconciliation mechanism was introduced by [1] and is an optimistic variant of [8] "Transactional Escrow Method". Escrow relies on guaranties given to the transaction before the commit was executed, which is only possible for a certain class of transactions. OCC was introduced by [13], which did not gain much consideration in practice until SI, introduced by [14], has been implemented in an optimistic way. SI in general gained much attention through [6], [7], also in practice [15].

VII. OUTLOOK

The work carried out in [2] covers, in addition to the presented result, various aspects of O|R|P|E such as replication and runtime adaptation. However, a dynamic algorithm for an automatic classification of data would be advantageous. Also, a performance study that considers replication and runtime adaptation is still missing.

REFERENCES

[1] F. Laux and T. Lessner, "Transaction processing in mobile computing using semantic properties," in *Proceedings of the 2009 First International Conference on Advances in Databases, Knowledge, and Data Applications*, ser. DBKDA '09. IEEE Computer Society, 2009, pp. 87–94.

[2] T. Lessner, "-ORPE- a high performance semantic transaction model for disconnected systems," Ph.D. dissertation, University of the West of Scotland, 2014.

[3] *TPC BENCHMARK C, Standard Specification, Revision 5.11*, Transaction Processing Performance Council Std., February 2010.

[4] A. Thomasian, "Concurrency control: methods, performance, and analysis," *ACM Comput. Surv.*, vol. 30, no. 1, pp. 70–119, Mar. 1998.

[5] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[6] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making snapshot isolation serializable," *ACM Trans. Database Syst.*, vol. 30, no. 2, pp. 492–528, Jun. 2005.

[7] M. J. Cahill, U. Röhm, and A. D. Fekete, "Serializable isolation for snapshot databases," *ACM Trans. Database Syst.*, vol. 34, no. 4, pp. 20:1–20:42, Dec. 2009.

[8] P. E. O'Neil, "The escrow transactional method," *ACM Transactions On Database Systems*, vol. 11, pp. 405–430, December 1986.

[9] G. Weikum and G. Vossen, *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. Morgan Kaufmann, 2002.

[10] H. Garcia-Molina, "Using semantic knowledge for transaction processing in a distributed database," *ACM Trans. Database Syst.*, vol. 8, no. 2, pp. 186–213, Jun. 1983.

[11] S. H. Phatak and B. Nath, "Transaction-centric reconciliation in disconnected client-server databases," *Mob. Netw. Appl.*, vol. 9, no. 5, pp. 459–471, 2004.

[12] P. Graham and K. Barker, "Effective optimistic concurrency control in multiversion object bases," in *ISOOMS '94: Proceedings of the International Symposium on Object-Oriented Methodologies and Systems*, ser. Lecture Notes in Computer Science, E. Bertino and S. D. Urban, Eds., vol. 858. London, UK: Springer-Verlag, 1994, pp. 313–328.

[13] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Trans. Database Syst.*, vol. 6, no. 2, pp. 213–226, Jun. 1981.

[14] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A critique of ansi sql isolation levels," *SIGMOD Rec.*, vol. 24, no. 2, pp. 1–10, May 1995.

[15] D. R. K. Ports and K. Grittner, "Serializable snapshot isolation in postgresql," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1850–1861, Aug. 2012.

 ISBN: 978-1-61208-408-4

Towards Fuzzy Querying of NoSQL Document-oriented Databases

Fuzzy Mongo Query Language

Belhadj Kacem Abir
LR-SITI
ENIT, University of Tunis El Manar
Tunis, Tunisia
e-mail: belhadj.kacem.abir@gmail.com

Grissa Touzi Amel
LIPAH
FST, University of Tunis El Manar
Tunis, Tunisia
e-mail: amel.touzi@enit.rnu.tn

***Abstract*—When querying databases, users have become more demanding to express vague concepts. We speak then of flexible queries which have been the subject of several researches in the case of relational databases. Unfortunately, Not Only SQL (NoSQL) databases do not support this type of queries. In this paper, we focus on NoSQL oriented-document databases because they are the most appropriate for use in the context of a web applications where the role of fuzziness is crucial (e.g., social networks). We consider MongoDB oriented-document Database Management System (DBMS), which represents a leader in this market. Thus, we present an extension of the Mongo Query Language (MQL) to support fuzzy queries (fMQL) as well an extension of MongoDB architecture to support fMQL.**

Keywords-fuzzy queries NoSQL oriented-document database; MongoDB; MQL; fMQL.

I. INTRODUCTION

The end of the last century was marked by a significant evolution in information technologies. This evolution is characterized by an exponential growth of heterogeneous data volumes. Traditional relational databases have failed to scale with this huge data even after their physical distribution (distributed databases). To tackle these challenges, NOSQL databases [1] have emerged. Indeed, such databases offer: (1) flexible schemas, (2) high scalability, and (3) distribution of data.

We distinguish four categories of NoSQL databases, according to data storage: (1) key-value: As associative arrays in programming language, data is represented as a collection of key-value pairs. Among the NoSQL DBMS based on key /value storage, we can mention: REDIS [2], Berkeley DB [3], etc., (2) oriented column: A key point to a set of column, each having a value. As example of NoSQL DBMS based on column storage, we can mention: BigTable [4], Cassandra [5], etc., (3) oriented documents: This model is versioned documents that are collections of other key-value collections. The semi-structured documents are stored in formats like JavaScript Object Notation (JSON) or Binary JSON (BSON). Among the NoSQL DBMS oriented-document, we can mention: MongoDB [6], CouchDB [7], etc., and (4) graph-oriented: data are modeled as nodes connected with arcs. Neo4j [8] is a leader in this market.

NoSQL databases are dedicated for using internet applications. NoSQL oriented-documents databases are the most appropriate for web applications. Indeed, with this type of database and through AJAX [9], it is possible to exchange data in JSON format [10] documents directly (or with an intermediary which has a filtering and relaying role). That is why we focus on NoSQL oriented-document DBMS MongoDB. Mongo Query Language (MQL); as its name suggests is the query language of MongoDB. Figure 1 shows an MQL query which is displayed to return orders with a price lower than 500$ addressed to a NoSQL database describing a trading company.

Query

```
> db. Commande.find({ prix: { $lt: 500} })
```

Result (1 document)

```
>
{
ID_Cmd : <Cmd_120>
Date : 20/10/2014,
Prix: {
       total : 450
        },
Qnt :
                    {Qnt1 : 25
                    Qnt2 : 20
                    }
}
```

Figure 1. MQL query example.

However, this language does not support the expression of flexible query, for instance, to find orders whose prices are "low" or "well paid" customers where "low" and "well paid" are fuzzy predicates. This has been extensively studied in the case of relational databases. Indeed, many efforts have been made to create query specification mechanisms that allow users to express their preferences and manipulate words and phrases in query conditions. As

 ISBN: 978-1-61208-408-4

example of RDBMS supporting fuzziness, we mention: fSQL [11], fQuery [12], etc.

In [13], Castelltort and Laurent propose an approach for the implementation of flexible read queries over NoSQL graph DBMS Neo4j using a flexible language Cypherf which represents an extension of the Cypher language: the Neo4j query language. This solution is unfortunately restricted to only one type of NoSQL DBMS and has a high maintenance cost.

In this paper, we focus on fuzzy queries over the NoSQL oriented-document DBMS MongoDB by extending its query language MQL to support flexible queries where we express vague concepts. Thus, we speak of the fuzzy MQL (fMQL). In addition, we will propose a new architecture of MongoDB that supports fMQL.

The rest of the paper is organized as follows: Section 2 presents the basic concepts of MQL and flexible querying. Section 3 presents related researches and its limitations. Section 4 presents our perspective of extension of MQL language. Section 5 presents the new architecture of MongoDB to support fuzzy queries. We finish this paper with a conclusion and a presentation of some future works.

II. BASIC CONCEPTS

A. *MongoQuery Language (MQL)*

Select queries in MQL have the following syntax:

```
db.collection.find(<criteria>, <projection>)
```

TABLE I. THE COMMAND FIND() PARAMETERS

Parameter	Type	Description
Criteria	Optional Document	Specifies selection criteria using query operators. To return all documents in a collection, we should omit this parameter or pass an empty document ({}).
Projection	Optional Document	Specifies the fields to return using projection operators. To return all fields in the matching document, we should omit this parameter. The projection parameter have the following form: { field1: <boolean>, field2: <boolean> ... } The <boolean> value can be any of the following: • 1 or true to include the field. The find() method always includes the _id field even if the field is not explicitly stated to return in the projection parameter. • 0 or false to exclude the field.

The find() command allows selecting documents from a collection and returns a cursor on the selected documents that match the query criteria.

B. *Fuzzy queries*

A query is characterized as fuzzy when we can "express our preferences to order the more or less acceptable records found according to their adequacy to the query" [14].

To interpret and execute fuzzy queries, we had to extend the query languages, as well as the architecture of DBMS, to support such query languages. We speak then of Sqlf [11], fQuery [12], RankSql [15], etc.

In such systems, fuzziness in the queries is basically associated to fuzzy labels, fuzzy comparators (e.g., 'fuzzy greater than') and aggregation over clauses. Thresholds can be also defined for the expected fulfillment of fuzzy clauses. For instance, on a database describing employees, we can address such a typical fuzzy query: SELECT * FROM employee WHERE age IS 'young' AND salary IS 'well-paid' in order to find the young and well-paid employees where "well-paid" and "young" are fuzzy labels described by fuzzy sets (Figure 2 and Figure 3).

Figure 2. Fuzzy representation of « well-paid » predicate.

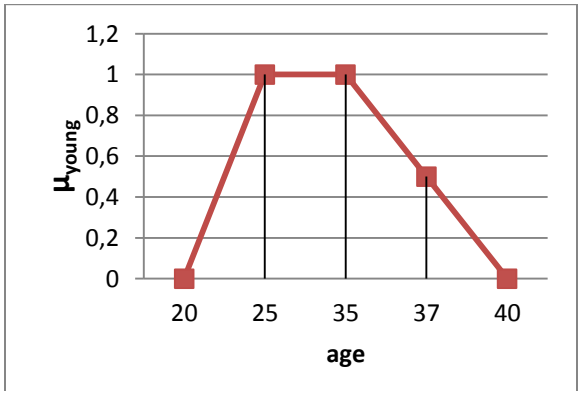


Figure 3. Fuzzy representation of « young » predicate.

Thresholds can be added for working withα−cuts, such as searching for employees where the degree young is greater than 0.7.

III. RELATED WORKS

A. Cypherf

Castelltort and Laurent [13] propose a perspective to extend the declarative way of querying the NOSQL graph DBMS Neo4j with the Cypher query language for dealing with vague queries. We speak then of Cypherf. Figure 4 shows a prototype under development, based on the extension of Cypher to support fuzziness.

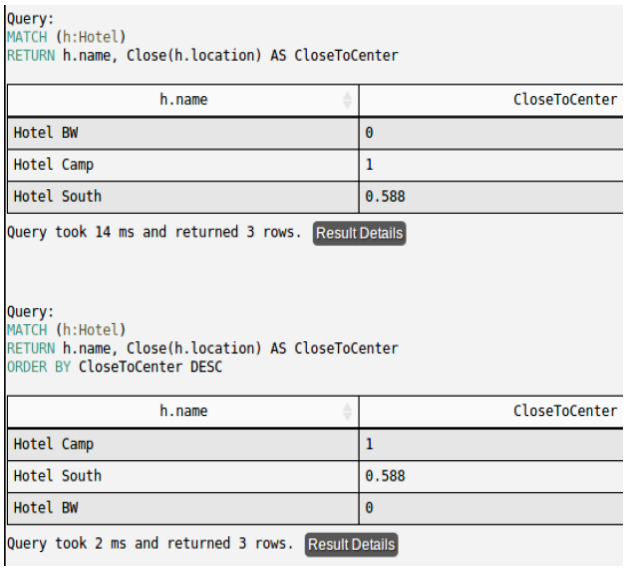


Figure 4. Prototype based on cypherf.

With Cypherf, fuzziness is handled over three levels: (1) properties, (2) nodes, and (3) relationships.

1) Cypherf over propeties

Dealing with fuzzy queries over properties is similar to the queries on relational databases. Such queries are defined by using linguistic labels and/or fuzzy comparators.

Such fuzzy queries impact the *START*, *MATCH*, *WHERE* and *RETURN* clauses from Cypher.

2) Cypherf over nodes

Dealing with fuzzy queries over nodes allows retrieving similar nodes. For instance, it is possible to retrieve similar employees.

3) Cypherf over relationships

Queries in this case are based on the graph structure in order to better exploit and benefit from it. In Cypher, the structure of the pattern being searched is defined in the MATCH clause. To deal with fuzziness over structure, fuzzy pattern matching considering chains and depth were defined.

B. Limits of existing solutions

Certainly, the solution proposed in [13] offers optimized performance, but it has a high cost of development and maintenance. This solution presents limitations:

- It is restricted to a single type of NoSQL bases, namely, graph-oriented. Indeed, Cypherf cannot query other NoSQL databases which are very popular and more used than Neo4j.
- There is not always a function that transcribes faithfully a preference expressed in natural language. More generally, a score function is inadequate when the desired preferences do not induce a total order on all objects.
- It does not consider the dependencies between the criteria for scheduling the result records knowing the order described in the query.

Apart from this solution, there is not an effective solution for flexible querying of other types of NOSQL databases. We propose a perspective to extend document NoSQL database MongoDB to support fuzziness in read queries.

IV. FUZZY MONGO QUERY LANGUAGE (FMQL)

We propose a description of the fuzzy query language fMQL, which is an extension of the MQL to support flexible queries. We are mainly interested in the extension of MQL to support linguistic labels and fuzzy comparators.

A. Linguistic labels in fMQL

If an attribute supports fuzzy processing, linguistic labels can then be defined. These labels will be preceded by the # symbol to distinguish them easily.

These labels will be replaced by the corresponding intervals that the minimum and maximum values are stored in a meta-knowledge database.

For instance, on a NoSQL document database describing employees, we can address such a typical fuzzy query:

```
db.employees.find(
{ Age: #young ,
Salary: #well-paid }
{Employee_id: 1,
Name: 1,
Surname: 1,
Age: 1,
Salary: 1,
Adress: 1,
 _id: 0}
)
```

This query allows to find the young and well-paid employees where “well-paid” and “young” are fuzzy labels described by fuzzy sets (Figure 2 and Figure 3).

TABLE II. FMQL QUERIES TRANSLATED IN MQL

Query	fMQL Query	MQL Query
List of young employees	`db.employees.find( { age:#young} )`	`db.employees.find( { age: { $gt: 20, $lte: 40 } } )`
List of well-paid employees	`db.employees.find( {salary:#well-paid } )`	`db.employees.find( {salary: { $gt: 750, $lte: 2000} } )`
List of young and well-paid employees	`db.employees.find( { age:#young, salary:#well-paid } )`	`db.employees.find( { age: { $gt: 20, $lte: 40 }, salary: { $gt: 750, $lte: 2000} } )`

In Table II, we have presented some fMQL queries translated into MQL.

B. Fuzzy comparators

In addition to typical comparators ($gt, $lt, $gte, $lte, etc.), fMQL includes fuzzy comparators. fMQL fuzzy comparators for fMQL are defined by the user.

In the case of the digital attributes, a fuzzy logic comparator can be defined by means of the distance measurement. Distance measurements allowed are difference and the quotient. The satisfaction degree of comparison is given by the membership of this distance to a user given fuzzy set. In case of scalar attributes, it is also possible to define fuzzy comparators by listing the related pairs with their corresponding satisfaction degrees. Comparison is always established between regular (crisp) data values.

As in MQL, fuzzy comparators can compare a column with a constant or two columns having the same type.

We define for fMQL 18 integrated fuzzy comparators (Table 2). Six of them are defined as possibility measures ($feq, $fgt, $fgte, $flt, $flte, $fdif). Two are purely fuzzy much greater then ($mgt) and much less then ($mlt). We define eight other comparators that have been conceived as the necessity measures counterpart of preceding possibility comparators.

TABLE III. TABLE TYPE STYLES

Possibility comparators	Necessity comparators	Description
$Fgt	$FNgt	Fuzzy greater than (necessity/possibility)
$Flt	$FNlt	Fuzzy less than (necessity/possibility)
$Fgte	$FNgte	Fuzzy greater or equal than(necessity/possibility)
$Flte	$FNlte	Fuzzy less or equal than(necessity/possibility)
$Feq	$FNeq	Fuzzy equal (necessity/possibility)
$Fdif	$FNdif	Différent (necessity/possibility) flou
$mlt	$Nmlt	Beaucoup plus grand (necessity/possibility)
$mgt	$Nmgt	Beaucoup plus petit (necessity/possibility)

V. EXTENSION OF MONGODB ARCHITECTURE

A. Architecture

1) Architecture type

The implementation of a fuzzy query system can be tackled in two types of architecture [16]: the low coupling and the high coupling. We have chosen the low coupling where new features are integrated through a software layer above the DBMS because this solution is a cheap and non-intrusive.

The concept is to create a high-level fuzzy language (fMQL) that will be used to generate MQL well-formed queries. The generated MQL queries will be executed by the existing MongoDB engine. Thus, MQL is used as a low level language to achieve fuzzy queries.

2) Bosc Architecture for flexible querying modeling

To extend the architecture of Mongdb to support fuzzy queries, we were inspired by the Bosc architecture [17] proposed for relational databases.

The approach proposed by Bosc (Figure 5) consists of using the capacities of the commercial DBMS (in particular their mechanisms of optimization) while adding a supplementary layer assuring the interface between flexible queries and Boolean queries [17][18].

 ISBN: 978-1-61208-408-4

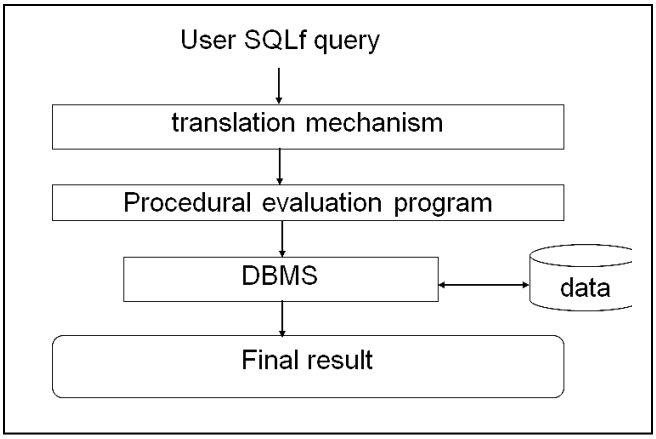


Figure 5. Bosc Architecture for flexible querying modeling.

As shown in Figure 5, the fuzzy query process is done by a transformation procedure located on top of the existing DBMS. The translation mechanism generates a SQL query addressed to the DBMS.

B. Architecture implementation

We propose an extension of the DBMS MongoDB architecture to support flexible data retrieval (fuzzy queries) following the architecture of Bosc. In this architecture, the layer fMQL_TO_MQL functions as an interface between a fuzzy query modeled in fMQL and its corresponding query modeled in MQL, as presented in Figure 6. This layer interacts with a Fuzzy Meta-Knowledges Base (FMB) which extends the DBMS dictionary in order to store all necessary information to describe fuzzy attributes and satisfaction degrees of fuzzy comparators.

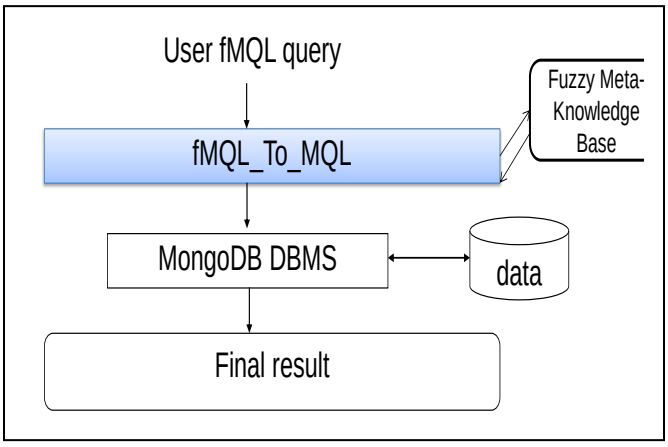


Figure 6. New MongoDB architecture to support fuziness.

The fMQL_TO_MQL layer is a tool that allows the automatically transformation of the fMQL query to its equivalent MQL query, while specifying the modifications that should be made at the level of the FMB. Its main idea is to replace the linguistic labels and fuzzy comparators of fuzzy queries with the Boolean expressions compatible with the MongoDB engine system. This task is done thanks to the procedure Translation_fmql_in_mql() shown in Figure7.

This procedure extracts the fMQL query criteria and cut the extracted part in several lines each containing one criterion then proceeds as follows:

- To replace linguistic labels, the procedure Translation_fmql_in_mql() should, first of all, extract the label proceeded by '#' from the criteria. This label is, then, replaced by the bounds of the corresponding interval, imported from the FMB (eg., The criteria'**age**:#young' is replaced by 'age: { $gt: 20, $lte: 40 }'. The values 20 and 40 correpond to the bounds of the interval corresponding to the label 'young'). Thus, we proceed as follows :

0) Begin
1) Find out the position of # character with whom begin linguistic label
2) Extract this label
3) Connect to FMB and import the corresponding interval of this label
4) Replace the label with the interval in the criterion
5) End

- To replace fuzzy comparators, the procedure Translation_fmql_in_mql() should, first of all, find out the position of the fuzzy comparator proceeded by '$F' from the criteria. Then, if it's a necessity comparator, it replaces the compared value V by the appropriate bounds interval depending on the satisfaction degree μ<1 defined by the user (eg., The criteria'**age**: {$Feq 20}' is replaced by 'age: { $gt: 18, $lte: 22 }' μ=0.1). Thus, we proceed as follows:

0) Begin
1) Find out the position of $F
2) If $F is followed by N
   Replace the compared value V with the interval [V-μ*V, V+μ*V] (where μ is the satisfaction degree specified by the user)
   Else
   Execute Two queries R1 and R2: (1) R1 drops the criterion containing $F and (2) R2 where we replace the compared value V with the interval [V-μ*V, V+μ*V] in the criterion containing $F (where μ is the satisfaction degree specified by the user)
3) Display the result
4) End

 ISBN: 978-1-61208-408-4

```
Traduction_fMQL_to_MQL(Var  Query :String)
Begin
Criteria = extract_critera (Query)
While (Criteria <>End_String) faire
Criterion=Extract(Criteria,'{', '}')
        If (Criterion contain '#') Then
        Label=Remove(Criterion,'#', ' ')
        Res= ConnexionFMB(Label)
        Interval_inf= Res[inf]
        Interval_sup= Res[sup]
        Label=Concat ('{$gte', Interva_inf,',$lte'
        Interva_sup,'}')
        Criterion=Insert(Criterion,'#', Label)
        Else
            If (Criterion contient '$FN') Then
            Value=Remove  (Criterion,'$FN', ' ')
            inf= Val(Value)+μ
            sup= Val(Value)-μ
            Value= Concat ('{$gte', inf,',$lte' sup,'}')
            Criterion =Insert(Criterion, $FN', Value)
            Else
            Remove (Criterion,'$F ', ',')
            EndIF
        EndIF
Loop
End.
```

Figure 7. Procedure Translation_fMQL_In_MQL.

The procedure Translation_fmql_in_mql() summarized in Figure 7 implements the algorithms of replacing fuzzy comparators and linguistic labels.

VI. CONCLUSION AND FUTURE WORK

Several applications need to manage fuzzy information and to make benefit their users from flexible queries. This need have been intensively studied in the case of relational databases but there is not an efficient solution to benefit from flexible queries in case of NoSQL databases.

In this paper, we have been interested to NoSQL oriented-document DBMS MongoDB because it is the most adapted NoSQL DBMS to web application where dealing with fuzziness is crucial.

Indeed, we have proposed to extend the MQL to fMQL that describes, in addition to the boolean MQL query, fuzzy queries whose predicates contain vague concepts such as linguistic labels and fuzzy comparators. Furthermore, we have proposed an extension of the MongoDB architecture to support fMQL by integrating a software layer on MongoDB translating fMQL queries to corresponding MQL queries.

As future perspectives, we plan to (1) extend the fuzzy concept to description manipulating language dealing with data stored in a NoSQL oriented document database, and (2) implement a finalized solution that allows flexible read and write querying.

REFERENCES

[1] R. Cattell, "Scalable SQL and NoSQL data stores," ACM SIGMOD Record, vol. 39, no. 4, 2011, pp. 12-27.

[2] J. L. Carlson, Redis in Action, vol. 79, Connecticut: Manning Publications Co., 2013.

[3] M. A. Olson, K. Bostic, and M. I. Seltzer, "Berkeley DB," USENIX Annual Technical Conference, FREENIX Track, 1999, pp. 183-191.

[4] F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," ACM Trans. Comput. Syst., vol. 26, no. 2, 2008, p. 205–218.

[5] E. Hewitt, Cassandra - The Definitive Guide: Distributed Data at Web Scale, California: Springer, 2011.

[6] K. Chodorow and M. Dirolf, MongoDB - The Definitive Guide: Powerful and Scalable Data Storage, O'Reilly, 2010.

[7] J. C. Anderson, J. Lehnardt, and N. Slater, CouchDB: the definitive guide, California: " O'Reilly Media, Inc.", 2010.

[8] J. J. Miller, "Graph Database Applications and Concepts with Neo4j," Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2013, pp. 131-142.

[9] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)," July 2006. [Online]. Available: https://tools.ietf.org/html/rfc4627. [Accessed January 2015].

[10] J. J. Garrett, "Ajax: A New Approach to Web Applications," 23 March 2007. [Online]. Available: http://www.adaptivepath.com/publications/essays/archives/000385print.php. [Accessed January 2015].

[11] P. Bosc and O. Pivert, "SQLf: a relational database language for fuzzy querying.," IEEE T. Fuzzy Systems, vol. 3, no. 1, 1995, pp. 1-17.

[12] J. Kacprzyk and S. Zadrozny, "Computing with words in intelligent database querying: standalone and Internet-based applications," Information Sciences, vol. 134, no. 1-4, 2001, pp. 71-109.

[13] A. Castelltort and A. Laurent, "Fuzzy Queries over NoSQL Graph Databases: Perspectives for Extending the Cypher Language," Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2014, pp. 384-395.

[14] P. Bosc, A. Motro, and G. Pasi, "Report on The fourth International Conference on Flexible Query Answering systems," ACM SIGMOD Record, vol. 30, no. 1, 2001, pp. 66-69.

[15] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song, "RankSQL: query algebra and optimization for relational top-k queries," Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005, pp. 131-142.

[16] A. Urrutia, L. Tineo, and C. Gonzalez, "FSQL and SQLf: Towards a standard in fuzzy databases," Handbook of Research on Fuzzy Information Processing in Databases, vol. 1, no. 1, 2008, pp. 270-298.

[17] P. Bosc and O. Pivert, "SQLf query functionality on top of a regular relational database management system," in Knowledge Management in Fuzzy Databases, Heidelberg, Springer, 2000, pp. 171-190.

[18] P. Bosc, L. Liétard, and O. Pivert, "Bases de données et flexibilité: les requêtes graduelles," TSI. Technique et science informatiques, vol. 17, no. 3, 1998, pp. 355-378.

 ISBN: 978-1-61208-408-4

Combining Distributed Computing and Massively Parallel Devices to Accelerate Stream Data Processing

David Bednárek, Martin Kruliš, Petr Malý, Jakub Yaghob, Filip Zavoral, and Jaroslav Pokorný
Faculty of Mathematics and Physics
Charles University in Prague, Czech Republic
email: {bednarek,krulis,malyp,yaghob,zavoral,pokorny}@ksi.mff.cuni.cz

***Abstract*—Data streaming systems have been successfully employed for various data processing tasks. Their main benefit is that they simplify the design of data-intensive applications and they introduce many opportunities for task, data, and pipeline parallelism. In this work, we are proposing an enhancement for data streaming systems that allows distributed processing of the data streams and also introduce parallel accelerators, which can be utilized for data parallel subtasks. The viability of our approach is verified by integrating the support for heterogeneous accelerators into the Bobox system, which is a parallel framework for data stream processing.**

Keywords–parallel processing; stream computing; distributed computing; parallel accelerators

I. INTRODUCTION

A significant part of the scientific community and high performance computing (HPC) community has become fascinated by the growing potential of GPU (graphics processing unit) accelerators in a number crunching applications. However, the GPUs excel only in specific tasks while in others their performance is inferior. In particular, tasks involving complex data structures struggle with the complexity of the GPU memory hierarchy. The utilization of GPUs is also limited by the necessity of transferring the data between the host system and the internal GPU memory, which brings additional overhead to the data processing. Consequently, complex data manipulation (which often occurs in database systems) is in most cases performed faster by CPUs than by GPUs and the fact that more efficient GPU algorithms exist for many subproblems is outweighed by the additional costs imposed by the GPU architectures.

At the same time, the growing size of data sets means that even applications that focus solely on numerical computations require sophisticated methods for data manipulation, which were previously known only in database systems. Although this does not necessarily mean that all applications must use a relational database or a database management system at all, an application must directly or indirectly use elaborate data structures to store its data on external media and to explicitly cache working sets of this data in operating memory. In addition, a scalable application must be able to balance the workload over a number of computing nodes, which results in intensive communication between the nodes conducted either directly or indirectly through a distributed file system or similar abstraction layer.

As a consequence of the growing potential of the GPUs and the expanding data sets, almost every computationally intensive application combines parts, which are best suited for GPUs, as well as parts that must be done by CPUs. In some cases, the performance of GPUs and CPUs on a particular subtask may be approximately equal; consequently, such a subtask may be subject of load balancing between both platforms.

These considerations lead to increased interest in heterogeneous computing systems, which combine both large number of CPUs (or at least a CPU with multiple cores), as well as one or more GPUs. In addition, many-core platforms like Intel Xeon Phi have emerged recently. They represent another step in CPU-GPU convergence trend which combines the generality of CPUs and the massive parallelism of GPUs. While the heterogeneous systems are already available as hardware, the software that is responsible for efficient hardware utilization is still immature. It is quite improbable that one solution will fit for every application, thus a number of thoroughly different approaches is being attempted.

In this work, we investigate the applicability of the stream data processing paradigm in heterogeneous computing platforms. The semantics of a stream naturally fits to the data flow between individual heterogeneous computing nodes, meaning both the communication between servers in a distributed system and the data movement between host memory and accelerators such as GPUs or Xeon Phi devices. Since the scheduler of such a system must be aware of all communication and synchronization inside the application, it requires that every communication between individual subtasks is explicitly expressed using streams. Although this restriction requires a different approach to application decomposition than traditional procedural programming, there are numerous examples of problems that were successfully modeled in the stream paradigm. In particular, almost all database systems convert every query to an execution plan, consisting of operators chained to a tree or directed acyclic graph, which almost exactly fits to the decomposition imposed by the stream systems. Given the necessity of database-like processing of large data sets, computational applications may often employ presented approach (or a very similar one) directly.

The paper is organized as follows. Section II summarizes related work and Section III revises most important facts about parallel hardware. The architecture of the distributed Bobox and necessary modifications of the existing implementation are

 ISBN: 978-1-61208-408-4

presented in Section IV. The integration process of dependent accelerators is proposed in Section V and Section VI concludes the paper.

II. Related Work

A. Bobox

The Bobox framework [1] was designed to support development of data-intensive parallel computations. The main idea behind Bobox is to divide large tasks into many simple tasks that can be arranged into a nonlinear pipeline while preserving transparency of the distribution logic. The tasks are executed in parallel and the execution is driven by the availability of data on their inputs. The developers do not need to be concerned about technical issues such as synchronization, scheduling, or race conditions [2].

The system can be easily used as a database execution engine; each query language requires its own frontend that translates a request (query) into a definition of the structure of the pipeline that corresponds to the query [3]. Bobox uses its own language called Bobolang [4] in which the execution plan in described. The bobolang code can be generated by database frontends, but it can also be created directly by the developer of a parallel application.

B. Streaming Systems

Stream processing has no formal definition and this term is used to describe a variety of systems [5]. One of the founding systems in this domain is StreamIt [6], which uses streaming in combination with general parallel data processing. It provides a domain specific language that follows the Synchronous Data Flow paradigm [7] and it is able to utilize various hardware architectures such as distributed multi-core systems.

Data stream management systems (DSMS) form a particular category of streaming systems. Systems as Borealis [8], STREAM [9], or NiagarsST [10] are used for real-time querying of data streams, which are usually continuous and infinite. These systems typically provide a high-level runtime API accompanied by either a set of predefined operators or by a specialized query language.

The class of hardware-specific streaming languages and systems is represented by StreamC/KernelC [11], which is designed for the Imagine processor [12] or BrookGPU [13] designed for GPUs.

The Auto-Pipe [14] is intended to simplify development of complex streaming applications on various architectures. It is based on specification of operators, resources, and a topology of these resources. The system provides an optimal mapping between the operators and the resources.

Intel Threading Building Blocks (TBB) [15] is a framework for parallel computing. It cannot be strictly regarded as a streaming system; however, it provides similar functionality through its Flow Graph component [16]. In addition, it provides basic algorithm templates, so it can be employed for parallel data processing in more general way. Another widely used framework is OpenMP [17] with an extension for data stream processing [18] that enables description of the pipeline stages of an algorithm. However, more complex constructions are tricky and rather difficult to implement.

The MapReduce programming model [19] is another example of a specific data streaming system with fixed number of stages. It is mainly intended for processing of large data in a distributed environment [20]. Despite the fact that it is considered a step back [21] in parallel database engines, this model gained significant attention since it can be easily utilized for large tasks and it provides sufficient scalability and robustness. Unfortunately, it does not support more complex execution plans like non-linear pipelines. This drawback is partially solved by the Apache Pig [22] environment together with the Pig-Latin language [23], which is able to transform non-linear pipeline into a sequence of MapReduce programs.

C. Accelerated Database Engines

Many-core parallel accelerators (especially GPUs) have proven useful for accelerating individual database operations. For instance, GPUTeraSort [24] is an algorithm developed to sort database rows, and demonstrated significant performance improvements over serial sorting methods. Similarly, the relational join operation has been successfully implemented for GPUs [25].

Using external procedures in big commercial databases, such as Oracle [26] or PostgreSQL [27], GPU hardware can be employed to accelerate certain critical operations. In [28], the authors propose a design of a GDB database accessed through a rich set of individual parallelizable operations.

III. Computing on Parallel and Distributed Architectures

In this section, we revise important facts related to the parallel computations and parallel architectures which became popular in the past few decades. We will mention mainstream architectures, which are employed widely in various applications.

A. CPU and Multi-CPU Architectures

A multicore CPU is perhaps the most typical representative of current parallel hardware. It comprises several physical cores, which are further divided into logical cores by means of hyper threading (Intel) or dual-core module (AMD) technology. Logical cores share most of the units of the physical core in order to improve utilization of the physical core. Most of the current CPUs employ three-level memory cache, which reduces the latency of operating memory. In the parallel processing, we must consider the fact that some levels of the cache are shared among cores. Therefore, the overall cache capacity available solely for individual cores is reduced; however, the shared cache can improve situations where the cores read the same memory or need to exchange small amounts of data among themselves.

Multiprocessor systems are composed of several multicore CPUs. The systems which use an independent memory controller are usually organized as *symmetric multiprocessing* systems. If each CPU has a memory controller integrated, the system has to be organized as *nonuniform memory architecture* (NUMA) and each CPU manages its part of the (shared) operating memory. In modern NUMA systems, the hardware is able to maintain cache coherency (ccNUMA); thus, the memory access is transparent. Nevertheless, the latency and bandwidth are not uniform across the memory range, since any access to memory managed by different CPU has to be routed through interprocessor communication grid.

 ISBN: 978-1-61208-408-4

B. Parallel Accelerators

Parallel accelerators, especially GPUs, form an important group of architectures used for parallel computing. An accelerator is usually an extension card connected to the host system which controls its utilization. Most accelerators are capable of performing computations concurrently to the host CPU; however, the host system is still responsible for providing (or at least passing on) the input data. In this work, we are particularly interested in GPUs and Intel Xeon Phi devices, which are typical representatives of parallel accelerators of the day.

GPU devices have become quite popular platform for data parallel processing. A GPU processor consists of several *symmetric multiprocessing units* (SMPs). Each SMP comprises multiple GPU cores, single instruction decoder and scheduler, L1 cache and shared memory. The cores of the SMP are tightly coupled since they share its internal resources and they all execute the same code in a SIMT (Single Instruction, Multiple Threads) fashion, i.e., in a way that all cores perform the same instructions. This model makes GPUs perfect devices for *data parallelism*, where the same routine is executed many times on different data objects.

The memory hierarchy of GPUs is significantly different from current CPUs. A GPU device has its own *global memory*, each SMP has *shared memory* which is much faster but accessible only by the cores of the SMP, and finally, each core has its own registers (assigned from the SMP registry pool). This hierarchy is particularly important for fine-tuning and optimizations of the GPU code. Furthermore, the GPU processor cannot access data in the *host memory* directly, but the data must be transferred to the global memory first. These transfers have to be carefully planned, so they will not present a bottleneck of the data processing. Fortunately, modern GPUs are capable of performing two concurrent data transfers (in and out, since PCIe bus is duplex) while the GPU processor is computing.

Intel Xeon Phi is a many-core device usually delivered in the form of accelerator card. It resembles GPUs in the main aspects, since the accelerator has its own memory and its processor is designed to be massively parallel. The most important difference is that the Xeon Phi processor is based on x86 architecture being used in common multicore CPUs. Therefore, the device is capable of running more general code and it even hosts an internal operating system, which makes it more independent on the host.

C. Distributed Computing

The performance scaling of individual servers (i.e., vertical scaling) has its limits, since we can incorporate only limited number of accelerators and employ only limited number of CPUs in a single host. In order to increase performance further, we need to utilize multiple servers, which are connected by some networking technology. This approach is called horizontal scaling.

Horizontal scaling brings new challenges, since the data transfers over the network require special handling and introduce additional possible bottlenecks to the whole system. Furthermore, incorporating many nodes in one system increases the chance of individual failures and also introduces new types of failures caused by communication.

In the remainder of this paper, we will assume that the system is composed of multiple nodes, which all employ some technology for message passing (e.g., MPI). On the other hand, we have no additional requirements on the individual nodes, and the whole environment could be quite heterogeneous. In other words, the distributed system may employ different servers with different configurations.

IV. Architecture of Distributed Bobox

The Bobox framework evaluates a data streaming program, which is expressed in a form of oriented graph called *execution plan*. Vertices of the execution plan (denoted *boxes*) perform individual data processing operations and each operation is implemented as a sequential routine in C++ language. The *edges* prescribe the data flow among the boxes. The data transfers on the edges are aggregated for efficiency reasons and transferred in bundles called *envelopes*. An envelope is a fragment of a data stream which consists of multiple rows, where each row has prescribed number of columns of prescribed type. Internally, an envelope uses column-oriented format, so it is organized as a tuple of columns where each column is an array of cells and columns of one envelope have the same number of items. The types and numbers of columns are defined in the *edge descriptor*, which is embedded in the execution plan specification. An example of execution plan is depicted in Figure 1.

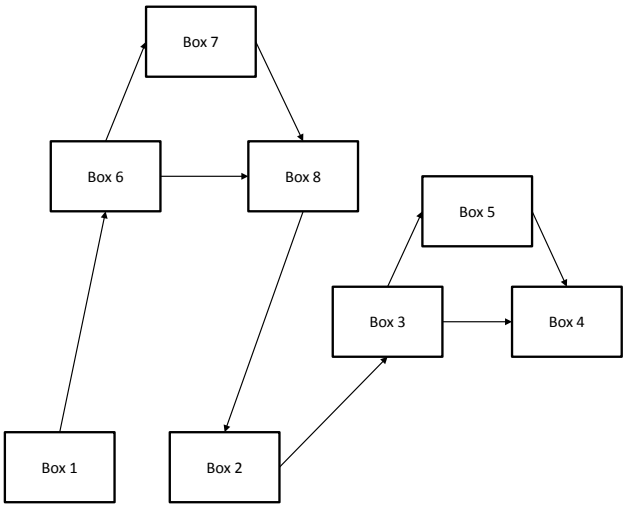


Figure 1. An example of Bobox execution plan

Manual construction of an execution plan in a procedural or object-oriented language would be tedious. Furthermore, the plan has to be tuned for the hardware configuration (e.g., number of available CPU cores) to achieve optimal performance. For these reasons, we have developed the Bobolang language [4], which simplifies the creation of execution plans. Bobolang is used for the definition of execution *model*. The model expresses the evaluation logic of a streaming data application, but it can generalize some details, which can be specified later (e.g., according to the hardware properties) when the model is instantiated into an execution plan. A model is either generated by a database front-end (e.g., a SPARQL engine [29]) or it may be created manually by a developer.

 ISBN: 978-1-61208-408-4

Modifications described in this paper require updates of both the parallel runtime that processes execution plans and the Bobolang language that describes the model of a streaming application. Individual issues of these updates are addressed in this section.

A. Considering Distributed Hardware

Distributed Bobox runs on a *cluster*, which is defined as a set of computers connected with each other by one *connection technology*. In this case, the term connection technology refers to the API used for communication regardless of its physical realization in hardware. Each computer connected to the cluster is denoted a *node*. A node may have one or more accelerators and the nodes in a cluster do not have to be homogeneous – i.e., nodes may be built on different architectures and may utilize variable numbers and types of accelerators. An accelerator is accessible locally from its node (which we also denote *host* when emphasizing its relation to the accelerator) and the node must provide either explicit management of the accelerator or extend the connection technology so the accelerator can be attached to the cluster as a nested node. An example of a cluster is presented in Figure 2.

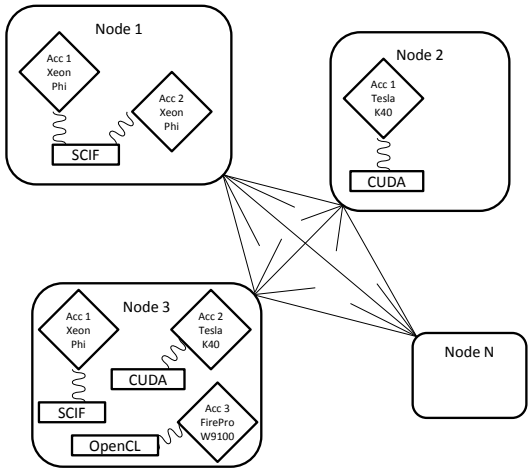


Figure 2. Example of cluster architecture from the Bobox perspective

For the purposes of our work, we will distinguish between *dependent* accelerators and *independent* accelerators. Dependent accelerators are fully controlled from the host system (i.e., from a code running in a CPU thread) thus they require additional management from our distributed framework. An example of dependent accelerator is a GPU, which can perform computations concurrently with the host system, but all computational tasks must be issued from a host CPU. Integration of dependent accelerators is detailed in Section V.

Independent accelerators can execute more complex code which is capable of managing the internal workload and which may also initiate communication and data transfers to and from the host system. Intel Xeon Phi is an example of independent accelerator, since it is operated internally by a Linux system and it can be perceived as a separate computer, which is interconnected with the host system via a PCI-Express bus. In the remainder of this work, the independent accelerators are treated the same way as cluster nodes.

Bobox relies on fine grained parallelism, so the execution plans of data queries [3][29] usually comprise hundreds or even thousands (single-threaded) boxes. Moreover, many front-end query compilers are capable of providing additional estimates regarding the amount of data being transferred on the edges. This information can be used to divide the execution plan into parts, which are then distributed over the nodes of the cluster.

This work focuses solely on the static distribution of the execution plans. Dynamic distribution and load balancing can be additionally employed; however, it raises many additional issues such as workload measurement, migration of box internal states, etc. Most of these details are beyond the scope of this paper.

An example of a static distribution of the execution plan is depicted in Figure 3. The plan is divided into three parts and distributed among three nodes. Nodes 1 and 2 are independent servers, while node 1 is also host for a Xeon Phi device.

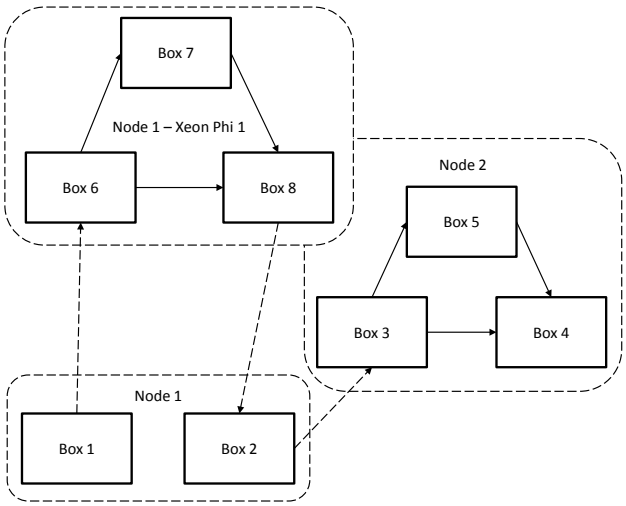


Figure 3. Example of an execution plan distributed on a cluster

The distribution of the execution plan requires additional method of envelope transfers. In its original parallel mode, the Bobox runtime assumes that all boxes can share data via the operating memory of the node. Therefore, passing an envelope from one box to another is only a matter of passing one memory pointer. Furthermore, shared memory allows immutable data to be shared among boxes.

In the distributed environment, the envelope must be hard-copied whenever crossing a border of a cluster node. In order to minimize modifications of the original runtime, we introduced two specialized boxes.

- A *send box* (S-box) consumes envelopes, performs data serialization, and send the data to another node using connection technology.
- A *receive box* (R-box) receives data from the connection technology, deserializes them, and emits them as envelopes.

The S-Boxes and R-boxes form inseparable pairs, where each pair represents one unidirectional peer-to-peer connection. Different pairs may use different types of communication technologies, but both boxes of a single pair must use the same

technology. The example from Figure 3, augmented with a set of corresponding send/receive boxes is presented in Figure 4. The solid arrows represent normal edges of the execution plan (i.e., where the envelopes are passed via shared memory) and the dashed arrows symbolize connections between transport boxes.

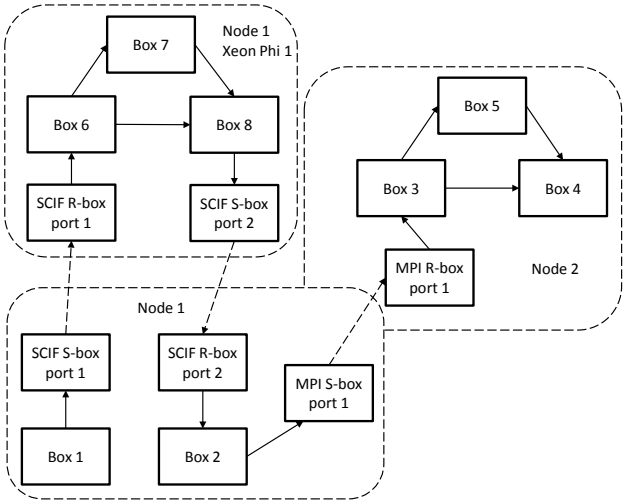


Figure 4. A distributed execution plan connected with communication boxes

B. Construction of Distributed Execution Plan

As mentioned before, the Bobox execution plan is not assembled manually by the programmer, but it is created automatically as an instance of a given application model. The model is written in Bobolang and it may also be generated from a query language such as SPARQL or SQL by a Bobox frontend module. In a single-node environment, the instantiation of a model into the execution plan is straightforward. In the distributed environment, the situation becomes much more complex, since the execution plan has to be divided among individual nodes and appropriate send and receive boxes must be added at the node boundaries. The situation gets even more complicated when dependent accelerators are being utilized in the system.

In order to maintain generality and sufficient level of abstraction, we have modified the execution plan instantiation process in the following way. A new intermediate plan abstraction called *bound model* is introduced and the term 'bound' emphasized that the model is designated only for the current cluster configuration. The bound model is also described in Bobolang and it is created from the model by the *concretization* process, which is performed by a *placement binder*. The complete Bobox abstraction overview and terminology is summarized in Figure 5.

The placement binder is a separate component which has complete information regarding the cluster topology, hardware properties of the nodes, and properties of the attached accelerators. Its concretization process transforms the original model to the bound model by replacing selected edges by S-box/R-box pairs and by replacing selected boxes or subgraphs by accelerator boxes that contain nested execution plans for selected accelerators. Furthermore, the placement binder assigns each box to a specific node in the cluster and each accelerator box to

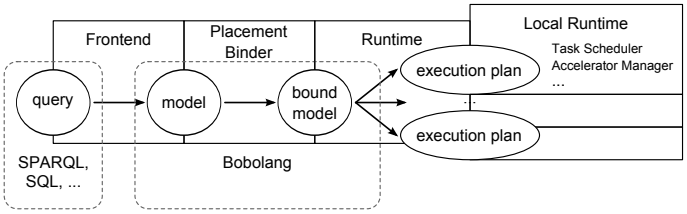


Figure 5. Bobox model abstraction and execution plan construction

an accelerator. These assignments are represented as Bobolang box annotations and they are recognized by the Bobox runtime that instantiates the models.

The bound model is subsequently distributed to every node of the cluster and passed to their respective Bobox runtimes. The local runtimes work in the same way as the single-node runtime, except that each runtime instantiates only those boxes (and their adjacent edges), which are designated to its respective node.

C. Technical Details

In the remainder of this section, we address several technical issues, which are essential for the distributed implementation. The first problem is caused by the diversity of addressing schemes that the underlying communication technologies use in S-box/R-box transfers (MPI uses message tags, SCIF uses ports, TCP/IP uses address:port pairs, etc.). Since the placement binder has to be independent on the communication technologies, it assigns each S-box and R-box an identification number called *port*. The runtime is responsible for mapping these port numbers to identification tokens of particular communication technologies. The mapping is stored in a distributed form, so the corresponding pairs of S/R-boxes can establish their data connections. The placement binder is responsible for correct port assignments (i.e., that there is exactly one S-box and one R-box for each issued port number).

The Bobox runtime relies on an external startup mechanism (e.g., such as Hydra or MPD, which are used by MPI technology), which is responsible for starting a Bobox runtime process on every node of the cluster. After the local runtime starts and detects the local accelerators, it starts the appropriate versions of the Bobox runtime on the independent accelerators and the accelerator management threads for each dependent accelerator. Finally, the local runtimes use group communication primitives to exchange their configurations, so that each node knows the configuration of the whole cluster.

Section IV-B describes the process of execution plan creation and distribution. The initial model can be submitted to Bobox via any node, since the placement binder is replicated on every host and the bound model is distributed via distributed communication primitives of the runtime. If necessary, the Bobox front-end modules responsible for translating database queries into models can be replicated as well and the Bobox can be used as platform for a distributed database management system.

The proposed solution is highly scalable and it is expected to be applicable for large clusters. It has no single point of

 ISBN: 978-1-61208-408-4

failure since the hosts are mutually interchangeable. Nevertheless, the current implementation does not handle node failures and reliability is a subject of our future work.

V. Parallel Accelerators

In the previous section, we have described our proposed solutions for distributed processing and for independent accelerators, which are treated as independent nodes of the cluster. In this section, we address the issues of the dependent accelerators.

A. Integrating Accelerators into System

Dependent accelerators are more complicated, since they need to be managed from the host system. In case of GPUs (which are typical representatives of dependent accelerators), there are two basic approaches that are supported by current frameworks such as CUDA [30] or OpenCL [31]:

- The GPU can be managed by a dedicated thread and all operations are invoked as blocking operations (i.e., the thread is resumed by the operating system, when the issued work on the GPU has concluded).
- The GPU operations are issued in asynchronous manner so the thread which has issued them can process other tasks. The GPU events (such as work completion) can be either polled by the host code, or reported via asynchronous callbacks executed in a service thread.

In our system, we primarily use the first approach, but the second approach is additionally employed in special cases. Bobox runtime (i.e., its task scheduler) already support blocking operations [32] without limiting CPU utilization. The thread dedicated for GPU workload management uses internal API functions to notify the task scheduler that the thread is about to invoke blocking operation. The scheduler can wake up a replacement thread which supplants the blocked thread in the thread pool, so the size of the thread pool always corresponds to the available computational cores.

Modern GPUs [33] are capable of overlapping data transfers with kernel execution or even executing multiple kernels if they consist of only a few thread blocks. However, the GPU driver may exploit these features only when the overlapping work has been issued without explicit synchronization at the host side. To comply with these requirements of GPU API, we have additionally employed asynchronous operations in special cases when the host code needs to issue overlapping tasks.

B. Exploiting Data Parallelism

The parallel accelerators present a potential complication regarding their optimal utilization. The original Bobox framework and its distributed version have presented themselves very efficient for the parallel processing of OLAP workload; however, the framework itself employs task parallelism, which can be used to process data parallel and pipeline parallel problems as well. On the other hand, current accelerators (especially GPUs) are suitable mainly for data parallel problems and other forms of parallelism are difficult to express or even encumbered with serious overhead.

In order to avoid this problem, the execution plan should be designed or generated in a way that allows maximal applicability of the accelerators. In other words, the data parallel parts of the execution plan should be primarily assigned to accelerators whilst the task parallel or even sequential parts of the plan should be processed by CPUs. In order to simplify the design of the application model, Bobox framework introduces two techniques:

- operator composition
- envelope aggregation

Operator composition is one of the key features, which has been present in Bobox since its first versions. A composed box is treated as regular box in the execution plan, but it holds a nested execution plan instead of C++ routine. We have extended this feature to designate parts of the execution plan for accelerator processing. These parts are represented as specialized operators called *accelerator boxes* and these boxes hold a nested execution plan dedicated solely for an accelerator. Furthermore, the accelerator box carries annotations that identify the target accelerator which is responsible for the processing of the box. These annotations can also hold additional execution details, such as number of accelerator threads or required internal buffer sizes. The whole idea is depicted in Figure 6.

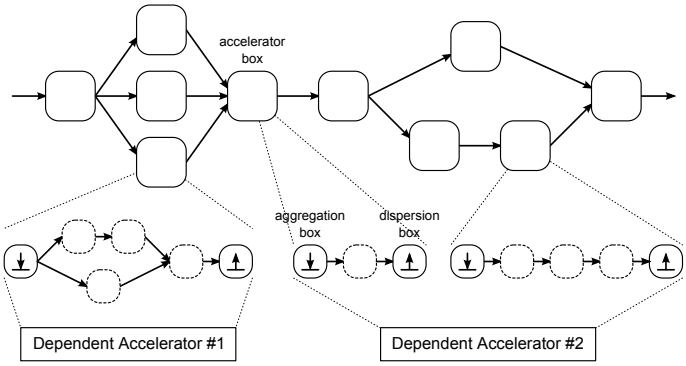


Figure 6. Designating parts of execution plan for accelerators

Regular boxes process data in portions called envelopes. Envelope size is carefully selected by the runtime based on the hardware properties of the target system – especially the CPU cache size. Accelerators need to process data in lager batches, in order to exploit the data parallelism principle. Therefore, an accelerator plan has an *envelope aggregator box* at its very beginning. This operator use heuristics and information from operator annotations to select the batch size for its current accelerator box. It aggregates data from multiple incoming envelopes and the aggregated envelope is transferred into the internal memory of the accelerator. Analogically, the plan represented in an accelerator box is ended by a *dispersion box*, which transfers the result data from the accelerator memory back to host memory and divides them into envelopes of regular size (suitable for CPU processing).

Finally, we need to specify how the accelerator plans are executed on accelerators. Even though the accelerator boxes are assigned to specific accelerators, the workload cannot be planned strictly statically since it is heavily data-driven. Therefore, the placement binder can employ oversubscription – i.e., assign multiple accelerator boxes to the same accelerator. The accelerator management thread plans the workload dynamically to maximize the overall utilization of the accelerator

 ISBN: 978-1-61208-408-4

and to increase the chance of data transfers and computations overlapping.

C. Memory Allocation

Perhaps the most problematic issue of the accelerator utilization is its memory allocation and management. Current accelerators are usually equipped with several gigabytes of internal memory, which is approximately $10 - 100\times$ smaller amount than the operating memory of modern NUMA servers. Therefore, an accelerator cannot replicate the entire content of the host memory and the data has to be copied only as necessary.

Internal execution plan of an accelerator box requires four buffers. The buffers may have complex internal structure, or even be composed from several memory blocks. However, we are omitting technical details and focusing on the purpose of these buffers. These buffers need to be allocated in the internal memory of the accelerator:

- buffer for input data,
- buffer for output results,
- buffer for internal data (operator state),
- and buffer for temporary data.

The input data buffer stores data from the aggregated envelopes which are transferred from the host. Analogically, output buffer is designated for data yielded by the accelerator box, which are transferred back to the host memory. Both these buffers are present in at least two instances, so that one pair of input/output buffers can hold data that are being processed by the accelerator, whilst the other buffers are engaged in concurrent data transfers (the following input envelope is being copied to the accelerator and the preceding result envelope is being copied to the host memory).

The buffer for internal data holds the current state of the operator which needs to be maintained across the processed envelopes. The consistency of the internal operator state has to be achieved by internal synchronization implemented in every part of the execution plan of the accelerator box. The scheduler ensures that at most one envelope is being processed by the accelerator at any time (except for the external data transfers).

The temporary buffers are designed to hold intermediate data that may be required by the execution plan of the accelerator and that cannot be accommodated by internal storage capacities of the computational units such as registers or shared memory of an SMP in case of GPUs. Unlike the internal state buffer, data in the temporary buffer does not have to be kept between the processing of two subsequent envelopes nor they have to be initialized before the first envelope is processed.

All buffers are allocated when the accelerator box receives its first incoming envelope. Furthermore, the internal data buffer has to be filled with the initial state of the operator. The buffers are disposed when the accelerator box receives termination envelope (the poisoned pill) and when the last aggregated envelope is processed.

D. Deadlocks

Multiple accelerator boxes can be assigned to a single accelerator and each one will attempt to allocate all its memory buffers when it receives first portion of data to process. If the overall size of the buffers required by all boxes assigned to an accelerator exceeds its available memory and if the data flow pattern requires that all the accelerators are working simultaneously, the overall execution will end by either failure or by a deadlock.

A naïve solution to this problem is to instruct the execution plan designer not to exceed the combined memory capacity of any accelerator, except in cases when boxes assigned to the same accelerator (and which possibly exceed its total memory capacity) are strictly divided by an implicit barrier such as a global sort operator placed between them in the execution plan. In this approach, a failure to allocate memory buffers for an accelerator box should be treated as a global failure of the whole execution plan and the responsibility of the frontend.

Unfortunately, the naïve solution limits the utilization of the accelerators quite severely. Hence, we propose a more elaborate model that allows allocator memory oversubscription and ensures successful execution. When an accelerator box fails to allocate its buffers, its execution is postponed until another box occupying the accelerator concludes its work and releases its buffers. Such behaviour may naturally lead to deadlock which has to either avoided or recovered. There are two possible approaches:

- The accelerator manager monitors the workload of all active boxes assigned to the accelerator and all requests from boxes that cannot be accommodated due to the limited memory capacity. When a deadlock is detected, the manager swaps internal state of selected accelerator box to the host memory and activate one of the waiting accelerator boxes. The box swapping can be even performed preemptively, when an accelerator box does not have any input data for some time. Let us emphasize, that only the internal data buffer needs to be swapped. If the box does not have any internal state or the internal state is relatively small, the box swapping can be very fast.
- Every accelerator box has an alternative backup execution plan that consist of regular boxes (i.e., boxes executed on CPU). If an accelerator box fails to allocate memory on an accelerator, it switches for the backup execution plan and starts processing its workload on the host system. Since the boxes used in the execution plans are usually developed so that Bobox does not require accelerators, constructing an alternative backup plan for each accelerator box should be rather simple.

Both presented approaches can also be combined. Furthermore, if the internal state of the accelerator box and its backup execution plan can be maintained in a binary compatible format, an accelerator box can switch between GPU plan and the CPU backup plan dynamically by swapping the internal state to/from the accelerator.

VI. Conclusions

The main contribution of this paper is a description of the architecture of a heterogeneous distributed system. The unique feature of the system is a combination of three modern computing platforms – multi-core CPUs, many-core accelerators like Intel Xeon Phi, and GPU-based accelerators. In particular, we investigated the differences between these two types of accelerators and suggested both static distribution of execution

 ISBN: 978-1-61208-408-4

plans, as well as dynamic swapping of work between hosts and their accelerators.

We are currently experimenting with all the described approaches; however, the dynamically evolving properties of accelerators requires larger empirical data to make clear conclusions on the strategy. In our future work, we will measure various types of workloads processed on various hardware configurations, in order to compare different load-distribution strategies.

ACKNOWLEDGMENT

This work was supported by the Czech Science Foundation (GACR), projects P103-13-08195S and P103-14-14292P, by Charles University Grant Agency (GAUK) project 122214, and by Specific Research project SVV-2014-260100.

REFERENCES

[1] D. Bednarek, J. Dokulil, J. Yaghob, and F. Zavoral, "Bobox: Parallelization Framework for Data Processing," Advances in Information Technology and Applied Computing, 2012, pp. 189–194.

[2] D. Bednárek, J. Dokulil, J. Yaghob, and F. Zavoral, "Data-flow awareness in parallel data processing," in Intelligent Distributed Computing VI. Springer, 2013, pp. 149–154.

[3] Z. Falt, D. Bednarek, M. Cermak, and F. Zavoral, "On Parallel Evaluation of SPARQL Queries," in DBKDA 2012, The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications, 2012, pp. 97–102.

[4] Z. Falt, D. Bednárek, M. Kruliš, J. Yaghob, and F. Zavoral, "Bobolang: A language for parallel streaming applications," in Proceedings of the 23rd international symposium on High-performance parallel and distributed computing. ACM, 2014, pp. 311–314.

[5] R. Stephens, "A survey of stream processing," Acta Informatica, vol. 34, no. 7, 1997, pp. 491–541.

[6] W. Thies, M. Karczmarek, and S. Amarasinghe, "StreamIt: A language for streaming applications," in Compiler Construction. Springer, 2002, pp. 179–196.

[7] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," Proceedings of the IEEE, vol. 75, no. 9, 1987, pp. 1235–1245.

[8] D. J. Abadi et al., "The Design of the Borealis Stream Processing Engine," in CIDR, vol. 5, 2005, pp. 277–289.

[9] A. Arasu et al., "STREAM: the stanford stream data manager (demonstration description)," in Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003, p. 665.

[10] D. Maier et al., "NiagaraST," 2010, http://datalab.cs.pdx.edu/niagara/ [Online, retrieved: February, 2015].

[11] A. Das, W. J. Dally, and P. Mattson, "Compiling for stream processing," in Proceedings of the 15th international conference on Parallel architectures and compilation techniques. ACM, 2006, pp. 33–42.

[12] P. Mattson, "A programming system for the imagine media processor," Ph.D. dissertation, Stanford University, 2002.

[13] I. Buck et al., "Brook for GPUs: stream computing on graphics hardware," in ACM Transactions on Graphics (TOG), vol. 23, no. 3. ACM, 2004, pp. 777–786.

[14] R. D. Chamberlain et al., "Auto-Pipe: Streaming applications on architecturally diverse systems," Computer, vol. 43, no. 3, 2010, pp. 42–49.

[15] J. Reinders, Intel Threading building blocks. O'Reilly, 2007.

[16] "Flow Graph," http://www.threadingbuildingblocks.org/docs/help/reference/flow_graph.htm [Online, retrieved: February, 2015].

[17] R. Chandra, Parallel programming in OpenMP. Morgan Kaufmann, 2001.

[18] A. Pop and A. Cohen, "A stream-computing extension to OpenMP," in Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers. ACM, 2011, pp. 5–14.

[19] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, 2008, pp. 107–113.

[20] R. Stewart and J. Singer, "Comparing fork/join and MapReduce," Technical Report HW-MACS-TR-0096, Department of Computer Science, Heriot-Watt University, Tech. Rep., 2012.

[21] M. Stonebraker et al., "MapReduce and parallel DBMSs: friends or foes?" Communications of the ACM, vol. 53, no. 1, 2010, pp. 64–71.

[22] "Apache Pig," http://pig.apache.org/ [Online; accessed 2014-03-18].

[23] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008, pp. 1099–1110.

[24] N. Govindaraju, J. Gray, R. Kumar, and D. Manocha, "Gputerasort: high performance graphics co-processor sorting for large database management," in Proceedings of the 2006 ACM SIGMOD international conference on Management of data. ACM, 2006, pp. 325–336.

[25] T. Kaldewey, G. Lohman, R. Mueller, and P. Volk, "Gpu join processing revisited," in Proceedings of the Eighth International Workshop on Data Management on New Hardware. ACM, 2012, pp. 55–62.

[26] "Oracle," http://www.oracle.com [Online, retrieved: February, 2015].

[27] "PostgreSQL," http://www.postgresql.com [Online, retrieved: February, 2015].

[28] B. He et al., "Relational query coprocessing on graphics processors," ACM Transactions on Database Systems (TODS), vol. 34, no. 4, 2009, p. 21.

[29] Z. Falt, M. Cermak, J. Dokulil, and F. Zavoral, "Parallel sparql query processing using bobox," International Journal On Advances in Intelligent Systems, vol. 5, no. 3 and 4, 2012, pp. 302–314.

[30] C. Nvidia, "Compute unified device architecture programming guide," 2007.

[31] A. Munshi et al., "The opencl specification," Khronos OpenCL Working Group, vol. 1, 2009, pp. 11–15.

[32] M. Kruliš, Z. Falt, D. Bednárek, and J. Yaghob, "Task scheduling in hybrid cpu-gpu systems," Informačné Technológie-Aplikácie a Teória, p. 17.

[33] NVIDIA, "Kepler GPU Architecture," http://datalab.cs.pdx.edu/niagara/ [Online, retrieved: February, 2015].

 ISBN: 978-1-61208-408-4

Accelerating Data Mining on Incomplete Datasets by Bitmaps-based Missing Value Imputation

Sameh Shohdy, Yu Su, Gagan Agrawal
Computer Science Department, The Ohio State University
Columbus, Ohio, USA, 43210
Email: {ahmedsa,su1,agrawal}@cse.ohio-state.edu

***Abstract*—Among all 'big data' research issues, the veracity challenge, which refers to the precision and accuracy of the data, has not received as much attention. Traditionally, it has been well known that problems related to data quality, such as incomplete, redundant, inconsistent, and noisy data pose a major challenge to data mining and data analysis. Particularly, we note that existing methods for handling missing values cannot scale to larger datasets. In other words, this particular veracity challenge has been addressed, but not in context of also handling volume (and possibly the velocity) challenge of 'big data'. This paper focuses on speeding up the missing values imputation process using the bitmap indexing technique. The research takes two directions: first, the bitmap indexing is used to directly access the required records for the imputation method (i.e., Direct Access Imputation (DAI)). Second, the bitmap indexing technique is used for missing value estimation using the pre-generated bitmap indexing vectors without accessing the dataset itself (i.e., Bitmap-Based Imputation (BBI)). Both approaches have been evaluated using different real and synthetic datasets, and four common imputation algorithms. We show how our bitmap-based methods can accelerate data mining classification of incomplete data while also maintaining precision.**

***Keywords*—Missing Values; Bitmap Indexing; Indexing as a Service.**

I. Introduction

In recent years, 'big data' has become one of the most important challenges in computing research. One of the most popular definitions of 'big data' involves four challenging aspects of dealing with data - *volume*, *velocity*, *variety*, and *veracity*. There has been a lot of work in recent years on addressing these challenges, with the volume challenge, and to a less extent, the velocity challenge, receiving most interest. Work on MapReduce [1] and related framework for handling massive data, as well the work on handling streaming data and *in-situ* data analysis have been topics of much investigation.

The *veracity* challenge, which refers to the *precision* and *accuracy* of the data, has not received as much attention. Traditionally, it has been well known that problems related to data quality, such as incomplete, redundant, inconsistent, and noisy data [2] pose a major challenge to data mining and data analysis. In fact, one of the most important steps in data mining is considered to be the *data preparation* step, which is the process of ensuring the quality of data by changing the original data into a suitable format for the analysis process. About 60% of data mining time is consumed in the data preparation process compared to only 20% in the actual data mining process [3].

Unfortunately, recent work on MapReduce and other 'big data' frameworks has not emphasized the challenges of data preprocessing, and particularly, the difficulty of applying known techniques on large-scale data. Consider the problem associated with handling Missing Values (MVs). Incomplete dataset or missing values can impact data analysis tasks. To avoid their negative impact, a popular approach is to *fill* the missing values with estimated values that can be calculated from the complete records of the same dataset [4]. Several studies have illustrated different methods for dealing with missing values, i.e., using the complete set of records to *impute* the missing values in the real datasets. Clark and Niblett explained a simple algorithm (i.e., the CN2 algorithm) for imputing missing values using the most common value of the same attribute [5]. A modified version of this method is the most common value of the missing value attribute restricted to a label (or concept) [6]. Another study performed by Grzymala-Busse suggested replacing the missing value with all the possible values that appears at the same attribute in all the dataset records [7]. Another approach called Global Closest Fit has also been proposed [8], where missing value in a specific record is taken from the corresponding values from the record that is most similar to this record in the entire dataset. Moreover, different studies have surveyed and compared several approaches for missing values imputation [8]–[12].

A common theme in all of the works in this area is that algorithms for imputing missing value cannot scale to larger datasets. In other words, this particular *veracity* challenge has been addressed, but not in context of also handling volume (and possibly the velocity) challenge of 'big data'. This paper addresses this shortcoming of existing work, and focuses on improving the scalability of methods. We achieve this by using indexing to accelerate missing value imputation - particularly, we use bitmap indexing, which can serve as a summary of datasets. A key advantage of this technique that low-cost bit-wise operations can be exploited for processing. In this work, we show how various missing value imputation methods can be accelerated using indexing, at two levels. First, like any indexing technique, bitmap indexing can be used to look-up relevant records, and thus, complete scans over the entire dataset are avoided. Second, one can use bitmaps as a summary of the entire dataset, and not access the dataset at all. These two methods are referred to as Direct Access Imputation (DAI), and Bitmap-Based Imputation (BBI), respectively. We evaluate both approaches extensively using different real and synthetic datasets, and four different imputation algorithms.

The rest of paper is structured as follows: In Section 2, a brief background on existing imputation methods and bitmap indexing is introduced. In Section 3, we present our proposed bitmap-based algorithms in detail. In Section 4, we present the performance evaluation of the proposed algorithms. Finally, we draw our conclusions in Section 5.

II. Background

This section provides key background on two important topics. The first is the existing algorithms for dealing with missing values, i.e., specifically the imputation methods. The second is bitmaps-based indexing.

A. Imputation Methods

The goal of any imputation algorithm is to estimate the missing value in a specific record using others complete records in the same dataset. As we stated earlier, this has been an active area of research and much work has been done. We give an overview of four existing algorithms.

1) Global Closest Fit Method (GCF): The main idea in this method is to replace the missing value by the value of corresponding attribute from a single record that is the most similar record to the record with the missing value. The most similar record is the one with the smallest *distance* from the current record, with the distance calculated as follows:

 ISBN: 978-1-61208-408-4

$$Distance(X, Y) = \sum D(x_i, y_i) \quad (1)$$

$$D(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \text{ is missing} \\ & \text{or } y_i \text{ is missing} \\ \frac{|x_i - y_i|}{r} & \text{if } x_i \neq y_i \end{cases} \quad (2)$$

Equation (1) is the general form for calculating the difference between any two records X and Y as a summation of the distance between values in each attribute for both records. Equation (2) can be used to calculate the distance between the two values for the same attribute in both X and Y records, where r is difference between maximum and minimum values of this attribute for the entire dataset – this corresponds to using what is referred to as the Manhattan distance, a commonly used metric [10].

2) Most Common Value (MCV): This is a simple method where the missing value can be replaced by the most common value in the missing value attribute. By scanning the entire dataset, we can find the single value of the attribute that occurs most frequently [2] [13].

3) Concept-based Most Common Value (CMC): The Concept-based Most Common Attribute Value method is a special case of the most common value method. This method assumes that records have labels associated with them. Using this information, the missing value is replaced by the most common value for the specific attribute, among the set of records with the same label [13].

4) Concept-based Mean Attribute Value (CMAV): The missing attribute value is replaced with the mean of values of the same attribute, among the records having the same concept. A further generalization of this method is the mean attribute value method, which is used to impute the missing value as the mean of all values in the same attribute, i.e., ignoring any possible labels [8] [11].

B. Bitmap Indexing Method

The implementations of each of the above techniques in the literature assume a small dataset that fits in memory, and involves simple scan(s) on all records to find values to substitute missing values. These implementations clearly have limitations while dealing with large datasets. Thus, we examine the use of indexing as a mechanism to accelerate imputation methods we have described above. There are several reasons for considering this approach. First, indexing is already implemented in most data stores that are handling large-scale data. Second, intuitively, it is easy to see that all or most of the methods described above can be accelerated using indexing support.

The particular indexing method we have chosen is bitmap indexing [14]–[16]. This section gives a brief overview of bitmap indexing and the reasons behind choosing it specifically to improve the performance of existing missing values imputations techniques.

In the simplest form, a single bit-vector is generated for each distinct value *v* in each attribute. The length of the bit-vector is equal to the number of data records in the dataset. If the value of the particular attribute in a record matches the value *v* this bit-vector corresponds to, the bit is set to 1, and is 0 otherwise. However, large number of distinct values for an attribute negatively impacts the bitmap indexing performance, because large number of bit-vectors need to be generated to cover all these distinct values. A typical solution for this challenge is to use the binning process. In the binning process, the attribute values are binned so that each bin represents a range of values [15] [17]. Table I illustrates an example of using bitmap indexing technique with/without binning process.

Bit-vectors can become extremely space consuming in the original form. Thus, several compression methods have been proposed to solve the space problem, which includes Byte-aligned Bitmap Code [18], Word-Aligned Hybrid code [19], Partitioned Word-Aligned Hybrid (PWAH) [20], and Position List Word-Aligned hybrid [21]. The main idea in these methods is to encode a possible large series of contiguous 0s (or 1s, though they are less likely) before storing them.

Using the bitmap indexing, a typically subsetting query can be processed by extracting a set of bit-vectors depending on the query conditions, and then the result can be exported by performing bit-wise AND and OR operations over these bit-vectors, which are normally supported very efficiently in the hardware [22].

TABLE I. AN EXAMPLE OF BITMAP INDEXING

Temp.	**Bitmap Indices**								
	Without Binning						With binning		
	23	25	28	30	33	35	23-25	28-30	33-35
23	1	0	0	0	0	0	1	0	0
25	0	1	0	0	0	0	1	0	0
28	0	0	1	0	0	0	0	1	0
28	0	0	1	0	0	0	0	1	0
30	0	0	0	1	0	0	0	1	0
25	0	1	0	0	0	0	1	0	0
23	1	0	0	0	0	0	1	0	0
33	0	0	0	0	1	0	0	0	1
33	0	0	0	0	1	0	0	0	1
35	0	0	0	0	0	1	0	0	1

There are several reasons for choosing bitmap to improve the imputation process. First, bitmap treats each attribute separately, and thus, is likely to support efficiently index for incomplete datasets [23]. In comparison, other popular indexing methods such as B-tree, B+-tree, and R-tree are designed to support queries on one or two attributes. Second, bitmap indices can be built on top of an existing data store, i.e, they do not require that data be reorganized under the index. Such data reorganization is extremely time consuming and may not be justified if the only or the major goal of indexing could be to support imputation of missing values. Third, missing values imputation operation is an approximate process. Thus, bitmap-based missing values imputation algorithms can be accelerated using bitmap approximation techniques, such as binning without loss of precision. Finally, during data mining, we typically target datasets that are not updated, and bitmaps are well suited for such datasets [24].

III. Proposed Imputation Methods

Our goal is to accelerate different imputation algorithms using bitmap indices. It turns out that there are two distinct ways in which bitmaps can be used for the imputation process, with trade-off between accuracy and processing speed. First, search for records that are similar to the record with missing value, or records with a particular label can be accelerated using bitmaps. Thus, a scan on the entire dataset can be replaced by look-up on specific parts of the dataset. Second, it is possible to use bit-vectors as an approximate summary of the entire dataset, and provide approximate answers using this information. We refer to the first strategy as the *Direct Access Imputation* method or DAI, whereas the second method is called *Bitmap-Based Imputation* method or BBI. Before discussing the approaches, we note that bitmaps have been used to support query processing in several systems [25]–[27]. Thus, if bitmaps are already being built to support query processing, implementing imputation method using them will not even involve additional overhead of index generation.

A. Direct Access Imputation (DAI)

Several imputation algorithms can be accelerated if we can directly access the required records to impute each missing value without full database scan. This is the idea of the DAI approach. We show how

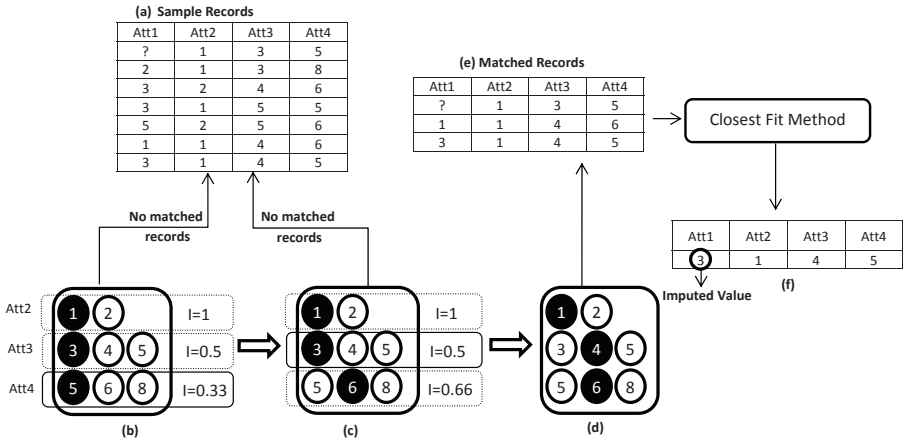


Fig. 1. Global Closest Fit Method using Bitmap Indexing - Impact Value of Each Attribute is Used to Select Records

to apply the DAI method in the case of three imputation algorithms: Global Closest Fit, Concept-based Most Common Attribute Value and Concept-based Mean Attribute Value , each of which was introduced in Section II.

1) Global Closest Fit Method: Recall that this method imputes the missing value from the record(s) that are closest to the record with the missing value, in the sense of the distance between the two records. The brute-force approach to implement this method involves retrieving the missing record and computing the distance between this record and all other records (i.e., Full Dataset Scanning method). By using bitmaps, we can retrieve only the set of records that have attributes values close to the target record values.

In performing a full-scan of the dataset, it is relatively easy to select the closest match with respect to other attributes. However, using an indexing method to find the closest match requires certain challenges. We need to keep identifying neighboring values of each attribute, so as to find the closest match or a good approximation. To enable the selection process, we use the notion of *impact* of a neighboring value of an attribute to the distance. The *impact* I of the next value v for each attribute A can be computed as: $I(A,v) = \frac{v-c}{r}$, where c is the current used value for attribute A and r is the difference between the maximum and minimum value for attribute A. The idea is that neighboring values of each attribute can contribute differently to the impact with respect to the distance. Records that have a neighboring value with the smallest *impact* should be searched first, because they will have a smaller distance.

An example of our approach is given by Figure 1 to solve this challenge. Subfigure (a) represents a set of records from any given dataset that has a missing value marked by *"?"*. Suppose that there are no records that have the same value for all other attributes as the record with the missing value. A second search process is conducted by selecting the next closest value (as recorded in a metadata file) for each attribute, and choosing the attribute and its value with the least *impact* – for example, $att4$ has the smallest *impact* as shown in sub-figure (b). In subfigure (c), we now need to retrieve records where the corresponding value is less than or equal to this particular value for the chosen attribute $att4 = 6$ and larger that the old value $att4 = 5$, while having the same value for the remaining attributes that were used for the previous search. If there are no matching records identified, the process of picking a new attribute using the *impact* value is repeated (see sub-figures (c) and (d)). Once the result bit-vector does retrieve a set of records, e.g., as shown in (e), these records should be examined, with the goal of obtaining the most closest fit record(s) to the record with the missing value. This last step is shown through the sub-figure (f).

The implementation of our approach has been in the context of a particular bitmap-based data management system [27]. This system supports database-like query subsetting over data in scientific data formats like NetCDF. The query processing module converts the user query request into a set of query conditions where each condition can be represented by a single bit-vector (i.e., the *condition bit-vector*).

Returning to the example shown in Figure 1, the challenge here is how to build a query that retrieves the records that are most closest to the record with the missing value in each step. For example, in sub-figure (a), we need to retrieve all records that exactly match the record with missing value. In this case, each distinct attribute query condition will be built to cover all the records in a range with the maximum and minimum values equalling the corresponding attribute value in the missing value record. Further, in sub-figure (b), we need to increase the internal range in $att4$ to cover the records that can be used in the imputation process. So, the query should be built to have a maximum and minimum values related to $att4$, i.e., $att4 > 5$ AND $att4 \leq 6$.

Algorithm 1 BuildQueryforGCF($record$)

```
1: for att= 1 → sizeof(record) do
2:     vals[att]=ReadValues(MetaDataFile,att)
3:     query_vals[att]= record[att];
4: end for
5: resultbitvector:= null
6: while resultbitvector is null do
7:     for att= 1 → sizeof(record) do
8:         min_diff:= ∞
9:         for i= 1 → sizeof(vals[att]) do
10:            diff=abs(vals[att].get(i)-query_vals[i])
11:            if diff < min_diff then
12:                min_diff=diff
13:                r=Max(att) − Min(att)
14:                impact[att]=min_diff/r
15:                candidate_vals[att]=vals[i]
16:            end if
17:        end for
18:    end for
19:    selected_att=PickMin(Impact)
20:    query_vals[selected_att]=candidate_vals[selected_att]
21:    vals[selected_att].remove(selected_value)
22:    query= GenerateQuery(record,att,query_vals)
23:    resultbitvector=RetrieveIndex(record,query);
24: end while
```

Fig. 2. Algorithm 1: Query Building Operation

The query building operation algorithm is formally described by Figure 2. The algorithm takes as input the record with missing value. In the line 2, distinct values for each attribute are read from its $metadata$ file into $vals$ vector. Line 3 initiates the $query\_vals$ array with the values of missing value record. Through the algorithm, for each attribute, all records that have a value between both $query\_vals$ value and $record$ value should be retrieved. The idea is to adapt the $query\_vals$ array values until the $resultbitvector$ retrieved any number of records. In lines 9-17 , for each attribute's distinct value, the impact of changing the old value in $query\_vals$ with this value is computed to determine the value of each attribute that makes the smallest change in the $impact$ value. In the line 19, $PickMin$ method is used to determine the attribute that with the smallest $impact$ value. In the line 20, The value of this attribute replaces the old value in the $query\_vals$ array. In line 21, this value is removed from $vals$ vector to guarantee a new value is selected the next time. In line 23, the query was generated using the missing value's $record$ and the $query\_vals$ array. If the $bitmapvector$ has no records, a next iteration is required.

The same strategy can be applied if binning process is used. The operation of changing a proper attribute value depending on the impact degree on the distance can be applied to the binned values by selecting a proper binned range of values rather than a single value. In this case, the impact degree can be computed by using the average value of the current range c and the average value of the candidate range v.

2) Concept-based Most Common Value Method: The method can be implemented using bitmaps as follows. After retrieving all missing values, a single query request is built for each record R that contains a missing value. The goal of the query will be to retrieve all the records that contain the same label as the record R. These retrieved records can be then be checked to determine the most common value for the attribute where the record R has a missing value. These operations are trivial to perform if binning is not used.

However, in the case of binning, for each missing value, the retrieved records will be the records that contain a label in the same binned range, as the label associated with the record R. For the last step, we can determine the most common bin, and then proceed to search for the most common value. Note that this may result in a different value, as compared to the value obtained by a brute-force algorithm. This is because the most popular bin may not contain the most commonly occurring value.

Note that implementation of most common value is also very similar - we simply focus on finding the most common value of the attribute involved across all records.

3) Concept-based Mean Attribute Value Method: Similar to the previous method, a single query request is built to retrieve all the records that have the same label as the record R that has a missing value. These records can be accessed directly and all the values of the attribute where R has a missing value can be retrieved into the memory. The estimated value is the mean of all retrieved values.

In the case of binning, certain calculations are required. The label attribute's metadata file should be checked to determine the binned range that contains the target record's label (i.e., the record with the missing value). Then, a simple query request is built to retrieve all the records within this binned range. These records are examined again to determine the records with the same label as the missing value record. The imputed value will be considered as the mean of all the values of the missing value attribute.

B. Bitmap-Based Imputation (BBI)

This approach involves imputing the missing values directly using the bitmap vectors, i.e., without accessing the dataset itself. We will explain the use of BBI approach in three imputation algorithms: Most Common Value, Concept-based Most Common Value, and the Concept-based Mean Attribute Value methods. An example of using the BBI methods in different imputation algorithms is given by Figure 3. The example uses a dataset with three attributes and a label.

1) Most Common Value Method: In this method, the key is to determine the number of records for each distinct value in the target attribute (i.e., attribute with missing value). The value with the largest number of records is the most common value. The bitmap bit-vectors can be used to determine the most common value in the following manner. First, distinct attribute values can be retrieved from the missing value attribute's metadata file. Second, for each distinct value, the related bitmap vector is retrieved from the index file. It is easy to count the number of records for each distinct value by counting the number of ones on each retrieved bit-vector. The bit-vector with the largest count of ones represent the bit-vector of the most common value in the target attribute.

Figure 3 (c) gives an example of retrieving each distinct value's bit-vector and the bit-vector with the largest number of ones reflect the most common value. However, the BBI approach cannot be used without approximation in the case of binning, because each bit-vector represents a set of values rather than a single value.

2) Concept-based Most Common Value Method: First, a query condition bit-vector is constructed to retrieve all the records with the same label as the missing value record's label. Second, for each distinct value in the target attribute's metadata file, a query request is built and passed to the bitmap query system to retrieve a bit-vector representing all the records equal to this value. A logic AND operation will be performed between all distinct values' bit-vectors and the label bit-vector. The number of ones per each distinct value's bit-vector is counted which represents the number of records with the same value and label. The value's bit-vector with the largest number of ones can be used as the imputed value for the missing value.

Similar to the most common value method, this strategy becomes less accurate in the case of using binning when constructing the bitmap index. Figure 3 (d) gives an example of using BBI approach with Concept-based Most Common Value algorithm.

3) Concept-based Mean Attribute Value Method: Similar to the previous method, a single condition bit-vector will be constructed to retrieve all the records that have a label matches with the label of the missing value record. For the missing value attribute, the imputed value is the mean of all the target records values of this attribute. In the case of binning the technique is changed. First, the label attribute's metadata file needs to be checked to determine the binned set that contains the missing value record's label. Second, we construct a query condition bit-vector to retrieve all the records within this set. These records can be checked to determine the records that have the same label of the missing value record. The imputed value will be the mean value for all the records with the same missing value record label. Figure 3 (e) also gives an example of this method. In the case of binning, the technique is changed. First, the label attribute's metadata file needs to be checked to determine the binned set that contains the missing value record's label. Second, we construct a query condition bit-vector to retrieve all the records within this set. These records can be checked to determine the records that have the same label of the missing value record. The imputed value will be the mean value for all the records with the same missing value record label.

IV. Experimental Evaluation

This section presents results from a series of experiments done to evaluate the proposed bitmap-based imputation approaches. Specifically, our proposed DAI and BBI imputation approaches are compared with the brute-force FSI imputation approach in two ways: execution speedup and imputed values' accuracy. In accuracy comparison, to focus on the impact of different approaches to missing value imputation on the data mining process, we used a popular classification algorithm (i.e., the CPAR algorithm [28]) to classify the dataset records after the missing values imputation process is applied using different approaches (i.e., FSI, DAI, and BBI), then the dataset noise ratio is evaluated in each case.

 ISBN: 978-1-61208-408-4

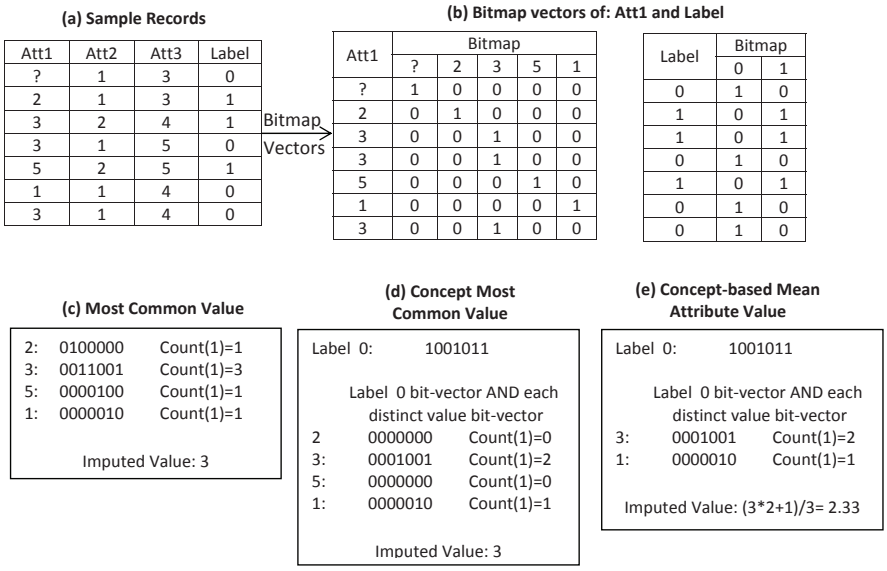


Fig. 3. Bitmap-based Imputation Method Example

All approaches were implemented in C++. The experiments were performed on Linux Red Hat 4.4.7-3 machine equipped with 2 quad-core Intel CPUs running at 2.53 GHz and with a 4 GB RAM.

A. Execution Speed Comparison

Execution speed experiments have been performed using two different datasets. The first dataset is what we refer to as the Berkeley dataset [29] (∼1GB), which is an earth surface temperature dataset. This dataset comprises seven attributes and nearly $3 \cdot 10^7$ records, and can fit into memory, and thus helping us evaluate the performance of the proposed approaches when datasets fit in memory. Second, we used a synthetic dataset (∼10GB), which was generated with nine numerical attributes and a label, and nearly $20 \cdot 10^7$ records. This dataset is used as an example of a large dataset, helping us evaluate the performance of proposed imputation methods on disk-resident datasets.

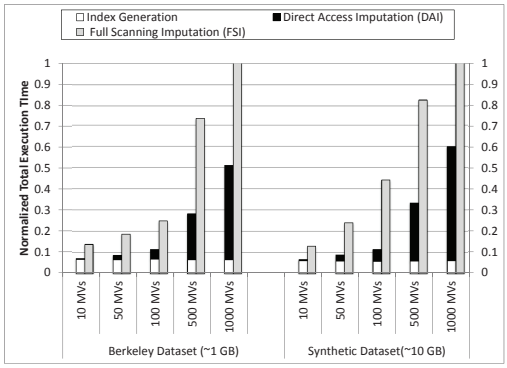


Fig. 4. Time Comparison between DAI and FSI Based Global Closest Fit Methods Using Different Numbers of Missing Values (MVs)

1) Global Closest Fit (GCF): Only DAI and FSI methods have been used to implement the Global Closest Fit imputation algorithm, as BBI cannot be applied. This experiment targets both datasets (i.e., Berkeley and the Synthetic dataset).A comparison between the execution speed in handling datasets' missing values on the GCF method with different number of missing values is shown in Figure 4. The number of missing values in the dataset varies from 10 to 1000, and they are distributed across different attributes. Note that our comparison is assuming that an index is generated specifically for the imputation process. However, in practice, an index may be generated in advance for a different reason, like need for supporting subsetting (or other types of) queries, and index generation time may be amortized. Returning to the Figure, the x-axis shows different number of missing values in both datasets while the y-axis shows the *normalized execution time*, which is computed as the fraction of the longest execution time. We can draw from the figure the following three observations. First, the total execution time of DAI method, even including the required time to generate index files for the target dataset, is less than the total execution time of the brute-force FSI method on both datasets. Second, with increasing number of missing values, the DAI's advantage becomes even more significant, when compared to the FSI method. Finally, the figure shows that the DAI method's relative performance is even better in the case of disk-resident datasets. The reason is as follows: the imputation time for a single value is the sum of the times for finding the most closest records to the missing value's record and the required time to retrieve these records, whether from the memory or the disk. In the case of the Berkeley dataset, the entire dataset fits into the memory, which means that the total execution time depends primarily on the time to find the most similar records. However, for a large dataset, disk operations are required to retrieve records from disk to memory.

2) Most Common Value (MCV): For the Most Common Value algorithm, Figure 5 shows the normalized total execution time of the algorithm using both BBI and FSI methods. Because all the missing values of the same attribute will be replaced by the most common value for this attribute, the performance with this algorithm depends on the number of attributes containing missing values, rather than the number of missing values itself. As we can see, BBI speeds up the imputation process compared to the brute-force FSI method. The BBI method depends only on the pre-generated bitmap vectors, so the increasing of dataset size does not affect the performance of the BBI method, unlike the FSI or the DAI methods. In other words, the performance gap between the total execution time in the FSI method and the BBI method increases as we consider a bigger dataset. However, BBI method becomes worse when the number of distinct values in the target attribute increase. In this case, the number of generated bitmap vectors increase which negatively affect the performance of the BBI method.

3) Concept-based Most Common Value (CMC): The Concept-based Most Common Attribute Value method can be implemented using both DAI and BBI methods. The normalized total execution

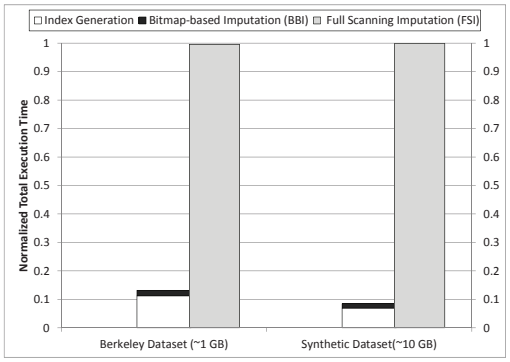


Fig. 5. Time Comparison between BBI and FSI Based Most Common Value Imputation

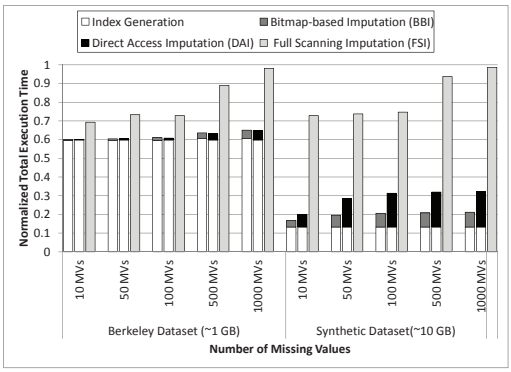


Fig. 7. Time Comparison between DAI, BBI, and FSI Based Concept-based Most Common Value Methods Using Different Numbers of Missing Values (MVs)

time for DAI, BBI, and FSI methods is shown in Figure 6 with different number of missing values in both target datasets. The figure shows that DAI and BBI methods speeds up the imputation process compared to the FSI method. The overall time is dominated by the index generation time. If the index can be pre-generated, the method can provide very large speedups. The figure also shows an important observation. In the Berkeley dataset, DAI satisfies a higher performance compared to BBI method with larger number of missing values. However, in the synthetic dataset, the BBI gives a higher performance. The reason of this behavior that BBI depends on the number of distinct values in the target attributes (i.e., in this case the missing value's attribute and the label attribute) while the DAI depends on the number of matched records for each missing value. In the Berkeley dataset, the number of distinct values in the target attributes is large compared to the synthetic dataset which negatively impact the BBI performance.

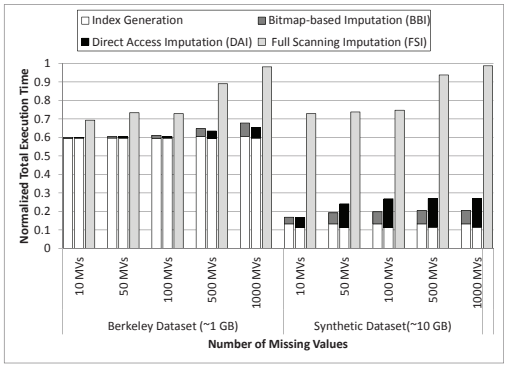


Fig. 6. Time Comparison between DAI, BBI, and FSI Based Concept-based Mean Attribute Value Methods Using Different Numbers of Missing Values (MVs)

4) Concept-based Mean Attribute Value (CMAV): The efficiency of implementing the Concept-based Mean Attribute Value algorithm with both DAI and BBI methods is shown by Figure 7. On both target datasets, the DAI and BBI methods are more faster than the FSI method with different number of missing values. Just like the case of the Concept-based Most Common Attribute Value method, in the Berkeley dataset the DAI is faster than BBI because of larger distinct values in the target attributes. However, in the synthetic dataset the performance gains from the BBI method is more significant, because the BBI method does not require any I/O processing compared to DAI method.

B. Impact of the Number of Distinct Values

A single bitmap vector is built for each distinct value in each attribute, unless binning is used. Increasing the number of distinct values in each attribute impacts both the index generation time and the proposed imputation methods based on bitmap vectors. To see this impact, we evaluate the performance of each imputation algorithm using both DAI and BBI methods with different number of distinct values in each attribute. Our experiment assumes that binning is not used. In this experiment, we generate five synthetic datasets with the same number of records (i.e., $20 \cdot 10^7$) and attributes (i.e., nine attributes and one label), but with different number of distinct values in each attribute (i.e., 100, 500, 1000, 5000, and 10000).

The performance of both CMC and CMAV methods with different number of distinct values is shown by Figure 8. It is not surprising that execution time increase with increasing number of distinct values. A more significant observation is that the BBI method becomes worse with very large number of distinct values. The BBI method depends on scanning the bitmap vectors to estimate the missing value, and increasing the number of distinct values increases the number of bitmap vectors generated, which slowdown the imputation process. Overall, it will be better to use binning when the number of distinct values is large.

C. Scalability

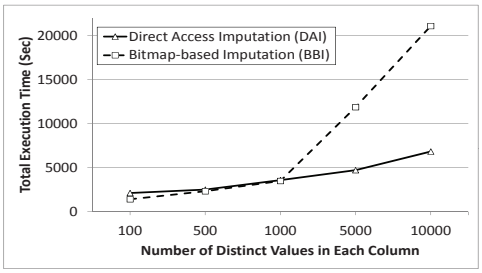
The goal of this experiment is to evaluate the scalability of our proposed methods when imputing missing values in datasets with increasing sizes. Five synthetic datasets are used in this experiment, i.e., 5, 10, 15, 20, and 30 GB. These datasets were randomly generated in two different manners: by increasing the number of attributes and by increasing the number of records. Each dataset was also generated to include 100 missing values distributed across different attributes. Because our goal is to evaluate the runtime scalability of proposed methods, we only recorded the total execution time of each imputation algorithm, without including the index generation time.

The results are shown in Figures 9 and 10. As shown in the figures, the execution times only increase linearly as the dataset size increases – either by increasing the number of records or by increasing the number of attributes in each record. This experiment also shows another observation, as illustrated particularly by Figures 10(b) and 10(c). Because the BBI method needs to analyse the indices files to impute the missing values, increasing the number of records negatively impacts the efficiency of BBI method compared to DAI method in both Concept-based Most Common value and Concept-based Mean Attribute imputation algorithms.

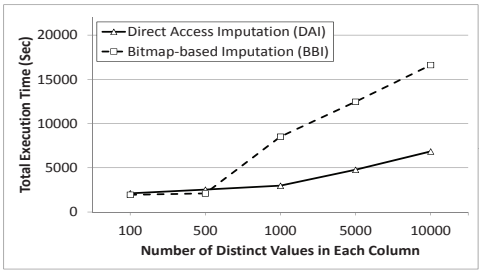
D. Imputation Method Impact on Data Mining

Since accuracy comparisons on synthetic datasets are not very meaningful, and because the size of the dataset was not very impor-

 ISBN: 978-1-61208-408-4

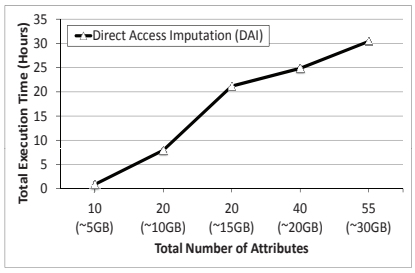


(a) Concept-based Most Common Value (CMC)

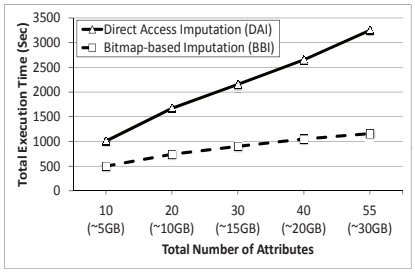


(b) Concept-based Mean Attribute Value (CMAV)

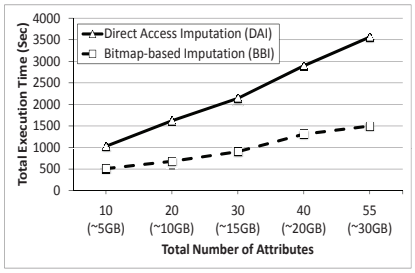
Fig. 8. Performance of Different Imputation Algorithms with Different Numbers of Distinct Values



(a) Global Closest Fit

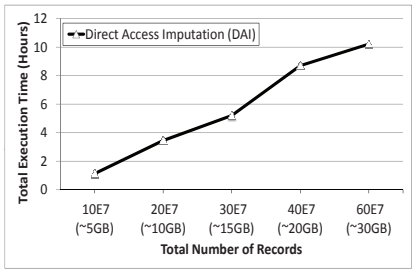


(b) Concept-based Most Common Value

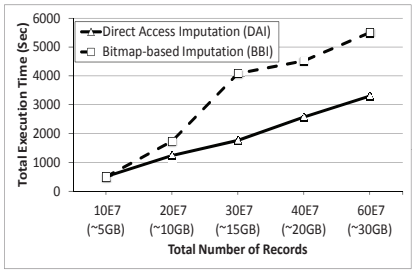


(c) Concept-based Mean Attribute Value

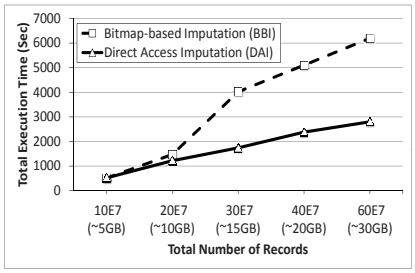
Fig. 9. Scalability: Influence of Increasing the Number of Attributes on DAI and BBI based Imputation Algorithms



(a) Global Closest Fit



(b) Concept-based Most Common Value



(c) Concept-based Mean Attribute Value

Fig. 10. Scalability: Influence of Increasing the Number of Records on DAI and BBI based Imputation Algorithms

tant, we used seven benchmarks datasets from the UCI repository [30] for accuracy comparisons. The latter are summarized in Table II; specifically, we show different characteristics of each dataset such as the number of records, attributes, and classes. We consider all the attributes as numerical attributes. In the case of symbolic attributes, we convert then into numeric format during the preprocessing stage.

TABLE II. UCI DATASETS DESCRIPTION

Dataset	No. records	No. attributes	No. classes
Glass Identification	214	10	7
German Credit Data	1000	20	2
Indian Liver Patient (ILP)	583	10	2
Mammographics	961	6	2
Pima Indians Diabetes	768	8	2
User Knowledge Modeling (UKM)	403	5	4
Wisconsin	699	10	2

As we stated above, all imputation methods are heuristics. Thus, the end goal of DAI and BBI methods is not to produce the same results as the full-scan methods, but to support the data mining process. With this observation, we evaluate different imputation algorithms with respect to the final results from a specific classification method. Particularly, a rule-based classification algorithm (i.e., the CPAR classifier) is used [11]. This algorithm depends on generating a set of predicative association rules from labeled records in a given dataset to classify unlabeled records in the same dataset. Comparing to other association rules-based classifiers, CPAR algorithm has several advantages such as: avoiding redundant rule generation by using dynamic programming, generating a small set of association rules, and achieving higher accuracy [28]. Because our main goal is the imputation process, we only use a single classification algorithm to compare the accuracy between different proposed imputation process.

Imputation using different methods is applied on datasets prior to classification, and subsequently, the classifier obtained is evaluated using a popular metric, which is the *Wilson's Noise Ratio* [31]. This metric identifies the noise rate in a given dataset, i.e., the ratio of records identified as *noisy* to the total number of records. In turn, a record is considered noisy if labelling it using the KNN algorithm (assigning it the most common label among the k-nearest records) leads to an incorrect label. An effective Imputation algorithm should

impute values that reduce the *Wilson's Noise Ratio*. In this experiment, we compare the *Wilson's Noise Ratio* for imputed datasets, after applying the FSI, DAI, and BBI methods, respectively. We used the standard K-Fold Cross-Validation technique with the CPAR algorithm to classify the given dataset records, i.e., the complete dataset after the missing value imputation operation is divided into K parts. Each part is used to train the CPAR model to predict the label for the other $K-1$ parts records. Each record is labeled with the common label predicated using the other nine parts.

We set up different parameters that are required by the CPAR algorithm based on the default values given in an earlier study [28] (i.e., totalweight=0.05, decayfactor=2/3, and min_gain=0.7). For the Wilson's Noise Ratio, we used $k = 5$. Comparing using Wilson's Noise Ratio, we can estimate the impact of our proposed approaches on the results of the CPAR classification algorithm.

A comparison between the brute-force approach and proposed imputation approaches (i.e., DAI and BBI) are shown by Table III with and without the binning strategy. We used seven datasets from the UCI repository for this purpose. For the seven datasets considered in our study, the noise ratio values for FSI, DAI, and BBI are almost the same for different imputation algorithms. This observation shows that using bitmaps, the imputed values lead to the same quality classifier as the brute-force FSI approach. We have also reported the noise ratio when imputation is not applied. As the Table III shows, for six of the seven datasets, very significant reduction in noise ratio is achieved by applying imputation methods. Overall, we can see that DAI and BBI methods not only improve the efficiency of imputation, but help improve the quality of the data mining process after imputation.

TABLE III. COMPARING IMPUTATION METHODS BASED ON WILSON'S NOISE RATIO

Dataset	Before Imput. (%)	Imput. Alg.	FSI (%)	DAI (%)		BBI (%)	
				W/O Bin.	W Bin.	W/O Bin.	W Bin.
Glass Identification	67.28	GCF	11.76	11.76	-	-	-
		MCV	11.76	-	-	11.76	-
		CMC	11.76	11.76	11.76	11.76	-
		CMAV	11.76	11.76	11.76	11.76	11.76
German Credit Data	70.0	GCF	10.88	10.88	-	-	-
		MCV	11.17	-	-	11.17	-
		CMC	10.88	10.88	10.88	10.88	-
		CMAV	10.88	10.88	10.88	10.88	10.88
Indian Liver Patient	28.64	GCF	37.69	37.69	-	-	-
		MCV	19.89	-	-	19.89	-
		CMC	19.89	19.89	19.89	19.89	-
		CMAV	19.89	19.89	19.89	19.89	19.89
Mammographics	53.69	GCF	51.20	51.20	-	-	-
		MCV	52.36	-	-	52.36	-
		CMC	52.26	52.26	49.63	52.78	-
		CMAV	52.78	52.78	52.78	52.78	52.78
Pima Indians Diabetes	34.89	GCF	9.23	9.23	-	-	-
		MCV	9.23	-	-	9.23	-
		CMC	9.23	9.23	9.23	9.23	-
		CMAV	9.23	9.23	9.23	9.23	9.23
User Knowledge Modeling	67.7	GCF	10.48	10.48	-	-	-
		MCV	7.26	-	-	7.26	-
		CMC	7.26	7.26	8.87	7.26	-
		CMAV	7.26	7.26	7.26	7.26	7.26
Wisconsin	34.48	GCF	11.76	11.76	-	-	-
		MCV	12.19	-	-	12.19	-
		CMC	11.76	11.76	11.76	11.76	-
		CMAV	11.76	11.76	11.76	11.76	11.76

V. Related Work

This section gives a brief overview of related work in missing values imputation and using bitmap to index incomplete datasets.

Several approaches have been proposed to avoid the negative impact of missing values in a dataset while performing data mining. The treatment involves estimating missing values, or building a different data view that can be used instead of the incomplete dataset [32]. Our work falls in the categoy of missing value imputation. Many existing approaches involve a direct imputation operation from the complete rows of the same dataset to estimate the missing values. Besides the methods our work builds on, i.e, the Most Common Attribute Value [13] and Global Closest Fit [10], other approaches are based on k-nearest neighbors [33] and k-means clustering [34]. Maximum likelihood approach is also a popular approach to estimate the missing values. In this approach, a statistical model is built on a given incomplete dataset, using a machine learning approach, and model parameters are estimated. For example, the Expectation-Maximization (EM) algorithm has been used to estimate the missing values [35]–[37]. Similarly, neural networks also can be used to build a data model to fill the missing values given incomplete dataset [36].

A somewhat different problem, improving query execution on incomplete datasets by modifying indexing methods, has also been studied [23] [38] [39]. One of these efforts is based on bitmap indexing [23]. The goal of our work is quite different, as we estimate missing values, in preparation for further analysis and mining of the dataset.

VI. Conclusion

Datasets arising in real applications tend to have many uncertainty or incompleteness issues, one of which is the problem of missing values. There are many existing methods for choosing a likely value for a missing value, to be able to support other queries or analysis on the dataset. However, a common theme in all of the work in this area is that algorithms for imputing missing value cannot scale to large datasets. This paper has addressed this shortcoming of the existing work. Particularly, we have shown how bitmaps can be used to accelerate missing value imputation. We have presented two approaches, including one in which bitmap indices can help reduce the number of records to be retrieved, and the second where only bitmaps are used to select a value to replace the missing value.

Through extensive evaluation, we have shown large performance improvements. Even after including index generation time, our methods are faster. However, if an index has already been built to support other functionality (such as supporting query processing), the improvements from our methods are very large. On other hand, the result of the data mining process stays approximately the same as with the original methods.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, 2008, pp. 107–113.

[2] L. L. Liu Peng, "A review of missing data treatment methods," International journal of intelligent information systems and Tech, 2005, pp. 412–419.

[3] K. J. Cios and L. A. Kurgan, "Trends in data mining and knowledge discovery," in Advanced techniques in knowledge discovery and data mining. Springer, 2005, pp. 1–26.

[4] D. F. Heitjan and S. Basu, "Distinguishing missing at random and missing completely at random," The American Statistician, vol. 50, no. 3, 1996, pp. 207–213.

[5] P. Clark and T. Niblett, "The cn2 induction algorithm," Machine learning, vol. 3, no. 4, 1989, pp. 261–283.

[6] I. Kononenko, I. Bratko, and E. Roskar, "Experiments in automatic learning of medical diagnostic rules," in International School for the Synthesis of Experts Knowledge Workshop, Bled, Slovenia, 1984.

[7] J. W. Grzymala-Busse, "On the unknown attribute values in learning from examples," in Methodologies for intelligent systems. Springer, 1991, pp. 368–377.

[8] J. W. Grzymala-Busse and M. Hu, "A comparison of several approaches to missing attribute values in data mining," in Rough sets and current trends in computing. Springer, 2001, pp. 378–385.

[9] S. Aslan, C. Yozgatlıgil, C. İyigün, İ. Batmaz, M. Türkeş, and H. Tatlı, "Comparison of missing value imputation methods for turkish monthly total precipitation data," in 9th International Conference on Computer Data Analysis and Modeling: Complex Stochastic Data and Systems, Minsk, Belarus, 2010, pp. 7–11.

 ISBN: 978-1-61208-408-4

[10] S. Gaur and M. Dulawat, “A closest fit approach to missing attribute values in data mining,” International Journal of advances in Science and Technology, vol. 2, no. 4, 2011, pp. 18–24.
[11] J. Luengo, S. García, and F. Herrera, “On the choice of the best imputation methods for missing values considering three groups of classification methods,” Knowledge and information systems, vol. 32, no. 1, 2012, pp. 77–108.
[12] L. Wohlrab and J. Fürnkranz, “A review and comparison of strategies for handling missing values in separate-and-conquer rule learning,” Journal of Intelligent Information Systems, vol. 36, no. 1, 2011, pp. 73–98.
[13] J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, and X. Zheng, “Handling missing attribute values in preterm birth data sets,” in Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. Springer, 2005, pp. 342–351.
[14] P. O’Neil and D. Quass, “Improved query performance with variant indexes,” in ACM Sigmod Record, vol. 26. ACM, 1997, pp. 38–49.
[15] K. Wu, W. Koegler, J. Chen, and A. Shoshani, “Using bitmap index for interactive exploration of large datasets,” in Scientific and Statistical Database Management, 2003. 15th International Conference on. IEEE, 2003, pp. 65–74.
[16] Y. Wang, Y. Su, A. Gagan, and T. Liu, “SciSD: Novel Subgroup Discovery Over Scientific Datasets Using Bitmap Indices,” Technical Report OSU-CISRC-3/15-TR03, Ohio State University, Tech. Rep., 2015.
[17] N. Koudas, “Space efficient bitmap indexing,” in Proceedings of the ninth international conference on Information and knowledge management. ACM, 2000, pp. 194–201.
[18] G. Antoshenkov, “Byte-aligned bitmap compression,” in Data Compression Conference, 1995. DCC’95. Proceedings. IEEE, 1995, p. 476.
[19] K. S. Kumar, M. Laxmaiah, and C. S. Kumar, “A compacted bitmap vector technique to evaluate iceberg queries efficiently,” International Journal, vol. 3, no. 6, 2013, pp. 412–418.
[20] S. J. van Schaik and O. de Moor, “A memory efficient reachability data structure through bit vector compression,” in Proceedings of the 2011 international conference on Management of data. ACM, 2011, pp. 913–924.
[21] F. Deliège and T. B. Pedersen, “Position list word aligned hybrid: optimizing space and performance for compressed bitmaps,” in Proceedings of the 13th International Conference on Extending Database Technology. ACM, 2010, pp. 228–239.
[22] Y. Su, Y. Wang, and G. Agrawal, “In-situ bitmaps generation and efficient data analysis based on bitmaps,” in HPDC. ACM, 2015.
[23] G. Canahuate, M. Gibas, and H. Ferhatosmanoglu, “Indexing incomplete databases,” in Advances in Database Technology-EDBT 2006. Springer, 2006, pp. 884–901.
[24] “Bitmap index vs. b-tree index: Which and when?” http://www.oracle.com/technetwork/articles/sharma-indexes-093638.html, [Online; accessed 1-January-2015].
[25] J. Chou, K. Wu, O. Rubel, M. Howison, J. Qiang, B. Austin, E. W. Bethel, R. D. Ryne, A. Shoshani et al., “Parallel index and query for large scale data analysis,” in High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for. IEEE, 2011, pp. 1–11.
[26] B. He, H.-I. Hsiao, Z. Liu, Y. Huang, and Y. Chen, “Efficient iceberg query evaluation using compressed bitmap index,” Knowledge and Data Engineering, IEEE Transactions on, vol. 24, no. 9, 2012, pp. 1570–1583.
[27] Y. Su, G. Agrawal, and J. Woodring, “Indexing and parallel query processing support for visualizing climate datasets,” in Parallel Processing (ICPP), 2012 41st International Conference on. IEEE, 2012, pp. 249–258.
[28] X. Yin and J. Han, “Cpar: Classification based on predictive association rules.” in SDM, vol. 3. SIAM, 2003, pp. 331–335.
[29] “Berkeley earth,” http://www.berkeleyearth.org, [Online; accessed 12-January-2015].
[30] C. Blake and C. J. Merz, “{UCI} repository of machine learning databases,” http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998, [Online; accessed 12-January-2015].
[31] D. L. Wilson, “Asymptotic properties of nearest neighbor rules using edited data,” Systems, Man and Cybernetics, IEEE Transactions on, no. 3, 1972, pp. 408–421.
[32] S. Parthasarathy and C. C. Aggarwal, “On the use of conceptual reconstruction for mining massively incomplete data sets,” Knowledge and Data Engineering, IEEE Transactions on, vol. 15, no. 6, 2003, pp. 1512–1521.
[33] G. E. Batista and M. C. Monard, “An analysis of four missing data treatment methods for supervised learning,” Applied Artificial Intelligence, vol. 17, no. 5-6, 2003, pp. 519–533.
[34] D. Li, J. Deogun, W. Spaulding, and B. Shuart, “Towards missing data imputation: A study of fuzzy k-means clustering method,” in Rough Sets and Current Trends in Computing. Springer, 2004, pp. 573–579.
[35] A. P. Dempster, N. M. Laird, D. B. Rubin et al., “Maximum likelihood from incomplete data via the em algorithm,” Journal of the Royal Statistical Society, vol. 39, no. 1, 1977, pp. 1–38.
[36] Z. Ghahramani and M. I. Jordan, “Learning from incomplete data,” Technical Report AI Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab, August 1995.
[37] W. R. Harvey, User’s guide for LSML76: Mixed model least-squares and maximum likelihood computer program. Ohio State University, 1977.
[38] B. C. Ooi, C. H. Goh, and K.-L. Tan, “Fast high-dimensional data search in incomplete databases.” in VLDB, vol. 98, 1998, pp. 357–367.
[39] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, “Skyline query processing for incomplete data,” in Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE, 2008, pp. 556–565.

 ISBN: 978-1-61208-408-4

Capturing the Structure of Internet of Things Systems with Graph Databases

for Open Bidirectional Multiscale Data Mediation

Dana Popovici, Gilles Privat
Orange Labs
Grenoble, France
email:dana.popovici@gmail.com, gilles.privat@orange.com

***Abstract*—The deep structure of Internet of Things (IoT) environments understood as complex Cyber-Physical Systems (CPS) is made up of all interwoven relationships of physical actuation, sensing, proximity, grouping and containment between their constituent subsystems. We describe and assess three solutions for capturing and exposing this structure as a persistent graph of matching links between the informational proxies that represent these subsystems. A graph database may provide an access-efficient persistence support for this graph, tightly coupled with the mediation platform. Alternatively, the Resource Description Framework (RDF) may be used directly as a standard, open and extensible means to represent this graph and its associated semantics, with triplestores as a persistent repository supporting queries with standard languages such as SPARQL. A third solution would be, if fine-grain hyperlinked REST interfaces are provided to subsystems or their states viewed as resources, to delegate the creation and maintenance of this graph to an external web crawler and search engine.**

Keywords*—*Internet of Things(IoT); Cyber-Physical Systems (CPS); Sensor-Actuator Networks; Ontologies; Linked Open Data; Resource-Oriented Architectures(ROA); Resource Description Framework (RDF)

I. INTRODUCTION

The Internet of Things (IoT) may, in its extended acceptation [1], reach beyond connected devices to encompass all kinds of physical entities, be they legacy appliances, passive items or subsets of space. These entities get identified and represented together with the attached devices through Internet of Things platforms [2] and may as such be the target of varied applications operating on top of these platforms. The information maintained by these platforms may comprise both real-time information about the state of the identified entities and devices as well as structural semi-static information about the relationships between these. We focus here on the solutions for the representation and management of the graph made up of all these relationships, which may correspond to the following:

- device used as primary sensor for an entity
- device used as secondary sensor for an entity
- device used as actuator for an entity
- device acting upon an entity as a side effect
- entity containing another
- entity adjacent to another
- device connected to another through the network

This clearly goes much beyond networked device management, to get closer to the structural representation of Cyber-Physical Systems (CPS) decomposed as interacting relevant subsystems. Typical examples of these IoT environments modeled as Cyber-Physical Systems could be smart homes, smart buildings and smart cities. We are interested in these rather than in more traditional one-of-a-kind industrial CPS because they stand to gain the most from the use of shared platforms instead of dedicated and vertically integrated design. We can take advantage of the generic character of the categories of entities/subsystems that make up these systems and of their invariance from one of their instances to another. All buildings are thus made up of rooms, corridors, floors, appliances and furniture items pertaining to broadly similar categories, while cities comprise streets, crossings, blocks, parking places, etc. These categories may be drawn from shared domain-specific ontologies and the entities we target as nodes of our extended IoT graph will be instances of these classes, providing a semantic reference for their eventual identification.

We present an architecture template for an IoT platform in the smart home, building or city domains in section II. We explain how the graph of relationships between entities can be the core of this platform in section III. We assess comparatively the two database solutions we propose, graph data base and RDF triplestore, in sections IV and V respectively and explain how they get interfaced to the platform in section VI. Section VII presents the alternative to having just a RESTful interface to the platform provided as a complete set of hyperlinked resources and delegating the capture of the corresponding graph to an external web crawler.

II. ARCHITECTURE OF AN INTERNET OF THINGS PLATFORM

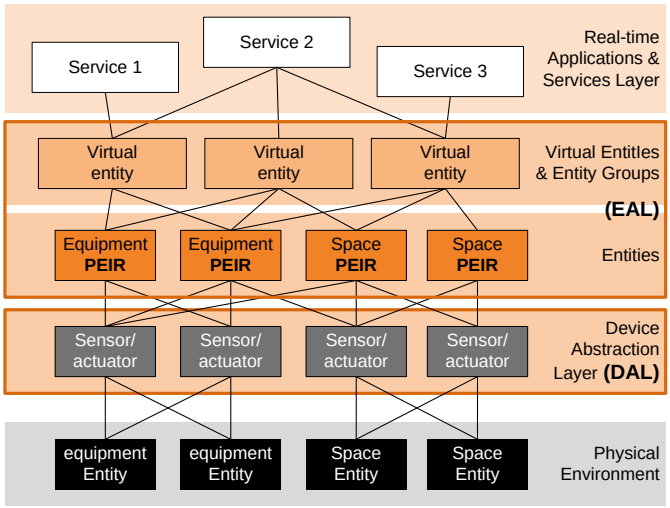


Figure 1. IoT platform architecture

An IoT platform mediates data between the target physical environment and applications, both upwards by fusing /aggregating/abstracting sensor data, and downwards by passing on commands to actuators[2][3]. The architecture of such an IoT platform uses, in our approach, software modules called “Physical Entity Informational Representatives” (PEIR) that serve as proxy for the individual entities, making up an Entity Abstraction Layer (EAL). This may comprise additional

 ISBN: 978-1-61208-408-4

levels of abstractions when these entities get regrouped as "virtual entities" according to functional or physical criteria (Figure 1). Beneath this, the Device Abstraction Layer (DAL) provides a uniform interface to networked devices (sensors & actuators) that serve as physical intermediaries to the entities, regardless of protocols and technologies. DAL and EAL each have their own REST interface for a full decoupling of the two.

III. CAPTURING THE DEVICES & SUSBSYSTEMS INSTANCE GRAPH

Our approach does partially disregard the heterogeneity and actual complication of individual entities (subsystems) by mapping them to simple or even simplistic generic models drawn from domain ontologies as proposed above. This makes it possible to focus on the more relevant complexity of the overall system, as it results from the composition of these individual subsystems and, crucially, the web of relationships of different kinds into which they are caught.

We will use throughout this paper the example of a smart building, renting office space to a number of companies as a prototype IoT environment. Each floor may comprise similar spaces and pieces of equipment, but there is a multitude of relationships to account for between these physical entities and other relevant virtual entities (e.g., sets of offices rented by the same company). Moreover, certain devices or entities can be used for purposes other than their primary function, generating contextual information through indirect relationships, for example a computer acting as presence sensor (if someone is typing, the office is occupied). Figure 2 shows a small subset of the different types of relationships captured by the graph representing our smart building example as an IoT system, showing parts of its three interconnected sub-graphs: the nodes of which are respectively ontology categories, physical entities and devices. The ontologies are imported from online repositories, but a local copy is needed on the IoT platform for efficient access. The external ontologies being referenced may be upper ontologies, domain ontologies (e.g., for smart home/building/city), and transversal device ontologies. The links between nodes are also of diverse types, representing either semantic relationships (e.g., instanceOf, subClassOf) or structural (e.g., isOn, actuated by) relationships. The resulting overall graph is thus heterogeneous and unites sub-graphs with different types of nodes, linked by different types of arcs.

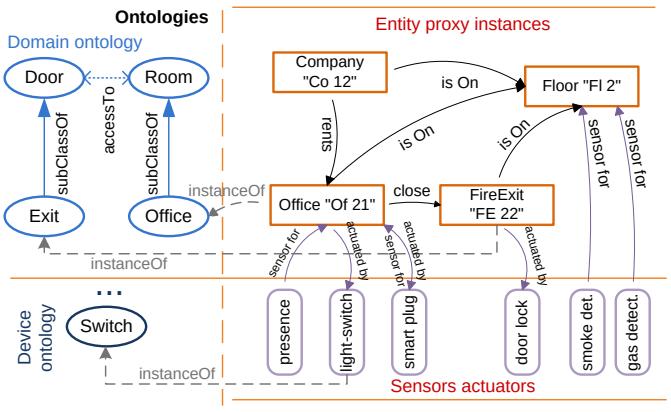


Figure 2. Example smart building graph

Contrary to the closed & fixed systems, which are the usual target of embedded systems design, these relationships and the systems configuration they capture will vary over time, as new entities will appear, move or be moved.

To give just a tiny example of the use of this graph by a very simple Smart Building IoT application, we could imagine a situation where an alert from a gas detector triggers turning off electrical equipment that belongs to a given company in the corresponding area. A query could be used to find the smart plugs that should be turned off for the company "Co 12". Expressed in a SPARQL-like language, it would be:

```
SELECT ?smart_plug WHERE
{?smart_plug location:adjacent ?gas_detect;
?gas_detect state:hasState Alert;
?smart_plug co:belongs "Co 12".}
```

IV. GRAPH DATABASE REPRESENTATION

The first solution that we investigate for capturing the structure of the IoT system through the relationships between physical entities that it represents is a graph database. This type of database focuses on the software objects that are the nodes of the graph and provides optimized algorithms for the traversal of the graph, with the benefit that nodes connected through directed paths are fast and easy to retrieve and unrelated nodes are not traversed. Property graphs provide rich information about both nodes and relationships, through key-value-pairs that describe them. A solution such as Neo4j, an open-source graph database implemented in Java, offers both an embedded and a server version. Neo4j queries are expressed in Cypher, an SQL-inspired, declarative language that describes patterns on the graph.

A. *Integrating database engine into the IoT platform*

A graph database seems well suited for the task of capturing the complex connections between the IoT sub-systems and entities. Based on the target platform there are several aspects to consider, among them the scale and latency requirements. Our approach includes two abstraction layers that are not necessarily hosted on the same machine, resulting in the distribution of nodes between these two platforms. The scale of the graph may vary widely, from smart homes with a few hundreds of nodes, up to smart buildings and smart cities. Even for the smaller scale of a smart home, it would still be possible to have two different platforms, typically a HomeLAN gateway and a dedicated home automation server. Tight coupling between the core of IoT platform (the Entity Abstraction Layer made up of all PEIRs) and the database (with nodes matching the PEIRs), could be achieved by hosting both on the same platform.

The database is built and updated from two sources, the IoT platform, during configuration and reconfiguration phases, and the domain ontologies stored in online repositories. Although graph databases can represent semantic information, this requires another step to import them from the OWL format with the OWL API.

B. *Advantages and disadvantages of a graph database*

The most important advantage of the graph database solution is, for our purposes, besides their query performance and scalability, their potential tight integration with the IoT platform, if both could share the same Java execution environment as proposed before. The disadvantage of using such a database is the added effort of its integration to the platform, both for representing structural relationships when they are discovered and for semantic relationships that need to be transformed and included from ontology repositories. Reasoning from the graph (e.g., inferences drawn from combinations of structural and semantic relationships) is not

 ISBN: 978-1-61208-408-4

"native" to graph databases such as Neo4j, requiring one more step of data transformation to RDF triples and the use of a reasoner such as Pellet.

V. RDF REPRESENTATION & TRIPLESTORE-BASED PERSISTENCE

Representing the relationships of entities in the IoT platform through RDF and storing the information in triplestores (the corresponding "native" solution) could be considered as the "opposite" solution of a graph database. RDF natively represents semantics and needs to be extended to include structural information whereas the graph database does the contrary. RDF is a W3C standard and represents relations (statements) as triples: subject, predicate, object. RDF databases are a prominent standards-based NoSQL solution. This representation is the basis of Linked Open Data, offering a means to publish structured and semantically annotated data that supports database-style queries. It is also one of the data models used for the representation of OWL-based ontologies. All these advantages can be used to our benefit, as our approach includes importing ontologies and sharing the generated data.

A. Exporting the database as RDF graph

Depending on the target system, the data generated and stored in the triplestore should be made available to the general public or to a subset of authorized stakeholders. Some of the information could be useful if exposed directly as Linked Open Data. For example, in a smart city, car parks with their location and number of available parking places could be shared. A smart building might have a public list of companies that are currently renting offices, as well as the number of available offices for rent. It is especially appropriate to share data for the smart city, but even some of the data from the smart home might be of interest to the general public. For example, temperature information or luminosity, as measured by outside weather stations can be shared and used to compute mean values for a city or area. It should be noted though that in the smart home and smart building domains the access to the RDF graph should be tightly restricted, as most of the information it contains is strictly private.

B. Query languages, rule specification & reasoning tools

Representing the system graph as RDF triples can profit from the existing tools to better exploit the information. Powerful open source triplestores exist today, such as Jena, Sesame, Virtuoso, Bigdata and many others. Most of them support SPARQL, a standardized and interoperable query language that is fairly simple to use. Some of the frameworks provide more precise functions, including reasoning tools for the stored data, and may even be built for that purpose, like Ontotext GraphDB, an OWL-based triplestore.

In this paper, we use the example of Jena, one of the most popular triplestores. Although its native persistence implementations are not the best ones from a scaling point of view, Jena can be used to interface several other triplestores. It also provides good inference support through several reasoners and their rules. It includes a generic rule reasoner, as well as an RDFS, OWL and a transitive reasoner for different levels of inference. Other advantages include the SPARQL server that is provided out of the box (Fuseki for Jena), allowing distant access to the graph.

These tools are used to enrich the knowledge stored in the graph, and help support smart environment-specific requirements. In a smart environment, the applications target not only devices and physical entities, but also entity groups and virtual entities. Some of the groups are static or semi-static, requiring them to be represented in the graph at all times (e.g. the group of offices being rented by a company). Other groups might be created in an ad-hoc fashion when they are needed, by querying the RDF database. This, plus the reasoning tools available, opens the IoT platform to unlimited possibilities.

C. Advantages and disadvantages of RDF databases

When compared to other NoSQL databases, RDF triplestores have some important advantages. Data is represented by a simple, uniform, standard model, which allows for portability and interoperability. This also means that the inner graph representation is vendor-independent, and it can easily be imported to another database. Having a high-level declarative query language is another major advantage.

Compared to graph databases, there are several differences. RDF can be considered as a graph, but is relation-centric (as opposed to node-centric) and is composed solely of labeled arcs. Representing undirected edges would require coupling two arcs in opposite directions between the corresponding nodes. Another issue concerns literal properties that are objects in the RDF representation, causing the resulting graph to have "leaf nodes".

Alternatively to "native" RDF triplestores, the persistent storage of RDF graphs may be provided through multiple methods, including tuple stores, graph databases and even traditional SQL-based databases. Graph databases such as Neo4j store data directly as a graph and thus benefit from optimized traversal algorithms. In general, native RDF databases offer slower performance than other solutions (especially graph databases) and they scale badly, making them a solution that is not perfectly adapted to applications with harder requirements.

VI. INTEGRATING THE DATABASES WITH THE IOT PLATFORM

We have presented two solutions for the representation of the relationships between entities of an IoT platform and compared their relative performance. In this section, we wish to address some of the issues that are shared between them and can be discussed jointly. Implementations of both RDF and graph databases can be either tightly or loosely coupled with the IoT platform. We use as example two representing databases, Jena for triplestores and Neo4j for graph database. Both offer Java APIs that can be used through direct invocations and that imply the database should be on the same machine as the rest of the IoT platform. This raises several questions about possible implementation solutions. As explained in the previous sections, we consider at least two separate abstraction layers, which might be on different machines, one for the connected devices and another for the entity representation. It seems more appropriate in this case to host the database on the same machine as the entity abstraction layer. In this way, the database can still store information coming from the device abstraction layer, but it will be more tightly coupled with the entities. Given that IoT applications target the entity level, the database enriches the quality of the applications through semantic and structural information. This solution might work very well in a smart home IoT system, but less so in a smart city, where the abstraction layers are physically distributed.

The second option, once again available for both Jena and Neo4j, is to use servers and a RESTful interfaces over HTTP.

 ISBN: 978-1-61208-408-4

Jena has the Fuseki server that can be used for SPARQL queries and Neo4j is first and foremost a server. This option prompts another choice about the location of the server with regard to the abstraction layers: should we use a regular cloud-based solution, or use "edge of cloud" ("fog") computing [4] which extends cloud computing to embedded platforms on the outer edges of the network. Fog computing combines the benefits of centralized data processing and proximity to the end-user, enabling flexible and virtualizable platform support for real-time applications of the IoT. Regarding the application domains that we consider, it seems that fog/edge of cloud solutions could provide a good tradeoff while being equally adapted for different types of systems: smart home, smart building or smart city.

VII. EXPORTING THE GRAPH FROM HYPERLINKED ROA INTERFACES

Both of the previously presented solutions imply an effort to create and maintain the structural relationship graph for the IoT system within the platform and keep it consistent with the internal operational representation at all times, thus placing a heavy burden on the software design of the platform. A third option is more in line with a full "web of things" [5] approach. Assuming we provide a full REST [6] interface to the platform, which means each entity of the Entity Abstraction Layer, each state of these entities and each device of the Device Abstraction Layer would be a resource in the REST sense, each with its own URI and exposed hyperlinks to other resources it interacts with, we could dispense with creating and maintaining a database of the corresponding graph of hyperlinks inside the IoT platform and delegate it to another platform or third party general-purpose web tools. This is what search engines have done for the original web of documents: mapping it as a graph and exploiting the structural properties of this graph for information retrieval. This would also fit better with a preference for of a minimal and loosely-coupled IoT platform.

In a "pure" ROA interface, (corresponding to the third level of the Richardson Maturity Model, "HATEOAS" [6]), no global functional description is required, all resources are self-descriptive and provide their own URI that can be interpreted by applications without requiring any "out-of-band" information or prior knowledge, with their behavior, semantic mappings, associated resources and sub-resources accessible through hyperlinks. In our architecture, these hyperlinks correspond to the relationships between entity groups, entities and devices as maintained by the graph database proposed in the previous two approaches, with additional links corresponding to the allowed state transitions between states viewed as sub-resources. Non-functional and semantic information is attached to each entity as read-only resources, accessible together with entity states through HTTP GET, while the controllable states can be updated through corresponding HTTP PUT. In this approach, the graph representation of the system does exist, but only implicitly through these hyperlinks, just as is the case in the web viewed as a graph. All the information that may be required from applications is in principle available by traversing the graph made up of these hyperlinks, providing the equivalent of interface introspection, discovery and dynamic service composition from more traditional SOA approaches. Just as the public web itself, this minimal web of things platform requires external or third party tools to provide the equivalent of the functionalities that are natively provided by the databases provided in the previous two approaches, especially responding to database-like queries. The graph structure could thus be extracted by using a crawler tool similar to web crawlers that would systematically and exhaustively traverse the graph of REST hyperlinks to recreate a fully indexed, searchable data structure, possibly using a graph database of its own. This database could in turn respond to queries such as the one mentioned in section III (possibly with restrictions on the primitives involved), providing direct links to the target entities for applications to monitor or control through the corresponding resources addressed individually through their REST-compliant URIs, thus making it possible for these applications to bypass the database when direct control will be involved.

VIII. CONCLUSION

Most present-day Internet of Things applications are limited to monitoring and data collection. If they involve control, it is usually not part of their automatic operation and occurs through human operators. The architecture we propose here is clearly designed to go beyond these in order to support bidirectional data mediation for applications that involve direct real-time automatic control of the same entities being monitored. With this in view, the proposed graph representation is essential in determining the perimeter of entities that may be impacted by a given control action, to avoid undesirable or potentially cascading and catastrophic side effects of any action, which could be done by just tracing directed paths of relevant actuation relationships through this graph. More fundamentally, a platform based on such a representation is intended to be a Cyber Physical System platform, not only an IoT platform, and the proposed graph representation should be understood as representing this CPS as a complex system, in keeping with received graph-based modeling approaches for such systems where node (subsystem) models may be grossly simplified as long as the complexity and structural accuracy of their interrelationships and of their physical grounding is fully accounted for.

ACKNOWLEDGMENTS

We are most grateful to Mengxuan Zhao and Laurent Lemke, who have contributed to shape and evolve the infrastructure on which this work is based, to Didier Donsez and Florence Maraninchi, for many in depth discussions around the issues presented here, and to Wenbin Li who has taken on the challenge to follow up on this work.

REFERENCES

[1] G. Privat, "Extending the Internet of Things", Communications & Strategies, Digiworld Economic Journal , vol 87, 2012, pp101-119

[2] G. Privat, M. Zhao, and L. Lemke, "Towards a Shared Software Infrastructure for Smart Homes, Smart Buildings and Smart Cities", International Workshop on Emerging Trends in the Engineering of Cyber-Physical Systems, Berlin, April 14, 2014

[3] Z. Hu, G. Privat, S. Frenot and B. Tourancheau, "Self-configuration of Home Abstraction Layer via Sensor-Actuator Network", Ambient Intelligence. Springer Berlin Heidelberg, 2011, . pp146-150.

[4] F Bonomi, R Milito, J Zhu and S Addepalli, "Fog computing and its role in the internet of things", Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012. p. 13-16

[5] D. Guinard, V. Trifa and E. Wilde, "A resource-oriented architecture for the web of things", Internet of Things (IOT),. IEEE, 2010. p. 1-8. .

[6] L. Richardson and S. Ruby, "RESTful web services" O'Reilly Media, 2008.

 ISBN: 978-1-61208-408-4

Towards Implementing Semantic Literature-Based Discovery with a Graph Database

Dimitar Hristovski*, Andrej Kastrin†, Dejan Dinevski‡ and Thomas C. Rindflesch§
*Faculty of Medicine, University of Ljubljana, Ljubljana, Slovenia
†Faculty of Information Studies, Novo mesto, Slovenia
‡Faculty of Medicine, University of Maribor, Maribor, Slovenia
§National Library of Medicine, Bethesda, MD, USA

***Abstract*—Literature-based discovery (LBD) combines known facts in the scientific literature to generate discoveries, or hypotheses. Potential discoveries have the form of relations between concepts; for example, in the biomedical domain (on which we concentrate), a drug may be determined to treat a disease other than the one for which it was intended. We view the domain knowledge underpinning LBD as a network consisting of a set of concepts along with the relations connecting them. In the study presented here, we used SemMedDB, a database of semantic relations between biomedical concepts extracted with SemRep from MEDLINE. SemMedDB is distributed as a MySQL relational database, which is not optimal for dealing with network data. We transformed and uploaded SemMedDB into a Neo4j graph database, and implemented the basic LBD discovery algorithms with the Cypher query language. We conclude that storing the data needed for semantic LBD is facilitated by a graph database. Also, implementing LBD discovery algorithms is conceptually simpler with a graph query language when compared with standard SQL.**

Keywords–Data science; Databases; Data mining; Semantics; Literature-based discovery.

I. INTRODUCTION

The corpus of biomedical papers in online bibliographical repositories, nowadays referred to as the bibliome, is of considerable size and complexity. The amount of biomedical literature available is growing at an explosive speed, but a large amount of useful information in it remains undiscovered [1]. For example, MEDLINE contains over 24 million references to biomedical journals, with approximately 3000 references added each day. Exploiting this information effectively crucially depends on linking information from diverse sources into coherently interpretable knowledge. In this regard, development of automated knowledge discovery tools is of utmost importance. Computer-based methods can greatly complement manual literature management and knowledge discovery from biomedical data [2].

Literature-based discovery (LBD) is a mature text mining methodology for automatically generating hypotheses for scientific research by uncovering hidden, previously unknown relationships, from existing knowledge. The LBD methodology was pioneered by Swanson [3], who proposed that dietary fish oils might be used to treat Raynaud's disease because they lower blood viscosity, reduce platelet aggregation, and inhibit vascular reactivity. Swanson's hypothesis was validated by DiGiacomo et al. [4]. Swanson's approach is based on the assumption that there exist two nonintersecting scientific domains. Knowledge in one domain may be related to knowledge in the other, without the relationship being known. The methodology of LBD relies on the idea of concepts relevant to three literature domains: X, Y, and Z (Figure 1). For example, suppose a researcher has found a relationship between disease X and a gene Y. Further suppose that a different researcher has studied the effect of substance Z on gene Y. The use of LBD may suggest an XZ relationship, indicating that substance Z may potentially treat disease X. Many researchers have replicated Swanson's discoveries using various approaches: Gordon and Lindsay [5], [6], Webber et al. [7], Hristovski et al. [8], Srinivasan [9], Cameron et al. [10]. For a recent review of LBD tools and approaches, see Hristovski et al. [11].

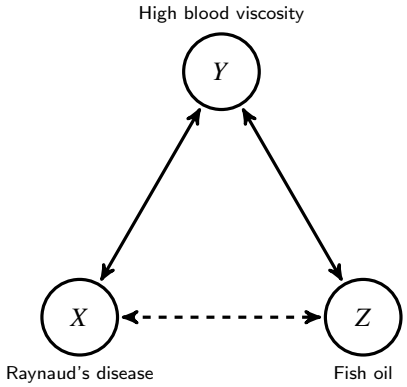


Figure 1. Swanson's XYZ discovery model

Current knowledge of a particular biomedical domain can be viewed as a set of concepts and the relationships among them [12]. For example, in pharmacogenomics relations among genes, diseases, and chemical substances constitute an important part of knowledge. These associations can be naturally represented as a graph consisting of nodes and edges, where the former represent concepts and the later relationships.

The great majority of existing LBD systems are co-occurrence based. Co-occurrence represents the simplest way to capture associations between biomedical concepts (nodes), but it does not express the meaning of the relationship between those concepts. Even widely used document retrieval systems, such as PubMed, typically have no access to the meaning of the text being processed [13]. To fill the gap between raw text and its meaning Rindflesch [14] developed the SemRep system. SemRep is a rule-based, symbolic natural language processing system that recovers semantic propositions from the biomedical research literature. The system relies on domain knowledge in the Unified Medical Language System (UMLS) [15] to provide partial semantic interpretation in the form of predications consisting of UMLS Metathesaurus concepts as arguments and UMLS Semantic Network relations as predicates. SemRep uses a partial semantic analysis based

 ISBN: 978-1-61208-408-4

on the SPECIALIST Lexicon [16] and MedPost tagger [17]. Each noun phrase in this analysis is mapped to a concept in the Metathesaurus using MetaMap [18]. Both syntactic and semantic constraints are employed to identify propositional assertions. For example, SemRep extracts three predications from the text "dexamethasone is a potent inducer of multidrug resistance-associated protein expression in rat hepatocytes":

1) Dexamethasone STIMULATES Multidrug Resistence-Associated Proteins
2) Multidrug Resistance-Associated Proteins PART_OF Rats
3) Hepatocytes PART_OF Rats

SemRep extracts 30 predicate types expressing assertions in clinical medicine (e.g., TREATS, ADMINISTERED_TO), substance interactions (e.g., INTERACTS_WITH, STIMULATES), genetic etiology of disease (e.g., CAUSES, PREDISPOSES), and pharmacogenomics (e.g., AUGMENTS, DISRUPTS). The program has been run on all of MEDLINE and the extracted predications deposited in a MySQL database named SemMedDB [19] updated quarterly and available to researchers.

In the last decade, various NoSQL (often interpreted as Not only SQL (Structured Query Language)) technologies for storing data have emerged. The term NoSQL refers to schemaless database technology such as key-value stores (e.g., Apache Cassandra), document stores (e.g., MongoDB), and graph databases (e.g., AllegroGraph, OpenLink Virtuoso, Neo4j). In this paper, we examine the Neo4j graph database as an alternative for LBD on semantic relations extracted from MEDLINE, because Neo4j is particularly useful for storing data structured as a graph.

II. METHODS

A. Exporting data from SemMedDB

SemMedDB contains detailed information about all the semantic relations extracted with SemRep from MEDLINE. The general form of the semantic relations is Subject-Relation-Object, for example Dopamine-TREATS-Parkinson Disease. This particular relation was extracted 1080 times from different text sentences. We will refer to each of these 1080 extractions as a semantic relation instance. In other words, we say that Dopamin-TREATS-Parkinson Disease is a semantic relation with 1080 instantces. For LBD we were interested in aggregated semantic relations, which means that for each semantic relation we wanted to have the corresponding number of instances. In SemMedDB, there is no single place where we could find aggregated semantic relations. Therefore, we exported only the semantic relation instances in a text CSV (Comma-Separated Values) format. For each instance, the following fields were exported: subject_concept_id, subject_name, relation_type, object_concept_id, and object_name. We exported all the 52 616 158 semantic relation instances.

B. Aggregating the semantic relations and loading into Neo4j

Originally, we thought of aggregating the semantic relations with shell tools, such as Awk, sort and uniq. However, we decided to put Neo4j to a test. We decided to load and aggregate the data in Neo4j with the `LOAD CSV` command, which is used for loading external text data (Figure 2). This command is part of Cypher, which is Neo4js graph query language. The command reads the input file line by line. Each input line is split into fields. After loading, we used the `MERGE` command for the subject part of the instance. If the concept that appeared as subject was not found, then, a node with label "Concept" was created with the corresponding concept_unique_id, name and semantic type, and the frequency counter was set to 1. All this was done with the `ON CREATE` part of the `MERGE` command. However, if there already was a node for the subject concept, then only its frequency counter was increased with the `ON MATCH` part of the `MERGE` command. A similar procedure was repeated for the object concept of the semantic relation instance with another `MERGE` command. And finally, a `MERGE` command was used for the semantic relation itself. If there was no relation between the current subject and object nodes, a new relation was created with the relation type, and the frequency counter was set to 1 within the `ON CREATE` part of the commend. If the relation already existed then its frequency counter was increased with the `ON MATCH` part. After the end of this procedure, we expected to have a graph in which the nodes were the subjects and/or objects of the semantic relations, with the nodes having relations between them that corresponded to the semantic relations between the arguments. This procedure worked well with a few million relation instances. Unfortunatelly, it was not able to load the entire set of semantic relation instances, despite the fact that we used the `USING PERIODIC COMMIT` command as instructed in the Neo4j documentation.

We then used a few other approaches. For example, loading and aggregating only the nodes as described above worked well and finished successfully. However, we could not load all the relations at once. We then tried loading the relations in batches by adding corresponding `START` and `LIMIT` parameters to the `LOAD CSV` command. In addition to this being too slow, there was another problem with this command. It does not allow setting the label of a node based on a field value in the input line, which we wanted to do. We wanted the semantic type of the nodes to become one of its labels. For example, we wanted to have nodes such as "c:Concept:dsyn" which in Neo4j terminology is read as node "c" with labels "Concept" and "dsyn" (where "dsyn" is our abbreviation for the concept semantic type "Disease or Syndrome").

```
LOAD CSV FROM 'semmed_sub_rel_obj.txt'
  AS line
WITH line
MERGE (c1:Concept {cui: line[0]})
ON CREATE SET c1.name=line[1],
  c1.type=line[2], c1.freq=1
ON MATCH SET c1.freq = c1.freq + 1
MERGE (c2:Concept {cui: line[4]})
ON CREATE SET c2.name=line[5],
  c2.type=line[6], c2.freq=1
ON MATCH SET c2.freq = c2.freq + 1
MERGE (c1)-[r:Relation {type:
  line[3]}]->(c2)
ON CREATE SET r.freq = 1
ON MATCH SET r.freq = r.freq + 1;
```

Figure 2. Loading semantic relation instances with the `LOAD CSV` Neo4j command.

 ISBN: 978-1-61208-408-4

We next tried to aggregate the instances with several Awk scripts and with the sort and join shell commands. During this process, for each semantic relation, all the instances of that semantic relation were replaced with a single text line, which, in addition to the instance level fields, also contained the total number of instances. We noticed that sometimes the same concepts appeared with different semantic types in different instances. All these semantic types were also aggregated as a comma-delimited list. To load the data into Neo4j we used a stand-alone batch-import tool [20]. This tool requires two files as input: a file describing the nodes, and a file describing the relations between the nodes. From the aggregated relation file prepared in the previous step we prepared the required two files with two more Awk scripts. Additionally, for each node, these scripts calculated in the number of relations in which the node (biomedical concept) occurs as an argument (subject or object). That number becomes a node property. The batch import tool for Neo4j then loaded the prepared files into a new database very quickly. It was also able to create labels for the nodes from the list of semantic types for each node. Finally, in Neo4j, we created an index on the node properties "name" and "cui", which is an abbreviation for "concept unique identifier" (in the UMLS).

III. RESULTS

Currently, there are $269\,047$ unique concepts and $14\,150\,952$ distinct relationships between them in our Neo4j graph database. These relationships originated from $52\,616\,158$ SemMedDB semantic relation instances. We illustrate the use of this database with some LBD examples.

LBD is conducted using one of two modes: open and closed discovery. In open discovery, a concept X (e.g., a disease) is used to start, and the task is to find a new discovery Z (e.g., a drug) regarding X through some intermediate concept Y (e.g., a gene associated with X). In closed discovery both the starting concept X and final concept Z are know in advance. The goal is to supply Y an explanation for the relation between X and Z. Closed discovery can be used, for example, to elucidate statistically determined relations between X and Z, which do not have an explanation.

Figure 3 shows the most general LBD implementation with a Cypher query. The relations between the concepts X, Y and Z can be matched regardless of the direction of the relations. We require a relation between X and Y, as well as a relation between Y and Z, but we are interested only in those X and Z concepts that are not already related. If we instantiate X with a specific concept (e.g., Curcumin) and leave Z uninstantiated, then we have an open discovery mode. If we instantiate both the X and Z concepts with particular concepts then we have closed discovery mode.

```
MATCH (x:Concept)--(y:Concept)--
  (z:Concept)
WHERE NOT (x)--(z)
RETURN x, y, z;
```

Figure 3. The most general LBD implementation in Cypher.

UMLS semantic types are often used in LBD for concept filtering or for working with a whole class of concepts. Figure 4 shows how we can use semantic types for node (concept) filtering. We implemented concept semantic types as Neo4j labels. In our current implementation, each concept has the label "Concept" and additionally all its semantic types appear as additional labels. The query in Figure 4 searches for drugs (concepts with semantic type "phsu" (abbreviation for Pharmacologic Substance) which are related to some concepts Y which are related to some diseases (concepts with semantic type "dsyn" (abbreviation for Disease or Syndrome)). Moreover, we add an additional constraint that the drug is not already used to treat the disease. This is a declarative query for discovering novel treatments. It can be used in several ways depending on what kind of input data is provided. If a specific drug is provided, the query finds diseases that have not been treated with that particular drug before. If a specific disease is provided, the query finds drugs that have not been used to treat that particular disease before. And finally, in this query we show how we can rank the potentially novel treatments by the count of intermediate concepts Y, as it is usually done in LBD.

```
MATCH (drug:Concept:phsu)-[r1]->(y)-
  [r2]->(disease:Concept:dsyn)
WHERE NOT (drug)-[:TREATS]->(disease)
RETURN drug, disease, count(y) AS y_count
  DESC;
```

Figure 4. A generic Cypher query for finding novel treatment rela-tions between drugs and diseases.

A discovery pattern [21] reduces the number of false positive discoveries and supports explanation by stipulating a set of conditions which enhance the likelihood for a good discovery candidate. Such conditions refer to the semantic types of concepts and the relations between them. We show here how to implement the LBD discovery pattern "inhibit the cause of the disease", which can be used to find novel treatments or explain why certain drugs might be beneficial for certain diseases [22]. This discovery pattern is more specific than the one shown before because it also takes into account the semantic relations between the concepts. The idea of the discovery pattern is to find drugs that inhibit some genes that are etiologically related to a disease. Additionally, we are interested only in drugs that have not been already used to treat the disease. In this Cypher query, "phsu", "gngm" and "dsyn" are UMLS semantic type abbreviations; and drug, gene and disease are variables that are instantiated to particular values when the query is run. Figure 5 shows a generic implementation of this discovery pattern.

```
MATCH (drug:phsu)-[:INHIBITS]->
  (gene:gngm)-[:CAUSES]-> (disease:dsyn)
WHERE NOT (drug)-[:TREATS]->(disease)
RETURN drug, gene, disease;
```

Figure 5. Generic Cypher implementation of the "inhibit the cause of the disease discovery pattern".

We would like to show how more advanced versions of this discovery pattern can be implemented. In Figure 6, first,

we find all the drugs in the "Antipsychotic Agents" class by using the ontological relation IS_A. We restrict the class of diseases to neoplasms by using the semantic type "neop".

```
MATCH (drug:Concept:phsu)-[:ISA]->
  (m:Concept {
    name:"Antipsychotic Agents"})
WITH drug
MATCH (drug)-[:INHIBITS]->
  (gene:gngm)-[:CAUSES]->(s:neop)
WHERE NOT (drug)-[:TREATS]->(s)
RETURN drug, count(distinct gene),
  count(distinct s);
```

Figure 6. More specific version of the "inhibit the cause of the disease" discovery pattern.

IV. Discussion

We faced some chalenges when using Neo4j and its declarative graph query language Cypher. The `LOAD CSV` Cypher command, although elegant and consise, was not able to load our data in a reasonable amount of time. There is confusion regarding indexing in the current version of Neo4j (2.1.6) because two types of indexes exist: "schema indexes" and "legacy indexes". The "schema indexes" are recommended by Neo4j, however they do not provide full text indexing and they only index node properties and not relation properties. The "legacy indexes" are not recommended by Neo4j, but they provide full text indexing and relation properties can also be indexed. However, they are more cumbersome to create and use. We did not evaluate how fast Cypher was when answering queries. Our subjective observation was that it was fast with a small number of starting nodes and no aggregation. The queries become much slower when dealing with a set of starting nodes and when aggregation was required.

In the future we will conduct performance evaluation of the proposed approach in terms of execution speed and memory consumption. We will also implement the same algorithms in a RDF triple store such as Virtuoso, and compare its performance to traditional relational database (MySQL) and Neo4j.

V. Conclusion

Research in LBD can be facilitated by considering the relevant literature as a graph of interacting semantic predications, such as those extracted from MEDLINE using SemRep. Implementing this graph using a graph database such as Neo4j has several advantages over the use of SQL technology for this task. We have illustrated this advantage by showing how the graph query language Cypher naturally supports the use of discovery patterns, a powerful mechanism for limiting the number of false positive "discoveries" that must be human reviewed and for providing explanation.

Acknowledgment

This work was supported by the Slovenian Research Agency and by the Intramural Research Program of the U.S. National Institutes of Health, National Library of Medicine.

References

[1] L. Cheng, H. Lin, F. Zhou, Z. Yang, and J. Wang, "Enhancing the accuracy of knowledge discovery: a supervised learning method," BMC Bioinformatics, vol. 15, no. Suppl 12, 2014, p. S9.

[2] D. Rebholz-Schuhmann, A. Oellrich, and R. Hoehndorf, "Text-mining solutions for biomedical research: Enabling integrative biology," Nature Reviews. Genetics, vol. 13, no. 12, 2012, pp. 829–839.

[3] D. R. Swanson, "Fish oil, Raynaud's syndrome, and undiscovered public knowledge," Perspectives in Biology and Medicine, vol. 30, no. 1, 1986, pp. 7–18.

[4] R. A. DiGiacomo, J. M. Kremer, and D. M. Shah, "Fish-oil dietary supplementation in patients with Raynaud's phenomenon: a double-blind, controlled, prospective study." The American journal of medicine, vol. 86, no. 2, 1989, pp. 158–64.

[5] M. D. Gordon and R. K. Lindsay, "Toward discovery support systems: A replication, re-examination, and extension of Swanson's work on literature-based discovery of a connection between Raynaud's and fish oil," Journal of the American Society for Information Science, vol. 47, no. 2, 1996, pp. 116–128.

[6] R. K. Lindsay, "Literature-based discovery by lexical statistics," Journal of the American Society for Information Science, vol. 50, no. 7, 1999, pp. 574–587.

[7] M. Weeber, H. Klein, L. T. W. De Jong-Van Den Berg, and R. Vos, "Using concepts in literature-based discovery: Simulating Swanson's Raynaud-fish oil and migraine-magnesium discoveries," Journal of the American Society for Information Science and Technology, vol. 52, no. 7, 2001, pp. 548–557.

[8] D. Hristovski, J. Stare, B. Peterlin, and S. Dzeroski, "Supporting discovery in medicine by association rule mining in Medline and UMLS." Studies in health technology and informatics, vol. 84, no. Pt 2, 2001, pp. 1344–8.

[9] P. Srinivasan, "Text mining: Generating hypotheses from MEDLINE," Journal of the American Society for Information Science and Technology, vol. 55, no. 5, 2004, pp. 396–413.

[10] D. Cameron, O. Bodenreider, H. Yalamanchili, T. Danh, S. Vallabhaneni, K. Thirunarayan, A. P. Sheth, and T. C. Rindflesch, "A graph-based recovery and decomposition of Swanson's hypothesis using semantic predications." Journal of biomedical informatics, vol. 46, no. 2, 2013, pp. 238–51.

[11] D. Hristovski, T. Rindflesch, and B. Peterlin, "Using literature-based discovery to identify novel therapeutic approaches," Cardiovascular & Hematological Agents in Medicinal Chemistry, vol. 11, no. 1, 2013, pp. 14–24.

[12] M. E. Bales and S. B. Johnson, "Graph theoretic modeling of large-scale semantic networks," Journal of Biomedical Informatics, vol. 39, no. 4, 2006, pp. 451–464.

[13] T. Rindflesch and H. Kilicoglu, "Semantic MEDLINE: An advanced information management application for biomedicine," Information Services & Use, vol. 31, no. 1-2, 2011, pp. 15–21.

[14] T. C. Rindflesch and M. Fiszman, "The interaction of domain knowledge and linguistic structure in natural language processing: Interpreting hypernymic propositions in biomedical text." Journal of Biomedical Informatics, vol. 36, no. 6, 2003, pp. 462–477.

[15] O. Bodenreider, "The Unified Medical Language System (UMLS): integrating biomedical terminology." Nucleic acids research, vol. 32, no. Database issue, 2004, pp. D267–70.

[16] A. T. McCray, S. Srinivasan, and A. C. Browne, "Lexical methods for managing variation in biomedical terminologies." in Proceedings of the Eighteenth Annual Symposium on Computer Application in Medical Care, J. G. Ozbolt, Ed. Washington, DC: Hanley & Belfus, 1994, pp. 235–239.

[17] L. Smith, T. Rindflesch, and W. J. Wilbur, "MedPost: A part-of-speech tagger for bioMedical text," Bioinformatics, vol. 20, no. 14, 2004, pp. 2320–2321.

[18] A. R. Aronson and F.-M. Lang, "An overview of MetaMap: historical perspective and recent advances." Journal of the American Medical Informatics Association : JAMIA, vol. 17, no. 3, 2010, pp. 229–236.

[19] H. Kilicoglu, D. Shin, M. Fiszman, G. Rosemblat, and T. C. Rindflesch, "SemMedDB: A PubMed-scale repository of biomedical semantic predications," Bioinformatics, vol. 28, no. 23, 2012, pp. 3158–3160.

 ISBN: 978-1-61208-408-4

[20] "Neo4j (CSV) Batch Importer," visited on 2015-05-14. [Online]. Available: https://github.com/jexp/batch-import

[21] D. Hristovski, C. Friedman, T. C. Rindflesch, and B. Peterlin, "Exploiting semantic relations for literature-based discovery." AMIA Annual Symposium proceedings, 2006, pp. 349–53.

[22] C. B. Ahlers, D. Hristovski, H. Kilicoglu, and T. C. Rindflesch, "Using the literature-based discovery paradigm to investigate drug mechanisms." AMIA Annual Symposium proceedings, 2007, pp. 6–10.

 ISBN: 978-1-61208-408-4