Actes de la conférence BDA 2016 Gestion de Données – Principes, Technologies et Applications

Ladjel Bellatreche, Farouk Toumani

BDA 2016

Gestion de Données – Principes, Technologies et Applications

32e anniversaire
15-18 novembre 2016, Poitiers, Futuroscope

Actes de la conférence BDA 2016
Conférence soutenue par l'ISAE-ENSMA, l'Université de Poitiers, la région Nouvelle Aquitaine, le laboratoire LIAS et la SIF

Site de la conférence : `https://bda2016.ensma.fr`
Actes en ligne : `https://hal.inria.fr/BDA2016`

Message des organisateurs

La 32ème édition de la conférence sur la "Gestion de Données - Principes, Technologies et Applications" (BDA 2016) a eu lieu à l'Ecole Nationale Supérieure de Mécanique et d'Aérotechnique (ISAE-ENSMA), à Poitiers du 15 au 18 novembre 2016. Ces actes regroupent les versions courtes des articles présentés lors de cette conférence.

L'émergence du phénomène des données massives (bigdata) a pour effet de changer en profondeur la manière dont les différentes phases du processus d'acquisition et de valorisation des données sont mises en œuvre. Les données modernes sont volumineuses et souvent semi-structurées, incomplètes, imprécises, bruitées, inconsistantes, dynamiques et/ou fortement connectées. L'exploitation de ce type de données par des applications issues de domaines scientifiques et métier très variés pose alors de nombreux défis pour la communauté de recherche en informatique tant sur le plan fondamental qu'appliqué. BDA 2016 a été un lieu privilégié pour présenter des travaux récents et rendre compte des défis et des avancées scientifiques et industrielles dans ce domaine en pleine effervescence.

Le programme scientifique a comporté trois conférences invitées, une cabale Wikipédia, 27 articles longs, 9 articles de doctorants et 6 démonstrations. Une originalité de la conférence BDA est de proposer deux catégories pour les articles longs : articles originaux et articles déjà acceptés ou publiés dans une conférence internationale de renom. Cette deuxième catégorie a pour vocation à attirer les meilleurs papiers de la communauté française et donner à leurs auteurs l'opportunité de présenter leurs travaux devant la communauté de recherche nationale.

Nous tenons à remercier tous les auteurs pour leur excellente contribution, toute l'équipe pour l'organisation des journées BDA'2016, Pierre Senellart et le comité de sélection des démonstrations, Serge Abiteboul et Yassine Ouhammou pour l'organisation de la cabale Wikipedia-BDA et enfin tous les membres du comité de programme de BDA 2016.

Ladjel Bellatreche, ISAE-ENSMA, Président du comité d'organisation
Farouk Toumani, Université Clermont Auvergne Président du comité de programme

Table des matières

1 Comités de BDA 2016 **6**

2 Conférenciers invités **8**

2.1 Big Data Management and Scalable Data Science : Key Challenges and (Some) Solutions
Volker Markl 8

2.2 Counting and enumerating for query answering : a survey
Arnaud Durand 8

2.3 Querying Evolving Data Graphs
Evaggelia Pitoura 9

3 Wikipedia-BDA : cabale Wikipedia **9**

4 Résumés des articles longs **11**

4.1 Inferray : fast in-memory RDF inference
Julien Subercaze, Christophe Gravier, Jules Chevalier and Frederique Laforest 11

4.2 Recherche Sociale, Structurée et Sémantique
Raphaël Bonaque, Bogdan Cautis, François Goasdoué et Ioana Manolescu 13

4.3 FURQL : une extension floue du langage SPARQL
Olivier Pivert, Olfa Slama et Virginie Thion 14

4.4 Une approche par dissimilarité pour la caractérisation de jeux de données
William Raynaut, Chantal Soulé-Dupuy, Nathalie Vallès-Parlangeau, Cedric Dray et Philippe Valet 15

4.5 gMark : Génération de Graphes et de Requêtes Dirigée par le Schéma
Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George H. L. Fletcher, Aurélien Lemay et Nicky Advokaat 17

4.6 Un partitionnement d'arêtes à base de blocs pour les algorithmes de marche aléatoire dans les grands graphes sociaux
Yifan Li, Camelia Constantin et Cédric Du Mouza 19

4.7 RDF Query Relaxation Strategies Based on Failure Causes
Géraud Fokou, Stéphane Jean, Allel Hadjali et Mickaël Baron 21

4.8 Optimising SPARQL query processing in distributed knowledge graphs
Abdoul Macina, Johan Montagnat et Olivier Corby 23

4.9 SPARQL query processing with Apache Spark
Hubert Naacke, Olivier Curé et Bernd Amann 24

4.10 Towards Differentially Private Community Detection
Hiep Huu Nguyen, Abdessamad Imine et Michaël Rusinowitch 26

4.11 Conception d'une base de données capteurs à l'aide de contraintes spatio-temporelles
Manel Charfi, Yann Gripay et Jean-Marc Petit 28

4.12 Scientific Workflow Execution with Multiple Objectives in Multisite Clouds
Ji Liu, Esther Pacitti, Patrick Valduriez, Daniel de Oliveira et Marta Mattoso 30

4.13 Lignages efficaces sur les instances quasi-arborescentes : Limites et extensions
Antoine Amarilli, Pierre Bourhis et Pierre Senellart 32

4.14 Calculer et Compresser le Skycube Négatif
Nicolas Hanusse, Patrick Kamnang Wanko et Sofian Maabout 33

4.15 Extending CloudMdsQL with MFR for Big Data Integration
Carlyna Bondiombouy, Boyan Kolev, Patrick Valduriez et Oleksandra Levchenko 35

4.16 Access Control Enforcement for Selective Disclosure of Linked Data
Tarek Sayah, Emmanuel Coquery, Romuald Thion et Mohand-Saïd Hacid 37

4.17 Federated SPARQL Query Processing with Replicated Fragments
Gabriela Montoya, Hala Skaf-Molli, Pascal Molli et Maria Esther Vidal 39

4.18 Fast Parallel Mining of Maximally Informative k-Itemsets in Big Data
Saber Salah, Reza Akbarinia et Florent Masseglia . . . 41
4.19 Inférence de Schémas pour Données JSON Massives
Mohamed-Amine Baazizi, Houssem Ben Lahmar, Dario Colazzo, Giorgio Ghelli et Carlo Sartiani . . . 43
4.20 Hybrid algorithms for candidate keys enumeration for a relational schema
Karima Ennaoui et Lhouari Nourine . . . 45
4.21 Filtres pour jointures et requêtes récursives en MapReduce
Thuong-Cang Phan, Thi-To-Quyen Tran et Laurent D'Orazio . . . 46
4.22 Interactive mapping refinement with user data examples
Angela Bonifati, Ugo Comignani, Emmanuel Coquery et Romuald Thion . . . 48
4.23 A scalability comparison study of data management approaches for smart metering systems
Houssem Chihoub et Christine Collet . . . 50
4.24 Impact de la réplication sur l'équilibrage de charge dans les bases de données distribuées
Noël Gillet, Nicolas Hanusse et Frédéric Lalanne . . . 52
4.25 Impact of Time on Detecting Spammers in Twitter
Mahdi Washha, Aziz Qaroush et Florence Sedes . . . 54
4.26 Query-Oriented Summarization of RDF Graphs
Šejla Čebirić, François Goasdoué et Ioana Manolescu . . . 55
4.27 An updated dashboard of Complete Search FSM Implementations in Centralized Graph Transaction Databases
Rihab Ayed, Mohand-Saïd Hacid, Rafiqul Haque et Abderrazek Jemai . . . 57

5 Articles de doctorant-e-s **59**
5.1 Architecture, concepts et services d'un système d'indexation de données distribuées pour l'observation à large échelle en écologie marine
Romain David . . . 59
5.2 Privacy Preserving Query Processing in the Cloud
Sakina Mahboubi . . . 61
5.3 A Database Model for Time Series
Cyrille Ponchateau . . . 63
5.4 Exécution de requêtes distribuées sous contraintes d'anonymat
Axel Miche . . . 65
5.5 AstroSpark - Towards a Distributed Data Server for Big Data in Astronomy
Meriem Brahem . . . 67
5.6 Gestion des données manquantes dans les grands entrepôts de données géo référencées : Application aux données agricoles
Nestor Koueya . . . 69
5.7 Towards a benchmark for Database exploration
Mahfoud Djedaini . . . 71
5.8 Stratégies de divulgation de lien en ligne pour les réseaux sociaux
Younes Abid . . . 73
5.9 Analyse de concepts formels pour la classification de triplets RDF
Justine Reynaud . . . 75

6 Démonstrations **78**
6.1 Génération de Requêtes pour les Bases de Données Orientées Graphes
Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George H. L. Fletcher, Aurélien Lemay, Nicky Advokaat . . . 78
6.2 Estocada : Stockage Hybride et Ré-écriture sous Contraintes d'Intégrité
Rana B. AL-Otaibi, Francesca Bugiotti, Damian Bursztyn, Alin Deutsch, Ioana Manolescu et StamatisZampetakis . . . 79

6.3 Demonstration of the CloudMdsQL Multistore System
Boyan Kolev, Carlyna Bondiombouy, Patrick Valduriez, Ricardo Jiménez-Peris, Raquel Pau José Pereira 82
6.4 FleDDi : Highlighting the Flexibility of Data Dissemination Systems
Sébastien Dufromentel, Sylvie Cazalens, François Lesueur et Philippe Lamarre 83
6.5 SPARQLGX : Une Solution Distribuée pour RDF Traduisant SPARQL vers Spark
Damien Graux, Louis Jachiet, Pierre Genevès, Nabil Layaïda 87
6.6 Exploring the evolution of science through interactive phylomemetic topic maps
Samuel Castillo, Hubert Naacke, Bernd Amann, David Chavalarias 89

1 Comités de BDA 2016

Présidente des Journées
Ladjel Bellatreche, LIAS / ISAE-ENSMA, Poitiers, France

Président du comité de Programme
Farouk Toumani, LIMOS / Université Blaise Pascal, Clermont-Ferrand, France

Président du comité des démonstrations
Pierre Senellart, Télécom Paris Tech, France

Comité d'organisation
Mickael Baron, ISAE-ENSMA, Poitiers, France
Brice Chardin, ISAE-ENSMA, Poitiers, France
Patrick Girard, Université de Poitiers, France
Stéphane Jean, Université de Poitiers, France
Amin Mesmoudi, Université de Poitiers, France
Yassine Ouhammou, ISAE-ENSMA, Poitiers, France
Claudine Rault, ISAE-ENSMA, Poitiers, France
Okba Barkat, ISAE-ENSMA, Poitiers, France
Nassima Benammar, ISAE-ENSMA, Poitiers, France
Selma Bouarar, ISAE-ENSMA, Poitiers, France
Lahcène Brahimi, ISAE-ENSMA, Poitiers, France
Anh Toan Bui Long, ISAE-ENSMA, Poitiers, France
Géraud Fokou, ISAE-ENSMA, Poitiers, France
Abdallah Khelil, ISAE-ENSMA, Poitiers, France
Yves Mouafo, ISAE-ENSMA, Poitiers, France
Thanh Dat Nguyen, ISAE-ENSMA, Poitiers, France
Cyrille Ponchateau, ISAE-ENSMA, Poitiers, France

Comité de Programme
Nicolas Anciaux, INRIA
Khalid Belhajjame, PSL, Université Paris-Dauphine, LAMSADE, France
Christophe Bobineau, Institut Polytechnique de Grenoble
Pierre Bourhis, CNRS LIFL/INRIA Lille, France
Amel Bouzeghoub, Télécom SudParis CNRS UMR 5157 SAMOVAR, France
Bogdan Cautis, Université Paris-Sud, France
Radu Ciucanu, University of Oxford, UK
Dario Colazzo, LAMSADE, Université Paris-Dauphine, France
Christine Collet, Grenoble Institute of Technlogy, France
Jérôme Darmont, Université de Lyon, France
Cedric Du Mouza, CNAM, Paris, France
Frédéric Flouvat, Universiy of New Caledonia, France
Walid Gaaloul, Computer Science Department Télécom SudParis, France
Francois Goasdoue, Université Rennes 1, France
David Gross-Amblard, Université Rennes 1 - ISTIC / IRISA, France
Mohand-Said Hacid, Université Claude Bernard Lyon 1 - UCBL, France
Abdelkader Hameurlain, IRIT Laboratory, Université de Toulouse, France
Hélène Jaudoin, Université Rennes 1, France
Stephane Jean, LIAS / ISAE-ENSMA and Université de Poitiers, France
Lotfi Lakhal, Université d'Aix-Marseille, LIF, France
Sofian Maabout, LaBRI. Université de Bordeaux, France

Florent Masseglia, INRIA, Montpellier, France
Boughanem Mohand, IRIT Université Paul Sabatier Toulouse, France
Pascal Molli, Université de Nantes - LINA, France
Amedeo Napoli, LORIA Nancy, France
Benjamin Nguyen, INSA Centre Val de Loire, France
Noël Novelli, LIF UMR CNRS 7279 – Université d'Aix-Marseille, France
Marie Pailloux, Université Blaise Pascal, France
Pascal Poncelet, LIRMM Montpellier, France
Philippe Pucheral, INRIA et UVSQ, France
Federico Ulliana, Université Montpellier 2, France

Comité des démonstrations

Anciaux Nicolas, INRIA, France
D'Orazio Laurent, LIMOS/ISIMA, France
Gançarski Stéphane, LIP6/UPMC, France
Masseglia Florent, INRIA/LIRMM, France
Preda Nicoleta, PRISM/UVSQ, France
Termier Alexandre, LIG/Université Joseph Fourier Grenoble 1, France
Teste Olivier, IRIT/Université Toulouse 2 Le Mirail, France

Organisation Cabale Wikipedia-BDA

Serge Abiteboul, INRIA & ENS, Paris, France
Yassine Ouhammou, LIAS/ISAE-ENSMA, Poitiers, France

Edition des actes

Mickael Baron, LIAS / ISAE-ENSMA, Poitiers, France

Site Web

Mickael Baron, LIAS / ISAE-ENSMA, Poitiers, France

Poster

Brice Chardin, LIAS / ISAE-ENSMA, Poitiers, France

2 Conférenciers invités

2.1 Big Data Management and Scalable Data Science : Key Challenges and (Some) Solutions
Volker Markl

Volker Markl is a Full Professor and Chair of the Database Systems and Information Management Group at the Technische Universität Berlin (TUB) and an Adjunct Full Professor at the University of Toronto. He is Director of the Intelligent Analytics for Massive Data Research Group at DFKI and Director of the Berlin Big Data Center. In addition, he serves as the Secretary of the VLDB Endowment. His current research interests include new hardware architectures for information management, scalable processing and optimization of declarative data analysis programs, and scalable data science. To date, Volker has presented over 200 invited talks in numerous industrial settings, major conferences, and research institutions worldwide. Furthermore, he has authored and published over 100 research papers at world-class scientific venues. Between 2010-2016, he was Speaker and Principal Investigator of the Stratosphere Research Unit funded by the German Research Foundation (DFG), which resulted in numerous top-tier publications, as well as the Apache Flink big data analytics system. In 2014, he was named one of Germany's leading digital minds (Digitale Köpfe) by the German Informatics Society. Prior to joining TUB, he was a Research Staff Member and Project Leader at the IBM Almaden Research Center in San Jose, California.

Abstract : The shortage of qualified data scientists is effectively limiting Big Data from fully realizing its potential to deliver insight and provide value for scientists, business analysts, and society as a whole. Data science draws on a broad number of advanced concepts from the mathematical, statistical, and computer sciences in addition to requiring knowledge in an application domain. Solely teaching these diverse skills will not enable us to on a broad scale exploit the power of predictive and prescriptive models for huge, heterogeneous, and high-velocity data. Instead, we will have to simplify the tasks a data scientist needs to perform, bringing technology to the rescue : for example, by developing novel ways for the specification, automatic parallelization, optimization, and efficient execution of deep data analysis workflows. This will require us to integrate concepts from data management systems, scalable processing, and machine learning, in order to build widely usable and scalable data analysis systems. In this talk, I will present some of our research results towards this goal, including the Apache Flink open-source big data analytics system, concepts for the scalable processing of iterative data analysis programs, and ideas on enabling optimistic fault tolerance.

2.2 Counting and enumerating for query answering : a survey
Arnaud Durand

Arnaud Durand works in the mathematics department of Paris-Diderot University. His research interests are in computational complexity, finite model theory and more generally logic in computer science. In the recent years, we worked on algorithmic and complexity issues for query answering. He received is PhD from Caen University in 1996, was associate professor in Paris 12 university from 1999 to 2005 and since 2005 is professor in Paris-Diderot University.

Abstract : Counting and enumerating the solutions of a problem are two natural algorithmic tasks that also make sense in databases where they relate to problems such as probabilistic reasoning or data aggregation. However, it is well-known that counting is usually hard even when deciding the existence of a solution is easy. Hence, finding efficient algorithms to count the answer set of some queries or to generate all tuples of the solution one after the other is challenging. In this talk, I will survey some simple algorithmic and complexity issues related to counting and enumerating in query answering and give an overview of recent results.

2.3 Querying Evolving Data Graphs
Evaggelia Pitoura

Evaggelia Pitoura is a Professor at the Computer Science and Engineering Department of the University of Ioannina, Greece where she also leads the Distributed Management of Data (DMOD) lab. She received a BSc degree from the University of Patras, Greece, and an MSc and PhD degree from Purdue University, USA. Her research interests are in the general area of data management with a recent focus on social networks and data exploration based on diversity, preferences and time. Her publications include more than 150 articles in international journal and conferences and a highly-cited book on mobile computing. She has served or serves on the editorial board of VLDBJ, TKDE, DAPD and as a group leader, senior PC member, or co-chair of many international conferences (including PC chair of EDBT 2016 and co-chair of ICDE 2012). She is the recipient of three best paper awards (ICDE 1999, DBSocial 2013, PVLDB 2013), a Marie Currie Fellowship (2009) and two Recognition of Service Awards from ACM.

Abstract : Graphs form a natural model for expressing relationships and interactions between entities. Most large graphs, including those modeling social and cooperation networks, evolve over time. In our research, we look into interesting new ways of exploring the history of such evolving data graphs. In this talk, I will present our recent work on finding nodes reachable through time, durable graphs patterns and Best-Friends-Forever (BFF) set of nodes. I will also provide a short survey of related research in exploring time-related information in data graphs and highlight directions for future work.

3 Wikipedia-BDA : cabale Wikipedia

Organisation : Serge Abiteboul et Yassine Ouhammou
La Société Informatique de France et Wikimedia France s'associent pour enrichir les articles de Wikipedia dédiés à l'informatique, en français. À l'heure où l'école, le collège et le lycée ouvrent plus largement leurs portes à l'informatique, il nous a paru important de participer à l'enrichissement de la première encyclopédie en ligne au monde.

Des formateurs ont été aux côtés des participants pour les initier à la contribution sur Wikipedia. Cette journée a été une occasion unique pour découvrir les rouages de Wikipédia.

Programme de la cabale

— 08h15-08h30 : Accueil des participants
— 08h30-08h40 : Introduction à la Cabale Informatique de France (par Serge Abiteboul)
— 08h40-09h15 : Introduction générale à Wikipédia (par Clément Coubronne)
— 09h15-10h30 : Edit-a-thon, au boulot.
— 10h30-10h45 : Pause café
— 10H45-12h45 : Edit-a-thon, au boulot
— 12h45-13h30 : Repas
— 13h30- ... : Edit-a-thon, suite ...

Résumés des articles longs

Inferray: fast in-memory RDF inference

[The full version of this paper has been published in the VLDB 2016 conference]

Julien Subercaze, Christophe Gravier
Jules Chevalier, Frederique Laforest
Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516
F-42023 Saint Etienne, France

1. INTRODUCTION

As defined by the W3C standard, the elemental constituents of the Resource Description Framework (RDF) data model are RDF terms. The set of RDF terms is broken down into three disjoint subsets: URIs (or IRIs), literals and blank nodes. The RDF triple is a 3-tuple of RDF terms, and is commonly denoted by $\langle s, p, o\rangle$ for $\langle subject, property, object\rangle$. Each RDF triple represents an atomic "fact", where p describes the relationship between s and o. RDF datasets are commonly conceptualized as directed labeled graphs, where s and o are vertices and p are edges. There has been a growing interest to design efficient Semantic Web databases known as triple stores, to handle potentially large volumes of RDF data and to support SPARQL, a mature, feature-rich query language for RDF content.

An inference layer coupled with a triple store provides an interesting feature over aggregated Web data. Inference is specified within a family of recommendations by the W3C, ranging from simple entailment regime like RDFS to more complex ones like OWL Full [6]. Special efforts have been made to specify *working* logics for real-world systems with more affordable complexity, like simplified RDFS [7], ρdf, and RDFS-Plus. In practice, systems usually perform incomplete RDFS reasoning and consider only rules whose antecedents are made of two-way joins [4]. Hence, inference engines such as Jena [5] and OWLIM [3] propose using lightweight flavors of RDFS. ρdf is another common meaningful subset of RDFS. RDFS-Plus [2] provides a framework that allows the merging of datasets and the discovery of triples of practical interest.

In general, inference schemes are inherently divided into two main approaches, backward-chaining and forward-chaining. Backward-chaining performs inference at query time. It is well suited to frequently changing data. Forward-chaining exhaustively makes explicit the implicit data.Inferred triples are explicitly written into the triple store. With this materialization, inferred data can be consumed without integrating the inference engine with the runtime query engine. Herein, we focus on forward-chaining inference. In the rest of the paper, unless otherwise stated, inference means forward-chaining inference.

Most of the research on forward-chaining has focused on decidability and fragment selection. The theory of processing per se has generally been relegated to secondary status. Regardless of the algorithm used for inference, execution involves numerous memory accesses. In the presence of limited memory bandwidth, it results in CPU stalling – as the CPU waits for the data required while it is being fetched from the main memory. In contrast, a predictive memory access pattern guides the prefetcher to retrieve the data correctly in advance, thus enabling a better usage of memory bandwidth.

In this paper, we revisit the problem of in-memory inference, with a focus on sequential memory access and sorting algorithms. Our system Inferrayuses widely dynamic arrays of 64-bit integers. We demonstrate that the performance of sequential memory access can be exploited to design and implement an efficient forward-chaining inference system.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. INFERRAY: OVERVIEW

We design Inferray with the objective of enhancing inference performance through an approach that both minimizes duplicate generation and leverages memory bandwidth using sequential memory accesses.

The global inference process is described in Algorithm 1. Inferray performs two steps.

In a first step, transitive closures are computed using a different data layout than for the second step (Line 2). The transitive closure cannot be performed efficiently using iterative rules application since duplicate generation rapidly degrades performance. To tackle this issue, we use the very efficient algorithm from Nuutila. This closure is applied on schema data for RDFS (`subClass` and `subProperty`), and is also applied to every transitive property as well as for `sameAs` for RDFS-Plus.

In a second step, Inferray applies rules iteratively, until a fixed-point is reached (Line 4). To derive new triples using rules, Inferray takes two inputs: existing triples and newly-inferred triples, except for the very first iteration, where existing and newly-inferred storages are identical (Line 3). For the rule-based inference stage, we observe that triple inference is identical to the process of performing joins and inserts. Due to the selectivity of the rules for all the frag-

ments supported by Inferray, the vertical partitioning approach [1] is well suited to perform inference efficiently using sort-merge joins. Inferred triples are stored in the structure `inferred`. They may contain triples that are already in the `main` structure: the duplicates. Duplicates are consequently removed from `inferred`, then these new triples are added in `main` and `new` to continue the process. The rule sets supported by Inferray are of various complexities. RDFS and ρdf require only two-way joins that are efficiently performed. However, several rules of RDFS-Plus contain multi-way joins that may hinder performance.

Algorithm 1: Inferray: main loop

Input: Set of triples *triples*, triples stores: $main, inferred, new$
Output: Set of triples with inferred triples

```
1 main ← triples;
2 transitivityClosures();
3 new ← main;
4 while new ≠ ∅ do
5 |   inferred ← infer(main, new);
6 |   new ← inferred \ main;
7 |   main ← main ∪ new;
8 end
```

To perform sort-merge joins efficiently, property tables that contain pairs of $\langle subject, object \rangle$ ($\langle s, o \rangle$ in the rest of the paper) must be sorted on $\langle s, o \rangle$ and possibly on $\langle o, s \rangle$. Inferray must also maintain sorted and duplicate-free property tables after each iteration of rules.

The cornerstone of Inferray is the sorting step that takes place first when inferred triples are merged into `main` and `new`, and second, when a property table is required to be sorted on $\langle o, s \rangle$ for inference purpose. To sort these lists of pairs of integers efficiently, we use dense a numbering technique to reduce the entropy. It allows us to design efficient algorithms that outperform state-of-the-art generic solutions on our use case. A key observation is that inferencing corresponds to increasing the density of the graph, no new vertices are added. In property tables, except for `Same-as`, subjects are of the same nature (property or not) as are objects (either property or non-property). Subject/Object pairs are sorted lexicographically. We aim at keeping numbers dense for properties on one side, and for non-properties on the other. The number of properties is negligible compared to the number of non-properties. Dense numbering is performed at loading time. Each triple is read from the file system, dictionary encoding and dense numbering happen simultaneously. We split our numbering space, i.e., $[0; 2^{64}]$, at 2^{32} – considering that in practice the number of properties is smaller than the number of resources. We assign the properties numbers in decreasing order and resources in an increasing order. For instance, the first property will be assigned 2^{32}, the second $2^{32} - 1$, and the first resource will be assigned $2^{32} + 1$, thereby, keeping dense numbering for both cases. To access the array of property tables, we perform a simple index translation.

Counting sort is an integer sorting algorithm that shows excellent performance for small values ranges, in other words, when the entropy is low. Sorting pairs with counting sort is not a well-studied problem, therefore we devise an adaptation of the algorithm to fulfill our requirement. due to space limitation, we cannot present this algorithm here. Readers interested in details should read the full version of this paper. The underlying idea of our counting sort for pairs is to keep the histogram principle for sorting the subjects (assuming we sort on $\langle s, o \rangle$) – while being able to sort in linear time the objects associated with each subject value (i.e., keys of the histogram). For this purpose, we store the objects in a single array. By using their positions in the unsorted array, combined with histogram values, we are able to sort subarrays corresponding to subjects. Finally, we build the sorted array by combining the histogram and the object sorted arrays.

3. CONCLUSION

In this paper, we present algorithms and data structures that drive Inferray, a new forward-chaining reasoner. The main technical challenges are to perform efficient transitive closure, to foster predictable memory access patterns for mechanical sympathy, and to efficiently eliminate duplicates. The first challenge is addressed by a transitive closure computation performed before the inference algorithm. The second challenge is tackled by a sort-merge-join inference algorithm over sorted arrays of fixed-length integers that favors sequential memory access. Duplicate elimination is handled by scanning arrays after efficient sorting, using efficient sorting algorithms for low entropy. Our exhaustive experiments, presented in the long version of this paper, demonstrate that leveraging solutions for these issues lead to an implemented system that outperforms existing forward-chaining reasoners on complex fragments. This paper comes with an open-source implementation of Inferray, which will be of the utmost practical interest to working ontologists.

4. REFERENCES

[1] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *PVLDB*, pages 411–422, 2007.

[2] D. Allemang and J. Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[3] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov. OWLIM: A family of scalable semantic repositories. *Semantic Web*, pages 33–42, 2011.

[4] E. L. Goodman and D. Mizell. Scalable in-memory RDFS closure on billions of triples. In *SSWS*, pages 17–31, 2010.

[5] B. McBride. Jena: Implementing the RDF Model and Syntax Specification. In *SemWeb*, 2001.

[6] W3C. OWL 1.1 Tractable Fragments. `http://www.w3.org/Submission/owl11-tractable/`, Dec. 2006.

[7] W3C. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`, 2014.

Recherche Sociale, Structurée et Sémantique

Raphaël Bonaque
INRIA & U. Paris-Sud, France
raphael.bonaque@inria.fr

Bogdan Cautis
U. Paris-Sud & INRIA, France
bogdan.cautis@lri.fr

François Goasdoué
U. Rennes 1 & INRIA, France
fg@irisa.fr

Ioana Manolescu
INRIA & U. Paris-Sud, France
ioana.manolescu@inria.fr

RÉSUMÉ EN FRANÇAIS

Les contenus sociaux comme les blogs, les tweets et les sites de nouvelles sont riches en informations interconnectées. Nous identifions un ensemble de conditions nécessaires à une exploitation complète de ces contenus et présentons un nouveau modèle de données, S3, qui est le premier à les satisfaire. S3 capture les relations *sociales* entre les utilisateurs, ainsi qu'avec les contenus sociaux, mais aussi la *structure* et la *sémantique* de ces contenus.

Nous présentons le premier algorithme top-k de recherche par mots-clés prenant en compte les dimensions sociales, structurelles et sémantiques, et nous prouvons formellement sa terminaison et sa correction. Des expériences sur de vrais réseaux sociaux montrent l'efficacité et les avantages qualitatifs de notre algorithme grâce l'exploration conjointe des dimensions sociales, structurelles et sémantiques de S3.

FURQL : une extension floue du langage SPARQL

Olivier Pivert
Univ. Rennes 1, IRISA
Lannion, France
Olivier.Pivert@irisa.fr

Olfa Slama
Univ. Rennes 1, IRISA
Lannion, France
Olfa.Slama@irisa.fr

Virginie Thion
Univ. Rennes 1, IRISA
Lannion, France
Virginie.Thion@irisa.fr

La publication de données ouvertes liées sur le web est un phénomène en pleine croissance. L'étude des modèles et langages permettant l'exploitation de ces données s'est donc grandement intensifiée ces dernières années. Les données publiées sont généralement de nature hétérogène et ne présentent pas de régularité structurelle. De plus, elles sont souvent porteuses de notions graduelles. Dans ce contexte, il est nécessaire de pouvoir proposer des langages de requête aussi flexibles que possible. Nous avons proposé une extension du langage SPARQL, fondée sur la théorie des ensembles flous, permettant (1) d'interroger une *extension floue du modèle de données RDF* dans laquelle les triplets sont porteurs de notions graduelles, et (2) d'exprimer des *préférences floues* portant non seulement sur les données mais également sur la *structure* du graphe de données, que celui-ci soit flou ou non.

BDA 2016, Poitier, France.

Une approche par dissimilarité pour la caractérisation de jeux de données

William Raynaut
IRIT UMR 5505, UT1, UT3
Universite de Toulouse
william.raynaut@irit.fr

Chantal Soule-Dupuy
IRIT UMR 5505, UT1, UT3
Universite de Toulouse
chantal.soule-dupuy@irit.fr

Nathalie Valles-Parlangeau
IRIT UMR 5505, UT1, UT3
Universite de Toulouse
nathalie.valles-parlangeau@irit.fr

Cedric Dray
INSERM, U1048
Universite de Toulouse
cedric.dray@inserm.fr

Philippe Valet
INSERM, U1048
Universite de Toulouse
philippe.valet@inserm.fr

ABSTRACT

La caractérisation de jeu de données reste un verrou majeur de l'analyse de données intelligente. Une majorité d'approches à ce problème agrègent les informations décrivant les attributs individuels des jeux de données, ce qui représente une perte d'information. Nous proposons une approche par dissimilarité permettant d'éviter cette agrégation, et étudions son intérêt dans la caractérisation des performances d'algorithmes de classification et dans la résolution de problèmes de méta-apprentissage.

Keywords

Caractérisation de jeux de données, Dissimilarité, Méta-attributs, Méta-apprentissage, Sélection d'algorithmes

1. INTRODUCTION

L'émergence du phénomène de données massives crée un besoin grandissant en analyse de données, et bien souvent, cette analyse doit être conduite par des experts de différents domaines ayant peu d'expérience en science des données. Afin de leur permettre de tout de même exploiter efficacement leurs données, divers travaux ont proposé des méthodes d'assistance intelligente à l'analyse de données [7]. La caractérisation de jeu de données, problème apparu avec les premières ébauches de méta-apprentissage [2], constitue encore l'un des verrous majeurs de l'assistance intelligente à l'analyse de données.

Dans le cadre général du méta-apprentissage, le problème de caractérisation de jeu de données consiste en la définition d'un ensemble de propriétés de jeu de données (ou méta-attributs) permettant leur caractérisation précise qui doit de plus être utilisable par des algorithmes de méta-apprentissage. Afin de se conformer aux prérequis de la plupart des algorithmes de méta-apprentissage, ces propriétés sont généralement agrégées en vecteurs d'attributs de taille fixe, ce qui peut représenter une importante perte d'information [4]. Nous étudions la possibilité que les limitations des techniques courantes de caractérisation de jeu de données soient l'un des obstacles majeurs à la bonne performance de la sélection d'algorithmes. Nous nous concentrons en particulier sur la définition d'une représentation des jeux de données permettant d'utiliser toute l'information disponible pour leur caractérisation.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. APPROCHE

Considérons deux jeux de données, **A** et **B** illustrés en figure 1. **A** décrit 12 attributs de 100 individus, et **B** 10 attributs de 200 individus. On souhaite comparer les résultats de 5 mesures statistiques et informationnelles relevées sur les attributs individuels de ces jeux de données (comme illustré sur le second attribut de **A**).

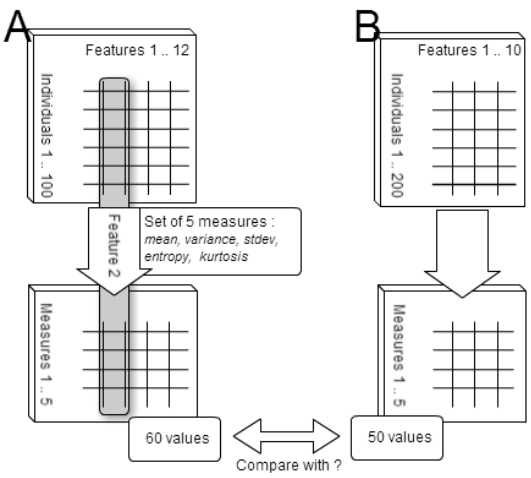


Figure 1: Propriétés d'attributs individuels

L'information complète que l'on souhaite comparer est donc un vecteur de 60 valeurs pour **A** et de 50 pour **B**. Une approche classique [3] serait de faire une moyenne de chaque méta-attribut selon les différents attributs des jeux de données, perdant ainsi l'information caractérisant individuellement chaque attribut.

Notre approche est de comparer les attributs de **A** et **B** par paires les plus similaires, comparant les attributs en surnombre à d'hypothétiques attributs vides. L'hypothèse émise ici est qu'un attribut absent équivaut à un attribut dont aucune valeur n'est connue. Pour en revenir à l'exemple, la comparaison des 5 mesures s'effectuera donc entre l'attribut de **A** et l'attribut de **B** les plus similaires *selon ces mêmes mesures*, puis sur les seconds plus similaires et ainsi de suite, pour finir par comparer les mesures relevées sur les deux attributs surnuméraires de **A** avec leur valeur sur un hypothétique attribut vide de **B**. Cette comparaison par paires permet de s'affranchir de l'ordre de présentation des attributs, qui ne recèle aucune information, se concentrant sur la topologie réelle du jeu de données.

3. VALIDATION

Afin d'évaluer l'utilité de la dissimilarité proposée, une procédure de validation en deux étapes a été mise en place. Tout d'abord, le potentiel de la dissimilarité proposée pour l'apprentissage est évalué en utilisant les méthodes d'estimation présentées dans [8]. Ceci a permis de montrer que l'application d'une dissimilarité prenant en compte les attributs individuels des jeux de donnée permet de prédire plus facilement si un algorithme d'apprentissage donné sera *approprié* à un nouveau jeu de données.

Dans un second temps, une importante expérience de méta-apprentissage évalue la dissimilarité proposée dans un large panel de scénarios. On y voit que l'introduction de la dissimilarité dans un important ensemble d'algorithmes de méta-apprentissage a un impact généralement positif sur leurs performances.

Bien que pouvant résulter en cas d'application, ces expériences se veulent préliminaires. Une dissimilarité particulière y est évaluée, principalement pour étudier l'intérêt de la démarche. Devant les résultats positifs obtenus, des expériences plus diverses sont en cours pour comparer diverses variants de dissimilarité dans des scénarii réalistes.

4. CONCLUSION

Nous avons proposé une fonction de dissimilarité entre jeux de données présentant un ensemble de propriétés désirables, et capable d'employer des méta-attributs caractérisant des attributs particuliers de ces jeux de données. Nous avons montré qu'elle permet de caractériser l'adéquation d'algorithmes de classification avec des jeux de données plus efficacement que des distances traditionnelles, et qu'elle peut être employée avec de bonnes performances dans le contexte de classification au méta-niveau.

De nombreuses pistes d'amélioration restent cependant à explorer. Tout d'abord, notre dissimilarité permet d'utiliser des méta-attributs caractérisant des attributs particuliers des jeux de données, mais diverses expériences [1] ont montré que les propriétés d'un attribut *dans le contexte des autres attributs* sont au moins aussi importantes. Il serait alors intéressant de permettre l'utilisation de tels méta-attributs relationnels, comme la covariance, ou l'information mutuelle, par la dissimilarité. D'autre part, bien que divers et provenant d'approches très différentes, les méta-attributs employés dans nos expériences ne couvrent pas complètement l'état de l'art en la matière. Ces dernières années ont été riches en contributions introduisant de nouveaux méta-attributs [6, 5], dont l'utilisation pourrait révéler l'intérêt d'une approche par dissimilarité dans de nouveaux contextes. Enfin, comme l'efficacité de l'approche par dissimilarité apparaît très dépendante du contexte (comme c'est souvent le cas en apprentissage et méta-apprentissage), il pourrait être intéressant de concevoir une méthode d'évaluation de méta-attributs considérant leurs diverses natures (globaux, liés à un attribut, relationnels...). Il serait alors possible de caractériser l'utilité des divers méta-attributs dans une variété de situations et donc d'approfondir notre connaissance du problème de méta-apprentissage.

Dans le cadre de l'assistance intelligente à l'analyse de données, un atout particulier de notre approche est qu'elle permettrait une caractérisation unifiée des expériences d'analyse de données. En effet, disposant d'une quelconque représentation du processus d'analyse de données et de ses résultats, il serait possible de l'intégrer dans la dissimilarité, permettant la comparaison directe d'expériences complètes. Il s'agit là d'un premier pas vers de nouvelle approches d'assistance intelligente à l'analyse de données, permettant notamment l'utilisation directe d'heuristiques pour la découverte et recommandation de processus d'analyse adaptés.

5. REFERENCES

[1] Brown, G., Pocock, A., Zhao, M.J., Luján, M.: Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. The Journal of Machine Learning Research 13(1), 27–66 (2012)

[2] Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. Machine learning 54(3), 187–193 (2004)

[3] Kalousis, A.: Algorithm selection via meta-learning. Ph.D. thesis, Universite de Geneve (2002)

[4] Kalousis, A., Hilario, M.: Model selection via meta-learning: a comparative study. International Journal on Artificial Intelligence Tools 10(04), 525–554 (2001)

[5] Ntoutsi, I., Kalousis, A., Theodoridis, Y.: A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees. In: SDM. pp. 810–821. SIAM (2008)

[6] Peng, Y., Flach, P.A., Brazdil, P., Soares, C.: Decision tree-based data characterization for meta-learning. IDDM-2002 p. 111 (2002)

[7] Serban, F.: Toward effective support for data mining using intelligent discovery assistance. Ph.D. thesis (2013)

[8] Wang, L., Sugiyama, M., Yang, C., Hatano, K., Feng, J.: Theory and algorithm for learning with dissimilarity functions. Neural computation 21(5), 1459–1484 (2009)

gMark : Génération de Graphes et de Requêtes Dirigée par le Schéma

Guillaume Bagan
CNRS LIRIS
guillaume.bagan@liris.cnrs.fr

Angela Bonifati
Université Lyon 1 & CNRS LIRIS
angela.bonifati@univ-lyon1.fr

Radu Ciucanu
Université Blaise Pascal, Clermont-Ferrand
ciucanu@isima.fr

George H. L. Fletcher
TU Eindhoven
g.h.l.fletcher@tue.nl

Aurélien Lemay
Université Lille 3 & INRIA
aurelien.lemay@inria.fr

Nicky Advokaat
TU Eindhoven
n.advokaat@student.tue.nl

ABSTRACT

Les jeux de données représentés par des graphes de grande taille sont omniprésents dans les domaines applicatifs actuels. C'est pourquoi les bases de données orientées graphes jouent un rôle de plus en plus important. Dans l'étude de ces systèmes, il est vital que la communauté scientifique ait à sa disposition des solutions pour générer des jeux de données de référence comprenant des instances de base de données et des requêtes ayant des propriétés prévisibles et contrôlables. Dans cet article, nous présentons les principes à la fois théoriques et d'ingénierie de gMark, un système générique de génération de graphes et de requêtes basé sur une gestion flexible des schémas et des requêtes. Une contribution centrale de gMark est sa capacité à viser et à contrôler la diversité des propriétés à la fois des instances de graphes générés et des requêtes correspondantes à celles ci. Une autre innovation est la capacité à générer des requêtes récursives basées sur des expressions régulières de chemin, un paradigme important dans les requêtes sur les graphes. Nous illustrons à la fois la flexibilité et l'applicabilité de gMark en montrant ses capacités à générer des graphes et des requêtes de haute qualité, et sa capacité à exploiter des schémas définis par l'utilisateur dans plusieurs domaines applicatifs différents.

1. INTRODUCTION

Le problème. Nous étudions ici le problème de la génération conjointe d'un graphe et d'un jeu de requêtes associé à partir d'un schéma en vue d'analyses expérimentales des systèmes de base de données. Notre étude est motivée par l'omniprésence des données graphiques dans les applications modernes telles que les réseaux sociaux, les données biologiques ou encore les bases de données géographiques pour donner quelques exemples. En réponse à cette problématique, la recherche et le développement de systèmes sachant gérer des volumes massifs de données structurées sous forme de graphe sont particulièrement actifs. Cela va de systèmes de base de données orientés graphes à des systèmes plus spécialisés de représentation de connaissance. Les systèmes gérant de manière native les graphes, tels que Neo4j[1], proposent ainsi leur propres modèles déclaratifs de données et leur propre langage de requête, en portant une attention particulière à l'optimisation des requêtes ainsi qu'aux performances en temps et en espace. Par contraste, d'autres systèmes plus génériques, tels que LogicBlox [2], se reposent sur des solutions déclaratives qui couvrent une variété plus large de cas d'utilisation. Par ailleurs, des systèmes de représentation de connaissance, comme Virtuoso[2], implémentent le standard RDF pour les modèles de graphes et le langage de requête SPARQL pour gérer les requêtes navigationnelles et récursives dans toute leur complexité sur des systèmes de données du Web sémantique, même à grande échelle.

Suite à cette évolution, la génération de graphes et de requêtes synthétiques, mais aussi les solutions d'étalonnage pour les systèmes de bases de données ont proliféré. Ce mouvement a commencé au sein de la communauté Web sémantique [1,6] et plus récemment au sein de la communauté base de données, notamment suite aux travaux du conseil LDBC [5]. Ces deniers ont notamment été les premiers à proposer un système de génération de base de données permettant de comparer les systèmes de bases de données graphes, avec une approche se concentrant sur les goulots d'étranglements dans l'évaluation des requêtes et qui permet de régler finement les paramètres des données et des requêtes considérées. En se basant sur un schéma fixe et un ensemble de requêtes bien définies, cela a permis à la communauté de se concentrer sur les fonctionnalités cruciales en optimisation de requête et/ou en calcul parallèle, et a réduit la confusion et les résultats parfois non comparables des systèmes évalués.

Notre solution. Nous proposons une approche complémentaire dans laquelle le focus n'est pas dans la conception individuelle des requêtes mais plutôt sur une description globale du type de requêtes souhaitées. Cette approche globale permet un contrôle sur la diversité à la fois des instances de graphes et des requêtes générées, ce qui nous permet de faire varier à la fois les propriétés structurelles des données mais aussi d'adapter les requêtes générées à un domaine ou une application particulière. Nous montrons

[1] http://neo4j.com
[2] http://virtuoso.openlinksw.com

que cette approche basée sur une description globale des requêtes permet de générer des scénarios d'évaluation et de réaliser des expériences différentes de ceux proposés par les systèmes actuels qui sont centrées sur des requêtes individuelles. En effet, nous pensons que cette approche est importante dans les contextes où des requêtes multiples (mais suivant des mêmes propriétés globales) ont besoin d'être considérées ensembles. Cela apparaît dans de nombreuses situations, comme par exemple l'optimisation multi-requêtes, le paramétrage de base de données basées sur des schémas d'utilisation, la découverte de correspondance entre bases de données ou encore la réécriture de requêtes dans les systèmes d'intégration de données.

Nous présentons gMark, un système opérationnel qui offre un cadre pratique permettant de réaliser cette approche centrée sur un paramétrage global. gMark combine une approche de génération d'instances de graphes basées sur un système de schéma à la fois souple et précis, avec une méthode sophistiquée de génération de requêtes. Le système complet est disponible en *open-source*[3] [3] et est prêt pour être utilisé par la communauté. Nous notons par ailleurs que le coeur du système de génération de requêtes est basé sur une méthode novatrice que nous proposons ici et qui permet d'estimer la sélectivité d'une requête de graphe. C'est, à notre connaissance, la première de ce genre. Nous pensons que cette méthode a par ailleurs un intérêt indépendant qui peut s'appliquer à d'autres contextes tels que l'optimisation de requêtes, l'inférence de requêtes, ou plus généralement à l'analyse des requêtes dans les données.

Principes de conception. Afin de permettre un large spectre de systèmes et de domaines, gMark vise à couvrir un maximum de fonctionnalités rencontrées couramment dans les systèmes de gestion de requêtes de graphes, d'analyse de graphes et de validation. L'interdépendance entre les caractéristiques clés suivantes caractérise l'architecture du gMark qui est, à notre connaissance, le premier système à satisfaire tous ces critères.

Support intégré de définition de schéma. Un but majeur de gMark est d'offrir une large diversité de contraintes de schéma définies par l'utilisateur dans la génération de graphes. En général, gMark est indépendant du domaine, bien qu'il puisse être utilisé pour simuler une grande variété de domaines réalistes. Ainsi, la génération des instances de graphes passe par la description d'un schéma optionnel, que nous appelons une *configuration de graphe*, qui inclut l'énumération des prédicats (i.e. les étiquettes des arêtes du graphe), et des types de noeuds (i.e. étiquettes de noeuds) qui peuvent apparaître dans le graphe, avec leurs différentes propriétés dans les instances générées.

Contrôle de la diversité de l'instance de graphe et des requêtes. Etant donné une configuration de graphe, un des défis est d'exploiter cette information pour générer des requêtes. Les approches actuelles en génération de requêtes se reposent sur l'instance de graphe pour générer des requêtes ayant le comportement désiré. Néanmoins, cette approche n'est pas faisable pour des graphes de taille importante ou moins bien structurés. Nous argumentons que la génération de requêtes doit venir avant tout de la configuration du graphe plutôt que de l'instance elle-même afin de maîtriser au mieux le comportement des requêtes générées. De ce point de vue, gMark supporte un large ensemble de paramètres pour à la fois le générateur de graphes et le générateur de requêtes. Par exemple, la capacité de juger des performances dans l'exécution de requêtes navigationnelles et récursives, sans dépendre d'un ensemble fixe de motifs de requêtes. A notre connaissance, les solutions existantes ne supportent pas un ensemble aussi configurable – et aisément extensible – de paramètres.

Indépendance vis-à-vis du langage et du système. Un autre principe de conception important de gMark est son indépendance par rapport aux langages de requêtes. En effet, gMark supporte différents formats de sortie, à la fois pour les graphes mais aussi pour les requêtes, incluant N-triples pour les données, mais aussi SQL, SPARQL, Datalog et openCypher pour les langages de requêtes. gMark est également extensible facilement à d'autres formats de sorties.

Large champ applicatif. Enfin, gMark vise à supporter un large champ de domaines applicatifs. Par exemple, nous montrons qu'il est facile d'adapter les scénarios de trois autres systèmes de générations (modulo les fonctionnalités communes) afin d'obtenir des configurations pour gMark, auxquelles nous pouvons ajouter des fonctionnalités propres à gMark : le réseau social de LDBC [5], les données du générateur SP2Bench [6], et le jeu de test SPARQL de Waterloo (WatDiv) [1]. Dans l'article complet, nous discutons en détail de l'expressivité de gMark vis-à-vis de ces scénarios.

Contributions. Notre article gMark [4] présente la conception à la fois théorique et pratique de notre approche, ainsi qu'une première analyse empirique. Nos contributions majeures incluent les points suivants.

• Nous formalisons le problème de la génération de graphes et de la description globale d'un jeu de requêtes. Nous montrons que le problème de génération d'un graphe satisfaisant un ensemble de contraintes n'est en général pas satisfaisable.

• Nous fournissons une présentation en profondeur des principes de conception de gMark à la fois pour la génération des graphes et des ensembles de requêtes, la notion la plus notable étant le support pour les requêtes récursives et l'estimation de la sélectivité des requêtes générées.

• Nous montrons empiriquement les capacités de gMark à couvrir diverses types de graphes et d'ensembles de requêtes, indiquant notamment la précision de l'estimation de la sélectivité et le passage à l'échelle du générateur.

• Nous présentons une analyse expérimentale d'une sélection de moteurs de requêtes de graphes représentative de l'état de l'art en utilisant gMark. Cette comparaison permet de mettre en lumière d'importantes limitations des moteurs de requêtes actuels, notamment en matière de gestion des requêtes récursives.

[3] https://github.com/graphMark/gmark

2. REFERENCES

[1] G. Aluç and et al. Diversified stress testing of RDF data management systems. In *ISWC*, pages 197–212, 2014.

[2] M. Aref and et al. Design and implementation of the LogicBlox system. In *SIGMOD*, pages 1371–1382, 2015.

[3] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat. Generating flexible workloads for graph databases. *PVLDB*, 9(13):1457–1460, 2016.

[4] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat. gMark: Schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering*, 2017 (à paraître). http://arxiv.org/abs/1511.08386.

[5] O. Erling and et al. The LDBC social network benchmark: Interactive workload. In *SIGMOD*, pages 619–630, 2015.

[6] M. Schmidt and et al. SP2Bench: A SPARQL performance benchmark. In *ICDE*, pages 222–233, 2009.

Un partitionnement d'arêtes à base de blocs pour les algorithmes de marche aléatoire dans les grands graphes sociaux.

Yifan LI
LIP6, Université Paris 6
Paris, France
yifan.li@lip6.fr

Camelia Constantin
LIP6, Université Paris 6
Paris, France
camelia.constantin@lip6.fr

Cedric du Mouza
CEDRIC Lab. - CNAM
Paris, France
dumouza@cnam.fr

Les algorithmes basés sur les marches aléatoires, tels que *personalized PageRank* [6] et *personalized SALSA* [3], se sont révélés efficaces pour les systèmes de recommandations personnalisés grâce à leur faculté de passer à l'échelle. Certaines propositions récentes s'appuient par ailleurs sur des marches aléatoires multiples démarrant de *chaque sommet* du graphe, par exemple l'algorithme *Fully Personalized PageRank* utilisant l'approximation de MonteCarlo [2].

Le partitionnement de graphe est d'un intérêt primordial dans le domaine de la recherche du traitement distribué de graphe. Il joue un rôle de plus en plus important à la fois dans les calculs centrés sur les sommets, comme dans le modèle de *Pregel*, et dans l'évaluation de requêtes. Les résultats récents montrent que le partitionnement des arêtes (vertex-cut) s'avère être plus efficace [4, 5] que le partitionnement des sommets (edge-cut) traditionnellement utilisé pour les calculs sur des graphes réels comme les graphes des réseaux sociaux. Par conséquence, plusieurs systèmes de calcul de graphe basés sur cette approche ont été proposés, tels que PowerGraph (GraphLab2) [5] et GraphX [8]. Toutefois, leurs stratégies de partitionnement de graphe sont génériques et ne dépendent pas des algorithmes utilisés pour les différents calculs.

Ils distribuent ainsi les arêtes uniformément dans les partitions soit de manière aléatoire, c'est-à -dire en utilisant une fonction de hachage sur les identifiants des sommets comme dans Giraph [1] et GraphX, ou grâce à un algorithme gourmand ou dynamique comme dans PowerGraph et GPS [7]. Par ailleurs, contrairement aux algorithmes comme PageRank, où les messages transmis entre les sommets ne sont que des scores ou des valeurs de rang (et donc sont de petites tailles), l'exécution d'algorithmes tels que des marches aléatoires multiples a un coût de communication plus important puisque (i) certaines informations supplémentaires sur les chemins des marches aléatoires doivent également être transmis (donc les messages sont de taille plus importante) et (ii) plus d'un message (marche) débute à partir de chaque sommet simultanément. Dans cette situation, la réduction du nombre de communications est cruciale pour la performance du calcul. C'est particulièrement vrai dans un cluster informatique de grande taille avec une infrastructure complexe de réseau machine-machine, ou dans un système de calcul à mémoire distribuée comme Spark où la communication pourrait être un goulet d'étranglement pour atteindre de bonnes performances.

Nous proposons dans cet article une nouvelle stratégie de partitionnement de graphe (de ses arêtes) basée sur des blocs, qui fournit une distribution équilibrée des arêtes et réduit les coûts de communication pour les calculs utilisant des marches aléatoires. A notre connaissance, il s'agit de la première fois qu'une stratégie de partitionnement dédiée à des algorithmes de marches aléatoires est proposée dans le modèle de Pregel. Enfin, les expériences montrent que notre partitionnement a apporté des améliorations significatives concernant les coûts de communication et les temps d'exécution des différents algorithmes s'appuyant sur des marches aléatoires.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

Stratégie de partitionnement par blocs

Dans l'approche de partitionnement d'arêtes, un sommet est éventuellement alloué à plusieurs partitions et les communications entre les partitions se produisent lors de la mise à jour des différentes répliques (miroirs des sommets) à chaque super-étape de Pregel. Par conséquent, le facteur de réplication des sommets (VRF) est souvent utilisé comme mesure de communication.

Par ailleurs dans les réseaux sociaux, il existe plusieurs clusters (communautés) et notre objectif est de tirer partie de cette caractéristique topologique lors de notre construction de blocs.

Un bloc correspond à un sous-graphe fortement connecté, par exemple une communauté dans le réseau social. Dans l'approche de Pregel, nous considérons le bloc comme un ensemble d'arêtes qui sont "proches" d'une autre, et ces blocs deviennent les unités constitutives de chaque partition dans le calcul, mais aussi les unités d'allocation pour la charge de travail sur les machines. Nous proposons de calculer un ensemble de K blocs en explorant le graphe. Une arête est allouée à un bloc en fonction de sa distance par rapport à ce bloc.

Pour la distance nous adaptons l'inverse P-distance de chaque sommet. La P-distance permet de mesurer la connectivité: plus nombreux et plus courts sont les chemins entre deux sommets, plus ils sont proches topologiquement dans ce graphe.

La distance $dist_v(v_1, v_2)$ depuis un sommet v_1 à un som-

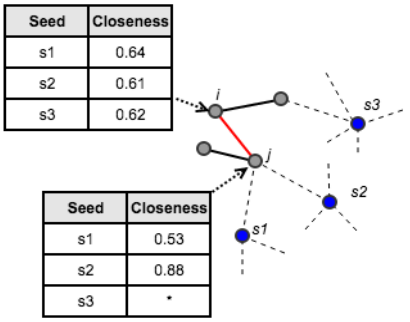


Figure 1: Exemple d'allocation d'arêtes

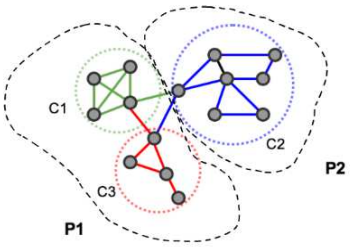


Figure 2: Exemple de blocs et de partitions

met v_2 dans un graphe dirigé est alors définie par:

$$dist_v(v_1, v_2) = \sum_{p \in P_{v_1 v_2}} (1-\alpha)^{k_p} \cdot \prod_{i=0}^{k_p - 1} \frac{1}{outDeg(v_{p,i})} \quad (1)$$

où $P_{v_1 v_2}$ représente l'ensemble des chemins entre v_1 et v_2, k_p et la longueur du chemin p, $v_{p,i}$ est le sommet i du chemin p et $\alpha \in (0, 1)$ est la probabilité de téléportation, *c.à.d.*, la probabilité de retourner au sommet original et $outDeg(v_{p,i})$ est le degré sortant du sommet $v_{p,i}$.

Un score agrégé des distances des deux sommets d'une arête par rapport à un seed nous donne le score de distance d'une arête par rapport à ce seed.

Notre processus d'allocation consiste d'abord en une exploration du graphe en largeur (BFS) à partir d'un ensemble prédéfini de K seeds. Pour chaque arête rencontrée nous mettons à jour sa distance par rapport à tous les blocs. Quand l'étape d'exploration est terminée, nous allouons chaque arête au bloc le plus proche.

EXAMPLE 1. *Considérons l'exemple de la figure1. Supposons que nous ayons déjà calculé les distances des sommets i et j de l'arête e par rapport à chacun des 3 seeds s_1, s_2 et s_3. Observons que la valeur '*' signifie que le sommet ne peut pas être atteint par le seed s_3 lors de l'étape de BFS. Nous faisons la somme des scores pour i et j par rapport à chaque seed et on obtient les scores suivants, 0.64+0.53, 0.61+0.88 et 0.62+0.0, soit 1.17, 1.49 et 0.62 pour respectivement les seeds s_1, s_2 et s_3. Par conséquent on choisira d'allouer l'arête e au seed s_2.*

Algorithmes de fusion et d'éclatement des blocs

Eclatement d'un bloc.

Puisque l'allocation des arêtes aux blocs n'est basée que sur un critère de distance, certains blocs peuvent avoir une taille excédant la taille maximale autorisée pour une partition. Par conséquent, nous proposons une stratégie d'éclatement simple d'un bloc. Supposons que la taille d'une partition p_i est $(\beta-1)\lambda\frac{|E|}{P} \leq |Edge(p_i)| < \beta\lambda\frac{|E|}{P}$. Nous appliquons alors notre algorithme de construction de blocs à la partition p_i avec de nouveaux seeds pour la diviser en sous-blocs. Nous pouvons éventuellement itérer le processus pour n'importe lequel des sous-blocs qui dépasse la taille de la partition.

Fusion de blocs.

Notre construction de blocs peut également entraîner la production de certains blocs dont la taille est inférieure à la taille minimale (em c-à-d $eta\frac{|E|}{P}$. Pour un tel bloc, nous réaffectons ses arêtes sans considérer son seed d'origine. Observons que cette stratégie peut conduire à des éclatements de blocs.

Allocation des blocs aux partitions.

Nous avons adopté une allocation de blocs en tenant compte uniquement du critère de taille du bloc, pour obtenir un partitionnement équilibré. Nous proposons un algorithme gourmand simple mais efficace. Les blocs sont triés par taille décroissante. Nous attribuons le plus grand bloc non-alloué à la partition dont la taille actuelle est la plus petite. Nous itérons cette stratégie jusqu'à ce que tous les blocs soient alloués. Par conséquent, cette allocation est en $O(|\mathcal{B}|)$ où $\mathcal{B}$ représente l'ensemble des blocs.

EXAMPLE 2. *La figure 2 présente un exemple illustrant notre approche en 2 étapes. Tout d'abord, nous regroupons les arêtes en fonction de leur distance par rapport aux seeds (ici trois seeds). Nous obtenons trois communautés (blocs d'arêtes): c1, c2 et c3. Ensuite nous fusionnons les blocs pour obtenir des partitions de taille similaire. Dans cet exemple nous construisons la partition P1, composée de c1 et c3, et la partition P2 correspond au seul bloc c2.*

1. REFERENCES

[1] Apache. Giraph.

[2] B. Bahmani, K. Chakrabarti, and D. Xin. Fast Personalized PageRank on MapReduce. In *SIGMOD*, pages 973–984, 2011.

[3] B. Bahmani, A. Chowdhury, and A. Goel. Fast Incremental and Personalized PageRank. *PVLDB*, 4(3):173–184, 2010.

[4] F. Bourse, M. Lelarge, and M. Vojnovic. Balanced Graph Edge Partition. In *KDD*, pages 1456–1465, 2014.

[5] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In *OSDI*, pages 17–30, 2012.

[6] G. Jeh and J. Widom. Scaling Personalized Web Search. In *WWW*, pages 271–279, 2003.

[7] S. Salihoglu and J. Widom. GPS: a Graph Processing System. In *SSDBM*, pages 22:1–22:12, 2013.

[8] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. GraphX: a Resilient Distributed Graph System on Spark. In *GRADES*, page 2, 2013.

RDF Query Relaxation Strategies Based on Failure Causes*

Geraud Fokou
LIAS/ISAE-ENSMA & University of Poitiers
1, Avenue Clement Ader
86960 Futuroscope Cedex, France
geraud.fokou@ensma.fr

Stephane Jean
LIAS/ISAE-ENSMA & University of Poitiers
1, Avenue Clement Ader
86960 Futuroscope Cedex, France
stephane.jean@ensma.fr

Allel Hadjali
LIAS/ISAE-ENSMA & University of Poitiers
1, Avenue Clement Ader
86960 Futuroscope Cedex, France
allel.hadjali@ensma.fr

Mickael Baron
LIAS/ISAE-ENSMA & University of Poitiers
1, Avenue Clement Ader
86960 Futuroscope Cedex, France
mickael.baron@ensma.fr

1. INTRODUCTION

Recent projects like DBPedia [11] or Knowledge Vault [3] have created Knowledge Bases (KBs) with millions of facts represented in the RDF format. Despite their large size, KBs face a significant amount of incomplete factual knwoledge, which makes query answering over them often unsuccessful. For instance, a recent study on SPARQL endpoints [12] shows that ten percent of the submitted queries between May and July 2010 over DBpedia returned empty answers.

Relaxation of the failing queries is one of the cooperative techniques used to retrieve alternative results to serve the users' needs. In the context of RDF, current approaches generate multiple relaxed queries using different techniques such as logical relaxation based on RDFS entailment and RDFS ontologies [1,5,9,10], query rewriting rules [2], statistical language models [4] or matching functions [8]. Most of these approaches compute the similarities between the obtained relaxed queries and the user failing query and then proceed to the execution of the relaxed queries in a similarity-based ranking order. A major drawback of the above approaches is the fact they relax the user query without knowing its *failure causes (FCs)*.

In our previous work [6], we have addressed the issue of finding the FCs of an RDF failing query by computing a set of *Minimal Failing Subqueries (MFSs)* and argued that they provide the user with a clear explanation about the reasons of the empty answer retrieved. In this paper, we investigate the idea of using MFSs for the purpose of query relaxation process. The main idea is that MFSs can speed up this relaxation process by avoiding executing relaxed queries that still contain one or several FCs. This approach applies both for the user query, as well as for the failing relaxed queries. However, as enumerating the MFSs of a query is an NP-Hard problem [7], identifying them could be sometimes disadvantageous since their computation time may be greater than the execution time of the relaxed queries avoided thanks to them. Thus, we show that there is a tradeoff between not knowing any MFS and identifying the MFSs of each relaxed query. To do so, we propose three strategies that leverage different levels of information about the MFSs of the user query and its relaxed queries as well. The first is MFS Based Search (MBS), the second is Optimized MFS-Based Search (O-MBS) and the last is Full MFS-Based Search (F-MBS).

1.1 MFS-Based Search (MBS)

This strategy is based on the following assertion: Let Q' be a relaxed query of Q. If Q' does not relax at least one triple pattern of each MFS of Q, then Q' is failing.

PROOF. If there is one MFS of Q denoted Q^* such as none of its triple patterns has been relaxed in Q', then $Q^* \subseteq Q'$. A query that contains a failing query, also fails. By definition of an MFS, Q^* is a failing query. Thus Q' is also failing. □

So, the MFSs of the failing query allow us to identify some relaxed queries that will necessarily fail.

1.2 Optimized MFS-Based Search (O-MBS)

In the previous approach, we only use the MFSs of the initial query to prune the search space. The idea behind the *Optimized MFS-Based Search* (O-MBS) is that the MFSs of the initial query Q gives some clues on the MFSs of a relaxed query Q' of Q. Intuitively, a relaxed query Q' of Q fails if and only if at least one MFS of Q has not been repaired in Q' or if there is a failing sub-query in Q' that was not minimal in Q. More formally, let M_Q be an MFS of Q, we denote by $M_Q^{\uparrow Q'}$ the query that corresponds to M_Q in Q'. By extension, we denote by $mfs^{\uparrow Q'}(Q)$ the queries corresponding to the MFSs of Q in Q' and for any MFS $M_{Q'}$ of Q' there is an MFS M_Q of Q such that $M_Q^{\uparrow Q'} \subseteq M_{Q'}$.

PROOF. Let $M_{Q'}$ be an MFS of Q'. By definition $M_{Q'}$ is failing. As $M_{Q'}^{\uparrow Q} \subseteq M_{Q'}$, $M_{Q'}^{\uparrow Q}$ is also failing. So, $M_{Q'}^{\uparrow Q}$ contains an MFS M_Q of Q and thus $M_Q^{\uparrow Q'} \subseteq M_{Q'}$. □

*The complete version of this paper has been published in the 13th European Semantic Web Conference (ESWC 2016), May 31th - June 2th, 2016, Heraklion, Crete, Greece

BDA 15 au 18 novembre 2016, Futuroscope, Poitiers - France.

Thus, if an MFS has been repaired, there can still be some queries that include this repaired MFS and fail. Identifying these new MFSs is not easy. Indeed, the number of queries that include the repaired MFSs is exponential in terms of the number of query triple patterns. Thus, the O-MBS strategy is only based on the MFSs that are not repaired.

1.3 Full MFS-Based Search (F-MBS)

In the previous strategy, all the MFSs of an explored relaxed query are not necessarily discovered. This strategy proposes to compute the complete set of MFSs. Here, we want to evaluate if it is worth computing the set of MFSs of each node of the query relaxation graph, i.e., if this computation time is acceptable in comparison with the number of relaxed queries that are pruned thanks to the discovered MFSs. This strategy called *Full MFS-Based Search* (F-MBS) is based on the two following assertions:

1. If all the queries $Q \in mfs^{\uparrow Q'}(Q)$ are failing, then $mfs(Q') = mfs^{\uparrow Q'}(Q)$.
2. Any MFS $M_{Q'}$ of Q' contains the triple patterns that are shared by the queries of $mfs^{\uparrow Q'}(Q)$

Thanks to these assertions, F-MBS extends the O-MBS strategy as follows. For each relaxed query Q', we execute all the MFSs of $mfs^{\uparrow Q'}(Q_0)$, where Q_0 is the last explored query such that Q' relax Q_0. If all these queries are failing, then $mfs(Q') = mfs^{\uparrow Q'}(Q_0)$. Otherwise, we execute an optimized version of the LBA algorithm [6] to find the MFSs of Q'. As in the previous strategies, the queries that include at least one of the identified MFSs of Q' are pruned from the query relaxation graph.

2. CONCLUSION AND DISCUSSION

In this paper, we have proposed three strategies to relax an RDF query. Their originalities is that they use different levels of information about the FCs of the initial query and its relaxed queries. In the first strategy, named MBS, only the FCs of the initial query are used to prune from the search space all the relaxed queries that include FCs. The second strategy named O-MBS extends the previous one by searching the FCs that remain in the relaxed query. In this strategy, all the FCs of a relaxed query are not necessarily discovered. Our last strategy named F-MBS fills this gap by using an optimized version of a previous work algorithm to find the FCs of the relaxed queries.

We have run several experiments on the LUBM benchmark with two triplestores to compare these strategies with a state-of-the-art strategy, which consists in executing the relaxed queries in their ranking order. In these experiments, our best strategy O-MBS outperforms it by more than a factor of 2. O-MBS is a good compromise between MBS, which often does not use enough information about the FCs, and F-MBS which uses too much information about them.

This paper opens many perspectives. As our approach is defined for conjunctive RDF queries, we plan to extend it to support other SPARQL queries. Studying the relevance of our strategies when they are applied on other query relaxation models that use different relaxation operators is another perspective. As our strategies gather a lot of information about the failure of many queries, we intend to design an interactive approach based on our strategies. Finally, we plan to investigate whether the FCs of a query could be used in conjunction with other cooperative techniques that aim at handling the empty-answer problem.

3. REFERENCES

[1] A. Calí, R. Frosini, A. Poulovassilis, and P. Wood. Flexible querying for sparql. In *ODBASE'14*, volume 8841, 2014.

[2] P. Dolog, H. Stuckenschmidt, H. Wache, and J. Diederich. Relaxing rdf queries based on user and domain preferences. *IJIIS*, 33(3):239–260, 2009.

[3] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *ACM SIGKDD*, pages 601–610, 2014.

[4] S. Elbassuoni, M. Ramanath, and G. Weikum. Query relaxation for entity-relationship search. In *ESWC'11*, pages 62–76, 2011.

[5] G. Fokou, S. Jean, and A. Hadjali. Endowing semantic query languages with advanced relaxation capabilities. In *ISMIS*, pages 512–517, Roskilde, Denmark, 2014.

[6] G. Fokou, S. Jean, A. Hadjali, and M. Baron. Cooperative techniques for SPARQL query relaxation in RDF databases. In *ESWC'15, Portoroz, Slovenia, Proceedings*, volume 9088 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2015.

[7] P. Godfrey. Minimization in Cooperative Response to Failing Database Queries. *International Journal of Cooperative Information Systems*, 6(2):95–149, 1997.

[8] A. Hogan, M. Mellotte, G. Powell, and D. Stampouli. Towards fuzzy query-relaxation for rdf. In *ESWC'12*, pages 687–702, 2012.

[9] H. Huang, C. Liu, and X. Zhou. Approximating query answering on rdf databases. *World Wide Web*, 15(1):89–114, January 2012.

[10] C. A. Hurtado, A. Poulovassilis, and P. T. Wood. Query Relaxation in RDF. *Journal on Data Semantics X*, 10:31–61, 2008.

[11] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, March 2015.

[12] M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, and A. N. Ngomo. LSQ: the linked SPARQL queries dataset. In *The Semantic Web - ISWC*, pages 261–269, 2015.

Optimising SPARQL query processing in distributed knowledge graphs

Abdoul Macina
Univ. Nice Sophia Antipolis
I3S laboratory
Sophia Antipolis, France
macina@i3s.unice.fr

Johan Montagnat
CNRS
I3S laboratory
Sophia Antipolis, France
johan.montagnat@cnrs.fr

Olivier Corby
Inria
I3S laboratory
Sophia Antipolis, France
olivier.corby@inria.fr

Abstract. An increasing number of linked knowledge bases are openly accessible over the Internet. Distributed Query Processing (DQP) techniques are developed to query multiple knowledge bases coherently. However, a precise semantics has to be defined to specify the precise expected results, and query performance issues arise.

In this paper, we define SPARQL DQP over distributed RDF graphs. We propose a SPARQL and RDF-compliant DQP semantics. We improve performance through query planning heuristics that generate Basic Graph Pattern-based sub-queries designed to maximise the parts of the query processed by the remote endpoints. We evaluate our DQP engine considering a query set representative of most common SPARQL clauses and different data distribution schemes. Results show a significant reduction of the number of remote queries executed and the query execution time while preserving completeness.

SPARQL Query Processing with Apache Spark

[Extended Abstract]

Hubert Naacke
Sorbonne Universités, UPMC
Univ Paris 06, UMR 7606,
LIP6
Paris, France
hubert.naacke@lip6.fr

Olivier Curé
LIGM (UMR 8049), CNRS,
ENPC, ESIEE, UPEM,
Marne-la-Vallée, France.
olivier.cure@u-pem.fr

Bernd Amann
Sorbonne Universités, UPMC
Univ Paris 06, UMR 7606,
LIP6
Paris, France
bernd.amann@lip6.fr

ABSTRACT

The number and size of linked open data graphs keep growing at a fast pace and imposes semantic RDF data services and triple stores to adopt "big data" solutions. Distributed query processing is one of these solutions for guaranteeing scalability, high availability and fault tolerance. Distributed RDF triple store implementations are rarely built from scratch but are rather designed on top of an existing big data processing infrastructure. Following this line of work, we propose and compare five SPARQL query processing approaches with different distributed join algorithms on top of Apache Spark, a state of the art cluster data processing engine. Experimentation on real-world and synthetic data sets emphasizes that our two new approaches are outperforming the other ones on all major query shapes, *i.e.*, star, snowflake, chain and their composition.

1. INTRODUCTION

Due to the Linked Open Data (LOD) initiative, new RDF content is continuously created to enrich publicly available data sets containing several billions triples. This evolution generates an increasing demand for querying very large RDF data sets efficiently, using the SPARQL query language. SPARQL queries are often part of a more complex data analysis workflow which could benefit from data sharing across the workflow steps. Multiple research and development efforts aim to design SPARQL query processing solutions that smoothly integrate into general multi-purposes cluster computing platforms, such as Apache Spark, to enable unified data management, to benefit from scalable and fault-tolerant data manipulation primitives, and to leverage on recent advances in high performance computing.

In this article, we address the problem of processing SPARQL queries over large RDF data sets on top of Apache Spark, a modern scalable data processing cluster platform. Remote data access is the major bottleneck in such environments since it implies important communication overhead compared to local data access. Many cluster-based solutions [4, 2] therefore aim to improve data locality by distributing and replicating data such that most of the query operations evaluate locally without any transfer. Related works rely on exhaustive indexing techniques [5] and complex data partitioning algorithms *e.g.*, graph clustering, and assume a priori knowledge of the query workload to organize the data with respect to the most frequent queries. On the opposite, our solution trades locality for a shorter data preparation delay. We avoid replication, indexing, complex partitioning, and data maintenance. Consequently, our solution puts less pressure on the memory space such that (i) the data set is partitioned without replication, (ii) the time to load a new data set into the platform and prepare it for querying purpose is shorten, (iii) the query plan is composed of the primitive operators that the platform provides, and (iv) the query plan generates few data transfers.

Our contributions are as follows: (i) a formalization for evaluating the cost of SPARQL query processing in a distributed setting, (ii) the design and comparison of five Spark-based SPARQL query processing solutions using our framework, (iii) an evaluation and validation of our framework using real-world and synthetic data sets. To the best of our knowledge, this is the first in-depth analysis of SPARQL query processing with Apache Spark at this level of detail.

2. DISTRIBUTED SPARQL PROCESSING

We consider a data set of $(subject, property, object)$ triples. To reduce the data preparation overhead, we distribute the data using hash based partitioning on the triples' subject. Therefore two triples having the same subject are stored at the same cluster node. For example, on Figure 1, for a person, say P_2, the three triples about his lab, his age, and his name are stored together. Subject partitioning allows for local join on the subject. For instance, join on the $?P$ variable can be processed without any data transfer.

In general, the query plan for evaluating a basic graph pattern query such as a chain, snowflake or complex query shown on Figure 1 is a tree of join operations. To evaluate a binary join in a cluster computing platform, there are two commonly used algorithms: (i) the partitioned join ($Pjoin$) implies to distribute the two data sets, whereas (ii) the broadcast join ($Brjoin$) transfers the smaller data set to every compute nodes, in order to access the larger one locally. Our solution is based on communication cost estimations for choosing the best join algorithm to use for evaluating the joins in a given query plan. Based on a for-

BDA 2016, 15 au 18 Novembre, Poitiers, France.

mal data-transfer cost model, we apply a greedy strategy to build a query plan in two steps: the first step groups star patterns into transfer-free (local) joins, which are iteratively composed in the second step by choosing at each iteration, the join algorithm with minimal cost.

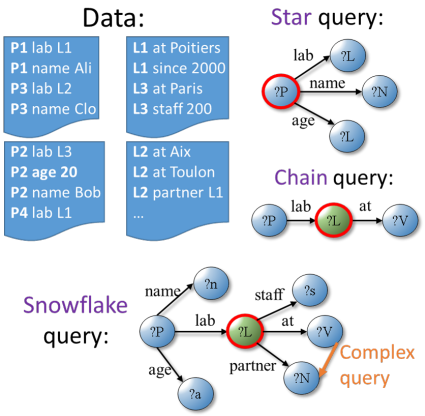


Figure 1: Hash-based data partitioning and query patterns

3. SPARQL ON SPARK

We present an in depth investigation of five alternative methods to process SPARQL queries on top of Apache Spark, each method depending on a particular Spark data layer (DF, RDD). Figure 2 shows the query processing features supported by each method. The Spark implementations of the first three methods are rather straightforward but have limited support for query optimization and fail to build query plans containing successive broadcast joins (*Brjoin*). The main drawback is that this yields rather inefficient plans for chain queries, as we have demonstrated it based on our cost model.

The last two methods, labeled as *hybrid*, are part of our contributions. They support general query plans that combine both types of joins (*Pjoin* and *Brjoin*). For these methods we also have optimized the local data access through an efficient merged selection strategy. Finally, the last method complies with Spark built-in data compression which further reduces data transfers.

	Method	co-partitioning	Join plan	Merged selection	Query Optimizer	Data Compression
Spark interface	SPARQL RDD		Pjoin			
	SPARQL DF	v 1.5	Pjoin,BrJoin1		poor	
	SPARQL SQL	v 1.5	Pjoin,BrJoin1		cross prod	
Our solutions	**Hybrid RDD**		Pjoin,BrJoin+		cost based	
	Hybrid DF		Pjoin,BrJoin+		cost based	

supported | not supported

Figure 2: Qualitative comparison of five SPARQL on Spark methods

4. EXPERIMENTAL VALIDATION

We conducted a series of experiments with real and synthetic data sets of medium/large size over a 17 nodes (300 cores) cluster. We executed the snowflake query named Q_8 from the LUBM benchmark [3] over the medium (100M triples) and the large (1 billion triples) data sets. Figure 3 shows that our solution achieves higher gain for larger datasets. The gain on response time is up to 3 times for compressed (*i.e.*, DF) data, and 4.7 times for uncompressed (*i.e.*, RDD) data. We also compared our solution with the state-of-the-art S2RDF [5] solution, using the WatDiv benchmark [1] along with the corresponding 1 billion triples data set. We found that our solution brings an additional performances gain of 71% for star queries, 32% for snowflake queries, and 214% for complex queries. See the companion web site[1] for more details.

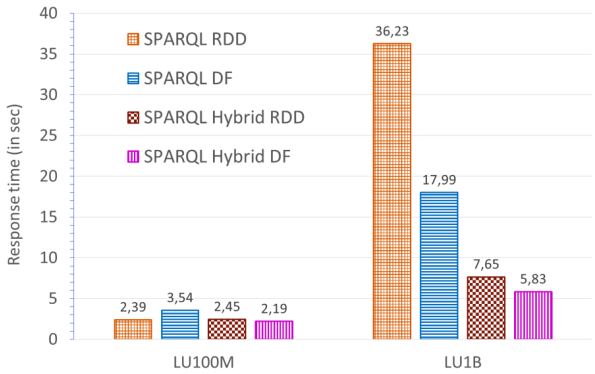


Figure 3: Performance benefit for snowflake query Q_8 from the LUBM Benchmark [3]

5. REFERENCES

[1] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of RDF data management systems. In *Int'l Semantic Web Conf. (ISWC)*, pages 197–212, 2014.

[2] F. Goasdoué, Z. Kaoudi, I. Manolescu, J. Quiané-Ruiz, and S. Zampetakis. Cliquesquare: Flat plans for massively parallel RDF queries. In *IEEE Int'l Conf. on Data Engineering (ICDE)*, pages 771–782, 2015.

[3] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[4] J. Huang, D. J. Abadi, and K. Ren. Scalable SPARQL querying of large RDF graphs. *Proc. of the VLDB Endowment*, 4(11):1123–1134, 2011.

[5] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, and G. Lausen. S2RDF: RDF querying with SPARQL on spark. *Proc. of the VLDB Endowment*, 9(10), 2016.

[1]https://sites.google.com/site/sparqlspark/home

Towards Differentially Private Community Detection

[Extended Abstract]

Hiep H. Nguyen
LORIA/INRIA Nancy
France
huu-hiep.nguyen@inria.fr

Abdessamad Imine
LORIA/INRIA Nancy
France
abdessamad.imine@loria.fr

Michaël Rusinowitch
LORIA/INRIA Nancy
France
michael.rusinowitch@inria.fr

ABSTRACT

Complex networks usually expose community structure with groups of nodes sharing many links with the other nodes in the same group and relatively few with the nodes of the rest. This feature captures valuable information about the organization and even the evolution of the network. Over the last decade, a great number of algorithms for community detection have been proposed to deal with increasingly complex networks. However, the problem of doing this in a private manner is rarely considered.

In this paper, we solve this problem under differential privacy, a prominent privacy concept for releasing private data. We propose several schemes to tackle them from two perspectives : input perturbation and algorithm perturbation. We choose Louvain method as the back-end community detection for input perturbation schemes and propose the method LouvainDP. For algorithm perturbation, we design ModDivisive using exponential mechanism with the modularity as the score. We have thoroughly evaluated our techniques on real graphs of different sizes and verified that they outperform the state-of-the-art.

1. DESCRIPTION

Graphs represent a rich class of data observed in daily life where entities are described by nodes and their connections are characterized by edges. Apart from microscopic (node level) and macroscopic (graph level) configurations, many complex networks display a mesoscopic structure, i.e. they appear as a combination of components fairly independent of each other. These components are called *communities*, modules or clusters and the problem of how to reveal them plays a significant role in understanding the organization and function of complex networks. Over the last decade, a great number of algorithms for community detection (CD) have been proposed to address the problem in a variety of settings (for a comprehensive survey, see [4]).

These approaches, however, are adopted in a non-private manner, i.e. a data collector (such as Facebook) knows all the contributing users and their relationships before running CD algorithms. The output of such a CD, in the simplest form, is a clustering of nodes. Even in this case, contributing users privacy may still be put at risk.

In this paper, we address the problem of CD from the perspective of differential privacy [3]. This privacy concept offers a formal definition of privacy with a lot of interesting properties : no computational/informational assumptions about attackers, data type-agnosticity, composability and so on [6]. As far as we know, the problem is quite new and only mentioned in the recent work [7] where Mülle et al. use a sampling technique to perturb the input graph so that it satisfies differential privacy before running the conventional CD algorithms. This technique (called *EdgeFlip*) is classified as *input perturbation* in differential privacy literature (the other two categories are *algorithm perturbation* and *output perturbation*). Similarly, *TmF* approach [9] can apply to the true graph to get noisy output graphs as in [7]. These schemes can be followed by any exact CD algorithm to get a noisy clustering satisfying differential privacy.

Our main contributions are the new schemes *LouvainDP* (input perturbation) and *ModDivisive* (algorithm perturbation). LouvainDP is a high-pass filtering method that randomly groups nodes into supernodes of equal size to build a noisy weighted supergraph. LouvainDP is guaranteed to run in linear time. ModDivisive is a top-down approach which privately divides the node set into the k-ary tree guided by the modularity score at each level. The main technique used in ModDivisive is the Markov Chain Monte Carlo (MCMC) to realize the exponential mechanism [5]. We show that ModDivisive's runtime is linear in the number of nodes, the height of the binary tree and the burn-in factor of MCMC. The linear complexity of LouvainDP and ModDivisive enables us to examine million-scale graphs in minutes. The experiments show the high modularity and low distortion of the output clusters by LouvainDP and ModDivisive.

1.1 Differential Privacy

Essentially, ϵ-differential privacy (ϵ-DP) [3] is proposed to quantify the notion of *indistinguishability* of neighboring databases. In the context of graph release, two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are neighbors if $V_1 = V_2$, $E_1 \subset E_2$ and $|E_2| = |E_1| + 1$. The definition of ϵ-DP for graph data is as follows.

DEFINITION 1.1. *A mechanism $\mathcal{A}$ is ϵ-differentially private if for any two neighboring graphs G_1 and G_2, and for any output $O \in Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(G_1) \in O] \leq e^{\epsilon} Pr[\mathcal{A}(G_2) \in O]$$

BDA 2016, 15 au 18 Novembre, Poitiers, France.

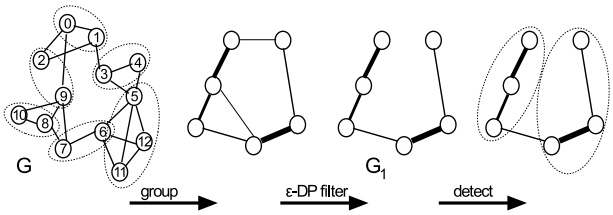


Figure 1: LouvainDP

Laplace mechanism [3] and Exponential mechanism [6] are two standard techniques in differential privacy. The latter is a generalization of the former. Laplace mechanism is based on the concept of *global sensitivity* of a function f which is defined as $\Delta f = \max_{G_1,G_2} ||f(G_1) - f(G_2)||_1$ where the maximum is taken over all pairs of neighboring G_1, G_2. Given a function f and a privacy budget ϵ, the noise is drawn from a Laplace distribution $Lap(\lambda) : p(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$ where $\lambda = \Delta f/\epsilon$.

For non-numeric data, the exponential mechanism is a better choice. Its main idea is based on sampling an output O from the output space $\mathcal{O}$ using a score function u. This function assigns exponentially higher probabilities to outputs of higher scores. Let the global sensitivity of u be $\Delta u = \max_{O,G_1,G_2} |u(G_1, O) - u(G_2, O)|$.

1.2 Community Detection in Graphs

There is a vast literature on community detection in graphs [4]. *Modularity* [8] is a popular quality metrics of community detection. It is based on the idea that a random graph is not expected to have a modular structure, so the possible existence of clusters is revealed by the comparison between the actual density of edges in a subgraph and the density one would expect to have in the subgraph if the nodes of the graph were connected randomly. *Louvain* method by Blondel et al. [1] is a top performance scheme for modularity maximization. It scales very well to graphs with hundreds of millions of nodes.

1.3 LouvainDP

The basic idea of LouvainDP is to create a noisy weighted graph G_1 from G by grouping nodes into supernodes of equal size k. Then we apply the filtering technique of Cormode et al. [2] to ensure only $O(m)$ noisy weighted edges in G_1. Finally, we run the exact Louvain method on G_1.

Figure 1 illustrates the three main steps of LouvainDP. Given the fact that Louvain method runs empirically in linear time [1], LouvainDP also has a time complexity of $O(m)$.

1.4 ModDivisive

In contrast with the agglomerative approaches (e.g. Louvain method) in which small communities are iteratively merged if doing so increases the modularity, our ModDivisive is a divisive algorithm in which communities at each level are iteratively split into smaller ones. Our goal is to heuristically detect cohesive groups of nodes in a private manner. There are several technical challenges in this process. The first one is how to efficiently find a good split of nodes that induces a high modularity and satisfies ϵ-DP at the same time. The second one is how to merge the small groups to larger ones. We cope with the first challenge by realizing an exponential mechanism via MCMC (Markov Chain Monte-Carlo) sampling with the modularity as the score function.

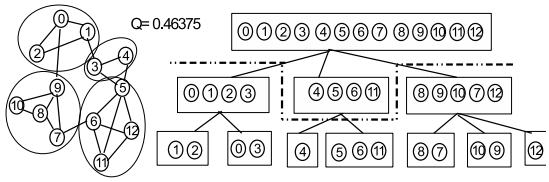


Figure 2: Example of ModDivisive with $k = 3$

The second challenge is solved by dynamic programming.

We design ModDivisive as a *k-ary* tree (Fig.2), i.e. each internal node has no more than k child nodes. The root node (level 0) contains all nodes in V and assigns arbitrarily each node into one of the k groups. Then we run the MCMC over the space of all partitions of V into no more than k nonempty subsets. The resultant subsets are initialized as the child nodes (level 1) of the root. The process is repeated iteratively for each child node at level 1 and stops at level $maxL$. Fig.2 illustrates the idea with $k = 3$ for a graph of 13 nodes.

1.5 Conclusion

We give a big picture of the problem ϵ-DP community detection within the two categories : input and algorithm perturbations. We propose LouvainDP and ModDivisive as the representatives of input and algorithm perturbations respectively. By conducting a comprehensive evaluation, we reveal the advantages of our methods. ModDivisive steadily gives the best modularity and avg.F1Score on large graphs while LouvainDP outperforms the remaining input perturbation competitors in certain settings. The existing algorithm perturbation scheme gives low modularity clusterings and the input perturbation schemes like EdgeFlip and TmF hardly deliver any good node clustering except EdgeFlip on a medium-sized graph. For future work, we plan to extend our work for directed graphs and overlapping community detection under differential privacy.

2. REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[2] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In *ICDT*, pages 299–311. ACM, 2012.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *TCC*, pages 265–284, 2006.

[4] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3) :75–174, 2010.

[5] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE, 2007.

[6] F. D. McSherry. Privacy integrated queries : an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[7] Y. Mülle, C. Clifton, and K. Böhm. Privacy-integrated graph clustering through differential privacy. In *PAIS 2015*, 2015.

[8] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113, 2004.

[9] H. H. Nguyen, A. Imine, and M. Rusinowitch. Differentially private publication of social graphs at linear cost. In *ASONAM 2015*, 2015.

Conception d'une base de données capteurs à l'aide de contraintes spatio-temporelles

Manel Charfi
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
manel.charfi@insa-lyon.fr

Yann Gripay
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
yann.gripay@insa-lyon.fr

Jean-Marc Petit
Université de Lyon, CNRS
INSA-LYON, LIRIS, UMR5205
F-69621, Villeurbanne, France
jean-marc.petit@insa-lyon.fr

1. CONTEXTE & PROBLÉMATIQUE

De nos jours on a de plus en plus de capteurs qui ont tendance à apporter confort et facilité dans notre vie quotidienne. Ces capteurs sont faciles à déployer et à intégrer dans une variété d'applications (monitoring de bâtiments intelligents, aide à la personne,...). Ces milliers (voire millions) de capteurs sont de plus en plus envahissants et génèrent sans arrêt des masses énormes de données qu'on doit stocker et gérer pour le bon fonctionnement des applications qui en dépendent. A chaque fois qu'un capteur génère une donnée, deux dimensions sont d'un intérêt particulier : la dimension temporelle et la dimension spatiale. Ces deux dimensions permettent d'identifier l'instant de réception et la source émettrice de chaque donnée. Chaque dimension peut se voir associée à une hiérarchie de granularités qui peut varier selon le contexte d'application.

Dans notre travail on se focalise sur les capteurs dans le cadre des bâtiments intelligents et les applications qui requièrent le stockage à long terme des flux de données issues de ces capteurs. Notre objectif est de mettre en œuvre une approche qui vise à contrôler la stockage des données capteurs en ne gardant que les données jugées pertinentes selon la spécification des granularités spatio-temporelles représentatives des besoins applicatifs. Nous empruntons l'approche déclarative développée dans les années soixante-dix pour la conception de bases de données et on l'étend au flux de données des capteurs. Nous pensons que des contraintes temporelles augmentées avec la dimension spatiale sont nécessaires dans notre contexte pour saisir les contraintes sous-jacentes.

Ayant un ensemble de capteurs émettant des flux de données, notre problématique est :
Comment construire une base de données capteurs pour des applications qui requièrent le stockage à long terme ?

2. PROPOSITION

Nous nous basons sur l'intuition que les granularités spatio-temporelles combinées aux Dépendances Fonctionnelles (DFs)

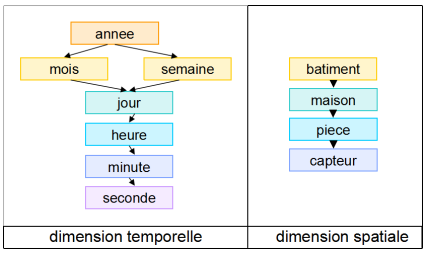


Figure 1: Dimensions spatiale et temporelle

[3] classiques peuvent jouer un rôle majeur pour exprimer quelles sont les données pertinentes à conserver pour une utilisation à long terme. De nombreuses extensions des DFs aux bases de données temporelles ont été proposées [4, 2, 5, 6]. Mais aucun d'entre elles n'étend complètement les DFs aux dimensions temporelles et spatiales. Dans [6], la notion de *Roll-Up Dependencies* (RUD) généralise les DFs temporelles à d'autres dimensions. Les aspects algorithmiques, comme la fermeture des attributs par rapport à un ensemble de contraintes, ne sont pas étudiés pour RUD, comme nous le faisons dans notre travail. A notre connaissance, ce problème n'a pas été étudié en profondeur dans la littérature.

On introduit alors les Dépendances Fonctionnelles Spatio-Temporelles (DFSTs) qui étendent les DFs aux dimensions spatiale et temporelle. Les DFSTs servent à exprimer les relations entre les données en prenant en compte des granularités spatio-temporelles. Les DFSTs sont inspirées des Dépendances Fonctionnelles Temporelles (DFT) [1].

Par exemple, considérons les dimensions spatio-temporelles présentées dans la figure 1 et le schéma de flux de capteurs : $capteursTemperature(temperature, localisation, temps)$.
On peut exprimer le fait qu'un utilisateur considère que « la température de chaque pièce reste la même sur chaque heure » en utilisant la DFST :
$localisation^{piece}, temps^{heure} \rightarrow temperature$.

En fait, comme les résultats requis concernent les données sur une longue période de temps (par exemple quelques mois, plusieurs années), les flux de capteurs peuvent être approximés grâce aux DFSTs pour réduire l'espace de stockage et augmenter l'efficacité des requêtes applicatives.

Nous proposons une technique de normalisation basée sur deux étapes principales : d'abord, le calcul d'une couverture minimale de DFSTs, ensuite, la production de schémas spatio-temporels pour la base de données capteurs. À l'issue

de la phase de normalisation, on obtient un premier schéma de base de données qu'on appellera dans la suite *schéma abstrait*. A ce stade, la question qui se pose est : quelle valeur choisir (quelle minute de l'heure et quel capteur parmi ceux de la pièce) ?
Afin de choisir la valeur représentative de chaque couple de granules spatio-temporelle, on a définit la notion d'Affectation Sémantique de Valeur (ASV), inspirée des hypothèses sémantiques [1] dans les bases de données temporelles. En effet, une ASV est une sorte d'annotation de schéma qui permet de préciser avec une fonction d'agrégation pour chaque attribut la valeur à garder. Par exemple, on peut préciser que pour l'attribut *temperature* si on doit garder une seule valeur par pièce par heure, il faut choisir la première (ou la moyenne, le maximum ...). Cela veut dire que pour une pièce i à l'heure j on gardera la première valeur de température reçue à une minute appartenant à l'heure j par un capteur qui appartient à la pièce i. Les ASVs dépendent du contexte d'application. Elles peuvent présenter des fonctions d'agrégation plus complexe (par exemple la moyenne des 3 premières valeurs appartenant à un domaine de validité). Ces fonctions peuvent être définies afin d'éviter les problèmes de données incomplètes et bruitées. L'ajout des ASVs au schéma abstrait mène à l'obtention du *schéma concret* qui peut être intégré dans des SGBDR classiques.

Dans notre travail, nous nous intéressons aussi au chargement des données à la volée à partir des flux de capteurs. Pour ce faire, on réutilise l'information sémantique relative au choix de la donnée dans chaque intervalle spatio-temporel contenue dans les ASVs. Notre système crée alors pour chaque ASV un *data wrapper*. Chaque *data wrapper* observe le flux de données du capteur concernant son attribut. Ensuite, il identifie les valeurs pertinentes en fonction de ses granularités et de sa fonction d'agrégation.

3. PROTOTYPE & EXPÉRIMENTATIONS

Nous avons implémenté un prototype qui gère les deux niveaux de prise en compte des flux de données des capteurs : la conception du schéma de la base de données et le chargement à la volée des données à partir des flux des capteurs. Ce qui nous a permis de mener des expériences avec des flux de données, synthétiques et réelles, provenant de bâtiments intelligents. Nous avons comparé notre solution avec la solution de référence et nous avons obtenu des résultats prometteurs en termes de performance de requêtes et d'utilisation de mémoire.

Figure 2 contient une capture d'écran de la phase de cration d'une DFST par l'utilisateur. Comme nous pouvons le voir, les DFSTs peuvent être naturelles et simples à définir. En effet, l'utilisateur n'a qu'à cocher les attributs concernés et sélectionner les granularités sans se soucier de la syntaxe.

Parmi les expérimentations que nous avons déroulées afin de vérifier notre approche, nous avons évalué le taux d'erreur et vérifié si l'approximation dégrade les résultats de l'évaluation des moyennes de température. Dans cette expérimentation nous avons utilisé des données réelles issues de capteurs installés dans notre campus. Nous nous sommes intéressés à la différence entre les résultats sur les données brutes (base de données contenant toutes les valeurs reçues par le flux de capteurs) et une base de données capteurs (base de données créée et maintenue en suivant notre approche). Les résultats de la comparaison des moyennes obtenues à partir d'une requête Q_1 sur les données brutes et des moyennes obtenues

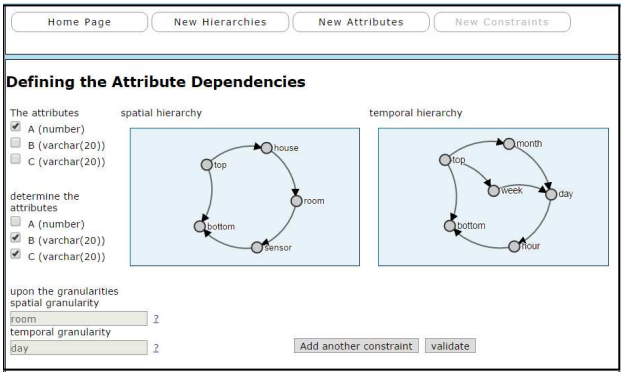


Figure 2: Création d'une DFST

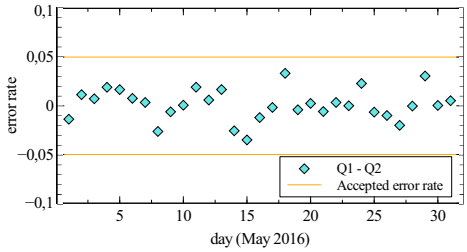


Figure 3: Différence entre les résultats de Q_1 et Q_2

à partir d'une requête Q_2 sur les données pertinentes sont donnés dans Figure 3. On peut voir que la différence est maintenue sous un taux d'erreur de 5%, ce qui peut être considéré comme acceptable dans un contexte réel.

4. REMERCIEMENTS

Ce travail a été financé par une allocation doctorale de recherche de la Région Rhône-Alpes.

5. REFERENCES

[1] C. Bettini, S. Jajodia, and S. Wang. *Time granularities in databases, data mining, and temporal reasoning*. Springer, 2000.

[2] C. S. Jensen, R. T. Snodgrass, and M. D. Soo. Extending existing dependency theory to temporal databases. *Knowledge and Data Engineering, IEEE Transactions on*, 8(4) :563–582, 1996.

[3] M. Levene and G. Loizou. *A guided tour of relational databases and beyond*. Springer Science & Business Media, 2012.

[4] V. Vianu. Dynamic functional dependencies and database aging. *Journal of the ACM (JACM)*, 34(1) :28–59, 1987.

[5] X. S. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems (TODS)*, 22(2) :115–170, 1997.

[6] J. Wijsen. Temporal fds on complex objects. *ACM Transactions on Database Systems (TODS)*, 24(1) :127–176, 1999.

Scientific Workflow Execution with Multiple Objectives in Multisite Clouds

Ji Liu
MSR-INRIA Joint Centre
Inria and LIRMM
University of Montpellier
Ji.Liu@inria.fr

Esther Pacitti
University of Montpellier
Inria and LIRMM
Esther.Pacitti@lirmm.fr

Patrick Valduriez
MSR-INRIA Joint Centre
Inria and LIRMM
University of Montpellier
Patrick.Valduriez@inria.fr

Daniel de Oliveira
Institute of Computing
Fluminense Federal University
danielcmo@ic.uff.br

Marta Mattoso
COPPE
Federal University of Rio de Janeiro
marta@cos.ufrj.br

1. INTRODUCTION

Large-scale *in silico* scientific experiments typically take advantage of Scientific Workflows (SWfs) to model data operations. A SWf is the assembly of scientific data processing activities with data dependencies among them. An activity is the description of a piece of work that forms a logical step within a SWf representation. A SWf Management System (SWfMS) is the tool to manage SWfs [3]. In order to execute SWfs efficiently, SWfMSs typically exploit High Performance Computing (HPC) resources in a cluster, grid or cloud environment. Clouds have become an interesting solution for SWf execution because of various advantages. A cloud is typically made of several sites (or data centers), each with its own resources and data. Thus, in order to use more resources at different sites, SWfs could also be executed in a distributed manner at different sites. Nowadays, the computing resources or data of a cloud provider such as Amazon or Microsoft are distributed at different sites and should be used during the execution of SWfs. As a result, a multisite cloud is an appealing solution for large scale SWf execution. As defined in [2], a multisite cloud is a cloud with multiple sites, each at a different location (possibly in a different region) and being explicitly accessible to cloud users, typically in the data center close to them for performance reasons. In addition, there is a stored data constraint, *i.e.* some data cannot be moved to other sites because of big amounts or proprietary issues, in a multisite cloud.

To enable SWf execution in a multisite cloud, the execution of each activity should be scheduled to a corresponding site. Then, the scheduling problem is to decide where to execute the activities. In general, to map the execution of activities to distributed computing resources is an NP-hard problem. The scheduling problem may have multiple objectives, *i.e.* multi-objective, *e.g.* reducing execution time or monetary cost *etc.* Thus, the multisite scheduling problem must take into account the impact of resources distributed at different sites, *e.g.* different bandwidths, data distribution and costs to use Virtual Machines (VMs) at different sites. To the best of authors' knowledge, there is no solution to execute SWfs in a multisite cloud while considering both multiple objectives and dynamic VM provisioning. The related work either focuses on static VM provisioning, single objective or single site execution. Static VM provisioning refers to the use of the existing VMs (before execution) for SWf execution without modifying VMs during execution. In addition, existing cost models are not suitable for the SWfs that have a big part of sequential workload.

In this short paper (see [4] for the extended version), we propose a general solution based on multi-objective scheduling to execute SWfs in a multisite cloud with the following main contributions: the design of a multi-objective cost model, SSVP VM provisioning approach, ActGreedy scheduling algorithm and an extensive experimental evaluation.

2. FRAGMENT SCHEDULING

In this section, we present the multisite SWfMS architecture, cost model, SSVP and ActGreedy.

The architecture of a multisite SWfMS is composed of four modules: workflow partitioner, multisite scheduler, single site initialization, and single site execution. The workflow partitioner partitions a SWf into fragments. After SWf partitioning, the fragments are scheduled to sites by the multisite scheduler. After scheduling, in order to avoid restarting VMs for the execution of continuous activities, all the activities scheduled at the same site are grouped as a fragment to be executed. Then, the single site initialization module prepares the execution environment for the fragment, using two components, *i.e.* VM provisioning and multisite data transfer. At each site, the VM provisioning component deploys and initializes VMs for the execution of SWfs. Finally, the single site execution module starts the execution of the fragments at each site. This can be realized by an existing single site SWfMS, *e.g.* Chiron [5].

BDA 2016, 15 au 18 Novembre 2016, Poitiers, France.

It is difficult to estimate the execution time and monetary costs for the whole SWf even with a scheduling plan according to existing cost models [1] since it is hard to generate a VM provisioning plan for each site with global desired execution time and monetary costs. As shown in Formula 2.0.1, we decompose the cost model as the sum of the cost of executing each fragment.

$$Cost(Sch(SWf, S)) = \sum_{wf_i \in SWf}^{Schedule(wf_i, s_j)=1} Cost(wf_i, s_j) \quad (2.0.1)$$

In the cost model, we also take the cost to provision VMs and the sequential workload into consideration in order to estimate the cost more precisely. In addition, we take advantage of Amdahl's law to estimate the execution time.

We propose a single site VM provisioning algorithm, called SSVP, to generate VM provisioning plans, which minimizes the cost to execute a fragment at Site s. First, SSVP calculates a near-optimal number of virtual CPUs to instantiate based on the cost model. Then, it optimizes the provisioning plan to reduce the cost to execute a fragment at Site s. Afterward, SSVP calculates the cost to execute the fragment at the site based on the cost model and improves the provisioning plan by inserting a new VM, modifying an existing VM or removing an existing VM. If the cost to execute the fragment at Site s can be reduced by improving the provisioning plan, the provisioning will be updated, and the improvement of provisioning plan continues.

Finally, we propose our multisite scheduling algorithms, *i.e. ActGreedy*, which schedules the most suitable site to each fragment. In ActGreedy, all the fragments of a SWf are not scheduled at beginning. During the scheduling process, all the fragments are scheduled at a corresponding site, which takes the minimum cost with the consideration of the stored data constraint while the cost is the minimum compared to other sites. As a result, the cost of executing a SWf in a multisite cloud is minimized. ActGreedy schedules fragments of multiple activities. ActGreedy can schedule a pipeline of activities to reduce data transfer between different fragments, *i.e.* the possible data transfer between different sites. A pipeline is a group of activities with a one-to-one, sequential relationship between them. In addition, ActGreedy makes a trade-off between time and monetary costs by using SSVP. ActGreedy schedules the available fragments, while choosing the best site for an available fragment rather than choosing the best fragment for an available site.

3. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of ActGreedy algorithm. All experiments are based on the execution of a SciEvol SWf in Microsoft Azure multisite cloud with three sites. We compare ActGreedy with LocBased [2] and SGreedy [1], as well as with two general algorithms, *i.e.* Genetic and Brute-force. In the experiments, workflow partitioner, multisite scheduler and single site initialization are simulated, but the execution of fragments is performed in a real environment by Chiron.

The results show that LocBased corresponds to up to 21.75% higher cost than ActGreedy and SGreedy takes up to 74.51% higher cost than ActGreedy based on our proposed cost model. Based on an existing cost model, the advantage of ActGreedy can reduce the total cost up to 10.7% compared with LocBased and 17.2% compared with SGreedy. In addition, when the weight of reducing execution is low, ActGreedy may correspond to bigger execution time but when the weight is high, ActGreedy corresponds to less execution time. The execution with ActGreedy always takes less monetary cost compared with LocBased (up to 14.12%) and SGreedy (up to 17.28%). In addition, the experiments also show that the scheduling time of Genetic and Brute-force is much longer than ActGreedy (up to 577 times and 128 times). For instance, with more than 22 activities or 10 sites, the scheduling time of both Genetic and Brute-force exceeds the execution while the scheduling time of ActGreedy remains small.

4. CONCLUSION

In this paper, we proposed a general solution based on multi-objective scheduling to execute SWfs in a multisite cloud (from the same provider). We first proposed a novel multi-objective cost model, based on which, we proposed a dynamic VM provisioning approach, namely SSVP, to generate VM provisioning plans for fragment execution. The cost model aims at minimizing two costs: execution time and monetary costs. Our proposed fragment scheduling approach that is ActGreedy, allows for considering stored data constraints while reducing the cost based on our multi-objective cost model to execute a SWf in a multisite cloud. We evaluated our approaches by executing SciEvol in Microsoft Azure cloud. The results show that since it makes a good trade-off between execution time and monetary costs based on SSVP, ActGreedy leads to the least total normalized cost, which is calculated based on our multi-objective cost model, than LocBased (up to 21.75%) and SGreedy (up to 74.51%) approaches. Additionally, ActGreedy scales very well, *i.e.* it takes a very small time to generate the optimal or near optimal scheduling plans when the number of activities or sites increases, compared with general approaches, *e.g.* Genetic and Brute-force.

References

[1] D. de Oliveira, K. A. C. S. Ocaña, F. Baião, and M. Mattoso. A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. *Journal of Grid Computing*, 10(3):521–552, 2012.

[2] J. Liu, V. Silva, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow partitioning in multi-site clouds. In *BigDataCloud'2014: 3rd Workshop on Big Data Management in Clouds in conjunction with Euro-Par 2014*, page 12, 2014.

[3] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, pages 1–37, 2015.

[4] J. Liu, E. Pacitti, P. Valduriez, D. de Oliveira, and M. Mattoso. Multi-objective scheduling of scientific workflows in multisite clouds. *Future Generation Computer Systems*, 63:76–95, 2016.

[5] E. S. Ogasawara, J. Dias, V. Silva, F. S. Chirigati, D. de Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience*, 25(16):2327–2341, 2013.

Lignages efficaces sur les instances quasi-arborescentes: Limites et extensions (Résumé)

Antoine Amarilli
Télécom ParisTech,
Université Paris-Saclay, France
antoine.amarilli@telecom-paristech.fr

Pierre Bourhis
CRIStAL, UMR 9189, CNRS,
Université Lille 1, France
pierre.bourhis@univ-lille1.fr

Pierre Senellart
DI, École normale supérieure,
PSL Research University, France
pierre.senellart@ens.fr

Il est généralement infaisable (#P-difficile) d'évaluer des requêtes sur les bases de données probabilistes. Des résultats de dichotomie ont permis d'identifier [5–7] quelles requêtes (dites *safe*) peuvent être évaluées efficacement, en rattachant cela à des représentations du lignage [8]. Nous avons précédemment montré [2], à l'aide de techniques différentes, que l'évaluation de requêtes arbitraires en logique monadique du second ordre est faisable en temps linéaire sur les bases de données probabilistes, à condition de borner la *largeur d'arbre* des instances.

Dans ce travail, nous étudions les limites et les extensions possibles de ce résultat. Nous montrons d'abord, pour l'évaluation probabiliste de requêtes, qu'il est *nécessaire* de borner la largeur d'arbre pour assurer la faisabilité de MSO : en effet, il y a même des requêtes FO dont l'évaluation probabiliste est infaisable sur n'importe quelle classe de graphes de largeur d'arbre non bornée qui soit efficacement constructible. Cette dichotomie s'appuie sur des bornes polynomiales récentes pour l'extraction de graphes planaires comme mineurs [4] ; elle implique des bornes inférieures pour des problèmes non-probabilistes analogues comme l'évaluation de requêtes et de comptage d'assignements sur des familles closes par sous-instances. Nous montrons ensuite comment notre résultat de faisabilité peut s'expliquer en termes de lignage : on peut représenter le lignage d'une requête MSO sur une instance quasi-arborescente comme un circuit quasi-arborescent, un OBDD de taille polynomiale, ou une d-DNNF de taille linéaire. En revanche, nous pouvons étendre notre premier résultat de nécessité aux représentations de lignage, et exhiber une UCQ avec inégalités telles que, pour n'importe quelle famille de graphes de largeur d'arbre non bornée, le lignage ne peut pas être représenté par un OBDD de taille polynomiale ; nous pouvons même caractériser les requêtes qui ont cette propriété. Nous montrons enfin comment notre approche sur les instances quasi-arborescentes permet d'expliquer la faisabilité de l'évaluation pour les requêtes safe *sans inversion* : leurs instances d'entrée peuvent être récrites pour borner leur largeur d'arbre.

L'article complet (en anglais) a été présenté à PODS'16 et est disponible en ligne [3]. Des compléments sont disponibles dans la thèse de doctorat du premier auteur [1].

1. BIBLIOGRAPHIE

[1] A. Amarilli. *Leveraging the Structure of Uncertain Data*. PhD thesis, Télécom ParisTech, 2016. 2016-ENST-0021. https://tel.archives-ouvertes.fr/tel-01345836.

[2] A. Amarilli, P. Bourhis, and P. Senellart. Provenance circuits for trees and treelike instances. In *ICALP*, 2015. https://arxiv.org/abs/1511.08723.

[3] A. Amarilli, P. Bourhis, and P. Senellart. Tractable lineages on treelike instances : Limits and extensions. In *PODS*, 2016. https://arxiv.org/abs/1604.02761.

[4] C. Chekuri and J. Chuzhoy. Polynomial bounds for the grid-minor theorem. In *STOC*, 2014. https://arxiv.org/abs/1305.6577.

[5] N. Dalvi and D. Suciu. The dichotomy of conjunctive queries on probabilistic structures. In *PODS*, 2007. https://arxiv.org/abs/cs/0612102.

[6] N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *JACM*, 59(6), 2012. https://homes.cs.washington.edu/~suciu/jacm-dichotomy.pdf.

[7] R. Fink and D. Olteanu. A dichotomy for non-repeating queries with negation in probabilistic databases. In *PODS*, 2014. https://www.cs.ox.ac.uk/dan.olteanu/papers/fo-pods14.pdf.

[8] A. K. Jha and D. Suciu. Knowledge compilation meets database theory : Compiling queries to decision diagrams. *Theory Comput. Syst.*, 52(3), 2013. https://homes.cs.washington.edu/~suciu/camera_ready.pdf.

Calculer et Compresser le Skycube Négatif

Nicolas Hanusse
LaBRI. Université de Bordeaux. CNRS
France
hanusse@labri.fr

Patrick Kamnang Wanko
LaBRI. Université de Bordeaux. CNRS
France
pkamnang@labri.fr

Sofian Maabout
LaBRI. Université de Bordeaux. CNRS
France
maabout@labri.fr

ABSTRACT

Given a table T with D dimensions, the skycube of T is the union of all skylines obtained by considering each of the subsets of D (subspaces). The number of these skylines is exponential w.r.t D. To make the skycube practically useful, two lines of research have been pursued so far : the first one aims to propose efficient algorithms for computing it and the second one considers either that the skycube is too large to be computed in a reasonable time or it requires too much memory space to be stored. They therefore propose skycube summarization techniques to reduce time and space consumption. Intuitively, previous efforts have been devoted to compute or summarize the following information : "for every tuple t, list the skylines where t belongs to". In this paper, we consider the complementary statement, i.e., "for every tuple t, list the skylines where t *does not belong to*". This is what we call the *negative skycube*. Despite the apparent equivalence between these two statements, our analysis and extensive experiments show that these two points of views do not lead to the same behavior of the related algorithms. More specifically, our proposal shows that (i) the negative summary can be obtained much faster than state of the art techniques for positive summaries, (ii) in general, it consumes less space, (iii) skyline queries evaluation using this summary are much faster, (iv) the positive skycube can be obtained much more rapidly than state of the art algorithms, and (v) it can be used for a larger class of queries, namely k-domination skylines.

The Negative Skycube

Skyline queries are a special case of preference queries whose aim is to retrieve a subset of records that are not worse than any other record. These are also called *Pareto optimum* tuples. In a multidimensional setting, users are allowed to chose a subset of attributes along which the tuples are compared. To optimize these queries, the skycube structure has been proposed. Because of the exponential number of queries, the skycube materialization is time and space consuming. Therefore, two research directions have been pursued so far : sophisticated algorithms to speed up the skycube computation have been proposed and on another hand, summarization techniques have been defined to reduce the skycube space consumption while keeping query execution time acceptable. Regarding summarization, some techniques assume the skycube as part of the input while others do provide a summary procedure directly from ground data.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

Id	(P)rice	(D)istance	(A)rea	(W)ifi
h_1	100	10	20	No
h_2	10	100	20	Yes
h_3	100	10	25	Yes

Figure 1: Hotels

The present work is based on the following observation : to state that a point p is part of the skyline, it should be *compared to all* other points. However, by comparing p to just a single other point q, we may derive a set of skylines, each of which is related to a dimensions subset, where p does not belong to. Simply said, our departure point is that finding the subsets of attributes (subspaces) w.r.t. which p is dominated should be faster than the converse. Once this set of subspaces is computed, obtaining its complementary is straightforward.

Storing for every tuple p, the whole set of subspaces where p is not in the skyline can be unfeasible in practice due to the large amount of memory space it requires. Therefore, we propose techniques to summarize this information. Let us consider the following example to illustrate our approach.

EXAMPLE 1. *Consider the* Hotels *table depicted in Figure 1. Cheaper, closer, wider and with Wifi hotels are preferred. Users are allowed to ask for skylines w.r.t any of the* 2^4-1 *non-empty subsets of* {**P**, **D**, **A**, **W**}. *By comparing* h_2 *to* h_1, *we find that* h_2 *dominates* h_1 *w.r.t* $subspaces_2 = \{$**P**, **W**, **PW**, **AP**, **AW**, **APW**$\}$ [1]. *By contrast, and after this comparison, we have no information about the subspaces where* h_1 *belong to their skyline. We need to compare* h_1 *to* h_3 *too. Once* h_1 *is compared to* h_3, *we derive a new set of subspaces where* h_1 *is dominated, i.e.,* $subspaces_3 = \{$**A**, **W**, **AW**, **AD**, **DW**, **AP**, **ADP**, **PW**, **ADW**, **APW**, **DPW**, **ADPW**$\}$. $subspaces_2 \cup subspaces_3$ *is the set of subspaces where* h_1 *is dominated. Its complement is the set of subspaces where* h_1 *is a skyline point, i.e.,* {**D**, **DP**}. *In practice, the number of subspaces associated to a tuple can be large. We*

1. We will see later that it is easy to derive this information without comparing h_1 to h_2 w.r.t. every subspace.

therefore propose techniques to summarize this set while keeping the efficacy of queries evaluation.

EXAMPLE 2. *Table 1 serves as example throughout the paper. Using this data set, users may ask for the* best *tuples (Skyline tuples) regarding every combination of the dimensions* $\{A, B, C, D\}$. *For instance,* $Sky(AB) = \{t_1, t_2\}$ *and* $Sky(ABCD) = \{t_2, t_3, t_4\}$.

Table 1: Dataset T

Id	A	B	C	D
t_1	1	1	3	3
t_2	1	1	2	3
t_3	2	2	2	2
t_4	4	2	1	1
t_5	3	4	5	2
t_6	5	3	4	2

Let us first give some preliminary definition.

DEFINITION 1 (DOMINANCE SUBSPACES). *Let* $t \in T$ *and* $X \subseteq \mathcal{D}$. X *is a* dominance subspace *for* t *iff* $t \notin Sky(X)$.

DEFINITION 2 (NEGATIVE SKYCUBE). *Let* $t \in T$ *and let* $DomSubspaces(t)$ *denote the dominant subspaces for* t. *The* negative skycube of T *is the set* $\{DomSubspaces(t) | t \in T\}$.

In other words, the negative skycube stores for every tuple t, the subspaces where it does not belong to their respective skyline.

EXAMPLE 3. *From the running example in Table 1, it is easy to check that* $DomSubspaces(t_1) = \{ABCD, ABC, ACD, BCD, AC, BC, CD, C, D\}$.

Clearly, if the negative skycube of T is available, then the computation of every skyline $Sky(X)$ is straightforward : for every tuple t, $t \in Sky(X)$ iff $X \notin DomSubspaces(t)$. Therefore, the negative skycube of T can be seen as the complement of its skycube.

Now we show how the negative skycube can be computed without using the naive solution which consists in first computing the *positive* skycube then derive from it the negative one. We first show how by comparing t to t' we obtain a set of subspaces belonging to $DomSubspaces(t)$.

DEFINITION 3. *Let* $t, t' \in T$. *We define a comparison function* $\mathcal{COMPARE}$ *as follows :* $\mathcal{COMPARE}(t, t') = \langle X|Y\rangle$ *such that* X *is the set of dimensions* D_j *such that* $t'[D_j] < t[D_j]$ *and* Y *is the set of dimensions* D_k *for which* t *and* t' *are equal. Then,* $t[D_\ell] < t'[D_\ell]$ *for every* $D_\ell \in D \setminus X \cup Y$.

To simplify the notation, we use $\mathcal{COMPARE}(t)$ to denote the set $\cup_{t' \in T}\{\mathcal{COMPARE}(t, t')\}$.

EXAMPLE 4. *From Table 1, we have* $\mathcal{COMPARE}(t_5, t_6) = \langle BC|D\rangle$ *because* $t_6[B] < t_5[B]$ *(3 vs. 4),* $t_6[C] < t_5[C]$ *(4 vs. 5) and both tuples are equal on dimension D (2). Note that* $\mathcal{COMPARE}(t_6, t_5) = \langle A|D\rangle$.

Obviously, if $\mathcal{COMPARE}(t, t') = \langle X|Y\rangle$ then $X \cap Y = \emptyset$.

DEFINITION 4 (COVERAGE). *Let* $\langle X|Y\rangle$ *be a pair of disjoint subspaces and let* Z *be a subspace. We say that* $\langle X|Y\rangle$ covers Z *iff* $Z \subseteq XY$ *and* $Z \cap X \neq \emptyset$.

EXAMPLE 5. $p = \langle AC|B\rangle$ *covers subspaces* A, AB, C, AC, BC *and* ABC. *Note that* B *is not covered by* p *because even if* $B \subseteq ACB$, $B \cap AC = \emptyset$.

The following proposition shows that the covered subspaces by the pair we obtain when t is compared to t' are precisely the subspaces where t' dominates t.

PROPOSITION 1. *Let* $t, t' \in T$ *and let* $\mathcal{COMPARE}(t, t') = \langle X|Y\rangle$. *Then* t' *dominates* t *in every and only subspaces* Z *covered by* $\langle X|Y\rangle$, *i.e.,* $t \notin Sky(Z)$.

In fact, $\mathcal{COMPARE}(t, t')$ is a concise summary of the set of subspaces for which t is dominated by t'. Hence, it is a summary of a subset of the subspaces where t does not belong to their respective skylines.

EXAMPLE 6. *From Table 1, the data structure returned by our algorithm is depicted in Table 2. Note that pairs* $\langle X|Y\rangle$ *where* $X = \emptyset$ *are not considered because they do not cover any subspace (see Proposition 1).*

Table 2: NSC of T

$Tuples$	$Pairs$
t_1	$\langle C\|ABD\rangle, \langle CD\|\emptyset\rangle, \langle D\|\emptyset\rangle$
t_2	$\langle D\|C\rangle, \langle CD\|\emptyset\rangle, \langle D\|\emptyset\rangle$
t_3	$\langle AB\|\emptyset\rangle, \langle AB\|C\rangle, \langle CD\|B\rangle$
t_4	$\langle AB\|\emptyset\rangle, \langle A\|B\rangle, \langle A\|\emptyset\rangle$
t_5	$\langle ABC\|\emptyset\rangle, \langle ABC\|D\rangle, \langle BCD\|\emptyset\rangle, \langle BC\|D\rangle$
t_6	$\langle ABC\|\emptyset\rangle, \langle ABC\|D\rangle, \langle ABCD\|\emptyset\rangle, \langle A\|D\rangle$

Reducing the size of NSC not only reduces memory consumption but also optimizes skyline queries evaluation. So the next problem we solve consists in finding a *minimal set* of pairs which covers exactly the subspaces covered by $\mathcal{COMPARE}(t)$.

EXAMPLE 7. *The minimization of Table 2 using our algorithm is depicted in Table 3. Note that the number of pairs decreases from 20 to 8.*

Table 3: List of pairs synthetizing dominance subspaces sets

Tuple	Associated list of pairs
t_1	$\langle C\|ABD\rangle, \langle CD\|\emptyset\rangle$
t_2	$\langle CD\|\emptyset\rangle$
t_3	$\langle AB\|C\rangle, \langle CD\|B\rangle$
t_4	$\langle AB\|\emptyset\rangle$
t_5	$\langle ABCD\|\emptyset\rangle$
t_6	$\langle ABCD\|\emptyset\rangle$

Acknowledgments

This work has been partially carried out as part of SPEED DATA and Programme IdEx Bordeaux - CPU (ANR-10-IDEX-03-02) from PIA (Projets d'Investissement d'Avenir) and PETASKY project from CNRS program MASTODONS.

Extending CloudMdsQL with MFR for Big Data Integration *

Carlyna Bondiombouy, Boyan Kolev, Patrick Valduriez, Oleksandra Levchenko
Inria and LIRMM
University of Montpellier, France
firstname.lastname@inria.fr

1. INTRODUCTION

Multistore systems [3] have been recently proposed to provide integrated access to multiple, heterogeneous data stores through a single query engine. Compared to multidatabase systems [6], multistore systems typically trade source autonomy for efficiency, using a tightly-coupled approach. In particular, much attention is being paid on the integration of unstructured big data (e.g. produced by web applications) typically stored in HDFS with relational data, e.g. in a data warehouse.

Existing solutions to integrate such unstructured and structured data do not directly apply to solve our problem, as they rely on having a relational view of the unstructured data, and hence require complex transformations. SQL engines, such as Hive, on top of distributed data processing frameworks are not always capable of querying unstructured HDFS data, thereby forcing the user to query the data by defining map/reduce functions.

Our approach is different as we propose a query language that can directly express subqueries that can take full advantage of the functionality of the underlying data processing frameworks. Furthermore, the language should allow for query optimization, so that the query operator execution sequence specified by the user may be reordered by taking into account the properties of map/filter/reduce operators together with the properties of relational operators. We respect the autonomy of the data stores and pay special attention to this throughout our experimental evaluation.

In this short paper (see [2] for the long version), we propose a functional SQL-like query language (based on CloudMdsQL) and query engine to retrieve data from two different kinds of data stores - an RDBMS and a distributed data processing framework such as Apache Spark or Hadoop MapReduce on top of HDFS - and combine them by applying data integration operators (mostly joins). However, users need to be aware of how data are organized across the data stores, so that they write valid queries. The query therefore contains embedded invocations to the underlying data stores, expressed as subqueries. As our query language is functional, it introduces a tight coupling between data and functions.

*Work partially funded by the European Commission under the Integrated Project CoherentPaaS [1].

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. QUERY LANGUAGE

The query language is based on a more general common query language, called CloudMdsQL [4], designed in the context of the CoherentPaaS project [1] to solve the problem of querying multiple heterogeneous databases (e.g. relational and NoSQL) within a single query while preserving the expressivity of their local query mechanisms. The common language itself is SQL-based with the extended capabilities for embedding subqueries expressed in terms of each data store's native query interface. The common data model respectively is table-based, with support of rich datatypes that can capture a wide range of the underlying data stores' datatypes, such as MongoDB arrays and JSON objects, in order to handle non-flat and nested data, with basic operators over such composite datatypes.

In this section, we introduce a formal notation to define Map/Filter/Reduce (MFR) subqueries in CloudMdsQL that request data processing in an underlying big data processing framework (DPF). Then we give an overview of how MFR statements are combined with SQL statements to express integration queries against a relational database and a DPF.

2.1 MFR Notation

An MFR statement represents a sequence of MFR operations on datasets. A dataset is considered simply as an abstraction for a set of tuples, where a tuple is a list of values, each of which can be a scalar value or another tuple. Although tuples can generally have any number of elements, mostly datasets that consist of key-value tuples are being processed by MFR operations. In terms of Apache Spark, a dataset corresponds to an RDD (Resilient Distributed Dataset - the basic programming unit of Spark). Each of the three major MFR operations (MAP, FILTER and REDUCE) takes as input a dataset and produces another dataset by performing the corresponding transformation. Therefore, for each operation there should be specified the transformation that needs to be applied on tuples from the input dataset to produce the output tuples. Normally, a transformation is expressed with an SQL-like expression that involves special variables; however, more specific transformations may be defined through the use of lambda functions.

2.2 Combining SQL and MFR

Queries that integrate data from both a relational data store and a DPF usually consist of two subqueries (one expressed in SQL that addresses the relational database and another expressed in MFR that addresses the DPF) and an integration SELECT statement. The syntax follows the CloudMdsQL grammar introduced in [5]. A subquery is defined as a named table expression, i.e. an expression that returns a table and has a name and signature. The signature defines the names and types of the columns of the returned relation. Thus, each query, although agnostic to the underlying data store' schemas, is executed in the context of an ad-hoc schema, formed by all named table expressions within the query. A named table expression can be defined by means of either an SQL SELECT statement (that the query compiler is able to analyze and possibly rewrite) or a native expression (that the query compiler considers as a black box and passes to the wrapper as is, thus delegating it the processing of the subquery).

3. EXPERIMENTAL VALIDATION

The goal of our experimental validation is to evaluate the impact of query rewriting and optimization on execution time. More specifically, we explore the performance benefit of using bind join under different conditions.

3.1 Datasets

We generated data to populate the PostgreSQL table `scientists`, the MongoDB document collection `publications`, and text files with unstructured log data stored in HDFS.

3.2 Experimental Results

To evaluate the impact of optimization on query execution, we use a cluster of the GRID5000 platform (www.grid5000.fr), with one node for PostgreSQL and MongoDB and 4 to 16 nodes for the HDFS cluster. The Spark cluster, used as both the DPF and the query processor, is collocated with the HDFS cluster. Each node in the cluster runs on 16 CPU cores at 2.4GHz, 64GB main memory, and the network bandwidth is 10Gbps. To demonstrate in detail the optimization techniques and their impact on the query execution, we prepared 1 query with different selectivity factors of the bind join condition. We execute the query in three different HDFS cluster setups - with 4, 8, and 16 nodes. Then we compare the execution times without and with bind join to the MFR subquery, which are illustrated in each query's corresponding graphical chart.

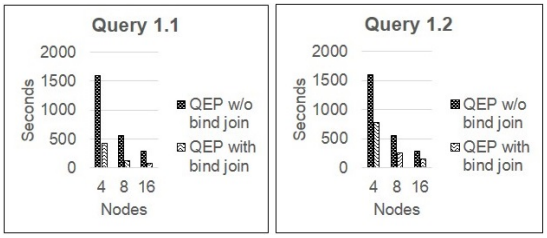
Query 1 involves all the data stores and aims at finding experts for publications of authors with a certain affiliation, it uses the MFR subquery `experts_alt` , which uses more sophisticated map functions, but makes only one shuffle, where the key is a keyword. For comparison, the MFR expression `experts` makes two shuffles, of which the first one uses a bigger key, composed of a keyword-author pair. Therefore, the corresponding Spark computation of Query 1 involves much smaller size of data to be shuffled, which explains its better overall efficiency and higher relative benefit of using bind join.

–Query 1.1: selectivity factor 10%

```
SELECT p.author, p.title, e.kw, e.expert
FROM scientists s, publications p, experts_alt e
WHERE s.affiliation = 'affiliation1'
AND p.author = s.name AND e.kw IN p.keywords
```

–Query 1.2: selectivity factor 30%

```
SELECT p.author, p.title, e.kw, e.expert
FROM scientists s, publications p, experts_alt e
WHERE s.affiliation IN ('affiliation1',
  'affiliation2', 'affiliation3')
AND p.author = s.name AND e.kw IN p.keywords
```



This experimental evaluation illustrates the query engine's ability to perform optimization and choose the most efficient execution plan. The results show the significant benefit of performing bind join in our experimental scenario, despite the overhead it produces (see [2]).

4. CONCLUSION

In this short paper, we proposed a functional SQL-like query language and query engine to integrate data from relational, NoSQL, and big data stores (such as HDFS). Our validation demonstrates that the proposed query language achieves the following requirements. First, it provides high expressivity by allowing the ad-hoc usage of specific map/filter/reduce operators through the MFR notation, as it was demonstrated with the `hdfs` subqueries. Second, it is optimizable as was demonstrated through performing bind join by rewriting the MFR subquery after retrieving the dataset from the MongoDB database. Our performance evaluation illustrates the query engine's ability to optimize a query and choose the most efficient execution strategy.

5. REFERENCES

[1] Coherentpaas project. `coherentpaas.eu`.

[2] C. Bondiombouy, B. Kolev, O. Levchenko, and P. Valduriez. Multistore big data integration with cloudmdsql. *TLDKS*, 28:48–74, 2016.

[3] C. Bondiombouy and P. Valduriez. Query processing in multistore systems: an overview. *International Journal of Cloud Computing (IJCC)*, 38, 2016.

[4] B. Kolev, C. Bondiombouy, P. Valduriez, R. Jiménez-Peris, R. Pau, and J. Pereira. The cloudmdsql multistore system. In *ACM SIGMOD Int. Conf. on Data Management*, pages 2113–2116, 2016.

[5] B. Kolev, P. Valduriez, C. Bondiombouy, R. Jiménez-Peris, R. Pau, and J. Pereira. Cloudmdsql: querying heterogeneous cloud data stores with a common language. *Distributed and Parallel Databases*, 34(4):463–503, 2016.

[6] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems, Third Edition*. Springer, 2011.

Access Control Enforcement for Selective Disclosure of Linked Data

Implantation d'Autorisations pour l'Exposition Sélective de Données Liées

Tarek Sayah
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205 F-69622, France
tarek.sayah@liris.cnrs.fr

Emmanuel Coquery
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205 F-69622, France
emmanuel.coquery@liris.cnrs.fr

Romuald Thion
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205 F-69622, France
romuald.thion@liris.cnrs.fr

Mohand-Saïd Hacid
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205 F-69622, France
mshacid@liris.cnrs.fr

Keywords

RDF, Autorisations, Données liées, Implantation

1. RÉSUMÉ

La problématique du contrôle des accès aux contenus RDF (*Resource Description Framework*) et de l'exposition sélective de l'information en fonction des privilèges des requérants devient de plus en plus importante comme l'illustre par exemple le nombre de mouvements *open data* des collectivités locales et gouvernements.

Notre principal objectif est d'encourager les entreprises et les organisations à publier leurs données RDF dans l'espace global des données liées. En effet, les données publiées peuvent être sensibles, et par conséquent, les fournisseurs de données peuvent être réticents à publier leurs informations, à moins qu'ils ne soient certains que les droits d'accès à leurs données par les différents profils d'utilisateurs sont appliqués correctement.

Dans cet article, nous nous intéressons à l'implantation efficace d'un contrôle d'accès pertinent pour les données RDF. Ce travail s'appuie sur une proposition de modèle de contrôle d'accès à grains fins pour les données RDF publié précédemment [1], où les autorisations sont exprimées sous la forme `GRANT`/`DENY` h `WHERE` b, où h est un triplet RDF avec variables sur lequel porte la décision d'accès (`GRANT`) ou de refus d'accès (`DENY`), et où b un ensemble de triplets avec variables, appelé *basic graph pattern* dans la littérature. À chaque profil utilisateur est associé un ensemble spécifique d'autorisations qui détermine le sous-ensemble des données auquel l'utilisateur a accès. Une fonction générique dite de *résolution des conflits* permet de traiter le cas où des décisions différentes sont applicables à un triplet concerné par plusieurs autorisations.

Nous proposons dans cet article une approche basée sur *l'annotation de données* pour *appliquer efficacement* les autorisations décrites. La preuve de concept que nous avons réalisée montre que notre implantation entraîne *un surcoût raisonnable durant l'exécution*. Notre approche repose sur un compromis entre une évaluation *statique*, calculée lors de la définition de la politique, et une évaluation *dynamique*, calculée lors de l'exécution d'une requête utilisateur.

L'évaluation statique consiste à calculer pour chaque triplet de la base l'ensemble des différentes autorisations qui lui sont applicables. En pratique, chaque autorisation est transformée en une requête `CONSTRUCT` SPARQL qui calcule l'ensemble de triplets sur lesquels chaque autorisation porte. Cette matrice entre autorisations et triplets est ensuite transposée puis matérialisée ligne par ligne dans le champ *Named Graph* de chaque triplet.

L'évaluation dynamique est implantée sous forme d'un déclencheur exécuté quand le moteur d'évaluation de requêtes SPARQL accède à la base de données. Lorsqu'un triplet est lu par le moteur, connaissant l'ensemble des autorisations qui le concerne, le déclencheur va calculer la décision finale `GRANT` ou `DENY` pour le sous-ensemble des autorisation de l'utilisateur qui pose la requête. Le triplet n'est effectivement transmis au moteur de requête que si cette décision est positive. Le moteur n'accède ainsi qu'à un sous ensemble des données propre à chaque profil utilisateur.

Sur la base de jeux de données synthétiques générés par le LUBM (*Lehigh University Benchmark*) allant jusqu'à 1,6 millions de triplets, nos expérimentations avec l'API Java Jena au dessus du *quadstore* Jena TDB, montrent que le surcoût dynamique est de l'ordre de 50% par rapport à une méthode *idéale* où le sous-graphe de chaque utilisateur serait matérialisé. Nous montrons également que le surcoût dynamique ne dépend pas du nombre d'autorisations. Enfin, les expérimentations confirment que pour neuf des requêtes proposées dans LUBM, le surcoût effectif de notre implantation ne varie qu'entre 5% et 40%.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

Notons que la méthode sus-citée n'est *idéale* que dans le sens où elle n'engendre pas de surcoût dynamique. En revanche, elle n'est pas intéressante en terme d'espace disque ni à l'évaluation statique, en effet, le sous-ensemble autorisé de chaque utilisateur doit être calculé puis matérialisé.

Le travaux futurs de cet article concernent la mise à jour incrémentale du graphe statique lorsque la base de données est modifiée, l'exploitation de l'ordre partiel des autorisations pour améliorer l'efficacité de l'évaluation statique et de nouvelles expérimentation.

Cet article a été publié dans les actes du 12th International Workshop on Security and Trust Management (STM) [2].

2. REFERENCES

[1] T. Sayah, E. Coquery, R. Thion, and M.-S. Hacid. Inference leakage detection for authorization policies over rdf data. In *29th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, pages 346–361, Fairfax, United States, July 2015. Acceptance rate of full papers 18/45.

[2] T. Sayah, E. Coquery, R. Thion, and M.-S. Hacid. Access control enforcement for selective disclosure of linked data. In *12th International Workshop on Security and Trust Management*, volume 9871 of *Security and Trust Management*, pages 1–17, Heraklion, Greece, Sept. 2016. Springer. Acceptance rate of full papers 13/37.

Federated SPARQL Query Processing with Replicated Fragments

Gabriela Montoya
LINA – Nantes University
Unit UMR6241 CNRS
Nantes, France
gabriela.montoya@univ-nantes.fr

Hala Skaf-Molli
LINA – Nantes University
Nantes, France
hala.skaf@univ-nantes.fr

Pascal Molli
LINA – Nantes University
Nantes, France
pascal.molli@univ-nantes.fr

Maria-Esther Vidal
Universidad Simón Bolívar
Caracas, Venezuela
mvidal@ldc.usb.ve

ABSTRACT

Federated query engines provide a unified query interface to federations of SPARQL endpoints. SPARQL endpoints that provide access to replicated data fragments, from different Linked Data sources, take advantage of customized data reorganization to offer new opportunities to optimize federated queries. However, existing federated query engines are not designed to support replication and replicated data can negatively impact their performance.

Although the problem of managing *overlapping* RDF data during federated query processing has been addressed at the triple pattern level, e.g., BBQ [3], DAW [6], the problem of managing *replication* in a federation of RDF datasets still remains open. Approaches, such as BBQ and DAW, rely on data summaries and a trade-off between summary size and accuracy. However, relying on data summaries has an intrinsic limitation, in order to make the approach feasible, the summary sizes are kept small and this naturally limits their accuracy. Therefore, these approaches may fail to correctly select non overlapping sources. Moreover, selecting sources at the triple pattern level, disregarding the structure of the query and how the triple patterns are connected through joins, limits the possible data transfer reduction. For example, two triple patterns with low selectivity may require to transfer a large amount of data if they are evaluated against two different endpoints, but only a little amount of data if evaluated in the same sub-query against the same endpoint.

In this paper [5], we formulate the source selection problem with fragment replication (SSP-FR). For a given set of endpoints with replicated fragments and a SPARQL query, the problem is to select the endpoints that minimize the number of tuples to be transferred. The FEDRA source selection algorithm solves SSP-FR. FEDRA steps are:

1. Selection of non redundant fragments.
2. Consolidation of relevant fragments.
3. Evaluation of basic graph patterns (BGPs) in as few endpoints as possible.

First, FEDRA relies on the fragment descriptions and query containment to identify relevant fragments for each query triple pattern and if the data of one relevant fragment is already provided by another, the fragment with the redundant data is pruned. Second, multiple relevant fragments for the same triple pattern are consolidated if they have been simultaneously replicated by at least one endpoint. This step facilitates the evaluation of multiple triple patterns in the same sub-query by existing federated query engines. Third, the source selection for the triples in the BGP that have only one relevant fragment is obtained using a reduction to a set covering problem instance. Given a set to cover and a set of collections with some elements from the set, a solution to the set covering problem finds the smaller group of collections that provides all the elements in the set. Each endpoint corresponds to a collection in the set covering problem, and each triple pattern corresponds to a set element to cover. The collection includes the element iff the endpoint has replicated the triple pattern relevant fragment. The use of an exact solution for the set covering problem makes our source selection approach NP-Hard. Therefore, we use an existing heuristic [4] to solve the set covering instance in order to evaluate the BGP in as few endpoints as possible. The use of this heuristic allows for obtaining a solution whose time is at worst linear on the number of query triple patterns and quadratic on the number of replicated fragments.

To illustrate, consider a federation of three endpoints that have replicated fragments from datasets DBpedia and LMDB (Table 1a), and a query Q composed of the four triple patterns in Table 2a. Endpoint descriptions and query containment are used to identify the relevant fragments (Table 2a). For example, the triple pattern $tp1$ has two relevant fragments: $f6$ and $f7$, and triple pattern $tp4$ has two relevant fragments: $f4$ and $f5$. Furthermore, fragment $f5$ is pruned because all the triples in $f5$ are also part of $f4$ and the com-

Table 1: Replicated fragments in a federation composed of endpoints $C1$, $C2$, and $C3$

(a) Replicated Fragments

F	Satisfied triple pattern	Dataset	Endpoints
f2	?film dbo:director ?director	DBpedia	C1,C2,C3
f3	?movie owl:sameAs ?film	LMDB	C2,C3
f4	?movie linkedmdb:genre ?genre	LMDB	C1,C3
f5	?movie linkedmdb:genre film_genre:14	LMDB	C2
f6	?director dbo:nationality dbr:France	DBpedia	C1
f7	?director dbo:nationality dbr:United_Kingdom	DBpedia	C2

Table 2: Relevant fragments and selected sources for Q. Fragments with redudant data are presented in gray

(a) Relevant Fragments to Q

Q triple pattern		RF	Endpoints
tp1	?director dbo:nationality ?nat	f6	C1
		f7	C2
tp2	?film dbo:director ?director	f2	C1,C2,C3
tp3	?movie owl:sameAs ?film	f3	C2,C3
tp4	?movie linkedmdb:genre ?genre	f4	C1,C3
		f5	C2

(b) Selected sources

TP	SS1	SS2	SS3
tp1	{C1,C2}	{C1,C2}	{C1,C2}
tp2	{C1,C2,C3}	{C1}	{C3}
tp3	{C2,C3}	{C2}	{C3}
tp4	{C1,C2,C3}	{C3}	{C3}
Triples to transfer	421,675	170,078	8,953

plete answer of $tp4$ can be obtained using $f4$. Table 2b shows three source selections, $SS1$-$SS3$, that can be used to evaluate Q and obtain its complete answer. Even if $SS1$ and $SS2$ minimize the number of selected endpoints per triple pattern, only $SS3$ minimizes the number of transferred tuples. Indeed, executing $tp1, tp2, tp3$ against replicated fragments that are located in the same data consumer endpoint greatly reduces the number of transferred tuples.

The approach proposed by Saleem et al. [6] aims to reduce the number of sources selected per triple pattern and may produce solutions as $SS2$. In $SS2$, the minimum number of sources per triple pattern has been selected, i.e., only one source has been selected per triple pattern. However, $SS2$ is not a solution to the SSP-FR problem because this selection assigns $tp2$, $tp3$, and $tp4$ to different endpoints, leading to transfer a large amount of data.

We implement Fedra on top of the state-of-the-art federated query engines ANAPSID [1] and FedX [7], and empirically evaluate their performance using six federations. Each federation replicates random fragments from one of the datasets: Diseasome, Sematic Web Dog Food, DBpedia Geo-coordinates, Linked Movie Database, WatDiv [2] (with 10^5 and 10^7 triples). Queries are composed of one BGP with between 1 and 14 triple patterns. We evaluated 100 random queries against each federation. We compare our approach, Fedra, with FedX and ANAPSID source selections, and also with DAW source selection. Further informations (implementation, results, setups details) are available at `https://sites.google.com/site/fedrasourceselection`.

Both Fedra and DAW significantly reduce the number of selected sources, however, the reduction achieved by Fedra is greater than the achieved by DAW. Figure 1 presents the number of transferred tuples for FedX. Fedra in combination with FedX transfers considerably less tuples than the other approaches in all the federation and most queries.

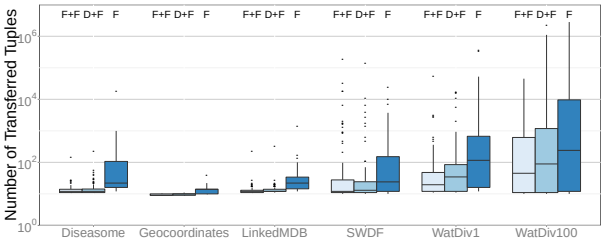


Figure 1: Number of Selected Sources and Transferred Tuples during execution with FedX (F) using Fedra (F+), DAW (D+), and the engine source selection

Even if Fedra source selection is aware of the replication, the combination of Fedra with existing query decomposition strategies does not allow to take full advantage of the replication knowledge to reduce the number of transferred tuples. Therefore, in the future, we will propose a query decomposition strategy that can effectively take advantage of replicated fragments in the federation.

Categories and Subject Descriptors

H.2.4 [**DATABASE MANAGEMENT**]: Systems—*Query processing*; H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Selection process*

Keywords

Semantic Web, Linked Data, Federated SPARQL Query Processing, Source Selection, Fragment Replication

1. REFERENCES

[1] M. Acosta, M. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In *ISWC 2011, Part I*, pages 18–34, 2011.

[2] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified Stress Testing of RDF Data Management Systems. In *ISWC 2014, Part I*, pages 197–212, 2014.

[3] K. Hose and R. Schenkel. Towards benefit-based RDF source selection for SPARQL queries. In *SWIM*, page 2, 2012.

[4] D. S. Johnson. Approximation Algorithms for Combinatorial Problems. In A. V. Aho et al., editors, *ACM Symposium on Theory of Computing*, pages 38–49. ACM, 1973.

[5] G. Montoya, H. Skaf-Molli, P. Molli, and M.-E. Vidal. Federated SPARQL Queries Processing with Replicated Fragments. In *ISWC 2015, Part I*, pages 36–51, 2015.

[6] M. Saleem, A.-C. N. Ngomo, J. X. Parreira, H. F. Deus, and M. Hauswirth. DAW: Duplicate-AWare Federated Query Processing over the Web of Data. In *ISWC 2013, Part I*, pages 574–590, 2013.

[7] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization Techniques for Federated Query Processing on Linked Data. In *ISWC 2011, Part I*, pages 601–616, 2011.

Mining Maximally Informative k-Itemsets in Massively Distributed Environments

Saber Salah*
Inria and LIRMM,
Zenith team,
University of Montpellier,
France,
saber.salah@inria.fr

Reza Akbarinia
Inria and LIRMM,
Zenith team,
University of Montpellier,
France,
reza.akbarinia@inria.fr

Florent Masseglia
Inria and LIRMM,
Zenith team,
University of Montpellier,
France,
florent.masseglia@inria.fr

ABSTRACT

The discovery of informative itemsets is a fundamental building block in data analytics and information retrieval. While the problem has been widely studied, only few solutions scale. This is particularly the case when i) the data set is massive, calling for large-scale distribution, and/or ii) the length k of the informative itemset to be discovered is high. In this paper, we address the problem of parallel mining of maximally informative k-itemsets (*miki*) based on joint entropy. We propose PHIKS (Parallel Highly Informative K-ItemSet) a highly scalable, parallel *miki* mining algorithm. PHIKS renders the mining process of large scale databases (up to terabytes of data) succinct and effective. Its mining process is made up of only two efficient parallel jobs. With PHIKS, we provide a set of significant optimizations for calculating the joint entropies of *miki* having different sizes, which drastically reduces the execution time of the mining process. PHIKS has been extensively evaluated using massive real-world data sets. Our experimental results confirm the effectiveness of our proposal by the significant scale-up obtained with high itemsets length and over very large databases.

La découverte d'itemsets informatifs est un élément fondamental dans l'analyse de donnés et la recherche d'information. Bien que le problème a été largement étudié, il y a peu de solutions qui passent à l'échelle. Ceci est particulièrement le cas lorsque i) les données sont de très grane taille, ce qui demande une distribution à grande échelle, et / ou ii) la longueur k des itemsets informatifs à découvrir est élevée. Dans cet article, nous abordons le problème de la fouille des k iems les plus informatifs (appelé *miki*) qui est calculé en considérant l'entropie conjointe des items. Nous proposons PHIKS (Parallel Highly Informative K-itemset), un algorithme parallèle d'extraction de miki. PHIKS rend le processus d'extraction de grandes bases de données à grande échelle (jusqu'à plusieurs téraoctets de données) rapide et efficace. Son processus d'extraction est constitué de seulement deux jobs parallèles. Avec PHIKS, nous proposons un ensemble d'optimisations significatives pour le calcul des entropies conjointes de miki ayant des tailles différentes, ce qui réduit considérablement le temps d'exécution du processus. PHIKS a été largement évaluée en utilisant de grands jeux de données réels. Nos résultats expérimentaux confirment l'efficacité de notre proposition qui passe à l'échelle pour de grandes valeurs de k et sur de très grandes bases de données.

*Saber Salah - This work was partially supported by the Inria Project Lab Hemera.

1. THE PHIKS APPROACH

Featureset, or itemset, mining [1] is one of the fundamental building bricks for exploring informative patterns in databases. Features might be, for instance, the words occurring in a document, the score given by a user to a movie on a social network, or the characteristics of plants (growth, genotype, humidity, biomass, etc.) in a scientific study in agronomics. A large number of contributions in the literature has been proposed for itemset mining, exploring various measures according to the chosen relevance criteria. The most studied measure is probably the number of co-occurrences of a set of features, also known as frequent itemsets [2]. However, frequency does not give relevant results for a various range of applications, including information retrieval [3], since it does not give a complete overview of the hidden correlations between the itemsets in the database. This is particularly the case when the database is sparse [4]. Using other criteria to assess the informativeness of an itemset could result in discovering interesting new patterns that were not previously known. To this end, information theory [5] gives us strong supports for measuring the informativeness of itemsets. One of the most popular measures is the joint entropy of an itemset. An itemset X that has higher joint entropy brings up more information about the objects in the database.

We study the problem of Maximally Informative k-Itemsets (*miki* for short) discovery in massive data sets, where informativeness is expressed by means of joint entropy and k is the size of the itemset [6, 7, 8]. *Miki* are itemsets of interest that better explain the correlations and relationships in the data. Example 1 gives an illustration of *miki* and its potential for real world applications such as information retrieval.

EXAMPLE 1. *In this application, we would like to retrieve documents from Table 1, in which the columns d_1, d_{10} are documents, and the attributes A, B, C, D, E are some features (items, keywords) in the documents. The value "1" means that the feature occurs in the document, and "0" not. It is easy to observe that the itemset (D, E) is frequent, because features D and E occur together in almost every document. However, it provides little help for document retrieval. In other words, given a document d_x in our data set, one might look for the occurrence of the itemset (D, E) and, depending on whether it occurs or not, she will not be able*

Features	Documents									
	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
A	1	1	1	1	1	0	0	0	0	0
B	0	1	0	0	1	1	0	1	0	1
C	1	0	0	1	0	1	1	0	1	0
D	1	0	1	1	1	1	1	1	1	1
E	1	1	1	1	1	1	1	1	1	1

Table 1: Features in the documents

to decide which document it is. By contrast, the itemset (A, B, C) *is infrequent, as its member features rarely or never appear together in the data. And it is troublesome to summarize the value patterns of the itemset* (A, B, C)*. Providing it with the values* $(1, 0, 0)$ *we could find the corresponding document* d_3*; similarly, given the values* $(0, 1, 1)$ *we will have the corresponding document* d_6*. Although* (A, B, C) *is infrequent, it contains lots of useful information which is hard to summarize. By looking at the values of each feature in the itemset* (A, B, C)*, it is much easier to decide exactly which document they belong to.* (A, B, C) *is a maximally informative itemset of size* $k = 3$*.*

Miki mining is a key problem in data analytics with high potential impact on various tasks such as supervised learning [9], unsupervised learning [10] or information retrieval [3], to cite a few. A typical application is the discovery of discriminative sets of features, based on joint entropy, which allows distinguishing between different categories of objects. Unfortunately, it is very difficult to maintain good results, in terms of both response time and quality, when the number of objects becomes very large. Indeed, with massive amounts of data, computing the joint entropies of all itemsets in parallel is a very challenging task for many reasons. First, the data is no longer located in one computer, instead, it is distributed over several machines. Second, the number of iterations of parallel jobs would be linear to k (i.e., the number of features in the itemset to be extracted [7]), which needs multiple database scans and in turn violates the parallel execution of the mining process. We believe that an efficient *miki* mining solution should scale up with the increase in the size of the itemsets, calling for cutting edge parallel algorithms and high performance evaluation of an itemset's joint entropy in massively distributed environments.

We propose a deep combination of both information theory and massive distribution by taking advantage of parallel programming frameworks such as MapReduce [11] or Spark [12]. To the best of our knowledge, there has been no prior work on parallel informative itemsets discovery based on joint entropy. We designed and developed an efficient parallel algorithm, namely Parallel Highly Informative K-itemSet (PHIKS in short), that renders the discovery of *miki* from a very large database (up to Terabytes of data) simple and effective. It performs the mining of *miki* in two parallel jobs. PHIKS cleverly exploits available data at each mapper to efficiently calculate the joint entropies of *miki* candidates. For more efficiency, we provide PHIKS with optimizations that allow for very significant improvements of the whole process of *miki* mining. The first technique estimates the upper bound of a given set of candidates and allows for a dramatic reduction of data communications, by filtering unpromising itemsets without having to perform any additional scan over the data. The second technique reduces significantly the number of scans over the input database of each mapper, i.e., only one scan per step, by incrementally computing the joint entropy of candidate features. This reduces drastically the work that should be done by the mappers, and thereby the total execution time.

PHIKS has been extensively evaluated using massive real-world data sets. Our experimental results show that PHIKS significantly outperforms alternative approaches, and confirm the effectiveness of our proposal over large databases containing for example one Terabyte of data.

2. ADDITIONAL AUTHORS

3. REFERENCES

[1] J. Han, *Data mining : concepts and techniques*. Elsevier/Morgan Kaufmann, 2012.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 1994, pp. 487–499.

[3] E. Greengrass, "Information retrieval: A survey," 2000.

[4] H. Heikinheimo, E. Hinkkanen, H. Mannila, T. Mielikäinen, and J. K. Seppänen, "Finding low-entropy sets and trees from binary data," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007, pp. 350–359.

[5] T. M. Cover, *Elements of information theory*. Hoboken, N.J: Wiley-Interscience, 2006.

[6] R. Gray, *Entropy and information theory*. New York: Springer, 2011.

[7] A. J. Knobbe and E. K. Y. Ho, "Maximally informative k-itemsets and their efficient discovery," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 237–244.

[8] C. Zhang and F. Masseglia, "Discovering highly informative feature sets from data streams," in *Database and Expert Systems Applications*, 2010, pp. 91–104.

[9] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Proceedings of International Conference on Emerging Artificial Intelligence Applications in Computer Engineering*, 2007, pp. 3–24.

[10] Z. Ghahramani, "Unsupervised learning," in *Advanced Lectures on Machine Learning*, 2004, pp. 72–112.

[11] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conf. on Hot Topics in Cloud Computing*, 2010, pp. 10–10.

Inférence de Schémas pour Données JSON Massives

Mohamed-Amine Baazizi
LIP6
Université Pierre et Marie Curie
Mohamed-Amine.Baazizi@lip6.fr

Houssem Ben Lahmar*
IPVS
University of Stuttgart
Houssem.Ben-Lahmar@ipvs.uni-stuttgart.de

Dario Colazzo
Université Paris-Dauphine,
PSL Research University,
CNRS, LAMSADE
75016 PARIS, FRANCE
dario.colazzo@dauphine.fr

Giorgio Ghelli
Dipartimento di Informatica
Università di Pisa
ghelli@di.unipi.it

Carlo Sartiani
DIMIE
Università della Basilicata
sartiani@gmail.com

Résumé.
Ces dernières années ont connu une large adoption de JSON en tant que format de représentation de données massives. Les données JSON sont généralement dépourvues de schémas puisqu'elles sont produites et gérées de manière flexible. Malgré cet avantage, l'absence de schéma présente de nombreux inconvénients : la correction des requêtes et des programmes ne peut être vérifiée de manière statique comme c'est la cas traditionnellement, les utilisateurs ne disposent d'aucun moyen le permettant de découvrir la structure des données sous-jacentes et, de manière plus générale, les techniques d'optimisations basées sur les schémas ne peuvent être appliquées.

Dans ce travail nous nous intéressons à l'inférence de schémas pour des données JSON massives. Notre première contribution consiste à proposer un langage de types pour JSON permettant de représenter la structure complexe des données analysées. Notre seconde contribution concerne le développement d'un algorithme d'inférence distribué et de son implantation dans Spark afin de garantir une exécution efficace sur des données volumineuses. Les résultats obtenus suite à une première étude expérimentale permettent de conclure que notre approche est satisfaisant en terme de temps d'exécution et de concision de schémas inférés.

1. INTRODUCTION

Les applications modernes sont amenées à analyser de larges volumes de données semi-strucutrées. Dans la plupart de ces applications et dans les applications communément appelées NoSQL, les données sont représentées en JSON, ce dernier étant un format particulièrement apprécié pour sa flexibilité et sa simplicité. Dans ce contexte, l'utilisation de schémas n'est pas obligatoire contrairement au cas relationnel. Ceci a pour avantage de permettre de déployer des applications de manière rapide sans avoir à se soucier du format des données échangées. Bien entendu, l'absence d'un schéma complique la plupart des opérations d'accès aux données à commencer par l'interrogation puisque l'utilisateur ne dispose d'aucune vue synthétique lui permettant de comprendre la structure sous-jacente des données. L'absence de schéma prive également les SGBD de tirer profit de techniques d'optimisation basées sur le schéma tels que la réécriture de requêtes ou la projection.

Dans ce travail nous proposons une approche permettant d'inférer, à partir d'une large collection de documents JSON, un schéma avec comme premier objectif de refléter la structure des données en entrée en tenant compte de l'imbrication des objets et du caractère obligatoire ou facultatif des attributs. Afin de permettre que ce schéma soit exploitable, notre technique d'inférence doit produire des schéma concis en évitant la redondance d'information. Enfin, afin de pouvoir manipuler de larges jeux de données, notre technique doit passer à l'échelle.

La technique que nous proposons s'appuie sur deux étapes qui se greffent dans un programme Map-Reduce de façon assez naturelle. Dans un premier temps, chaque document de la collection en entrée est analysé séparément dans le but d'en inférer un schéma. Cette étape se déroule de façon parallèle et est donc intégrée à la phase *Map*. Dans un second temps, les schémas produits dans la phase initiale sont fusionnés dans le but de produire le schéma global capturant la structure de tous les documents en entrée. Cette étape est intégrée à la phase *Reduce*.

La suite du document est consacrée à l'illustration de notre technique d'inférence. Faute d'espace, la présentation formelle est reléguée à l'article long [2].

*Houssem Ben Lahmar has been partially supported by the project D03 of SFB/Transregio 161 http://www.trr161.de/interfak/forschergruppen/sfbtrr161.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. APERÇU SUR L'INFÉRENCE DE SCHÉMAS

Afin d'illustrer notre technique, rappelons d'abord la syntaxe et la sémantique de JSON. Comme dans la plupart des modèles semi-structurés, JSON utilise des valeurs bases (null, booléens, entiers et chaînes de caractères) servant à stocker des informations atomiques et des valeurs complexes permettant de structurer les données soit dans un *object* ou

une *séquence*. Un object est un ensemble d'attributs, chaque attribut associe à une clé une valeur donnée. Une séquence est une liste (ordonnée) de valeurs quelconques.

Afin d'illustrer la syntaxe de JSON considérons le document d_1 ci-dessous.

```
{"A": 123,  "B": "The ...",  "C": false,
    "D": ["abc", "cde", "fr12"] }
```

Ce document est construit à partir d'un objet délimité par des accolades. Cet objet est composé des clés "A", "B", "C" et "D" auxquelles sont associés respectivement un entier, une chaîne de caractères, un booléen et une séquence. La séquence associée à "D" est délimitée par des crochets et est composée de trois chaînes de caractères.

Il est important de noter qu'il n'existe pas de langage de schéma standard pour représenter un schéma JSON. Néanmoins le langage Json-Schema [1] semble être le plus adopté dans la pratique et a d'ailleurs fait l'objet d'une étude formelle dans [3] visant à révéler ses différentes caractéristiques. Grosso modo, Json-Schema permet de représenter la structure d'un document JSON en exhibant sa structure, abstrait les valeurs atomiques en des types pré-définis, précise si les attributs sont facultatifs ou obligatoires et indique la cardinalité des éléments des séquences. Il a été démontré dans [3] que le problème de validation d'une instance par rapport à un schéma de ce langage est PTIME-hard. La problématique d'inférence de schémas n'a pas été considéré dans la littérature.

Dans notre étude où l'intérêt est d'inférer un schéma succinct à partir d'une collection de documents, nous proposons un langage de schéma permettant, de la même manière que Json-Schema, d'abstraire les valeurs atomiques vers des types atomiques, de représenter la structure des données en entrée et de savoir pour chaque attribut s'il est facultatif ou obligatoire. Le schéma s_1 correspondant au document d_1 est donc

```
{"A": Number,  "B": String,  "C": Boolean,
      "D": [String, String, String] }
```

Rappelons que l'un des objectifs de notre approche est de construire des schémas succincts. Ceci passe par l'encodage compact du contenu des séquences en introduisant l'étoile de Kleene. Ainsi, il sera possible de ne renseigner que les types distincts qui apparaissent dans une séquence. Dans notre exemple, le schéma s_1 devient

```
{"A": Number,  "B": String,  "C": Boolean,
      "D": [String*] }
```

Nous allons maintenant nous intéresser à l'opération de fusion qui prend en entrée deux schémas s et s' et doit produire un schéma capturant toutes les caractéristiques de s et s' tout en étant compact. Intuitivement, cette opération consiste à combiner les parties de s et s' qui sont identiques et à retourner les parties restantes telles quelles. Afin d'illustrer cette opération considérons le schéma s_2 ci-dessous qui sera fusionné avec s_1 décrit précédemment.

```
{"A": Number,  "B": Number,  "C": Boolean,
      "D": [Number*] }
```

Il est aisé de constater que les attributs identifiés par "A", "C" sont identiques dans les deux schémas et que, par conséquent, ils seront fusionnés dans le résultat. Les attributs identifiés par "B" et "D" sont, quant à eux, différents et devront subir un traitement particulier permettant de garder l'information sur la variation de leur structure dans s_1 et s_2. Pour ce faire, nous utilisons l'opérateur d'union (+) qui indique toutes les possibilités rencontrées lors de la fusion. Le résultat de la fusion de s_1 et s_2 est donc

```
{"A": Number,  "B": String+Number,  "C":
    Boolean,  "D": [(String+Number)*] }
```

Il est important de noter que le schéma retourné est un super-type des schémas s_1 et s_2 en entrée. Il est aussi important de souligner que pour permettre l'exécution de la fusion en Map-Reduce, la fusion est commutative et associative.

3. CONCLUSIONS ET PERSPECTIVES

L'approche décrite dans cet article est une première étape vers le développement d'une technique d'inférence de schémas pour une large collection de documents JSON. Le but principal est de fournir une vue succincte des données sous-jacentes afin de faciliter leur manipulation (stockage, interrogation, transformation). Bien que relativement simple, le langage de schéma utilisé permet de refléter les variations structurelles des données tout en offrant une vue succincte de celles-ci. Une étude expérimentale (consultable en [2]) a pu démontré que l'approche proposée remplit les objectifs fixés en amont : la concision et le passage à l'échelle. En guise de perspective, nous étudions la possibilité d'enrichir le langage de schéma d'information sur la cardinalité des données afin de mieux cerner les caractéristiques statistiques des données. Nous envisageons également d'étudier le compromis entre concision et précision des schémas produits en nous appuyant sur des cas d'études réels.

4. REFERENCES

[1] The json schema language. `http://json-schema.org/`.

[2] M.-A. Baazizi, H. B. Lahmar, D. Colazzo, G. Ghelli, and C. Sartiano. Schema inference for massive json datasets. *EDBT*, 2017 (to appear).

[3] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoc. Foundations of JSON schema. In J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 263–273. ACM, 2016.

Hybrid algorithms for candidate keys enumeration for a relational schema

Karima Ennaoui
LIMOS, Universite Blaise Pascal
Clermont-ferrand, France
ennaoui@isima.fr

Lhouari Nourine
LIMOS, Universite Blaise Pascal
Clermont-ferrand, France
nourine@isima.fr

RESUME

Cet article étudie le problème d'énumération des clés candidates d'un schéma relationnel de base de données. La notion de clés candidates est également connue sous le nom de générateurs minimaux en terminologie de treillis ou FCA.

Nous considérons le cas où l'entrée est un schéma relationnel, composé de paires de sous-ensembles d'attributs représentant les interrelations entre les attributs, à savoir un ensemble de dépendances fonctionnelles. Une clé candidate, dans un tel schéma, est un sous-ensemble minimal d'attributs dont le fermé par rapport aux dépendances fonctionnelles est égal à l'ensemble entier des attributs.

Lucchesi et Osborn ont donné en 1978 un algorithme polynomial incrémental avec un espace exponentiel pour énumérer toutes les clés candidates d'un schéma relationnel. Leur algorithme exploite la forte connectivité du graphe des clés candidates orienté, où chaque nœud correspond à une de ces clés et la fonction de transition est définie par une substitution d'un attribut par une dépendance fonctionnelle.

Saiedian et Spencer ont introduit en 1996 la notion de graphe d'attributs, en ramenant le problème d'énumération des clés candidates du schéma à énumérer les clés candidates des composantes fortement connexes du graphe.

L'existence d'un algorithme d'énumération des clés candidates en délai polynomial reste une question ouverte.

Dans un schéma relationnel d'une base de données, une dépendance fonctionnelle (P, Q) est dite unitaire si P est un singleton. Dans cet article, nous exploitons la présence de dépendances fonctionnelles unitaires qui forment un ordre partiel sur l'ensemble des attributs.

En considérant cet ordre, nous introduisons la notion de clé-idéal, qui est un idéal où les éléments maximaux forment une clé candidate. Nous distinguons les clé-idéaux minimaux au sens d'inclusion. Il existe une bijection entre l'ensemble des clé-idéaux et les clés candidates, qui prouve une équivalence linéaire entre le problème d'énumération des clés candidates et celui des clé-idéaux. Cependant, l'espace de recherche du dernier problème est réduit. En outre, nous montrons que le nombre de clé-idéaux peut être exponentiel par rapport au nombre de clé-idéaux minimaux.

Nous prouvons que s'il existe un algorithme qui énumère les clé-idéaux minimaux en délai et espace polynomial alors On peut énumérer toutes les clés candidates en délai et espace polynomial. Nous donnons également un algorithme en espace polynomial et délai polynomial incrémental pour énumérer les clé-idéaux minimaux. En conséquence, si le nombre de clé-idéaux minimaux est polynomial en fonction de la taille du schéma relationnel, alors il existe un algorithme en délai et espace polynomial pour énumérer toutes les clé candidates.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

Filtres pour jointures et requêtes récursives en MapReduce

Thuong-Cang Phan
Blaise Pascal University
CNRS, UMR 6158, LIMOS
Clermont-Ferrand, France
ptcang@isima.fr

Thi-To-Quyen Tran
Blaise Pascal University
CNRS, UMR 6158, LIMOS
Clermont-Ferrand, France
tttquyen@ctu.edu.vn

Laurent d'Orazio
Blaise Pascal University
CNRS, UMR 6158, LIMOS
Clermont-Ferrand, France
dorazio@isima.fr

ABSTRACT

Le traitement, l'analyse et la génération de données massives constituent un problème majeur au coeur de nombreuses applications (Internet, réseaux sociaux, santé, etc.). Des paradigmes pour le traitement de données massivement parallèles ont ainsi été proposés, le plus connu et l'un des plus répandu étant sans doute MapReduce. Malheureusement, MapReduce présente d'importantes limites en particulier pour des opérations nécessitant plus d'une source de données ou encore basées sur des processus itératifs. Ce papier s'intéresse à l'optimisation de la gestion de données dans des environnements massivement parallèles et plus précisément à l'optimisation de l'opération de jointure à l'aide de MapReduce. Nous introduisons de nouveaux filtres, le filtre d'intersection et le filtre de différence, qui visent à réduire les données intermédiaires (et donc la charge de travail), ainsi que le nombre de processus pour la jointure et les requêtes récursives. Des expérimentations à l'aide d'un banc d'essai permettent de démontrer les avantages de nos solutions.

Keywords

Data analysis, Cloud computing, Join, MapReduce, Intersection Bloom filter, Difference filter

1. INTRODUCTION

Dans certains domaines d'application, à l'image de la cosmologie, des réseaux sociaux ou encore de la publicité numérique, les volumes peuvent être si importants, qu'il est nécessaire de recourir à des environnements de traitement massivement parallèles (*Massively Parallel Processing* ou MPP) tels que MapReduce. Malheureusement, ce paradigme, à l'origine, ne considère par des opérations telles que la jointure[1] (opération commune dans de nombreuses requêtes) ou encore les requêtes récursives. Le problème est alors de proposer des solutions pour permettre de tels traitements, de manière efficace afin de réduire l'utilisation des ressources que ce soit en termes de consommation de bande passante ou d'utilisation processeur.

Ainsi ces dernières années, différentes approches ont été proposées [1, 2] afin de réaliser la jointure sur des noeuds de types Map ou alors de type Reduce. Malheureusement, ces approches imposent des contraintes fortes sur les données (un ensemble suffisamment petit pour être diffusé à tous les noeuds) ou leur organisation (tri selon l'attribut de jointure, placement des données sur des noeuds spécifiques), menant à de nombreux transferts de données (certains inutiles) et une charge de travail importante sur les machines ou alors nécessitant plusieurs phases (coûteuses) d'exécution. Le nombre de phases est également problématique pour le traitement des requêtes récursives à l'aide de MapReduce [4].

Cet article intéresse au traitement efficace de la jointure et des requêtes récursives en MapReduce. Il introduit ainsi des extensions du filtre de Bloom: (1) un filtre d'intersection permettant de filtrer des données non pertinentes (ne participant à l'opération) et ainsi réduire les transferts de données sur le réseau et la charge de travail sur les noeuds du centre de traitement de données et (2) un filtre de différence se substituant notamment à la condition d'arrêt de l'algorithme semi-naïf pour le traitement des requêtes récursives, limitant ainsi le nombre de phases MapReduce lors de l'exécution.

2. JOINTURES ET FILTRE D'INTERSECTION

Le filtre d'intersection IF (pour *Intersection Filter*) de deux ensembles E_1 et E_2, noté $IF(E_1 \cap E_2)$ est une structure de données probabiliste représentant une approximation de l'intersection d'ensembles. Il est utilisé pour déterminer si un élément e est commun à deux ensembles E_1 et E_2. La réponse est négative si e n'est pas commun. La réponse est positive si e est éventuellement commun, la structure présentant une probabilité de faux positif. Nous proposons trois modélisation pour IF à l'aide de filtres de Bloom: (1) pair de filtres de Bloom, (2) intersection de filtres de Bloom et (3) intersection de filtres de Bloom répartis.

En plus de IF, nous introduisons le concept de Filtre d'Intersection Etendu EIF (pour *Extended Intersection Filter*). L'objectif de cette structure est de vérifier l'appartenance d'un élément e à plusieurs ensembles E_1, ..., E_k en étudiant l'appartenance à des sous-éléments e_1, e_2, ..., e_n de e aux ensembles considérés.

Les filtres d'intersection sont particulièrement pertinents dans le cas de la jointure binaire, $R_1 \bowtie_{a1=a2} R_2$. Ils permettent lors du calcul d'une jointure de supprimer pour chacune

[1]Le terme jointure est ici utilisé comme abus de langage et se réfère au cas particulier de l'équi-jointure.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

des deux relations en entrée des éléments n'ayant pas de correspondance dans l'autre ensemble de données. Ceci a pour conséquence de réduire le processus de calcul et également le volume de données intermédiaires transitant sur le réseau.

Les filtres d'intersection peuvent également être utilisés pour la jointure chaînée. Il est bien sûr possible d'appliquer les IF des deux relations considérées successivement lors d'itérations de jointures binaires. Cette approche mène cependant au transfert de données non pertinentes lors des jointures intermédiaires. Par conséquent, nous proposons une autre solution qui est l'utilisation de cascade de jointures ternaires à l'aide de différents EIF, en anticipant comme pour la première approche la prochaine opération de jointure. Par exemple pour n ensembles de données, le nombre de jointures binaires est n-1 pour la première solution, contre (n-1)/2 pour la deuxième.

3. REQUÊTE RÉCURSIVES, ITÉRATION ET FILTRE DE DIFFÉRENCE

Le Filtre de Différence d'un ensemble E, noté $DF(E)$ (pour *Difference Filter*), est une structure probabiliste représentant une approximation des éléments n'appartenant pas à E. La réponse retournée est positive si l'élément x testé n'appartient pas à l'ensemble (élément disjoint). Elle est négative si x appartient à l'ensemble ou inconnue si x peut appartenir à l'ensemble. Ainsi, une propriété importante de la structure est qu'elle n'engendre pas de faux positif, même si elle permet des faux négatifs.

Un DF d'un ensemble S est composé d'un filtre de Bloom $BF(S)$ et d'une table de hachage à perte ou *Lossy Hash Table LHT*. $LHT(S)$ est utilisée pour détecter les faux positifs. Elle est constituée d'un tableau de seaux (*buckets*), dans lequel chaque seau contient une empreinte d'un élément de S. L'empreinte est créée en appliquant une fonction de hachage cryptographique h_c (par exemple, MD5 ou SHA-1) à un élément. Le fonctionnement de $DF(S)$ est le suivant. Un élément x de R est confronté à $BF(S)$. S'il n'est pas dans $BF(S)$, il est vrai que x n'appartient pas à S. Si par contre x appartient à $BF(S)$, il est envoyé à $LHT(S)$. Si x est dans $LHT(S)$, autrement dit l'empreinte $h_c(x)$ est identique au contenu de $LHT[h_p(x)]$ la réponse retournée est non, car l'élément appartient à S. Si l'empreinte est différente, la réponse est inconnue, avec deux cas de figure : l'élément n'est jamais présent ou l'élément a été écrasé par un autre.

L'utilisation conjointe de filtres de type DF et IF permet d'améliorer l'exécution de requêtes récursives à l'aide de MapReduce par rapport à l'approche de référence [5] basé sur l'algorithme 1 ou algorithme semi-naïve, en réduisant les transferts de données et le nombre de phases MapReduce.

Algorithm 1: Semi-naive algorithm

Require: K(x,y) : une relation
$F_0 \leftarrow$ vide
$i \leftarrow 0$
$\Delta F_0 \leftarrow$ K(x,y)
while $\Delta F_0 \neq$ vide **do**
i++
$F_i \leftarrow \Delta F_0 \cup ... \cup \Delta F_{i-1}$
$\Delta F_i \leftarrow \Pi\ (\Delta F_{i-1} \bowtie \mathrm{K})$ - F_i
end while

Un IF permet de supprimer les n-uplets de K et des ΔF_{i-1} (aux faux négatifs près) non concernés par la jointure $\Delta F_{i-1} \bowtie K$. DF est utilisé pour enlever des résultats issus de la jointures à l'itération i ceux produits lors des itérations précédentes. Autrement dit, il permet de calculer une approximation de ΔF_i=$\Delta F_i - F_{i-1}$ et de détecter le point fixe si l'ensemble résultat est vide, sans avoir recours à un processus MapReduce supplémentaire. De plus, un cache est utilisé en entrée de la phase Reduce pour stocker localement la relation K, qui est invariante.

Il est important de noter que le filtre de Bloom classique permet uniquement la représentation d'ensembles statiques dont la taille doit être estimée avant la construction et le déploiement. Dans le cas de l'algorithme semi-naïf, la taille de l'ensemble F_{i-1} augmente lorsqu'un nouvel élément est découvert dans ΔF_i. C'est pourquoi, nous proposons d'utiliser le filtre de Bloom dynamique [3] dont la taille peut être modifiée.

4. CONCLUSION

Nos recherches ont abordé la problématique des performances de certaines opérations en MapReduce. Nous avons proposé une première extension du filtre de Bloom appelée filtre d'intersection qui peut éliminer la plupart des éléments des ensembles de données d'entrée ne participant pas à la jointure, avant d'envoyer les données pour le traitement de l'opération. Une autre proposition est le filtre de différence, structure de données probabiliste visant à étudier des éléments disjoints. Ce filtre peut être appliqué à un grand nombre de problèmes, tels que la réconciliation, la déduplication, la correction d'erreur et en ce qui nous concerne la jointure récursive. Notre amélioration réduit de moitié le nombre de tâches effectuées (et les lectures de données associées), la génération des données intermédiaires et les communications. Ceci permet notamment une amélioration de l'évaluation de l'algorithme semi-naïf et par conséquent l'évaluation des requêtes récursives en MapReduce. Des expérimentations sur un centre de traitement de données privé et des jeux de données synthétiques ont mis en évidence l'intérêt de ces solutions.

5. REFERENCES

[1] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A Comparison of Join Algorithms for Log Processing in MaPreduce. In *SIGMOD*, pages 975–986, 2010.

[2] E. F. Codd. Relational Completeness of Data Base Sublanguages. *In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California*, 1972.

[3] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo. The Dynamic Bloom Filters. *TKDE*, 22(1):120–133, 2010.

[4] S. Idreos, E. Liarou, and M. Koubarakis. Continuous Multi-way Joins over Distributed Hash Tables. In *EDBT*, pages 594–605, 2008.

[5] M. Shaw, P. Koutris, B. Howe, and D. Suciu. Optimizing Large-Scale Semi-NaÃŕve Datalog Evaluation in Hadoop. In *Datalog in Academia and Industry*, number 7494 in Lecture Notes in Computer Science, pages 165–176. 2012.

Raffinement interactif de mapping à partir d'exemples de données fournis par l'utilisateur

Angela Bonifati
Université Lyon 1
angela.bonifati@univ-lyon1.fr

Ugo Comignani
Université Lyon 1
ugo.comignani@univ-lyon1.fr

Emmanuel Coquery
Université Lyon 1
emmanuel.coquery@univ-lyon1.fr

Romuald Thion
Université Lyon 1
romuald.thion@univ-lyon1.fr

Les mappings de schémas [2] sont des spécifications déclaratives des relations sémantiques entre des éléments d'un schéma source et d'un schéma cible. Ils sont typiquement décrits en logique du premier ordre et constituent les briques fondamentales dans les domaines de l'échange et de l'intégration de données (respectivement, la transformation d'une instance source en une instance cible ou la réécriture d'une requête sur une instance cible en un ensemble de requêtes sur les instances sources sous-jacentes). La spécification de ces mappings de schémas est une tâche ardue pour un utilisateur non-expert n'étant familier ni avec la sémantique et les langages de transformation de données, ni avec la manipulation des schémas utilisés. Cependant, la spécification de mappings sémantiquement corrects est une étape cruciale vers une résolution qualitative des problèmes d'échange et d'intégration de données.

Deux paradigmes ont été proposés pour faciliter la spécification de mappings de schémas aux utilisateurs non-experts. Le premier paradigme repose sur la spécification visuelle de mappings à l'aide d'une interface graphique ergonomique, e.g., [3]. Cependant, le problème majeur de ces approches est la dépendance à des outils spécifiques des primitives graphiques utilisées : en effet, une même spécification graphique peut être traduite par des outils différents en des mappings distincts et non comparables. Dans le second paradigme, qui est celui utilisé ici, on génère les mappings désirés à partir d'exemples de données représentatifs [1], i.e. une paire de sous ensembles des instances sources et cibles donnée par l'utilisateur. Dans ce cas, on attend de l'utilisateur qu'il fournisse des exemples de données valides et informatifs. La qualité de ces exemples a été définie dans la littérature comme étant reliée à la notion de solutions universelles (les solutions les plus représentatives des problèmes d'échanges de données). Ces solutions sont supérieures à des solutions arbitrairement choisies, car elles possèdent des homomorphismes vers l'ensemble des autres solutions. Concrètement, cette contrainte d'universalité signifie que les exemples de données ne doivent pas contenir de tuples excédentaires, et que chaque constante doit avoir été choisie consciencieusement de manière à éviter des interactions entre tuples qui ne seraient pas sémantiquement significatives. Dans ce papier, la contrainte d'universalité des exemples de données a été supprimée pour, au contraire, proposer une approche permettant l'utilisation de petits exemples de données arbitraires tels qu'ils peuvent être produits par un utilisateur non expert. Ces exemples de données sont ensuite traités et, à l'aide d'interactions utilisateur simples, raffinés afin d'obtenir le mapping correspondant aux attentes de l'utilisateur.

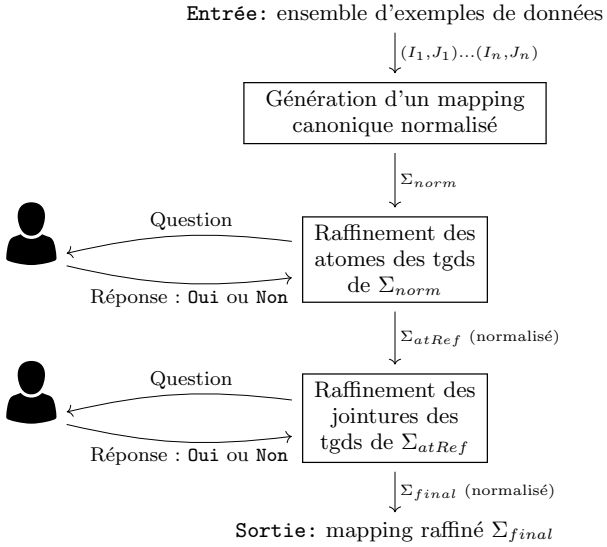


Figure 1: Spécification interactive de mapping

La figure 1 montre le processus de raffinement qui est présenté dans ce papier. L'utilisateur fournit en entrée du processus un ensemble d'exemples de données (I_1, J_1) ... (I_n, J_n) et, à la suite d'un ensemble de phases de raffinement via des interactions utilisateur, le système retourne un mapping Σ_{final}. Les exemples (I_1, J_1) ... (I_n, J_n) fournis en entrée sont formés d'une instance source I_k et d'une instance cible J_k. Chaque exemple sera habituellement de faible

BDA 2016, 15 au 18 Novembre, Poitiers, France.

(i) Instance source I :

T (*Trajet*) :

IdVolDepart	*IdVolRetour*	*IdAgence*
f0	f1	a0

A (*Agence*) :

IdAgence	*Nom*	*Ville*
a0	TC	Paris

CA (*CompagnieAerienne*) :

IdCompagnie	*Nom*	*Ville*
a1	AF	Paris

V (*Vol*) :

IdVol	*Depuis*	*Vers*	*IdCompagnie*
f0	Lyon	Paris	a1
f1	Paris	Lyon	a1

(ii) Mapping canonique :

$$T(\mathrm{idF}_0, \mathrm{idF}_1, \mathrm{idAg}) \wedge A(\mathrm{idAg}, \mathrm{nom}, \mathrm{v}) \wedge V(\mathrm{idF}_0, \mathrm{v}', \mathrm{v}, \mathrm{idAir}) \wedge V(\mathrm{idF}_1, \mathrm{v}, \mathrm{v}', \mathrm{idAir}) \wedge CA(\mathrm{idAir}, \mathrm{nom}', \mathrm{v}) \rightarrow \exists \mathrm{idC}_0, \mathrm{idC}_1, \mathrm{idC}_2, \mathrm{idC}_3, \mathrm{idC}_4, Co(\mathrm{idC}_0, \mathrm{nom}, \mathrm{v}) \wedge Dep(\mathrm{v}', \mathrm{idF}_0, \mathrm{idC}_1) \wedge Co(\mathrm{idC}_1, \mathrm{nom}', \mathrm{v}) \wedge Arr(\mathrm{v}, \mathrm{idF}_0, \mathrm{idC}_2) \wedge Co(\mathrm{idC}_2, \mathrm{nom}', \mathrm{v}) \wedge Dep(\mathrm{v}, \mathrm{idF}_1, \mathrm{idC}_3) \wedge Co(\mathrm{idC}_3, \mathrm{nom}', \mathrm{v}) \wedge Arr(\mathrm{v}', \mathrm{idF}_1, \mathrm{idC}_4) \wedge Co(\mathrm{idC}_4, \mathrm{nom}', \mathrm{v})$$

(iii) Instance cible J :

Dep (*Depart*) :

Ville	*IdVol*	*IdCompagnie*
Lyon	f0	P1
Paris	f1	P2

Arr (*Arrivee*) :

Ville	*IdVol*	*IdCompagnie*
Paris	f0	P3
Lyon	f1	P4

Co (*Compagnie*) :

IdCompagnie	*Nom*	*Ville*
P0	TC	Paris
P1	AF	Paris
P2	AF	Paris
P3	AF	Paris
P4	AF	Paris

(iv) Mapping final après raffinement :

$$\Sigma_{final} = \{A(\mathrm{idAg}, \mathrm{nom}, \mathrm{v}) \rightarrow \exists \mathrm{idC}_0, Co(\mathrm{idC}_0, \mathrm{nom}, \mathrm{v});$$
$$V(\mathrm{idF}_0, \mathrm{v}', \mathrm{v}_1, \mathrm{idAir}) \wedge CA(\mathrm{idAir}, \mathrm{nom}', \mathrm{v}_2) \rightarrow \exists \mathrm{idC}_1, Dep(\mathrm{v}', \mathrm{idF}_0, \mathrm{idC}_1) \wedge Co(\mathrm{idC}_1, \mathrm{nom}', \mathrm{v}_2);$$
$$V(\mathrm{idF}_0, \mathrm{v}', \mathrm{v}_1, \mathrm{idAir}) \wedge CA(\mathrm{idAir}, \mathrm{nom}', \mathrm{v}_2) \rightarrow \exists \mathrm{idC}_2, Arr(\mathrm{v}_1, \mathrm{idF}_0, \mathrm{idC}_2) \wedge Co(\mathrm{idC}_2, \mathrm{nom}', \mathrm{v}_2)\}$$

Figure 2: Exemple : exemple de données (I, J) (i) et (iii), resp. ; Mapping canonique (ii) et Mapping final (iv).

taille (de l'ordre d'une dizaine de tuples) et joue le rôle de référence dans la génération des interactions utilisateur.

Le processus est composé de trois phases détaillées ci-après :

Génération d'un mapping canonique Il s'agit d'une étape de prétraitement durant laquelle chaque exemple de donnée (I, J) est traité comme s'il était universel. Le mapping canonique, appelé Σ_{norm} dans la figure, est une unique assertion qui, lors du prétraitement, est scindée en un ensemble logiquement équivalent d'assertions plus petites.

Raffinement des atomes À la suite du prétraitement, pour chaque tgd dans Σ_{norm} un ensemble de questions booléennes est soumis à l'utilisateur afin d'éliminer les tuples superflus des exemples de données. Ces questions prennent la forme suivante : "Les tuples [sous-ensemble de I] sont-ils suffisants pour déduire les tuples [sous-ensemble de J]".

Raffinement des jointures Le raffinement des jointures est une étape similaire à la précédente de par les interactions utilisateur qui entrent en jeu. Le but de cette étape est de vérifier la signification des valeurs des tuples, plus précisément le but est de vérifier si les occurrences multiples d'une constante relèvent de jointures sémantiquement valides ou apparaissent de manière fortuite. À la fin de cette étape, le mapping final Σ_{final} est renvoyé à l'utilisateur.

Comme scénario, on considère l'exemple de données présenté en figure 2(i) et 2(iii).

On peut observer des ambiguïtés dans les instances de cet exemple de données, comme pour la constante `Paris` qui est présente à la fois en tant que ville du siège de l'agence du voyage et comme ville d'arrivée d'un vol. En conséquence, le mapping canonique obtenu à partir de l'exemple précédent, et présenté dans la figure 2 (ii), ne représente qu'une généralisation incorrecte de la sémantique. Par exemple, ce mapping implique que, dans l'ensemble des solutions, la ville d'arrivée d'un vol de départ d'un trajet doit également être la ville du siège social de la compagnie proposant ce trajet. Ainsi, notre approche a pour but de clarifier ces ambiguïtés en explorant les l'espaces des solutions possibles de manière efficace, afin d'obtenir un mapping désambiguïsé comme présenté en figure 2 (ii) .

Les contributions principales sont les suivantes :

— Nous définissons un processus de spécification de mapping à partir d'un ensemble d'exemples de données, et relaxons la contrainte d'universalité de ces exemples posés par les travaux précédents.

— Le cœur de ce processus est un processus de raffinement interactif durant lequel l'utilisateur doit répondre à des questions booléennes sur la validité de sous-ensembles des exemples qu'il a fournis en entrée. Ces questions sont générées à partir de l'exploration interactive de sup demi-treillis, et profitant d'un élagage dynamique de celui-ci pour conserver un nombre d'interactions raisonnablement faible.

— Nous prouvons que le processus de raffinement produit toujours un mapping plus général que le mapping canonique de départ. De plus, nous montrons que le mapping canonique peut toujours être obtenu par l'utilisateur si celui-ci choisit de valider systématiquement le supremum des sup demi-treillis explorés.

— La nature "raisonnable" de la quantité de questions posées à l'utilisateur pour inférer un mapping à été vérifiée expérimentalement sur sept jeux de données réels et classiques du domaine.

1. REFERENCES

[1] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Designing and refining schema mappings via data examples. In *Proceedings of the ACM SIGMOD 2011*, pages 133–144, 2011.

[2] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange : semantics and query answering. *Theoretical Computer Science*, 336(1) :89–124, 2005.

[3] L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *Proceedings of the ACM SIGMOD 2001*, pages 485–496, 2001.

A scalability comparison study of data management approaches for smart metering systems

Houssem Chihoub
Grenoble Laboratory of Informatics (LIG)
Grenoble Institute of Technology (INP)
Grenoble, France
houssem.chihoub@imag.fr

Christine Collet
Grenoble Laboratory of Informatics (LIG)
Grenoble Institute of Technology (INP)
Grenoble, France
christine.collet@grenoble-inp.fr

Smart grids are electrical grids that combine conventional and renewable energy resources, and rely on digital advances in smart metering, sensoring technologies, and two-way communication means in order to provide efficient management of energy. Recent years have seen many energy utilities invest heavily to transform their power grid infrastructures. For instance, in France ENEDIS is planning to deploy 35 million smart meters by the year 2021 [3]. Many utility providers (electricity, water, etc.) rely on smart metering and deploy Advanced Metering Infrastructures (AMIs) to guarantee an efficient management of their resources. The Advanced Metering Infrastructure (AMI) is the system that integrates smart meters, network protocols, and the meter data management system (MDM). In smart grids, data are generally sent from meters to head-ends (e.g. data concentrators) using low-throughput communication means such as the G3-CPL protocol. Meter data is further sent to the MDM where it should be stored. The MDM system insures data cleaning, storage, access, and processing.

With massive deployments of millions of smart meters and sensors throughout the power grid, more and more data are collected providing fine grain insights about client consumption profiles and the behavior of the power grid. Such data are critical towards more efficient management of energy resources and provides new level of efficiency for analytic-based applications such as predictive maintenance and demand forecasting. However, the frequency in which data are generated and collected from millions of sensors and smart meters at the scale of a country makes the task of data management and processing very complex. This complexity is due to the huge volumes of data collected over long periods of time to be stored, as well as the performance requirements on accessing and processing these data. MDM systems rely, generally, on legacy relational database systems adapted to manage meter data. As a result, many of these systems have started to exhibit major difficulties to cope with the data deluge of smart metering and sensoring and meet the scalability requirements of these systems. Fortunately, dealing with huge amounts of data is a common issue in the era of Big Data. In recent years, many data management and processing systems have emerged to cope with various challenges (volumes, velocity, variety, load variability, performance, etc.). These systems vary in their models and architectures, with many being specific to use cases such as graph processing and documents management. In this work, we investigate four approaches with their emerging models and systems for scalable data management and processing. We further evaluate their capacities to deal with massive meter data and compare their parallelism and distribution architectures, their data partitioning schemes, and their data models and query languages support as summarized in Table 1. Within the first approach, we investigate MapReduce based processing frameworks [5]. The second approach is based on next-generation of MapReduce computations with acyclic graph execution engines and in-memory processing. The third approach relies on NoSQL datastores with peer-to-peer architectures, which are known for their scale-out properties. The last approach consists of parallel relational database management systems (RDBMS) with massively parallel processing (MPP) models. Lately, parallel RDBMS models have been revisited to provide solutions to many Big Data problems at scale. As illustrative implementations, we rely on the following systems respectively: Hadoop [2], Spark [9], Cassandra [7], and Postgres-XL [4]. To evaluate these approaches for our smart meter data management case, we introduce a set of queries that exhibit the different processing types on smart meter data. These are queries that are mainly of three types: queries based on aggregation functions (such as the sum of measurements), selection and filtering queries based on some criteria (e.g. a measurement threshold), and bill computation queries that are based on some complex rules (e.g. tarif vert of EDF). Bill computation is considered to be one of the most complex type of processing for utilities providers and is a bottleneck for many meter data management systems [1].

We conduct a thorough experimental evaluation on Grid5000 [6], the French cloud and grid testbed. First, following the methodology in [8], we generate data for more than 4 million meters over a one year period with a measurement every hour and a total of more than 35 billion. Thereafter, we conduct several experiments on these data on up to 140 nodes to evaluate and compare data management approaches for meter data processing queries. Our extensive evaluation demonstrates that Postgres-XL outperforms all other systems for aggregation queries over meter data but exhibiting the worst data loading performance. Furthermore, we show that Spark on top of Cassandra provides

BDA 2016, 15 au 18 Novembre, Poitiers, France.

System	Distribution & Parallelism	Partitioning	Data model	SQL Support
Postgres-XL	Master/slave, MPP (massive parallel processing)	Range hashing, round robin	Relational	ANSI SQL:2008
Hadoop	MapReduce, master/slave	file chunks based	files	Hive SQL query engine
Spark and HDFS	MapReduce based, master/slave, acyclic graph execution engine	file chunks based	files and RDDs, DataFrames	SparkSQL query engine
Cassandra	peer-to-peer	consistent hashing (key-based)	column families	CQL (restricted number of operations)
Spark and Cassandra	peer-to-peer, MapReduce, acyclic graph execution engine	consistent hashing (key-based)	column families, RDDs, and DataFrames	SparkSQL query engine

Table 1: Data management and processing systems

an extremely fast response time for bill computation queries with massive data sizes and scales out with the increasing number of nodes. Our study reveals that overall, MapReduce based systems (Hadoop and Spark) tend to generate more data transfer throughout the processing cluster, which in turn delays response times for many queries. In addition, Spark processing performance is bound to memory size.

A main observation from the obtained results is that Postgres–XL is by far the solution that moves data around the least no matter the number of nodes. This is expected because of the MPP model where computation is moved to nodes rather than data, which explains its superior performance with aggregations and queries with many intermediate phases. Furthermore, our results demonstarate that Hadoop-Hive moves less data than Spark (with Spark SQL on top of HDFS and Cassandra). The main cause is that Spark SQL and Spark tend to create more transformations and thus more intermediate phases (where data is shuffled and moved) than Hive on top of Hadoop MapReduce. In fact, Spark with its rich programming API and acyclic graph engine allows users to make many more Map and Reduce steps than Hadoop Map Reduce. In contrast, Spark on top of Cassandra is outperforming all other systems with less than 1s response time for bill queries. This extremely fast response time is due to Cassandra's peer-to-peer architecture with consistent hashing. Therefore, when the meter ids (that are part of the row keys) are specified as query input, accessing data is straightforward based on the hash function. Moreover, with even data distribution there is a high level of parallelism to get data from different nodes simultaneously. The bill queries are the most complex and slow queries and are considered the main bottleneck for MDM systems within energy utilities. However, with the adequate data model and a suitable architecture such as exhibited by Apache Cassandra, these queries can become the fastest in the data management ecosystem.

Our main conclusion is that massively parallel processing (MPP) systems outperform other systems for many queries on meter data because it reduces data transfer. Furthermore, Cassandra with its consistent hashing scheme, is more suitable to many selection and filtering queries and providing extremely fast billing queries. In this context, future designs of meter data management systems (MDM) should focus on the minimization of data transfer. With massive volumes of data, any fraction of data to be moved introduces important delays in data processing. Furthermore, future efforts should focus on hybrid systems and models to achieve efficient data processing at scale given no current model is suitable for all types of processing.

Acknowledgment

This work has been funded by the partnership foundation Grenoble INP through the Industrial Excellence Chair ENEDIS in smart grids.

1. REFERENCES

[1] Using the hp vertica analytics platform to manage massive volumes of smart meter data. Technical report, HP Technical white paper, 2014.

[2] Apache hadoop, March 2016.

[3] Linky, le compteur communicant d'erdf, March 2016.

[4] Postgres-xl: Scalable open source postgresql-based database cluster, March 2016.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation*, OSDI'04, 2004.

[6] Y. Jégou, S. Lantéri, J. Leduc, et al. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. *Intl. Journal of High Performance Comp. Applications*, 2006.

[7] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 2010.

[8] X. Liu, L. Golab, W. Golab, and I. F. Ilyas. Benchmarking smart meter data analytics. In *EDBT: 18th International Conference on Extending Database Technology, Brussels, Belgium, March-23-27, 2015, Online Proceedings*.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, 2010.

Impact de la réplication sur l'équilibrage de charge dans les bases de données distribuées*

Noël Gillet
LaBRI, University of Bordeaux
noel.gillet@labri.fr

Nicolas Hanusse
LaBRI, CNRS
nicolas.hanusse@labri.fr

Frédéric Lalanne
LaBRI, University of Bordeaux
frederic.lalanne@labri.fr

ABSTRACT

Les bases de données distribuées, utilisées massivement dans le contexte actuel du big data, doivent gérer énormément de requêtes sur les données qu'elles stockent. Un des défis majeurs est de minimiser le temps de complétion de ces requêtes, c'est-à-dire le temps maximum nécessaire pour répondre à un ensemble de requêtes. Puisque chaque donnée (ou objet) est répliquée en plusieurs copies, une idée naturelle afin de réduire ce temps de complétion est de répartir les requêtes entre les différents noeuds pouvant les traiter. Cependant, la réplication est essentiellement utilisée dans les systèmes actuels à des fins de tolérances aux pannes et non pas pour améliorer les performances. Dans ce travail, nous étudions l'impact du facteur de réplication f sur le temps de complétion, et ce, même si la distribution des requêtes est laissée à un adversaire. Soit m objets dupliqués f fois, pour R requêtes de lecture ayant le même coût et devant être répartis entre n noeuds, le temps de complétion idéal est R/n. Nous donnons pour tout placement d'objets, une borne inférieure sur le temps de complétion optimal qui est $(R/n)(m/n)^{1/f}$. Nous proposons un algorithme distribué qui garantit un temps de complétion de $O((R/n)\log n)$ pour un placement d'objets aléatoire et pour toute distribution de requêtes en utilisant $\log n$ copies. Des expériences réalisées avec la base de donnée distribuée Apache Cassandra confirment l'intérêt de notre solution.

Keywords

Load Balancing, replication, distributed databases

1. INTRODUCTION

The more recent distributed databases like NoSQL databases are dedicated to massive data storage and have to answer to numerous queries. In those systems, data are replicated in order to improve the availability in case of crashes, failures, network maintenance. Moreover, one can hope that making several copies might improve the throughput or the completion time of the queries. In the other hand, handling too many copies is consuming in terms of memory but also costly in consistency mechanisms since the read and write operations can be done independently on several copies. In this paper, we only consider that the queries are *read-only*. Thus, we do not focus on the consistency but we aim at showing how to set up a good replication mechanism in order to get the smallest completion time for a sequence of *read queries*.

*This work has been carried out as part of the "SpeedData" (PIAO17298-398711) project supported by the French "Investissement d'Avenir" Program (Big Data - Cloud Computing topic) and Aquitaine Region.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. THE ALLOCATION PROBLEM

In the rest of the paper, we use the term of *object* as an abstract representation of the data stored by the system. We may consider for example that an object is simply a part of an horizontal partition of a table in a database. We consider a system of n nodes storing m *objects* in a distributed manner. Each object is replicated f times and these copies are spread over the nodes according to a placement strategy P. The system has to run a workload Q, that is a *sequence* of R read queries of *identical cost*. Each query for an object can only be handled by a subset of nodes called the *candidates nodes* which store a copy of the queried objetc. An allocation scheme consists to associate one of the candidates for each query. Let define the load $\ell_{P,A}(u)$ of a node u as the number of queries allocated to u with allocation A and object placement P. In the *allocation problem*, we aim at finding the placement P and allocation A that minimizes the *completion time* $T_{P,A}(Q)$ *i.e.* the maximum load of a node. Since the cost of queries are identical, we ideally want to get R/n queries per node. However, even for the specific case $R = m = n$ and $f = 2$, the completion time can be far from 1. For instance, assume that the query is *immediately assigned* to *the least loaded node*. It leads to a completion time of $\Theta(\frac{\log\log n}{\log f})$ for random queries ([2]) and is $\Theta(\log n)$ for some workload ([6]) for any immediate deterministic decision scheme. Another strategy is based on the *delayed assignment*: every query is sent to the f copies and a local decision is eventually done to treat the query. If we consider that all the queries are received at the same time, we fall in a traditionnal *off-line* settings where we can model our problem by an orientation of the queries hypergraph H in which vertices represent the nodes of the system and hyperedges represent the queries. We want to find an orientation that minimize the maximum indegree (one marked

extremity per hyperedge) of any vertex in H. For $f = 2$, the author of [5] shows that optimal maximum indegree $D(H)$ is equal to $\lceil \Delta \rceil$, where Δ is the maximum density over all subgraphs of H. The first polynomial optimal algorithm is given in [4] in a centralized system. An $(1+\epsilon)$-approximate solution is given in [3] and can be easily adapted in a synchronous distributed system in $O(\log n)$ rounds. However, even if this case the optimal allocation can be far from the ideal allocation. For instance, an *adversary* who is aware of the placement strategy can focus all the queries on objects stored by f nodes, leading to a completion time of R/f. Since replicate the data into each node is not an option in a big data environment, we need to design more sophisticated solutions to handle *any distribution*.

3. THEORETICAL CONTRIBUTIONS

In all our theoretical results, we consider an off-line settings where all the queries have been already received by the nodes in their *waiting queues*. We consider that a node is able to send a message to any other node of the system using routing mechanisms. Assume $m = n^\alpha$ with $\alpha \geq 1$ being a constant. We are interested in a workload Q of size $R \geq n$. We consider that the set of queries can be chosen by an *adversary*, knowing the copies placement, and are either *unitary* or *unpopular*. A set of queries is unitary (resp. unpopular) if there is at most 1 (resp. R/n) query per object. We propose to use a random placement for the objects and we denote by $T_{OPT}(Q)$ the optimal completion time for Q with this placement. Our contribution can be summarized as follows:

We give some theoretical bounds on $T_{OPT}(Q)$. We show that $T_{OPT}(Q) = O(R/n \cdot (m/n)^{1/f})$ for non popular queries. Moreover, we show that the optimal completion time $T^*(Q)$ over all possible placements is $\Omega(R/n \cdot (m/n)^{1/f})$ for a given replication factor f implying that
$T_{OPT}(Q) = \Theta(R/n \cdot (m/n)^{1/f})$.

We propose a distributed algorithm that combine multiple solution. First we use an allocation algorithm called `Average` (adapted from [3]) for unpopular queries. Briefly, a leader estimates the average load ℓ_{avr} of the system and broadcast this value. When a node u receives ℓ_{avr} and $\ell(u) \leq (1+\epsilon) \cdot \ell_{avr}$, every query of the queue is allocated to u. A deletion message for each allocated query is sent to the other nodes. Second we adapt the number of copies of an object based on its *popularity*. To finish, we use a simple round robin algorithm for the popular queries using the additional copies created by our copies management algorithm. We show that by choosing an initial replication factor $f = (\alpha - 1) \log n$ and by adding a linear number of additionnal copies, our algorithm guaranties a completion time of $O(R/n \log n)$, whatever the *queries distribution*. Our theoretical results on the optimal completion time comes mainly from the study of the underlying queries hypergraphs.

4. EXPERIMENTS

We have modified the distributed database system Apache Cassandra in order to compare `Average` in a *real asynchronous* distributed database whenever the objects are large. Queries are now inserted in sequence. Cassandra allocation is an immediate decision scheme based on the least response time. We also compare these algorithm with a simple allocation strategy called `Naive` for which a node treat the first query of its waiting queue and sends a deletion message to the others candidates. In the first experiment, queries are randomly chosen among the set of f nodes having the largest number of objects in common. The completion time is always improved as we increase the number of copies in this adversarial setting. In the second experiment, we take the real dataset Calgary and corresponding workload (a power-law like distribution [1]) keeping the 256 most popular objects but increasing there sizes to 128 MBytes. `Naive` and `Average` perform better than Cassandra (the least response time is less stable than the average load). If f is too large, a same query can be answered several times. In practice, `Naive` is often a bit more efficient than `Average` but it has no strong theoretical guarantee.

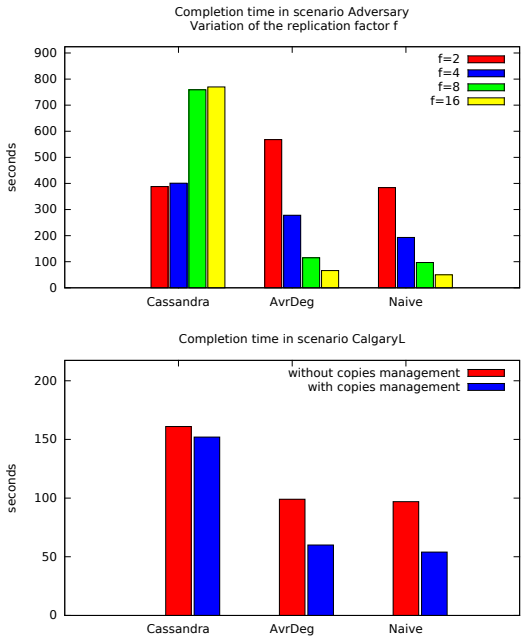


Figure 1: Experiments for $n = 32$ nodes, 256 objects of 128 MBytes and $R = 2048$ queries.

5. REFERENCES

[1] Traces in the internet traffic archive. `http://ita.ee.lbl.gov/html/traces.html`.

[2] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[3] M. Farach-Colton and M. Tsai. Computing the degeneracy of large graphs. In *LATIN*, volume 8392 of *Lecture Notes in Computer Science*, pages 250–260. Springer, 2014.

[4] H. N. Gabow and H. H. Westermann. Forests, frames and games: Algorithms for matroid sums and applications. In *STOC*, pages 407–421. ACM, 1988.

[5] A. Gyárfás. How to orient the edges of a graph. In *Combinatorics, I, Colloq. Math. Soc. János Bolyai*, volume 18, pages 353–364, 1978.

[6] G. Tang. *Fully Dynamic Graph Orientation*. Master thesis, 2014.

Impact of Time on Detecting Spammers in Twitter

Mahdi Washha
IRIT Laboratory
University of Toulouse
Toulouse, France
mahdi.washha@irit.fr

Aziz Qaroush
Birzeit University
Birzeit, Palestine
aqaroush@birzeit.edu

Florence Sedes
IRIT Laboratory
University of Toulouse
Toulouse, France
florence.Sedes@irit.fr

Abstract. Twitter is one of the most popular microblogging social systems, which provides a set of distinctive posting services operating in real time manner. The exibility in using these services have attracted unethical individuals, so-called "spammers", aiming at spreading malicious, phishing, and misleading information over the network, resulting non-ignorable problems related to real-time search and user's privacy. Although of Twitter's community attempts in breaking up the spam phenomenon, researchers have dived far infighting spammers through automating the detection process using the features concept combined with machine learning methods. However, the existing features are not effective enough to adapt the spammers' tactics due to ease of manipulation; rather than the graph features are not suitable for real-time filtering.

In this paper, we introduce the design of novel features suited for real-time filtering, distributed between robust statistical features considering explicitly the time of posting tweets and creation date of user's account as an only unmodifiable attribute overtime, and behavioural features that catch any potential posting behaviour similarity between different instances (e.g. hashtags) available in the user's tweets. The experimental results show that our new features are able to classify correctly the majority of spammers with an accuracy higher than 93% when using Random Forest learning algorithm, outperforming the accuracy of the state of features by about 6%.

Query-Oriented Summarization of RDF Graphs

Résumés orientés requêtes de graphes RDF

Šejla Čebirić
INRIA, France
sejla.cebiric@inria.fr

François Goasdoué
U. Rennes 1 & INRIA, France
fg@irisa.fr

Ioana Manolescu
INRIA, France
ioana.manolescu@inria.fr

RÉSUMÉ

RDF est le modèle de données du W3C, fondé sur les graphes, pour les applications du Web Sémantique. Nous étudions le problème du résumé de graphes RDF : étant donné un graphe RDF G, trouver un graphe RDF H_G résumant G aussi précisément que possible, tout en étant si possible plusieurs ordres de magnitude plus petit que le graphe original. Nos résumés sont destinés à aider l'exploration de graphes RDF, ainsi que la formulation et l'optimisation de requêtes.

Nous proposons quatre sortes de résumé de graphe RDF, obtenus comme des quotients de graphes dont les relations d'équivalence reflètent la similarité entre noeuds vis-à-vis de leurs types ou connexions. Nous étudions aussi s'ils possèdent les propriétés formelles de représentativité (H_G devrait représenter autant d'information de G que possible) et de précision (H_G devrait éviter, autant que possible, de refléter des informations qui ne sont pas dans G). Enfin, nous présentons des expériences faites sur plusieurs graphes RDF synthétiques ou issus d'applications réelles.

ABSTRACT

The Resource Description Framework (RDF) is the W3C's graph data model for Semantic Web applications. We study the problem of RDF graph summarization: given an input RDF graph G, find an RDF graph H_G which summarizes G as accurately as possible, while being possibly orders of magnitude smaller than the original graph. Summaries are aimed as a help for RDF graph exploration, as well as query formulation and optimization.

We devise four kinds of RDF graph summaries obtained as quotient graphs, with equivalence relations reflecting the similarity between nodes w.r.t. their types or connections. We also study whether they enjoy the formal properties of representativeness (H_G should represent as much information about G as possible) and accuracy (H_G should avoid, to the possible extent, reflecting information that is not in G). Finally, we report the experiments we made on several synthetic and real-life RDF graphs.

1. OUTLINE

The Resource Description Framework (RDF) is a graph-based data model promoted by the W3C as the standard for Semantic Web applications. Its associated query language is SPARQL. RDF graphs are often *large* and *varied*, produced in a variety of contexts, e.g., scientific applications, social or online media, government data etc. They are *heterogeneous*, i.e., resources described in an RDF graph may have very different sets of properties. An RDF resource may have: no types, one or several types (which may or may not be related to each other). *RDF Schema* (RDFS) information may optionally be attached to an RDF graph, to enhance the description of its resources. Such statements also entail that in an RDF graph, some data is **implicit**. According to the W3C RDF and SPARQL specification, **the semantics of an RDF graph comprises both its explicit and implicit data**; in particular, SPARQL query answers must be computed *reflecting both the explicit and implicit data*. These features make RDF graphs complex, both structurally and conceptually. It is intrinsically hard to get familiar with a new RDF dataset, especially if an RDF schema is sparse or not available at all.

In this work, we study the problem of *RDF summarization*, that is: given an input RDF graph G, find an RDF graph H_G which *summarizes* G *as accurately as possible, while being possibly orders of magnitude smaller* than the original graph. Such a summary can be used in a variety of contexts: to help an RDF application designer get acquainted with a new dataset, as a first-level user interface, or as a support for query optimization as typically used in semi-structured graph data management [2] etc. Our approach is *query-oriented*, i.e., a summary should enable static analysis and help formulating and optimizing queries; for instance, querying a summary of a graph should reflect whether the query has some answers against this graph, or finding a simpler way to formulate the query etc. Our approach is the first semi-structured data summarization approach focused on *partially explicit, partially implicit* RDF graphs.

In the sequel, Section 2 recalls the basics of the RDF data, schema and queries, and sets the requirements for our query-oriented RDF summaries. We then introduce a general concept of RDF summary in Section 3, then decline it into several distinct brands, in Section 4 and 5. We formally establish the properties of these summaries with respect to the representation of the graph they derive from, and analyze the complexity of building them; in all cases, this complexity is in the low-degree polynomials in the number of graph edges. We have fully implemented these summarization procedures; we describe our implementation and report experimental results, before discussing related works. *For space reasons, proofs for this paper's claims are delegated to [1].* Finally, as a picture is worth a thousand words, graphical representations of sample summaries can be found at: `https://team.inria.fr/cedar/projects/rdfsummary`.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. REFERENCES

[1] Š. Čebirić, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. http://pages.saclay.inria.fr/ioana.manolescu/RDFSummaries-2016.pdf, 2016.

[2] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.

An updated dashboard of Complete Search FSM Implementations in Centralized Graph Transaction Databases*

Rihab Ayed*
Université de Lyon, CNRS,
Université Lyon 1, LIRIS
UMR5205, 69622
Villeurbanne, France

Mohand-Saïd Hacid
Université de Lyon, CNRS,
Université Lyon 1, LIRIS
UMR5205, 69622
Villeurbanne, France

Rafiqul Haque
Université de Lyon, CNRS,
Université Lyon 1, LIRIS
UMR5205, 69622
Villeurbanne, France

Abderrazek Jemai
Université de Tunis El Manar
Faculté des Sciences de
Tunis, Laboratoire LIP2
Université de Carthage, INSAT
1080, Tunis, Tunisie

ABSTRACT

Graph mining is one of the most important approaches in data mining that transforms graph data into knowledge. Frequent Subgraph Mining (FSM) is a subcategory of Graph Mining. The objective of traditional FSM is to extract subgraphs, in a given dataset, whose occurrence counts (*aka* frequency) are above a specified threshold (Minimum Support Threshold). The extracted subgraphs, called Frequent Subgraphs, are useful for different applications (*e.g.,* graph indexing).

In CAIR project, we are investigating efficient and effective approaches for aggregated search over distributed repositories. One of the building blocks of our approach is graph indexing. The index data can be built by resorting to an FSM algorithm.

Given that there are many algorithms proposed for FSM in the literature, a first task was to classify the algorithms according to some relevant features (*e.g.,* Complete/Incomplete search, input graphs type, database setting, *etc.*).

Further, the existing studies devoted to benchmarking FSM algorithms are not satisfactory for our requirements due to a few reported ambiguities related to performance and lack of information about the specific cases of performances (*e.g.,* an algorithm X performs better than an algorithm Y for dense datasets and medium support threshold).

In this paper, we identified 32 algorithms from the literature which perform a complete search FSM for centralized graph transaction databases. We selected 6 algorithms with their respective 13 implementations using some useful filtering criteria. We conducted an experimental study of the selected implementations using datasets from the state of the art. We analyzed the behavior of the implementations according to the three commonly used parameters: (i) execution time, (ii) memory consumption and (iii) the number of frequent subgraphs. We varied two input parameters: datasets and minimum support threshold. The results of this study led to the selection of four implementations, namely Gaston Original, gSpan ParMol, gSpan Original and FSG Original for their efficiency regarding memory consumption, execution time and ability to finish the execution sucessfully with relatively large datasets or low support threshold values. A more general filtering returns two implementations out of 13 according to two criteria (*i.e.,* time or memory) : (i) Gaston Original or gSpan Original for critical memory applications, and (ii) Gaston Original for critical time applications. We noticed that by varying the size of the datasets and the minimum support as well as other environment variables (*e.g.,* labeling strategy), some specific implementations display specific performances. Hence, the implementations should be used in the cases for which they perform better.

*This research is performed within the scope of the CAIR (Contextual and Aggregated Information Retrieval) project that is funded by ANR (Agence nationale de la recherche) - http://www.irit.fr/CAIR

*Email address : rihab.ayed@liris.cnrs.fr

BDA 2016, 15 au 18 Novembre, Poitiers, France.

Articles de doctorant-e-s

Architecture, concepts et services d'un système d'indexation de données distribuées pour l'observation à large échelle en écologie marine

Dans le cadre du consortium IndexMeed (Indexing for Mining Ecological and Environmental Data)

R. David*, J.-P. Féral#, T. Tatoni$
Institut Méditerranéen de Biodiversité et d'Ecologie marine et continentale (IMBE)
Aix Marseille Université, CNRS, IRD, Université d'Avignon
Station Marine d'Endoume, 13007 Marseille, France. *romain.david@imbe.fr*

RESUME

Cet article présente la partie *architecture et services* de la thèse de R. DAVID, intitulé *Méthodes et outils pour le suivi à large échelle des habitats coralligène en Méditerranée : du système d'observation au système d'information*. Il présente les processus d'indexation et de qualification de données hétérogènes et distantes, basés sur des services WEB. Ceux-ci permettent de construire des graphes à partir de données et de thesaurus concernant les habitats coralligènes en Méditerranée orientale et occidentale et de les exploiter avec des objectifs d'indication et d'aide à la décision.

* étudiant en thèse, # encadrant principal, $ responsable du projet

Mots clefs

Indexation, qualification de données, traçabilité, système d'information décentralisé, fouille de donnée, écologie, graphes.

1. CONTEXTE

La révolution du *Big Data* en écologie tarde, alors qu'elle est considérée par la plupart des disciplines scientifiques et des industries produisant et utilisant de l'information comme la plus prometteuse des pistes de progrès et de découvertes. Les données utilisées par les scientifiques dans le domaine de l'écologie continuent de se diversifier et ne sont plus principalement produites par les institutions scientifiques, mais par des réseaux extérieurs d'acteurs et de compétences. De nouvelles disciplines (protéomique, métabolimique, métagénomique) complètent les prismes d'observation déjà multiples de la biodiversité. L'accessibilité de ces données est variable, et les processus de qualification qui évaluent leur utilisabilité et leur efficacité sont encore rares, a fortiori dans le domaine marin [1]. *A contrario*, la production de données scientifiques est de plus en plus financée sous condition de leur mise à disposition (depuis plusieurs décennies pour les données de biologie moléculaire, mais encore de façon balbutiante pour les données écologiques et environne-mentales), sans que des outils appropriés à une analyse intégrative soient proposés.

2. PROBLEMATIQUES

Des formats et des protocoles standards permettent d'interconnecter les bases de données dans le domaine de l'environnement. Des approches intégratives notamment en « écologie statistique » permettent des analyses à l'échelle globale [4]. Les approches sémantiques contribuent largement à améliorer leur interopérabilité. Il reste que les objectifs scientifiques spécifiques, les logiques d'organisation de projets et de collecte d'informations conduisent à une distribution décentralisée des données qui peut freiner la recherche dans le domaine de l'écologie. Dans ce contexte, comment i) analyser des données hété-rogènes situées dans des bases de données distantes ? ii) inter-calibrer des systèmes d'observations différents, créer des correspondances et intégrer certaines approxima-tions dans les correspondances entre systèmes descriptifs différents ? iii) mettre en évidence des relations entre des données d'observation les mettre en relation avec des patrons contextuels ? iv) favoriser l'ouverture et le partage et la valorisation des données ?

Les verrous scientifiques identifiés dans le cadre de ces travaux de thèse concernent notamment i) l'augmentation des fréquences et de la densité d'acquisition des observa-tions (méthodes de reconnaissance automatique, outils d'ac-quisition moins onéreux), ii) la diversification des objets et des descripteurs d'objet intégrés dans les graphes, iii) la normalisation des descripteurs de la donnée et les méthodes permettant d'intégrer les différents niveaux de qualité des données. De nouvelles approches, basées sur une architec-ture répartie d'informations normalisées, mettant en rapport des données quantitative et qualitatives, rendent possible l'investigation de questions de recherche complexes. Le pro-gramme européen CIGESMED (France, Grèce et Turquie - <www.cigesmed.eu>) et la récente structuration de l'obser-vation des habitats méditerranéens dénommés *coralligène* sert de cadre à cette étude.

3. DEVELOPPEMENTS

Un outil de représentation sous forme de graphes des données de différents champs disciplinaires (écologie, sociologie, économie) a été développé en vue d'élaborer des méthodes de création de scénarios par approches successives (coévolution de facteurs), basée sur des concepts actuel-lement décrits par les approches globales en écologie. L'objet de ce prototype est de construire des graphes paramétrables avec des données hétérogènes (de la molécule à l'écosystème, en passant par les traits de vie, jusqu'aux paysages et aux interactions homme-milieux) concernant l'écologie de cet habitat et d'analyser les données grâce à des algorithmes utilisés dans d'autres disciplines. Il est alimenté par des systèmes d'information distants (10 laboratoires), dont chaque enregistrement est indexé par le serveur, ainsi que les qualifications nécessaires pour créer des relations entre chaque type d'objets indexés. Ces qualifications sont décrites sous forme de micro-thésaurus.

4. RESULTATS PRELIMINAIRES

Les graphes sont construits *via* le prototype de visua-lisation de graphe (serveur WEB) à partir d'informations agrégées grâce à des points nodaux d'indexation et de qualification des données de contexte sur l'environnement littoral et marin méditerranéen [3], dans différentes disci-plines et à l'échelle méditerranéenne. Ces graphes sont paramétrables pour fouiller et visualiser ces données pluri-disciplinaires en mettant sur le même plan des données de

types écologiques, moléculaires et fonctionnelles (relations trophiques, traits fonctionnels), et le seront pour des aspects socio-écologiques, économiques. Les questionnements scientifiques possibles concernent l'écologie des systèmes observés (bon état écologique, correspondance de patrons de contextes et de données concernant les abondances relatives d'espèces comme dans la Figure 1), ou des systèmes d'observations (détection des biais dans la forma-tion des observateurs, expertise partielle dans les jeux de données, définition de la puissance de l'échantillonnage nécessaire, gestion des coûts associés).

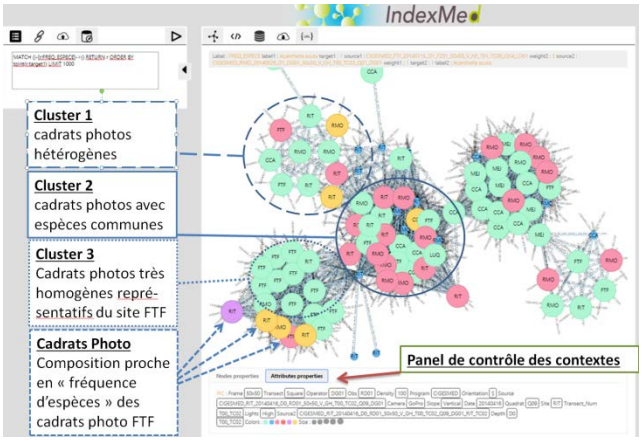


Figure 1 : Prototype de visualisation des données d'IndexMed utilisant dans cet exemple des fréquences d'espèces par photo. Les nœuds représentent les photos, les liens correspondent aux fréquences d'espèces, la grosseur des nœuds à différents paramètres environnementaux. Les photos proviennent de différents sites et les données indexées sont accessibles sur les systèmes d'information des partenaires, interrogées à distance par le prototype (JSON ou XML).

Les points nodaux d'indexation créés par le prototype sont *clonables* à volonté avec des règles d'enrichissement et de partage correspondant aux licences *creative common* du type «partage dans les mêmes conditions», autorisant les autres à reproduire, diffuser et modifier l'index, à condition qu'ils publient toute adaptation de l'index sous les mêmes conditions (open-source, open data). Ces règles devront favoriser l'alimentation de standards améliorant l'interopérabilité des données et favorisera la participation de nouveaux laboratoires en tenant compte de leurs capacités.

5. CONCLUSION ET PERSPECTIVES

L'architecture définie dans le cadre de ce travail permet de répliquer des « points nodaux d'indexation » dans différents domaines thématiques de l'écologie marine, permettant une qualification itérative des données compatible avec les systèmes normatifs internationaux (TDWG Taxonomic Data-base Working Group, recommandation INSPIRE), mais intégrant de manière générique tout nouveau système de qualification dit *métier* sous la forme de micro-thésaurus. La polysémie générée par l'adjonction de nouvelles disciplines doit être gérée par consensus sous la forme d'une interface (en cours de déploiement). De nouveaux champs disciplinaires sont intéressés au projet (EPD (European Pollen Database), ArkéoGIS (GIS en archéologie) ou GBIF (occurrences d'espèces terrestres) et renforcent l'aspect multidisciplinaire du projet.

Cette thèse a permis la création d'un consortium qui développe la culture des bases de données et leur utilisation efficace dans le milieu de la recherche en écologie. Sous l'impulsion du doctorant, il s'est étendu à plusieurs UMR de disciplines différentes. Lors du dernier séminaire (Journées du GRAAL - GRAphs and datamIning for environmentaL research), dans l'optique d'une internatio-nalisation, la communauté IndexMed (Inter-opérabilité des bases de données en écologie) forte de plusieurs dizaines de nouveaux membres est devenu IndexMeed (Indexing for Mining Ecological and Environmental Data). La prochaine étape sera la co-élaboration d'un projet de recherche (BiodivERsA, SeasEra, H2020).

6. REMERCIEMENTS

Ce travail est réalisé à l'IMBE avec le soutien de France Grille et de la FRB (Fondation pour la Recherche sur la Biodiversité). La construction du premier prototype du consortium IndexMed a été est financé par le *défi CNRS* "VIGI- GEEK[1]" et le PEPS Blanc CNRS INEE avec le projet "Charliee[2]". Les données utilisées pour cet article ont été obtenues par le biais du projet CIGESMED (www.cigesmed.eu) dont nous remercions chaque partenaire. L'architecture a été débattu en groupe de travail lors du séminaire «design your infrastructure» organisé par European Grid Infrastruc-ture http://www.egi.eu/ à Amsterdam (avril 2016) https://indico.egi.eu/indico/event/3025/. Et a été présentée lors du congrès IEMSs à Toulouse en Juillet 2016 [2]. http://www.iemss.org/sites/iemss2016. Nous remercions tous les membres actifs du consortium IndexMed pour leurs contributions et les GDR MaDICS et EcoStat pour leurs labellisations et leurs soutiens.

7. REFERENCES

[1] J. Barde, Mutualisation de données et de connaissances pour la Gestion Intégrée des Zones Côtières. Application au projet SYSCOLAG. Mathématiques [math]. U. Montpellier II - *Sciences et Techniques du Languedoc*, Nov 2006

[2] R. David, J.P. Féral, A-S. Archambeau, N. Bailly, C. Blanpain, V. Breton, A. De Jode, A. Delavaud, A. Dias, S. Gachet, D. Guillemain, J. Lecubin, G. Romier, C. Surace, L. Thierry de Ville d'Avray, C. Arvanitidis, A. Chenuil, M.E. Çinar, D. Koutsoubas, S. Sartoretto, T. Tatoni ; IndexMed projects : new tools using the CIGESMED DataBase on Coralligenous for indexing, visualizing and data mining based on graphs. In : Sauvage S, Sánchez-Pérez J-M., Rizzoli, AE (Eds.), *Proceedings of the 8th International Congress on Environmental Modelling and Software, Environmental modelling and software for supporting a sustainable future*, Vol. 3, pp.656-665, Toulouse, France. July 2016. ISBN : 978-88-9035-745-9.

[3] R. David, J.P. Féral, C. Blanpain, C. Diaconu, A Dias, S. Gachet, K. Gibert, J. Lecubin, C Surace, A first prototype for indexing, visualizing and mining heterogeneous data in Mediterranean ecology within the IndexMed consortium interdisciplinary framework. In: *SITIS 2015, 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Bangkok, Thailand, pp. 232-239, nov. 2015 doi: 10.1109/SITIS.2015.119.

[4] O. Gimenez, S.T. Buckland, B.J.T. Morgan, N. Bez, S. Bertrand, R. Choquet, S. Dray, M.P. Etienne, R. Fewster, F. Gosselin, B. Mérigot, P. Monestiez, J. Morales, F. Mortier, F. Munoz, O. Ovaskainen, S. Pavoin, R. Pradel, F.M. Schurr, L. Thomas, W. Thuiller, V. Trenkel, P. de Valpine, E. Rexstad. Statistical ecology comes of age. (2014) *Biology Letters* 10: 20140698.

[1] VIGI-GEEK : VIsualisation of Graph In transdisciplinary Global Ecology, Economy and Sociology data-Kernel

[2] CHARLIEE : CHAnger de Regard En Liant dans Indexmed l'Environnement et les Etoiles

Privacy Preserving Query Processing in the Cloud

Sakina Mahboubi*
Ph.D. thesis start date: Oct. 2015
Supervisers: Reza Akbarinia & Patrick Valduriez
Zenith team, Inria and LIRMM, University of Montpellier, France
sakina.mahboubi@inria.fr

1. INTRODUCTION

Nowadays, cloud data outsourcing provides users and companies with powerful capabilities to store and process their data in third-party data centers. However, when a user stores her sensitive data in a public cloud, they becomes vulnerable to several attacks, e.g., from the employees of the cloud provider.

One solution for protecting the user data against attacks is to encrypt the data before sending them to the cloud servers. Then, the challenge is to answer user queries over encrypted data. A naive solution for answering queries is to retrieve the encrypted database from the cloud to the client, decrypt it, and then evaluate the query over *plaintext (non encrypted)* data. This solution is not practical, in particular for large databases.

In this PhD thesis, we are interested in processing top-k queries on encrypted data. These queries have attracted much attention in several areas of information technology such as sensor networks [15], stream management systems [13, 12] and spatial data analysis [1, 3]. A top-k query allows the user to specify a number k, and the system returns the k tuples which are most relevant to the query. The relevance degree of tuples to the query is determined by a *scoring function*.

There have been many different approaches proposed for processing top-k queries over plaintext data. Two of the best known approaches are FA [5] and TA [7] that work on sorted lists of attribute values. These approaches, particularly TA, can find efficiently the top-k results because of smart strategies for deciding when to stop reading the database. However, TA, FA and all other efficient top-k approaches developed so far assume that the data are in plaintext, and there is no efficient solution capable of evaluating efficiently top-k queries over encrypted databases.

When we think about top-k query processing on encrypted data, the first idea that comes to mind is the utilization of a fully homomorphic encryption cryptosystem, e.g. [8], which allows to do arithmetic operations over encrypted data. this type of encryption allows to compute the overall score of data items over encrypted data. However, fully homomorphic encryption methods are very expensive in terms of encryption and decryption time. In addition, they do not allow to compare the encrypted data, and to find the top-k results.

We proposed efficient approaches, called EncFA and BuckTop, for processing top-k queries over encrypted data. We evaluated their response time over encrypted data with that of the TA algorithm over original (plaintext) data. Our results show that the response time of BuckTop over encrypted data is close to TA over plaintext data, and even better over some large databases.

In the rest of this paper, we fist define the problem we address. Then, we briefly introduce our proposed approach, and then we discuss the related work.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. PROBLEM DEFINITION

The problem which we address is top-k query processing over encrypted data. Let D be a database, and $e(D)$ be its encrypted version such that each data $c \in e(D)$ is the ciphertext of a data $d \in D$, i.e. $c = Enc(d)$ where $Enc()$ is an encryption function. The database $e(D)$ is stored in a remote server.

Given a number k and a scoring function f, our goal is to develop an algorithm A, such that when A is executed over the database $e(D)$, its output contains the ciphertexts of the top-k results, i.e. those that can be found by executing a correct top-k algorithm over the database D.

3. APPROACH

The architecture of our system for query processing over encrypted data is composed of two parts: *Trusted client* and *Service provider*.

Trusted client is responsible of user data encryption and decryption, user access control and secure key management. When a query is issued by a user, the trusted client checks the access rights of the user. If the user does not have the required rights to see the query results, then her demand is rejected. Otherwise, the issued query is transformed to a query that can be executed over the encrypted data and is sent to the service provider. the Trusted client decrypts the received results (calculated by the service provider) and returns to the user the k data item which are the response of the query launched.

Service provider stores the encrypted data, executes on them the algorithms provided by the trusted client, and returns the results to it.

We propose two approaches for processing top-k queries over encrypted data. The first approach, called *EncFA*, uses

a probabilistic (semantic) encryption scheme to encrypt the attribute values of the database items. It also uses a deterministic scheme for encrypting the IDs of data items. Then, it sends the encrypted database to the service provider. the data items after encryption must have the same order that they had before being encrypted.

The second approach, called *BuckTop*, is more efficient than EncFA. It uses a bucketization technique to partition the data items in each list into a set of buckets. also it uses tow types of encryption schema to encrypt the database; one is deterministic used to encrypt data item IDs and the other is probabilistic used to enrypt the attribute values of the data items. BuckTop includes a top-k query processing algorithm that works on the encrypted data of the buckets, and returns a set containing the top-k results. It also includes an efficient filtering algorithm that filters the false positives as much as possible in the server. We prove theoretically the correctness of the BuckTop approach.

4. RELATED WORK

A first important paper in top-k query processing is [5], which models the general problem of answering top-k queries using databases organized into lists of data items sorted by their local scores. One of the most efficient algorithms over sorted lists is the TA algorithm, which was proposed by several groups [6, 9]. However, all these algorithms assume that the data scores are available as plaintext, and not encrypted.

There have been also some proposed solutions for secure kNN query processing, e.g. [4, 2, 16]. The problem is to find k points in the space that are the nearest to a given point. This problem should not be confused with the top-k problem in which the given scoring function plays an important role, such that on the same database and with the same k, if the user changes the scoring function, then the output may change. Thus, the proposed solutions proposed for kNN cannot deal with the top-k problem.

The bucketization technique has been used in the literature for answering range queries over encrypted data, e.g. [10, 11] where Hore et al. use this technique, and propose optimal solutions for distributing the encrypted data of a database to the buckets in order to guarantee a good performance by reducing the number of false positives while preserving a high security level. The techniques developed in [10, 11] can be used in our system for an optimal distribution of the encrypted data in the buckets.

The only paper which we found about top-k query processing over encrypted data is [14] published in arXiv.org. The proposed architecture assumes the existence of two non-colluding servers s_1 and s_2 in two different clouds. One of the servers, say s_2, has the decryption keys, and the other one, say s_1, stores the data. Top-k query processing proceeds by using the TA algorithm and accessing the encrypted data in s_1, such that after reading each data in s_1, its encrypted local scores are sent to the server s_2 (using a special protocol) where they are decrypted and compared with the TA threshold. Our assumptions about the cloud are different. In our solution, we do not need to trust on any remote server, and during the top-k query processing, we do not decrypt the encrypted data in the cloud servers. In addition, the solution in [14] needs a lot of communications between remote servers (i.e., at least two messages after each sorted access). This solution that is not efficient and incurs a high latency in the query processing time.

to the best of our knowledge, in the literature there is no efficient solution for processing top-k queries over encrypted data. In this work, we propose such a solution.

5. REFERENCES

[1] L. Chen, J. Xu, X. Lin, C. S. Jensen, and H. Hu. Answering why-not spatial keyword top-k queries via keyword adaption. In *ICDE Conf.*, pages 697–708, 2016.

[2] S. Choi, G. Ghinita, H. Lim, and E. Bertino. Secure knn query processing in untrusted cloud environments. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 26(11):2818–2831, 2014.

[3] F. M. Choudhury, J. S. Culpepper, and T. K. Sellis. Batch processing of top-k spatial-textual queries. In *Second International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, pages 7–12, 2015.

[4] Y. Elmehdwi, B. K. Samanthula, and W. Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *ICDE Conf.*, pages 664–675, 2014.

[5] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.

[6] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS Conf.*, 2001.

[7] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[8] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009.

[9] U. Güntzer, W. Balke, and W. Kießling. Towards efficient multi-feature queries in heterogeneous environments. In *2001 International Symposium on Information Technology (ITCC)*, pages 622–628, 2001.

[10] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *VLDB J.*, 21(3):333–358, 2012.

[11] C. Li, M. Hay, G. Miklau, and Y. Wang. A data- and workload-aware query answering algorithm for range queries under differential privacy. *PVLDB*, 7(5):341–352, 2014.

[12] Z. Shen, M. A. Cheema, X. Lin, W. Zhang, and H. Wang. Efficiently monitoring top-k pairs over sliding windows. In *ICDE Conf.*, pages 798–809, 2012.

[13] X. Wang, Y. Zhang, W. Zhang, X. Lin, and Z. Huang. SKYPE: top-k spatial-keyword publish/subscribe over sliding window. *PVLDB*, 9(7):588–599, 2016.

[14] H. Z. Xianrui Meng and G. Kollios. Declarative cleaning of inconsistencies in information extraction. *arXiv:1510.05175v2*, 2016.

[15] H. Yang, C. Chung, and M. H. Kim. An efficient top-k query processing framework in mobile sensor networks. *Data Knowl. Eng.*, 102:78–95, 2016.

[16] B. Yao, F. Li, and X. Xiao. Secure nearest neighbor revisited. In *ICDE Conf.*, pages 733–744, 2013.

A Database Model for Time Series

From a traditional Data Warehouse to a Mathematical Models Warehouse

Cyrille Ponchateau
ISAE/ENSMA
ponchateau@ensma.fr

Ladjel Bellatreche
ISAE-ENSMA
bellatreche@ensma.fr

Carlos Ordonez
Huston University
USA
ordonez@cs.uh.edu

Mickael Baron
ISAE-ENSMA
mickael.baron@ensma.fr

ABSTRACT

Time series are series of values measured over time. They find their interest in numerous fields from finance (weekly sales total, stock price movements...) to medicine (medical imaging, electrocardiogram or medical surveillance in general) and physics (particle tracking). The automatic control team of our laboratory has a great experience in modeling the behavior of systems. The built models (differential equations) carry a physical knowledge of the studied systems. We are willing to take advantage of this expertise and knowledge, by storing and managing those models within a *models warehouse*, which is an adaptation of the data warehouse technology to models. In this paper, we first propose to use differential equations as a new time series representation and propose to adapt the data warehouse structure and its ETL process to the needs of equations storage. The new structure is called a *models warehouse*.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

Models Warehouse; Time Series; Differential equations storage; ETL; models storage

1. INTRODUCTION

Time series are series of values measured over time [1, 2]. They find their interest in numerous fields from finance (weekly sales total [2]...) to medicine (electrocardiogram [2] or medical surveillance in general [1]). Their diversity of use makes of great interest the research in the time series data mining field and the time series data miners task is really challenging. Indeed, time series can contain a great number of elements (high-dimensional) [2,3] and their values (observations) can be degraded by noise or outliers [1]. Moreover, the time series values are continuous, meaning that time series data miners have to compare time series and look for matching while coping with approximated values.

In the particular case of automatic control, time series are used to store results of experiments issued from the study of a system. The domain experts are able to derive laws/models (generally differential equations) from the time series raw data, that do not only describe the data, but provides a physical knowledge of the observed system. Our goal is to support the storing and exploitation of that knowledge by building a common repository of equations, that could organize equations to support their sharing and ease their retrieval.

In our proposal, we aim at taking advantage of domain expert expertise and knowledge by representing time series with their models (differential equations) and storing those models. Then, it would be possible to build a "library" of equations (like a list of different patterns) to compare new raw time series with. We chose to use a structure based on the data warehouse concept, enriched with some mathematical tools to fit the requirements of models storing and management. Such a structure was named *models warehouse*.

2. TIME SERIES REPRESENTATIONS: RELATED WORK

A lot of time series representations were introduced for numerous purposes and applications [2]. Some of the representations are said lossless, since they allow to recover the original time series data. However, the values calculated are, in fact, approximations of the original values. It is possible to recover the time series data from the equations, which makes them a lossless representation as well.

In addition, building and querying time series databases and comparing time series with each other is not a new issue. In order to have data stored in a form that meets the need of the processing method, some time series databases are NoSQL [4]. There are also relational time series databases, but they are usually not adapted to storing a time series, since they can have to store multiple series in a single ta-

BDA 2016, 15 au 18 Novembre, Poitiers, France.

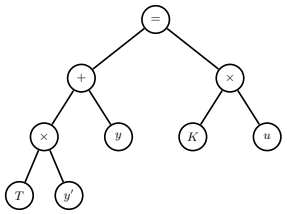


Figure 1: Binary tree of the equation 1

ble. Since, a differential equation describes a time series using variables, functions, numbers and operators (entities) related or associated to each other, they are more adapted to a storage in a relational database, than time series.

3. MODELS WAREHOUSE DESIGN

The equations are represented as binary trees. Here below is an example of a differential equation:

$$Ty'(t) + y(t) = Ku(t) \quad (1)$$

where T and K are real parameters defined as the time constant and the gain, respectively, $u(t)$ and $y(t)$ are the system inputs and outputs, respectively.

The Figure 1 gives the binary tree of the equation 1. The root node always contains the equality operator (=). To read the equation within the tree structure, we have to use a depth-first search in-order algorithm.

In order to compare equations and time series together, it is possible to generate a time series from an equation, by solving it numerically, with numerical algorithms, such as *Euler* and *Runge-Kutta* methods [5].

3.1 Models oriented ETL

The sources of the *models warehouse* includes both raw time series and equations already derived by domain experts (as shown on Figure 2), which implies that the *models warehouse* will have to cope with the two different types of data sources. Consequently, the ETL process of the *models warehouse* is split into two subprocesses, one specific to each type of sources.

In the models case, there are the three classical *Extract (E)*, *Transform (T)* and *Load (L)* modules. In fact, this subprocess aims at propagating the models from the sources to the DBMS, therefore it works exactly as any ETL process of a classical data warehouse. Whereas, in the time series case, the modules *Comparator (C)* and *Generator (G)* append themselves into the original ETL chain, between the *Transform (T)* and *Load (L)* modules. The role of this subprocess is to check if the data match with an equation (pattern) contained in the warehouse.

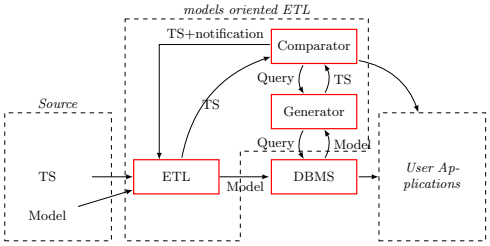


Figure 2: Generic models warehouse schema

A rough implementation of the *models warehouse* has been made using Java and Talend Open Studio. It was used to compare our solution with segmentation representations. The tests led to the following results: (i) in terms of storage space and approximation errors, using equations is as efficient as the segmentation methods, (ii) in terms of execution time, using equations is less efficient, but considering the method is supposed to support the analysis of the automatic experts, the warehouse efficiency has to be compared with their current efficiency (iii) and the storage of the input function (u in equation 1) can have a significant impact on the storage performance of the equations.

4. CONCLUSION AND FURTHER WORK

In this paper, a representation of time series based on differential equations was introduced. Then the notion of *models warehouse* aiming at storing and managing the equations of time series was detailed. A rough implementation was described in order to show how the *models warehouse* can be implemented and integrate mathematical tools to solve differential equations and compare equations with time series. The interest is to provide a storage structure, that can both centralize and organize equations and come with enough mathematical tools to perform some analysis with the stored equations.

The future research work will aim at (i) studying the possibility to use the PMML (Predictive Model Markup Language) standard, which is based on the XML language, to represent the equations (ii) and identify some mathematical tools to enhance comparison of raw time series with theoretical equations. In terms of implementations, the current work focuses on improving the introduced implementation by building the equations repository and implementing the models specific ETL subprocess, which would make a complete *models warehouse*.

5. REFERENCES

[1] P. Esling and C. Agón. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, Dec. 2012.

[2] T.-C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[3] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 3(3):263–286, 2000.

[4] D. Namiot. Time series databases. In *Proceedings of the XVII International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID/RCDL)*, pages 132–137, 2015.

[5] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 2002.

Exécution de requêtes distribuées sous contraintes d'anonymat

Axel Michel
Laboratoire d'Informatique Fondamentale d'Orléans
INSA Centre Val de Loire
Bourges, France
axel.michel@insa-cvl.fr

Benjamin Nguyen
Laboratoire d'Informatique Fondamentale d'Orléans
INSA Centre Val de Loire
Bourges, France
benjamin.nguyen@insa-cvl.fr

Philippe Pucheral
Laboratoire PRiSM
University of Versailles
Saint-Quentin en Yvelines
Versailles, France
philippe.pucheral@inria.fr

1. INTRODUCTION

De nombreux domaines scientifiques, allant de la médecine à la sociologie, ont une méthodologie basée sur les calculs statistiques sur des données personnelles. Avec l'expansion du Web et des bases de données massives qui le composent, les études statistiques sont devenues une science à part entière, transformant de grands ensembles de *micro-data* en connaissances. La plupart des études statistiques ont une utilité évidente telle que l'utilisation, par exemple, de données médicales pour améliorer la connaissance des maladies ou de données sur la consommation d'énergie dans le cas des *smart grids*. Dans toutes ces applications, les informations nécessaires sont acquises grâce à des données agrégées. La plupart du temps, les requêtes effectuées sur la base de données sont tout aussi efficaces sur des données anonymisées. Sans rentrer dans les polémiques qui entourent la sécurité comparée des différents protocoles d'anonymisation (*e.g.* k-anonymité, ℓ-diversité, differential privacy), tous partagent la caractéristique de procurer une protection uniforme aux individus. Ainsi, le même paramètre de *privacy* (k, ℓ, ou ϵ) s'applique à l'ensemble du groupe d'individus étudié, paramètre par ailleurs généralement ignoré de chacun.

Dans cet article, nous étudions le problème de la participation d'un individu (dit utilisateur) par la contribution de ses données personnelles à un calcul d'agrégation. On dit alors que l'utilisateur *participe* à ce calcul. Nous partons de l'hypothèse que chaque individu dispose de ses données dans un espace personnel sécurisé (appelé *Trusted Data Server* ou TDS). Un nombre croissant de propositions de *clouds* personnels ou serveurs personnels sécurisés fleurissent dans la littérature et commencent à apparaitre sur le marché (*e.g. Cozy Cloud*, *OwnCloud*). Sous cette hypothèse, nous proposons un modèle d'évaluation de requêtes statistiques dans lequel des contraintes d'anonymat sont définies par les utilisateurs. Notre contribution algorithmique porte sur l'évolution du système SQL/AA[5] permettant le calcul de requêtes distribuées sur une architecture de type *Trusted Cells* [1].

L'article est structuré comme suit. La section 2 présente le contexte de l'étude et énonce le problème à résoudre. La section 3 détaille la contribution et la section 4 conclut.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. CONTEXTE ET ÉNONCÉ DU PROBLÈME

Cette section rappelle les principes fondamentaux de l'architecture *Trusted Cells* [1] visant à préserver la vie privée des utilisateurs en permettant des calculs sur leurs données sans fuites d'informations. Nous nous focalisons ici sur les calculs d'agrégats lors de requêtes SQL.

2.1 L'architecture asymétrique des Trusted Cells

L'architecture asymétrique des *Trusted Cells* est une architecture basée sur la collaboration d'un grand nombre de TDS possédés par chaque utilisateur. Les TDS sont *trusted*, *peu disponibles* et *peu performants*.

Pour pallier au manque de disponibilité et de performance des TDS, on utilise une infrastructure de serveurs de support (SSI), *disponible*, *performante* (*e.g. cloud*) mais *untrusted*.

2.2 SQL/AA : Calcul de requêtes sur les TDS

SQL/AA est un protocole en trois phases permettant d'effectuer des calculs de requêtes SQL sur l'architecture des *Trusted Cells* [5] sans fuite d'informations. Lorsqu'un *querier* effectue une requête (*i.e.* parmis un jeu de requêtes dont les plans d'exécution ont déjà été calculés), la *phase de collection* permet à la SSI de récolter les données des TDS chiffrées, la *phase d'agrégation* permet aux TDS de calculer les agrégations (*e.g.* AVG, COUNT) de la requête et enfin, la *phase de filtrage* permet de calculer les clauses HAVING et de produire le résultat final envoyé au *querier*.

2.3 Rappel des principes d'anonymat

Pour protéger la vie privée des utilisateurs, de nombreux modèles d'anonymat existent tels que la *differential privacy* [2], le $k-$anonymat [4] ou la $\ell-$diversité [3]. Ce sont ces deux derniers modèles qui sont les plus utilisés de manière opérationnelle, c'est pourquoi nous nous y intéressons plus particulièrement dans cet article.

2.4 Énoncé du problème

Dans cet article nous étudions comment exécuter des requêtes SQL en respectant des contraintes d'anonymat définies par les utilisateurs. Le but étant de fournir un anonymat

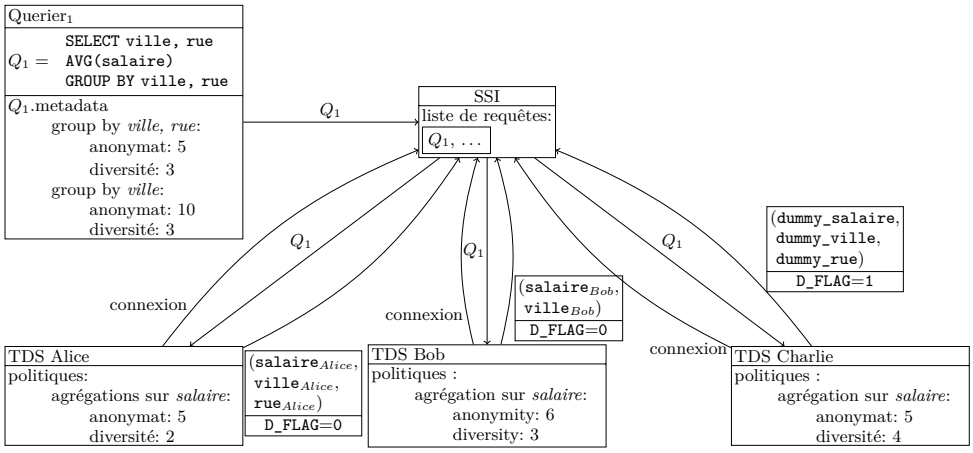


Figure 1: Exemple d'une phase de collection sous contraintes d'anonymat.

en fonction de contraintes différentes, de fournir aux utilisateurs plus de contrôle sur leurs données et leur assurer la protection de leur vie privée.

3. INTÉGRATION DE CONTRAINTES D'ANONYMAT DANS SQL/AA

Considérons dans cette section la simple requête SQL Q_1 calculant le salaire moyen des utilisateurs en les groupant par `ville` et par `rue` (voir la figure 1).

Supposons qu'*Alice* est une utilisatrice d'un TDS. Elle ne sait pas combien d'utilisateurs vont participer et souhaite que ses données figurent dans un groupe d'au moins cinq personnes partageant les même caractéristiques (*i.e.* 5-anonymat).

Modéliser l'anonymat en utilisant SQL

Principe général : La généralisation de données étant trop coûteuse pour les TDS, le *querier* peut déclarer des garanties d'anonymat dans les méta-données de la requête qui seront assurées par le protocole *SQL/AA*. De même, les utilisateurs fournissent des contraintes d'anonymat qui, si les garanties ne sont pas suffisantes, empêchent les TDS de partager les données des utilisateurs. Il faut noter que les garanties d'anonymat peuvent être assurées en ajoutant la clause `HAVING COUNT(*)>= 5` et la clause `HAVING COUNT(DISTINCT salaire) >= 3` à la requête pour assurer par exemple une garantie de 5-anonymat et de la 3-diversité.

Alice, souhaitant partager le montant de son salaire si la requête garantit un 5−anonymat, saura si elle peut participer à la requête Q_1 en examinant $Q_1.metadata$. Elle chiffrera ses données ainsi qu'un *flag* indiquant que ses données sont réelles puis les enverra à la SSI (comme décrit dans [5]).

Définition des contraintes d'anonymat : Chaque utilisateur définit un ensemble de politiques composées d'attributs sensibles (*e.g.* salaire) liés à des paramètres d'anonymat. Ces contraintes sont stockées dans la base de données du TDS de chaque utilisateur.

Comparaison des contraintes et garanties d'anonymat : Chaque TDS calcule la requête en utilisant sa propre vue des données. Si les garanties d'anonymat ne sont pas suffisantes, deux cas apparaissent. Le *querier* peut indiquer comment généraliser la requête (*e.g.* remplacer `GROUP BY ville, rue` par `GROUP BY ville`) dans les méta-données de la requête, donnant ainsi de meilleures garanties d'anonymat au détriment de la précision du résultat. Si ces garanties sont suffisantes le TDS répond par un tuple généralisé. Sinon le TDS renvoie un tuple *dummy* qui sera exfiltré du résultat final.

Dans l'exemple de la figure 1, les garanties d'anonymat proposées par le *querier* pour un groupe (`ville`, `rue`) sont trop faibles pour Bob (*i.e.* il souhaite un 6−anonymat). Mais celles proposées pour un groupe `ville` garantissent un anonymat qui convient à Bob. Pour ce qui est de Charlie, les garanties ne sont pas suffisante, il enverra donc de fausses données ainsi qu'un *flag* indiquant qu'elles le sont.

Afin de ne pas fausser les garanties d'anonymat, les agrégations des données généralisées sont calculées séparément des données non généralisées.

4. CONCLUSION

Les données sont généralisée sur un modèle peu flexible et impliquant des pertes de données non nécessaire. C'est pourquoi nous travaillons actuellement sur l'extension de ce modèle dans le but d'améliorer la généralisation de données. Nous cherchons à optimiser le détail du résultat et à réduire la perte de participants lors de la phase de collection.

5. REFERENCES

[1] T. Allard, N. Anciaux, L. Bouganim, Y. Guo, L. L. Folgoc, B. Nguyen, P. Pucheral, I. Ray, I. Ray, and S. Yin. Secure personal data servers : a vision paper. *PVLDB*, 3(1) :25–35, 2010.

[2] C. Dwork. Differential privacy. In *Proceeding of the 39th International Colloquium on Automata, Languages and Programming*, volume 4052, pages 1–12, 2006.

[3] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity : Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, page 24, 2006.

[4] L. Sweeney. k-anonymity : A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5) :557–570, 2002.

[5] Q. To, B. Nguyen, and P. Pucheral. SQL/AA : executing SQL on an asymmetric architecture. *PVLDB*, 7(13) :1625–1628, 2014.

AstroSpark - Towards a Distributed Data Server for Big Data in Astronomy

Mariem Brahem
DAVID lab., Univ. Versailles St Quentin, Versailles France, Paris Saclay University
mariem.brahem@uvsq.fr

Supervised by Stephane Lopes, Laurent Yeh, Karine Zeitouni
DAVID lab., Univ. Versailles St Quentin, Versailles France, Paris Saclay University
stephane.lopes@uvsq.fr, laurent.yeh@uvsq.fr, karine.zeitouni@uvsq.fr

ABSTRACT

Large amounts of astronomical data are continuously collected. As a result, support of scalable and high performance query processing of such data has become increasingly necessary. Apache Spark has been widely adopted as a successor to Apache Hadoop MapReduce to analyze Big Data in distributed frameworks. Despite its rich features, this framework can not be directly exploited towards processing astronomical data. In this work, we present AstroSpark, a distributed data server for astronomical data. AstroSpark extends Spark, a distributed in-memory computing framework, to analyze and query huge volume of astronomical data. It supports astronomical operations such as cone search, cross-match and histogram ... AstroSpark introduces data partitioning and optimization techniques to achieve high performance query execution.

Keywords

Astronomical Survey Data Management, Big Data, Query Processing, Spark Framework

1. CONTEXT AND PROBLEM DEFINITION

In recent years, there has been an accelerating explosion of astronomical data produced by advanced telescopes that can image enormous portions of the sky. For instance, Gaia an ESA mission [2], will face the challenge of dealing with an end mission volume of one Petabyte. Gaia is set to map our galaxy in three dimensions, to locate and characterize more than a billion of stars. Besides, traditional applications processed on a single machine cannot be used to query large size of data produced by the Gaia mission.
Distributed systems like Spark have become increasingly popular as a cluster computing models for processing large amounts of data in many application domains. Spark performs an in-memory computing, with the objective of outperforming disk-based frameworks such as Hadoop.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

However, these distributed frameworks do not provide efficient astronomical query processing capabilities, due to no data access optimization which involves intensive computing. Inspired by these observations, we propose AstroSpark a system that extends Spark towards a scalable, low-latency, cost-effective and efficient astronomical query processing framework.

2. ASTROSPARK PROPOSAL

AstroSpark in a nutshell adapts data partitioning (as shown in Figure 1) to efficiently processing astronomical queries. To this end, we apply a spatial-aware data partitioning, and first use linearization with the Healpix library [4] to transform a two dimensional data points into a single dimension value represented by a pixel identifier (Healpix ID). Healpix (Hierarchical Equal Area isoLatitude Pixelization) is a structure for the pixelization of data on the sphere. It is an available library maintained by the NASA. Using linearization with Healpix, we are able to manage a numerical range value. This is taken into account by many Spark function. The first benefit of using the Healpix library is that it is adapted to the spherical space, it does not allow space deformation. The second benefit is that it keeps data locality wich means that neighbouring points in the multi-dimensional space are likely to be close in the corresponding one dimensional space.

Astronomical objects are affected to IDs. In order to balance the partition sizes, we employ range partitioning and store the resulting partitions in HDFS. Partitioning is a fundamental component towards efficient and high performance processing of astronomical queries. Data partitioning enables query processing in parallel. For example, partitioning helps to prune out irrelevant partitions wich reduce computer resources and improve query performances.

Queries are expressed in Astronomical Data Query Language (ADQL)[1], a SQL-Like language improved with geometrical functions which allows users to express astronomical queries with a unified language. The query parser is extended to translate an ADQL query with astronomical functions and predicates into an internal algebric representation. Then, the query optimizer will enrich some prefiltering operators based on our spatial partitioning which make global filtering prune out irrelevant partitions.
AstroSpark extends the Spark SQL optimizer called Catalyst by integrating the particular logical and physical optimization techniques for the ADQL execution within the

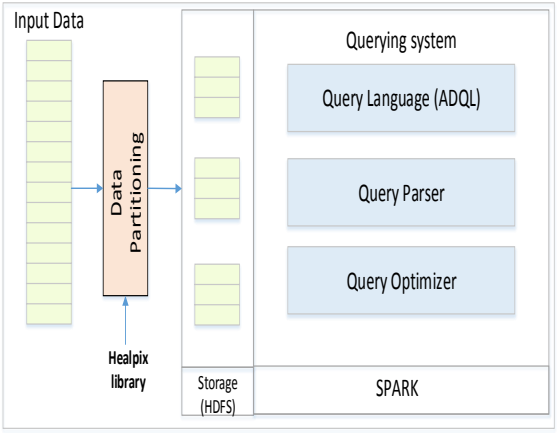


Figure 1: AstroSpark Architecture.

optimizer. AstroSpark focuses on three main basic astronomical operations:

- **Cone Search** is one of the most frequent queries in the astronomical domain, it returns a set of Stars whose positions lie within a circular region of the sky. A Cone is defined by a sky position and a radius around that position.
- **Cross-Matching** queries aim at identifying and comparing astronomical objects belonging to different observations of the same sky region in order to study the temporal evolution of the source.
- **Histogram** queries distribute the dataset into a specified number of groups and summarize astronomical information about each group.

Moreover, we can extend these queries to encompass other attributes, for instance, selection of sources within a certain angular distance from the specified center position (cone search) fitting as well limitations on a certain magnitude range and one particular spectral type.

3. RELATED WORK

Recent works have addressed the support of spatial data and queries using a distributed data server:

Spatialhadoop [3] is an extension to Hadoop that focuses on three basic operations range query, spatial join, k nearest neighbor (kNN) and supports spatial indexing using structures Grid, R-tree or R^+-tree.

MD-HBase [5] is a scalable multi-dimensional data store for Location Based Services (LBSs), which is an extension of HBase. MD-HBase supports a multi-dimensional index structure over a range partitioned Key-value store, builds standard index structures to support range and kNN queries.

GeoSpark [7] extends the core of Apache Spark to support spatial data types, indexes, and operations. GeoSpark provides native support for spatial data indexing (R-Tree and Quad-Tree) and query processing algorithms (range queries, kNN queries, and spatial joins over SRDDs).

SIMBA [6] is an extension of SPARK-SQL to support spatial queries and analytics over big spatial data. SIMBA builds spatial indexes over RDDs. It offers a programming interface to execute spatial queries (range queries, circle range queries, kNN, Distance join, kNN join).

All these systems are designed for the geo-spatial context that differs from the astronomical context in its data types and operations. These systems either do not provide a high-level query language or do not support a programming interface adapted to the astronomical context like ADQL. They do not exploit astronomical libraries which are suitable to the spherical coordinates system and do not use specific operations such as cone search queries, cross-match queries, and histogram queries.

4. CONCLUSION

This paper describes AstroSpark a distributed in-memory computing framework based on Spark for processing large-scale astronomical data. AstroSpark supports data partitioning with Healpix, offers an expressive query language in ADQL and extends the Spark SQL optimizer "Catalyst" to optimize astronomical query processing (cone search, cross-matching...). Our ongoing work focuses on data partitioning. For future work, we plan to add support for astronomical queries based on existing partitioned files. We envision to exploit partition pruning and query optimization techniques (caching techniques, cost based optimization, heuristics, auto-tuning, lazy-computation...) to efficiently execute astronomical queries. Experiments will focus on evaluating the data partitioning algorithm and proving the importance of partitioning for the efficient processing of astronomical queries.

5. ACKNOWLEDGMENTS

This work is partly founded by the CNES (Centre National d'Etudes Spatiales).

6. REFERENCES

[1] *ADQL*. `http://www.ivoa.net/documents/latest/ADQL.html`.

[2] *GAIA*. `http://www.cosmos.esa.int/web/gaia`.

[3] A. Eldawy and M. F. Mokbel. Spatialhadoop: A mapreduce framework for spatial data. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1352–1363. IEEE, 2015.

[4] K. M. Gorski, E. Hivon, A. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759, 2005.

[5] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi. *MD*-hbase: design and implementation of an elastic data infrastructure for cloud-scale location services. *Distributed and Parallel Databases*, 31(2):289–319, 2013.

[6] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo. Simba: Efficient in-memory spatial analytics. *In Proceedings of the 2016 ACM SIGMOD*, pages 1071–1085, 2016.

[7] J. Yu, J. Wu, and M. Sarwat. Geospark: A cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 70. ACM, 2015.

Gestion des données manquantes dans les grands entrepôts de données géo référencées : Application aux données agricoles.

Nestor Koueya est Doctorant en deuxième année au LIMOS[1] et à l'UBP[2] de Clermont Ferrand sous la direction d'Engelbert Mephu Nguifo, Sandro Bimonte et de Libo Ren.

1. Contexte et problématique

Aujourd'hui, grâce aux nouveaux moyens d'acquisition (données du Web, réseaux de capteurs, etc.) de plus en plus de sources de données spatiales sont disponibles pour des fins analytiques (« Spatial Big Data ») pour alimenter les modèles de simulation. Un entrepôt de données spatiales (EDS) est une « collection de données spatiales orientées sujet, intégrées, non volatiles et historisées, pour l'aide à la décision » (Bédard et al. 2001) exploitables pour alimenter les modèles (Truong et al., 2014). Ces données sont analysées en utilisant les opérateurs SOLAP qui permettent l'exploration en ligne des données entreposées selon le modèle spatio-multidimensionnel. Les opérateurs SOLAP intègrent des fonctions d'agrégation qui permettent la visualisation des données à différents niveaux de détails ou granularités. Au niveau des granularités fines, on retrouve les données détaillées ou micro données, alors que les données agrégées sont retrouvées au niveau des granularités élevées. Les données ou mesures agrégées résultent des calculs (somme, moyenne, etc.) opérés sur les données détaillées. Cependant, les valeurs incomplètes sont endémiques aux bases de données (Dyreson et al., 2003). Cette assertion est valable pour les EDSs. Leur présence peut influencer négativement la qualité des mesures agrégées (décisionnelles), puisque les résultats des analyses fondées sur des données incomplètes peuvent êtres inexacts (Dyreson et al., 2003 ; Rubin, 1976 ; Wohlrab et al. 2011 ; Wu et al. 2002).

La définition et l'implémentation des EDS pour le Spatial Big Data représentent une piste prometteuse et peu explorée car elle implique la redéfinition des concepts principaux des EDS classiques (stockage, operateurs d'analyse et restitution). En effet, le Spatial Big Data concerne les données géographiques issues des capteurs et modèles de simulation. Ces derniers produisent des grandes quantités de données hétérogènes à différentes échelles spatio-temporelles avec de nombreux problèmes de qualité (Bimonte, 2013). Dans ce cadre particulier, les méthodes pour la gestion de la qualité des données et en particulier l'incomplétude de données tel quel définies dans les EDSs classiques (Koueya et al., 2014) doivent être reformulées : « Quelles sont les données « utiles » pour l'estimation des données manquantes ?», « Comment exploiter la variété de données pour l'estimation des données manquantes ?», « Comment définir des méthodes d'estimation efficaces en temps de calcul ? », « Comment restituer aux décideurs les données estimées et les données originales dans les outils SOLAP ? ».

Dans ce projet, nous expérimenterons nos travaux sur l'évaluation des données manquantes pour l'analyse et la gestion des ressources des exploitations agricoles. Les données de base utilisées seront des gros volumes de données spatiotemporelles issus notamment du Web et de réseaux de capteurs. Un exemple d'utilisation des données est de prévoir les répercussions de changement de certaines pratiques sur les exploitations agricoles. Cette expérimentation se fera dans la suite des données collectées et identifiés dans précédents projets d'IRSTEA, tels que les projets Energetic et EDEN (Bimonte et al., 2013).

2. Actions réalisées

Dans un premier temps, nous avons fait une classification des méthodes d'imputation des faits détaillés manquants dans les entrepôts de données en deux groupes : Les méthodes verticales selon que les faits détaillés sont estimés à partir des agrégats et les méthodes horizontales selon qu'ils sont estimés à partir d'autres faits connus du même niveau de granularité. Les travaux sur la gestion des

[1] http://limos.isima.fr/

[2] http://www.univ-bpclermont.fr/

« Cette thèse est cofinancée par l'Union européenne dans le cadre du Fonds Européen de Développement Régional (FEDER) »

incomplétudes dans les bases de données (Green et al., 2006) peuvent être adaptés dans le cadre des entrepôts de données comme méthode horizontale parce qu'ils ne prennent pas en paramètres les données agrégées connus. Cependant, dans le cadre des entrepôts de données multi granulaires, il peut arriver que les mesures de certains faits soient stockées plutôt aux niveaux les moins détaillés (Iftikhar et al. 2010). Dans ce cas, le fait que les méthodes verticales ignorent les autres faits détaillés connus peut résulter plutôt à des résultats aberrants. Par ailleurs, ces mesures stockées aux niveaux les moins détaillés deviennent les contraintes d'agrégations. Ces contraintes sont distributives si la fonction d'agrégation relative est distributive (somme, …), algébriques si la fonction est algébrique (moyenne, …), et holistiques (distinct count, …). Toutefois les méthodes horizontales qu'elles proviennent des bases de données, des statistiques ou de la fouille de données ne prennent pas en compte ces contraintes dans le processus d'estimation de faits détaillés manquants. Pour faire face à cette problématique, nous avons également proposé un algorithme générique d'ajustement basé sur la programmation linéaire qui permet de tenir compte de ces contraintes. Les trois catégories des fonctions d'agrégation sont discutées. Une expérimentation a été faite sur les bases de données synthétiques et réelles afin de valider notre approche.

3. Actions futures

L'ajustement avec les contraintes d'agrégation holistiques ne peut pas être calculé à partir de résultats intermédiaires. Dans ce cas, il faut le calculer à partir des valeurs élémentaires correspondant au niveau de granularité le plus bas. Egalement tous les faits et membres connus ou non sont engagés dans le processus d'ajustement. Cette opération devient très couteuse en raison qu'elle dépend explicitement du volume et de la dimension des données. Pour gérer la volumétrie, nous définirons des prédicats de sélection des faits sémantiquement utiles, et des approches de réduction de dimension pour que seuls les membres ayant une importance participent au processus d'estimation et d'ajustement. Nous implémenterons également une variante de la méthode d'ajustement avec « Spark » pour faire face à la scalabilité. Nous comptons également entamer le second volet de cette thèse en exploitant la caractéristique spatiale des SOLAP, pour l'estimation des faits manquants.

4. Bibliographies

[1] Bédard, Y., et Han, J. (2001). *Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery,* Geographic Data Mining and Knowledge Discovery.

[2] Bimonte, S., Pradel L, M., Boffety , D., Tailleur , A., Andre, G., Bzikha, R., Chanet, JP. (2013). *A new sensor-based Spatial OLAP architecture centered on an agricultural farm energy-use diagnosis*. International Journal of Decision Support System Technology, vol. 5, n° 4,

[3] Dyreson, C. E, Pedersen, T. B. et Jensen, C. S. (2003). *Incomplete information in multidimensional databases*, In Multidimensional Databases, pages 282-309. Maurizio Rafanelli(ed), IDP.

[4] Green, T. J., Tannen V. (2006), *Models for Incomplete and Probabilistic Information,* EDBT'06, P.278-296.

[5] Iftikhar, N. et Pedersen, T. B. (2010). *Schema Design Alternatives for Multi-granular Data Warehousing*, 21th International Conference, DEXA.

[6] Koueya , N., Bimonte , S., Mephu Nguifo, E. - 2014. *Une nouvelle approche d'estimation pour les entrepôts de données multi-granulaires incomplètes*. EDA, Vichy, p. 129-144 16 p.

[7] Rubin, D.B. (1976). *Inference and Missing Data*. Biometrika, 29, 159-183.

[8] Truong, Thai M., Amblard, F., Gaudou, B. and Sibertin-Blanc, C. (2014). *To calibrate & validate an agent-based simulation model, an application of the combination framework of BI solution & Multi-agent platform*, 6th ICAART, Angers, France, 6-8 March.

[9] Wohlrab, L. et Furnkranz, J. (2011). *A review and comparison of strategies for handling missing values in separate-and-conquer rule learning*, J. of Intelligent Information Systems, 36(1) :73-98.

[10] Wu, X. et Barbara, D. (2002). *Learning missing values from summary constraints*, SIGKDD Explorations, Volume 4, Issue 1

Towards a benchmark for Database exploration

Mahfoud DJEDAINI
University de Tours, France
mahfoud.djedaini@univ-tours.fr

ABSTRACT

Supporting interactive database exploration (IDE) is a problem that attracts lots of attention these days. Exploratory OLAP (On-Line Analytical Processing) is an important use case where tools support navigation and analysis of the most interesting data, using the best possible perspectives. While many approaches were proposed, a recurrent problem is how to assess the effectiveness of an exploratory OLAP approach. We propose a benchmark to do so, that relies on an extensible set of user-centric metrics that relate to the main dimensions of exploratory analysis. This paper can be seen as a summary of [4], where we described the results obtained so far.

1. INTRODUCTION

A lot of approaches have been proposed in databases for helping users exploring the data [6]. These approaches are diverse, and use a bunch of different strategies to do so. Even if we focus to the specific domain of OLAP databases, these approaches are still numerous and various. Despite their diversity, the output they provide is pretty similar. It is either a query, a sequence of queries, or a set of cells of the database. Generally, each approach is evaluated by its authors independently, and the evaluation mostly focus on performance like time and energy consumption. However, to the best of our knowledge, the quality of these approaches is not evaluated, in the sense that how the user is helped by these systems is not measured. From this statement, we clearly identify two important investigation topics. First, how such Interactive Data Exploration (IDE) approaches could be evaluated by taking into consideration the user experience ? And then, how to provide a unified way of evaluating such systems so that we are sure that they are evaluated under the exactly same circumstances ? This is exactly what we are currently investigating in the context of OLAP. On the one hand, we are studying a way of scoring OLAP explorations by considering metrics that reflect the quality of user experience. On the other, and in parallel, we are working on a proposal of a unified benchmark protocol. The work we did for tackling these two problems has been published in a paper to TPCTC 2016 [4], a conference focused on databases benchmarking. The two next sections describe respectively each of these two points.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. OLAP EXPLORATION QUALITY

An OLAP exploration [2] is basically a sequence of OLAP queries over a database instance issued by a given user. We propose to score OLAP explorations using five categories of user-centric metrics borrowed from Exploraty search [10]. We implemented each category with a primary metric and a secondary to counterbalance it. The following sections describe the main metrics for each category. More details can be found in [4].

User engagement measures how engaged and invested is a user on a system. For this category, we borrow from web search two popular and intuitive metrics. Query Depth (QD) as primary metric, represents the number of queries. Query Focus (QF) as secondary metric, measures the degree to which an exploration is focused.

Information Novelty measures the quantity of Relevant New Information (RNI). We use a normalized entropy as primary metric to measure the quantity of information contained in each query result of the exploration. The secondary metric measures the Increase in View Area (IVA), i.e. the increase in the number of viewed cells.

Intuitively, a task consists in exploring a cube area around a group of cells. **Task completeness** is reached when the whole neighborhood around a cell, in the sense of OLAP operation, has been explored. A simple way of measuring it is with recall and precision. Recall (R) is the primary metric since, consistently with exploratory search, we consider OLAP navigation as a recall oriented activity. Precision (P) is then the secondary metric.

Measuring **task time** is done by adapting metrics of existing TPC benchmarks. The primary metric comes from the TPC-DS benchmark and measures the query frequency, i.e. number of queries per second (QPS). The secondary metric simply measures the elapsed time (TET) between the beginning and the end of an exploration.

Learning and cognition aims at evaluating the user knowledge. Knowledge Tracing (KT) [3] has been proposed originally in e-learning to evaluate students knowledge, based on a sequence of exercises that they have to solve. We adapt KT by considering as an exercise finding OLAP queries with high Information Novelty. The primary metric Learning (L) is then the knowledge level estimated by KT.

The secondary metric measures the Learning Growth Rate (LGR).

3. BENCHMARK

3.1 OLAP environment generation

An OLAP database (schema and instance) and a set of user sessions over it are first generated. We use by default the Star Schema Benchmark (SSB) [7], but the benchmark can be initialized with any other OLAP schema. We use CubeLoad [9] for automatically generating user sessions. CubeLoad generates realistic OLAP workloads, taking as input a cube schema and the desired number of sessions. Finally, a realistic database instance is generated with PDGF [8], a data generator that handles generation of skewed data. The obtained sessions are then clustered, using a distance based on a similarity measure tailored for OLAP sessions [2]. Finally, a Markov inspired generative model is learned from each cluster. A cluster then represents a user and the model represents his/her behavior.

3.2 Generating OLAP explorations

The evaluation protocol first provides a seed session, which is a set of seed queries representing part of a navigation, as a context for continuation of the navigation. Then, the user and the SUT play. The user smartly issues new queries using his/her model. The SUT uses its internal intelligence and the context (current query, user, ...) to propose new queries. Like in real cases, SUT propositions may or may not be included in the exploration, depending on the user choice. A SUT is allowed to play a given number of times, after which the process is stopped. The obtained exploration is then scored using the metrics described above. The same process is repeated a large number of times for a given SUT. Finally a SUT obtains a score for each metric, by averaging the scores for all the explorations for the given metric.

4. EXPERIMENTS AND FIRST RESULTS

4.1 Settings

A prototype of the benchmark is available as a Java API. SUTs can be evaluated by the prototype, by simply plugging them by implementing a Java interface. We used our prototype to conduct preliminary experiments, and obtained promising results. First, in order to test the benchmark ranking, we compared three basic SUTs that have simple behavior. 'Random', the one having the worst strategy, returns purely random next move suggestions. 'Naive' generates queries that are one OLAP operation away from the previous query. 'Cheater' uses 'insider information' in order to return good suggestions. Moreover, we tested two approaches from the literature, namely CineCube [5] and Falseto [1]. These approaches have a known behavior that we expect to retrieve through our metrics.

4.2 Results discusson

Regarding the three basic SUTs, the results globally allow us to rank 'Cheater' highest, followed by 'Naive' and 'Random' with the poorest performance. Concerning the two SUTs from the literature, an interesting thing is that the scores they obtain for each metric reflect pretty well their behavior. More than that, differents aspects of their behavior can be quantified by the metrics. When it comes to comparing existing approaches with basic SUTs, we retrieve a coherent ranking. The scores allow to globally rank both Falseto and Cinecube better than Naive and worse than Cheater, while being good in some points. Contrary to Random and Naive that do not seem to effectively support data exploration, Falseto and Cinecube are clearly of great help for the user.

5. PERSPECTIVES

So far we investigated how to evaluate OLAP exploration approaches. We proposed a set of user-centric metrics, together with a complete benchmark protocol, firmly relying on state of the art techniques. We currently have a prototype available, and we already conducted some preliminary tests. We are currently conducting more tests, by evaluating more SUTs from the literature. After that, an important thing is to study how this OLAP tailored benchmark could be adapted to relational databases. Indeed, the exploratory nature of OLAP provides some user friendly features and operators that cannot be found in standard relational databases. We also published a website (`http://www.info.univ-tours.fr/~marcel/benchmark.html`) where we plan to gather all the resources and documentation around the benchmark.

6. REFERENCES

[1] Julien Aligon, Kamal Boulil, Patrick Marcel, and Verónika Peralta. A holistic approach to OLAP sessions composition : The falseto experience. In *DOLAP 2014*, pages 37–46, 2014.

[2] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Similarity measures for olap sessions. *KAIS*, 39(2) :463–489, 2014.

[3] Albert T. Corbett and John R. Anderson. Knowledge tracing : Modelling the acquisition of procedural knowledge. *UMUAI*, 4(4) :253–278, 1995.

[4] Mahfoud Djedaini, Pedro Furtado, Nicolas Labroche, Patrick Marcel, and Veronika Peralta. Benchmarking exploratory olap. In *TPCTC*, 2016.

[5] Dimitrios Gkesoulis, Panos Vassiliadis, and Petros Manousis. Cinecubes : Aiding data workers gain insights from OLAP queries. *IS*, 53 :60–86, 2015.

[6] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, pages 277–281, 2015.

[7] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. The star schema benchmark and augmented fact table indexing. In *TPCTC*, pages 237–252, 2009.

[8] Tilmann Rabl, Meikel Poess, Hans-Arno Jacobsen, Patrick E. O'Neil, and Elizabeth J. O'Neil. Variations of the star schema benchmark to test the effects of data skew on query performance. In *ICPE'13*, pages 361–372, 2013.

[9] Stefano Rizzi and Enrico Gallinucci. Cubeload : A parametric generator of realistic OLAP workloads. In *CAiSE 2014*, pages 610–624, 2014.

[10] Ryen W. White and Resa A. Roth. *Exploratory Search : Beyond the Query-Response Paradigm.* Morgan & Claypool Publishers, 2009.

Stratégies de divulgation de lien en ligne pour les réseaux sociaux*

Younes Abid, Abdessamad Imine, Amedeo Napoli, Chedy Raïssi and Michaël Rusinowitch
INRIA Nancy and Lorraine University, France
firstname.lastname@inria.fr

ABSTRACT

Les réseaux sociaux en ligne soulèvent des questions éthiques et de confidentialité puisque ils fuient des informations, qui peuvent être sensibles, sur les utilisateurs. Cependant, la réalisation des attaques de la vie privée en ligne dans un délai raisonnable reste une tâche difficile. Dans cet article, nous abordons le problème de la divulgation rapide de nombreux liens d'amitié en utilisant uniquement les requêtes légitimes (à savoir, les requêtes et les outils fournis par le réseau social ciblé).

Keywords

Online Social networks, Privacy, Link disclosure attacks

1. INTRODUCTION

A social network can be defined as a website that allows users to create personal profiles in order to share information with their friends and acquaintances. Since their appearance at the end of the twentieth century, social networks have known an outstanding success and have become a global phenomenon. For instance, Facebook connects about 25% of humans in 2016 [1] and YouTube served videos to almost one-third of all connected people on the Internet [2]. With this rapid networks expansion, new scientific fields have emerged such as online social network analysis [6] creating a common domain of interest from sociology to mathematics and through computer science [5]. However, the emergence of social networks is also giving reasons to worry about privacy and ethics issues [8].

In order to mimic real (i.e., non-cybernetical) societal interactions, some social networks like Facebook, Linkedin and Viadeo support the creation of groups besides the profiles creation. As such, social networks can be modeled by two types of graphs as depicted by Figure 1.

∗. This work is funded by Fondation MAIF.

1. http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

2. https://www.youtube.com/yt/press/en/statistics.html

BDA 2016, 15 au 18 Novembre, Poitiers, France.

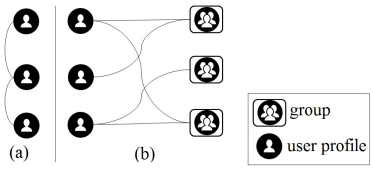


Figure 1: Social graphs : (a) unipartite friendship graph, (b) bipartite groups membership graph.

The friendship graph (a) is unipartite and models the friendship links between users while membership graph (b) is bipartite and models the membership links between users and groups. In [2] researchers propose a Partial Graph Profile Inference (PGPI) algorithm that exploit group memberships to infer profiles attributes. In [7], relational learning approaches and group memberships are used to infer sensitive attribute of users such as locations. In this work, we perform group uncover attack to disclose the group network of the target. Fast and accurate link disclosure attacks using only authorized requests are then performed based on this network.

2. MOTIVATIONS

The proposed class of attacks in this work finds its roots in the saying : *"birds of a feather flock together"*. Hence, disclosing sensitive information about social network users can be more accurate if friendship and group membership links are disclosed with certainty. In this work we put into question the visibility setting of friend list on social networks. And we prove that they can be bypassed.

3. CHALLENGES

Link disclosure attacks in online social networks can be misleading and time consuming. For instance, [4] show that homophilic attributes have significant influence on predicting friendship between users of Facebook. But, the number of homophilic profile can be very small. And disclosed links did not leak additional information about the target. On the other hand, the attacker can consider the friends of the target friends as potential friends and check these links. But, due to low degree of separation, tens of thousands of links need to be checked for each single target [1][3]. Furthermore, crawling tasks for online attacks must be fast and selective to avoid attacking deleted profiles and links and to do not miss newly created ones.

4. STRATEGY

We propose effective solutions to uncover the network groups of the target through bipartite graph traversal algorithm. The first step of the algorithm is to collect seed groups that can be the published groups by the target or suggested groups by OSN based on the target attributes. The second step is to collect the members of these groups from published membership lists. Next step is to collect groups from published groups lists of collected members and avoid crawling explored ones. The number of hops is the number of steps of collection of new groups.

Starting from the hypothesis that the target hides his links and since the visibility settings of links in a symmetric relationship are independently managed by both linked nodes (profile-profile link or profile-group link), it is sufficient to reach the friends and the groups of the target that publish their links in order to disclose the target links. Hence, we perform membership and friendship attacks to disclose links between the target and his uncovered network of groups and their members. We also perform mutual-friend attacks to disclose common friends between the target and the members of his uncovered network of groups.

5. RESULTS

We analysed the public densities of more than 22 000 Facebook groups. Where the public density of a group g is the ratio of published friendship links between its members to all possible friendship links between them. Results show that the public density decreases when the size of groups increases as depicted by figure 2. We have attacked online

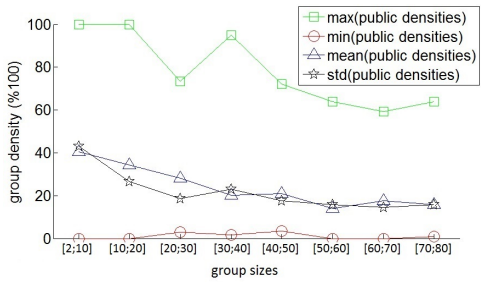


Figure 2: Variation of public density with respect to group size.

more than 1 000 active Facebook users that hide their friend lists.

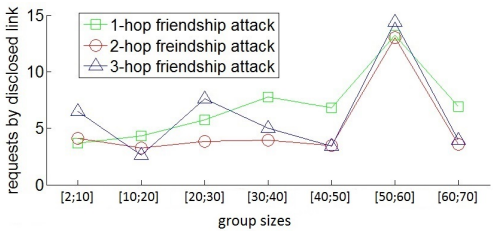


Figure 3: The average number of friendship requests to disclose one friendship link.

Figure 3 shows that the average number of friendship requests to disclose one link with certainty increases as the size of groups increases. And 2-hop attacks are more accurate than 1-hop and 3-hop attacks.

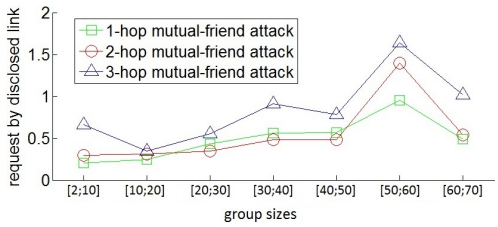


Figure 4: The average number of mutual-friend request to disclose one friendship link.

Figure 4 shows that the number of mutual-friend requests to disclose one friendship link is far less than the number of friendship requests depicted by Figure 3 as mutual-friend request returns a list of friends. However, since the target hides his friend list, it is only successful if both the distant group members and the mutual friends publish their friends lists. Moreover, it is not effective in the case of sparse networks since it does not disclose the friendship link between two users that did not have mutual friends even if they are friend.

6. REFERENCES

[1] R. Bakhshandeh, M. Samadi, Z. Azimifar, and J. Schaeffer. Degrees of separation in social networks. In *Proceedings of the Fourth Annual Symposium on Combinatorial Search, SOCS 2011, Castell de Cardona, Barcelona, Spain, July 15.16, 2011*, 2011.

[2] R. Y. Dougnon, P. Fournier-Viger, and R. Nkambou. Inferring user profiles in online social networks using a partial social graph. In *Advances in Artificial Intelligence - 28th Canadian Conference on Artificial Intelligence, Canadian AI 2015, Halifax, Nova Scotia, Canada, June 2-5, 2015, Proceedings*, pages 84–99, 2015.

[3] S. Edunov, C. Diuk, I. O. Filiz, S. Bhagat, and M. Burke. Three and a half degrees of separation. In *Research at Facebook*, 2016.

[4] I. Elkabani and R. A. A. Khachfeh. Homophily-based link prediction in the facebook online social network : A rough sets approach. *J. Intelligent Systems*, 24(4) :491–503, 2015.

[5] N. Memon and R. Alhajj. Social networks : A powerful model for serving a wide range of domains. In *From Sociology to Computing in Social Networks - Theory, Foundations and Applications*, pages 1–9. 2010.

[6] J. Scott. *Social Network Analysis*. Third edition. SAGE Publications, 2013.

[7] E. Zheleva and L. Getoor. To join or not to join : the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 531–540, 2009.

[8] E. Zheleva, E. Terzi, and L. Getoor. *Privacy in Social Networks*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2012.

Analyse de concepts formels pour la classification de triplets RDF

Justine Reynaud
LORIA
(CNRS, INRIA, U. Lorraine)
BP 239
54506 Vandoeuvre-les-Nancy
justine.reynaud@loria.fr

Yannick Toussaint
LORIA
(CNRS - INRIA - U. Lorraine)
BP 239
54506 Vandoeuvre-les-Nancy
yannick.toussaint@loria.fr

Amedeo Napoli
LORIA
(CNRS - INRIA - U. Lorraine)
BP 239
54506 Vandoeuvre-les-Nancy
amedeo.napoli@loria.fr

ABSTRACT

L'émergence du web des données et des *linked open data* donne lieu à de nouvelles problématiques d'abstraction, d'organisation et d'interprétation. Nous nous intéressons ici à la classification de triplets `RDF`. Pour cela, nous nous appuyons sur l'analyse de concepts formels (FCA) et ses extensions, notamment les structures de patrons. La FCA permet notamment de construire une représentation visuelle de la classification et permet à l'utilisateur de naviguer facilement au sein de celle-ci.

1. INTRODUCTION

Le web des données a permis la création de nombreuses bases de connaissances librement accessibles en ligne et liées les unes au autres. DBPedia est l'une d'entre elles. L'objectif de DBPedia est de représenter sous forme structurée les connaissances disponibles sur Wikipedia. Cette structure s'appuie sur des langages standards tels que `RDF`, `RDFS` et `OWL`.

On considère un ensemble de ressources sémantiquement liées entre elles. Différents types de ressources sont distingués : les *prédicats* correspondent à une relation binaire entre deux ressources, les *classes* représentent des ensembles de ressources, et les *instances* représentent les ressources qui ne sont ni des prédicats ni des classes. L'unité de base pour exprimer une connaissance est le triplet (sujet, prédicat, objet). Il peut exprimer des faits (`B_Obama néA Honolulu`), mais aussi structurer les ressources entre elles, notamment grâce aux prédicats `rdfs:subClassOf` et `rdfs:subPropertyOf` (notées par la suite `subC` et `subP` respectivement). Par exemple, `Capitale rdfs:subClassOf Ville` indique que toutes les capitales sont aussi des villes. Un exemple d'ensemble de triplets ainsi que les hiérarchies de classes et de prédicats associés sont présentés figure 1.

Ici, nous souhaitons classifier un ensemble de triplets en tenant compte des hiérarchies de classes et de prédicats impliqués. Nous considérons donc d'une part les ensembles de triplets à classifier, et de l'autre les hiérarchies.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

2. APPROCHES EXISTANTES

Des travaux existent dans le domaine du web sémantique (par exemple, typer automatiquement les ressources [9]) et dans le domaine des logiques de description notamment. Certain d'entre eux s'appuient sur la FCA [10].

La FCA [7] est une théorie mathématique qui permet de classifier des objets en fonction des attributs qu'ils partagent. Étant donné un ensemble G d'objets, un ensemble M d'attributs, et une relation binaire $I \subseteq G \times M$, $\mathbb{K} = (G, M, I)$ est un contexte formel. gIm s'interprète comme *g a pour attribut m*. Les correspondances de Galois (notées $.'$) sont définies comme suit : $A' = \{m \mid \forall a \in A, aIm\}$ avec $A \subseteq G$ et $B' = \{g \mid \forall b \in B, gIb\}$ avec $B \subseteq M$. Elles permettent de définir un concept formel (A, B) comme un sous-ensemble d'objets ($A \subseteq G$) partageant un sous-ensemble maximal d'attributs ($B \subseteq M$) et qui vérifie $A'' = A$ et $B'' = B$. A et B sont appelés *extent* et *intent* respectivement.

De nombreuses extensions ont été proposées afin de prendre en compte des données plus complexes. Dans [4], les auteurs proposent de considérer une *background knowledge*, une connaissance extérieure au contexte, en tenant compte d'une hiérarchie entre les attributs.

Les structures de patrons [6] sont une autre extension permettant de tenir compte de la hiérarchie entre les attributs. Au lieu de considérer un ensemble d'attributs, on considère un ensemble de descriptions partiellement ordonnées $(D, \sqcap)$. Chaque objet est associé à une description. À la place d'une intersection entre les ensembles d'attributs, une opération de similarité entre les descriptions est définie. Un concept a donc pour extent un ensemble d'objets et pour intent une description. Dans [3], une classification des sujets de chaque triplet reposant sur une structure de patrons est proposée. Étant donné un ensemble de triplets $\mathcal{B}$, les auteurs considèrent la description d'un sujet s comme l'ensemble des couples $(p, C(o))$ tels que $(s, p, o) \in \mathcal{B}$, où $C(o)$ représente la classe de o. La similarité entre deux sujets s_1 et s_2 est l'ensemble des couples (p, C) de façon à ce qu'il existe x, y tels que $(s_1, p, x), (s_2, p, y) \in \mathcal{B}$ et $C(x)$ `subC` C, $C(y)$ `subC` C. Cette approche s'appuie sur la hiérarchie des classes considérées, mais ne prends donc pas en compte la hiérarchie de prédicats. D'autres approches basées sur la FCA ont été proposées, en s'appuyant sur une représentation sous forme de graphes, notamment dans [5] et [8].

3. APPROCHE PROPOSÉE

Nous proposons une structure de patrons permettant de classifier les triplets en tenant compte des hiérarchies de

classes et de prédicats.

3.1 Définition

Étant donné un triplet (s, p, o), sa description est $\delta((s, p, o)) = (C(s), p, C(o))$. La similarité entre deux triplets $t_1 = (s_1, p_1, o_1)$ et $t_2 = (s_2, p_2, o_2)$ est $\delta(t_1) \sqcap \delta(t_2) = (lcs(C(s_1), C(s_2)), lcs(p_1, p_2), lcs(C(o_1), C(o_2)))$ où lcs (*least common subsumer*) désigne la super-classe la plus spécifique ou le super-prédicat le plus spécifique, selon que les ressources soient des classes ou des prédicats respectivement.

Cette approche peut-être généralisée aux ensembles de triplets. La description d'un ensemble de triplets est l'ensemble des descriptions de chacun des triplets qu'il contient : $\Delta(T) = \{\delta(t) \mid t \in T\}$. La similarité entre deux descriptions d'ensembles de triplets $\Delta(X)$ et $\Delta(Y)$ est l'ensemble des similarités entre chaque paire de triplet $t_x \in \Delta(X)$ et $t_y \in \Delta(Y)$ les plus spécifiques. Une description (s_i, p_i, o_i) est dite *plus spécifique* qu'une autre description (s_j, p_j, o_j) si et seulement si s_i `subC` s_j, p_i `subP` p_j et o_i `subC` o_j.

3.2 Exemple

Pour calculer la similarité entre les deux ensembles de triplets, on commence par calculer la similarité paire à paire. Lorsque l'un des *lcs* renvoie un élément $\top$, il n'y a pas de similarité. Par exemple,
δ(`B_Obama, présidentDe, USA`)$\sqcap\delta$(`Arkansas, étatDe, USA`)
= (`Président, présidentDe, Pays`)$\sqcap$(`État, étatDe, Pays`)
= (lcs(`Président, État`), lcs(`présidentDe, étatDe`),
lcs(`Pays, Pays`))
= ~~($\top$, $\top$, `Pays`)~~.
Le triplet résultant est alors ignoré.

La similarité entre les deux ensembles de triplets Figure 1b est l'ensemble de cinq triplets suivant :

```
Président présidentDe Pays   Président néA Ville
Ville villeDe État           Lieu situéDans Lieu
État étatDe Pays
```

Ces cinq triplets représentent des connaissances qui, bien que triviales, n'étaient pas explicites. Ce sont ces éléments qui vont permettre de classifier des ensembles triplets.

4. CONCLUSION

L'approche proposée permet de classifier des ensembles de triplets en tenant compte des hiérarchies de classe et de prédicat. Elle a été testée sur des exemples simples et des tests sur des triplets issus de DBPedia sont en cours. Le travail à venir consiste en une implémentation de la structure de patrons proposée. Il s'agira également d'optimiser le procédé, en s'appuyant notamment sur RMQ [2]. Cette approche pourra ensuite étendre [1] pour extraire des définitions dans DBPedia.

5. REMERCIEMENTS

La thèse de Justine Reynaud est co-financée par la Direction Générale de l'Armement et par la Région Lorraine.

6. REFERENCES

[1] M. Alam, A. Buzmakov, V. Codocedo, and A. Napoli. Mining Definitions from RDF Annotations Using Formal Concept Analysis. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, July 2015.

[2] M. Alam, A. Buzmakov, A. Napoli, and A. Sailanbayev. Revisiting Pattern Structures for Structured Attribute Sets. In *Proceedings of the 12th International Conference on Concept Lattices and Their Applications*, Clermont-Ferrand, France, Oct. 2015.

[3] M. Alam and A. Napoli. Interactive Exploration over RDF Data using Formal Concept Analysis. In *Proceedings of IEEE International Conference on Data Science and Advanced Analytics*, Paris, France, 2015.

[4] C. Carpineto and G. Romano. *Concept Data Analysis : Theory and Applications*. John Wiley & Sons, Chichester, UK, 2004.

[5] S. Ferré. A Proposal for Extending Formal Concept Analysis to Knowledge Graphs. In *Formal Concept Analysis*, volume LNCS 9113, pages 271–286, Nerja, Spain, June 2015.

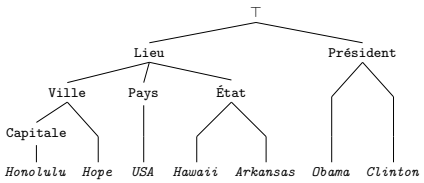
[6] B. Ganter and S. O. Kuznetsov. Pattern structures and their projections. In *Conceptual Structures : Broadening the Base, 9th International Conference on Conceptual Structures*, Stanford, CA, USA, July 30-August 3.

[7] B. Ganter and R. Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.

[8] J. Kötters. Concept lattices of RDF graphs. *Formal Concept Analysis and Applications*, page 81, 2015.

[9] A. G. Nuzzolese, A. Gangemi, V. Presutti, F. Draicchio, A. Musetti, and P. Ciancarini. Tìpalo : A tool for automatic typing of dbpedia entities. In *The Semantic Web : ESWC 2013 Satellite Events, Montpellier, France, May 26-30, 2013, Revised Selected Papers*, pages 253–257, 2013.

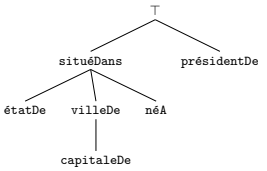
[10] B. Sertkaya. A survey on how description logic ontologies benefit from formal concept analysis. *CoRR*, abs/1107.2822, 2011.



(a) Hiérarchie `subC` et instances associées

```
Ensemble 1
B_Obama présidentDe USA .
B_Obama néA Honolulu .
Honolulu capitaleDe Hawaii .
Hawaii étatDe USA .
Ensemble 2
B_Clinton présidentDe USA .
B_Clinton néA Hope .
Hope villeDe Arkansas .
Arkansas étatDe USA .
```

(b) ABox



(c) Hiérarchie `subP`

Figure 1: La figure 1b représente deux ensembles de faits (correspondant à la ABox des logiques de description). La figure 1c (1a) représente la hiérarchie de prédicats (classes) étant donné la relation `subP` (`subC`). Les instances sont reliées aux classes auxquelles elles appartiennent dans la figure 1a. Cet exemple est inspiré de [5].

Démonstrations

Génération de Requêtes pour les Bases de Données Orientées Graphes

Guillaume Bagan
CNRS LIRIS
guillaume.bagan@liris.cnrs.fr

Angela Bonifati
Université Lyon 1 & CNRS LIRIS
angela.bonifati@univ-lyon1.fr

Radu Ciucanu
Université Blaise Pascal, Clermont-Ferrand
ciucanu@isima.fr

George H. L. Fletcher
TU Eindhoven
g.h.l.fletcher@tue.nl

Aurélien Lemay
Université Lille 3 & INRIA
aurelien.lemay@inria.fr

Nicky Advokaat
TU Eindhoven
n.advokaat@student.tue.nl

ABSTRACT

Les outils de gestion de bases de données orientées graphes sont actuellement en évolution permanente. Dans ce contexte, les outils permettant une génération de données ainsi que de scénarios de travail sont des éléments clés dans les études empiriques. Néanmoins, les générateurs actuels de graphe fournissent un support limité, voire inexistant, dans la génération de scénarios d'usage, ou s'en tiennent à un nombre fixe de cas d'études.

Afin de dépasser ces limitations, nous présentons gMark, le premier environnement de génération de graphes synthétiques et de scénarios d'utilisation indépendant à la fois du domaine et du langage de requête. Ses principales innovations sont : (i) un contrôle précis de l'instance de graphe généré et des requêtes correspondantes à partir de schémas définis par l'utilisateur ; (ii) le support d'un langage de requête expressif, incluant notamment la récursion ; et (iii) une estimation de la sélectivité des requêtes générées.

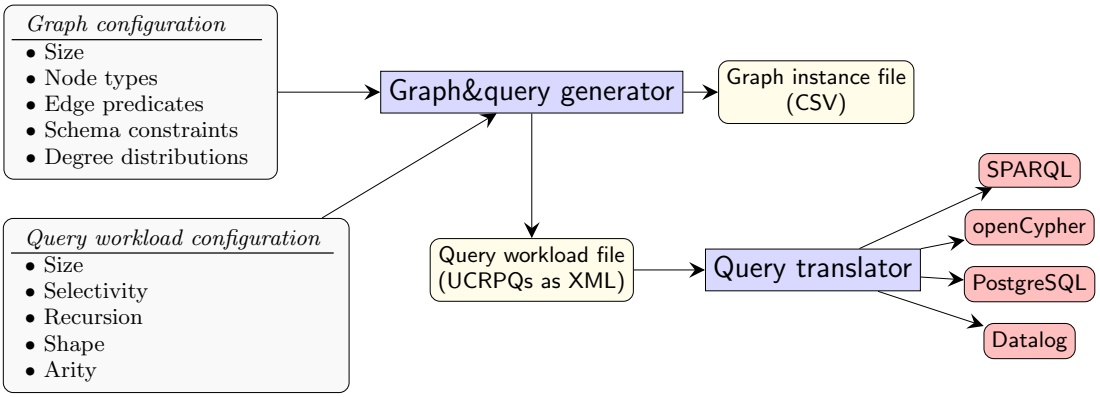
Durant la démonstration, nous avons illustré la capacité hautement paramétrable des graphes et des requêtes générés à travers différents schémas et différentes sélectivités choisies, et la variété des langages de requêtes supportés par le système. Nous avons montré également une comparaison des performances de quatre moteurs de requêtes de l'état de l'art, et comment les scénarios générés par gMark nous permettent de comprendre leurs forces respectives, et les évolutions que l'on pourrait souhaiter.

L'article de démonstration de gMark a été publié à *VLDB 2016* [1]. gMark est *open-source* (`https://github.com/graphMark/gmark`) et prêt à être utilisé par la communauté. L'article de recherche qui contient tous les détails techniques a été accepté dans le journal *TKDE* [2]. Nous présentons l'architecture de gMark dans la figure suivante.

1. REFERENCES

[1] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat. Generating flexible workloads for graph databases. *PVLDB*, 9(13):1457–1460, 2016.

[2] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat. gMark: Schema-driven generation of graphs and queries. *IEEE Transactions on Knowledge and Data Engineering*, 2017 (à paraître). `http://arxiv.org/abs/1511.08386`.



Estocada: Stockage Hybride et Ré-écriture sous Contraintes d'Intégrité

Rana B. AL-Otaibi
UC San Diego, USA
ralotaib@eng.ucsd.edu

Francesca Bugiotti
CentraleSupélec & INRIA, France
francesca.bugiotti@inria.fr

Damian Bursztyn
INRIA & École Polytechnique, France
damian.bursztyn@inria.fr

Alin Deutsch
UC San Diego, USA
alin@cs.ucsd.edu

Ioana Manolescu
INRIA & École Polytechnique, France
ioana.manolescu@inria.fr

Stamatis Zampetakis
INRIA, France
stamatis.zampetakis@inria.fr

ABSTRACT

La production croissante de données numériques a conduit à l'émergence d'une grande variété de systèmes de gestion de données (Data Management Systems, ou DMS). Dans ce contexte, les applications à usage intensif de données ont besoin *(i)* d'accéder à des données hétérogènes de grande taille ("Big Data"), ayant une structure potentiellement complexe, et *(ii)* de manipuler des données de façon efficace afin de garantir une bonne performance de l'application. Comme ces différents systèmes sont spécialisés sur certaines opérations mais sont moins performants sur d'autres, il peut s'avérer essentiel pour une application d'utiliser plusieurs DMS en même temps.

Dans ce contexte nous présentons ESTOCADA, une application donnant la possibilité de tirer profit simultanément de plusieurs DMSs et permettant une manipulation efficace et automatique de données de grande taille et hétérogènes, offrant ainsi un meilleur support aux applications à usage intensif de données. Dans ESTOCADA, les données sont reparties dans plusieurs fragments qui sont stockés dans différents DMSs. Pour répondre à une requête à partir de ces fragments, ESTOCADA est basé sur la re-écriture de requêtes sous contraintes; ces dernières sont utilisées pour représenter les différents modèles de données et la répartition des fragments entre les diffèrents DMSs.

Cette dèmonstration a ètè présentè lors de la conférence "International Conference on Data Engineering 2016" [6].

1. CONTEXT

There is significant consensus around the observation that the times where one system fits all data management needs are over [24]. Nowadays data-intensive applications often involve dealing with diverse datasets in terms of size and structure: relations flat or nested, complex-structure graphs, documents, and poorly structured logs, or even text data. Processing tasks to be run this data are also very varied: selective or bulk processing, structure traversal and aggregation, joins, grouping, pattern matching, advanced analytic processing using dedicated functions, etc.

Facing these needs, a wide variety of DMSs is now available to be used in data management applications. These systems include structured data*base* management systems from major vendors, which currently come in centralized or cloud edition, supporting traditional relational stores (disk- or memory-resident), but also novel formats such as JSON, RDF, graphs, text, etc. They have been joined by the large crowd of so-called NoSQL systems. More generally, numerous systems are competing for the glut of so-called "Big Data" applications; their capabilities (supported data format and operations) and their performance blueprint (in terms of speed and scale) makes each of them unique, and enable numerous optimization opportunities.

We propose to demonstrate ESTOCADA [5, 6], a platform providing applications with *transparent, optimized* data access to *diverse, heterogeneous storage systems*. ESTOCADA enables storing one dataset in a set of possibly overlapping fragments, while providing to the application access to this dataset in the native language most suited for the dataset, e.g., SQL if the data is relational (or object-relational), JSONiq if the data is in JSON documents, etc. At the heart of ESTOCADA lies a common modeling of the different data set and storage systems data models, data fragments, and also queries in an internal, expressive formalism based on relational queries and constraints (which, as we show, is rich enough to capture rich data structure including nesting, object identity, functional dependencies and more); query processing then starts by solving a problem of query rewriting under constraints, backed by an efficient recent algorithm [15]. Demo attendees will have the opportunity to try ESTOCADA on a set of systems of very varied nature, data models, and architectures; they will use different data fragmentation and queries and inspect the resulting query evaluation plans.

2. DEMONSTRATION OUTLINE

ESTOCADA can answer queries over an application dataset based on fragments stored in a variety of underlying DMSs, across distinct data models and platforms. The remaining

BDA 2016, 15 au 18 Novembre, Poitiers, France.

problem is to automatically recommend the way in which the data sets that an application needs to work with should be fragmented across such DMSs in order to maximize performance under specific storage or cost parameters.

To adapt to changes in the datasets, workload, and set of DMSs being used, we chose to internally *represent each data fragment as a materialized view over one or several datasets*; thus, query answering amounts to view-based query rewriting. As is well-known from prior work in data integration, this local-as-view approach allows the application to remain unchanged as the underlying data collections are modified. To enable query rewriting over and across different data models, we translate into an *internal pivot model* the declarative specification of the data stored in each fragment, as well as the incoming query, formulated in the application dataset model; specifically, our pivot model is based on relational conjunctive queries. Further, our reliance on views gives sound foundation to efficiency, as it guarantees the complete storage of data, and the correctness of the fragmentation and query answering.

We will show ESTOCADA in action on a set of scenarios closely derived from a large-scale online marketplace application scenario, on datasets obtained through the Big Data Benchmark [4], and server logs from several actual e-commerce platforms to which we gained access through the French R&D collaborative project Datalyse on Big Data analytics (`http://www.datalyse.fr`). To illustrate the scenarios we will use Postgres, Redis, and Spark as the underlying storage systems.

The demo attendee experience is as follows: **1.** Pick a dataset, which comes with a previously specified workload and a set of possible fragments; chose a subset of the fragments, view their specification in the source native language, and after translation to the pivot internal model. **2.** Pick a workload query and trigger its rewriting: inspect its translation in the pivot model, the output of the exploits the very significant performance savings brought by the recent provenance-aware C&B algorithm (PACB, in short) [15] rewriting algorithm, its translated form and finally the executable plan. **3.** Trigger the execution of the rewriting, which outputs a set of performance statistics split across the underlying DMS and ESTOCADA' runtime. We will provide for each dataset the specification of one fragment which stores it "as such" in a DMS of its native data model; this will enable comparing performance between the vanilla (one-store) execution and the one enabled by multiple stores.

3. RELATED WORK

Heterogeneous data integration is an old topic [8,13,20,22] but the remark "one-size does not fit all" [24] has been recently revisited [16, 21]. The performance benefits of using multiple stores together (a Hadoop one and a relational database) have been demonstrated in [19]; they select relational views to be materialized based on cost information, but do not handle multiple data models through a unified approach as we do. Polystores [9, 10] allow querying heterogeneous stores by grouping similar-model platform into "islands" and explicitly sending queries to one store or another; data sets can also be migrated by the users. This contrasts with our LAV approach where the data store variety is hidden to the application layer. The integration of "NoSQL" stores has been considered e.g., in [3] again in a top-down GAV approach without considering materialized views.

Adaptive stores for a single data model have been studied e.g., in [2, 7, 14, 17, 18]; views have been also used in [1, 23] to improve the performance of a large-scale distributed relational store. The novelty of ESTOCADA here is to support multiple data models, by relying on powerful query reformulation techniques under constraints.

Data exchange tools such as Clio [11, 12] allow migrating data between two different schemas. We aim at providing to the applications transparent data access to heterogeneous systems, relying on fundamentally different rewriting techniques.

View-based rewriting and view selection are grounded in the seminal works [13, 20]; the latter focuses on maximally contained rewritings, while we target exact query rewriting, which leads to very different algorithms. Further setting our work apart is the scale and usage of integrity constraints. Our pivot model recalls the ones described in [8, 22] but ESTOCADA generalizes these works by allowing multiple data models both at the application and storage level.

4. REFERENCES

[1] P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava, and R. Ramakrishnan. Asynchronous View Maintenance for VLSD Databases. In *SIGMOD*, 2009.

[2] I. Alagiannis, S. Idreos, and A. Ailamaki. H2O: a hands-free adaptive store. In *SIGMOD*, 2014.

[3] P. Atzeni, F. Bugiotti, and L. Rossi. Uniform access to NoSQL systems. *Information Systems*, 2014.

[4] Big Data Benchmark. `https://amplab.cs.berkeley.edu/benchmark/`.

[5] F. Bugiotti, D. Bursztyn, A. Deutsch, I. Ileana, and I. Manolescu. Invisible Glue: Scalable Self-Tunning Multi-Stores. In *CIDR*, 2015.

[6] F. Bugiotti, D. Bursztyn, A. Deutsch, I. Manolescu, and S. Zampetakis. Flexible hybrid stores: Constraint-based rewriting to the rescue. In *ICDE*, pages 1394–1397, 2016.

[7] D. Dash, N. Polyzotis, and A. Ailamaki. Cophy: A scalable, portable, and interactive index advisor for large workloads. *PVLDB*, 4(6), 2011.

[8] A. Deutsch and V. Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *VLDB*, 2003.

[9] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik. The BigDAWG polystore system. *SIGMOD Record*, 2015.

[10] A. Elmore, J. Duggan, M. Stonebraker, and al. A Demonstration of the BigDAWG Polystore System. *PVLDB*, 2015.

[11] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, 2003.

[12] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD*, 2005.

[13] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 2001.

[14] S. Idreos, M. Kersten, and S. Manegold. Database Cracking. In *CIDR*, 2007.

[15] I. Ileana, B. Cautis, A. Deutsch, and Y. Katsis. Complete yet practical search for minimal query reformulations under constraints. In *SIGMOD*, 2014.

[16] A. Jindal, J.-A. Quiané-Ruiz, and J. Dittrich. WWHow! Freeing Data Storage from Cages. In *CIDR*, 2013.

[17] M. Karpathiotakis, I. Alagiannis, T. Heinis, M. Branco, and A. Ailamaki. Just-in-time data virtualization: Lightweight data management with ViDa. In *CIDR*, 2015.

[18] A. Katsifodimos, I. Manolescu, and V. Vassalos. Materialized view selection for XQuery workloads. In *SIGMOD*, 2012.

[19] J. LeFevre, J. Sankaranarayanan, H. Hacigümüs, and al. MISO: souping up big data query processing with a multistore system. In *SIGMOD*, 2014.

[20] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB*, 1996.

[21] H. Lim, Y. Han, and S. Babu. How to Fit when No One Size Fits. In *CIDR*, 2013.

[22] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML queries on heterogeneous data sources. In *VLDB*, 2001.

[23] J. Shute, R. Vingralek, B. Samwel, and al. F1: A Distributed SQL Database That Scales. In *PVLDB*, 2013.

[24] M. Stonebraker and U. Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone. In *ICDE*, 2005.

Demonstration of the CloudMdsQL Multistore System

Boyan Kolev
Inria and LIRMM
Montpellier, France

Carlyna Bondiombouy
Inria and LIRMM
Montpellier, France

Patrick Valduriez
Inria and LIRMM
Montpellier, France

Ricardo Jiménez-Peris
LeanXcale and UPM
Madrid, Spain

Raquel Pau
Sparsity Technologies
Barcelona, Spain

José Pereira
INESC TEC and U. Minho
Braga, Portugal

The blooming of different cloud data management infrastructures, specialized for different kinds of data and tasks, has led to a wide diversification of DBMS interfaces and the loss of a common programming paradigm. This has turned multistore systems to a major topic in the nowadays cloud landscape.

In this demonstration, we present Cloud multidatastore query language (CloudMdsQL) [1], a functional SQL-like language, designed for querying multiple heterogeneous databases (e.g. relational and NoSQL) within a single query containing nested subqueries. Each subquery addresses directly a particular data store and may contain embedded invocations to the data store's native query interface. Thus, the major innovation is that a CloudMdsQL query can exploit the full power of local data stores, by simply allowing some local data store native queries to be called as functions, and at the same time be optimized based on a simple cost model, e.g. by pushing down select predicates or using bind join.

One of the major challenges in front of the CloudMdsQL language/engine is to allow joins across heterogeneous data stores and to be able to perform them in an efficient way. For this reason, we pay special attention to the use of bind joins and we apply this technique even when native queries are used.

This demonstration concentrates on a CloudMdsQL use case scenario: a social network analysis tool for marketing companies. The use case aims at finding the *communities in a social network, for a specific set of topics, with their top influencers*. Marketing companies are interested in discovering the people they need to convince about the quality of a specific brand. The dataset of this use case is a sample of Twitter, but it allows working with other social networks like Facebook or blogs. The application runs a Twitter listener of a set of topics in real-time; it modifies the database for each tweet it receives. The schema of this application contains a generic entity called *Document* to store text-items (tweets, messages, etc.), which can appear *copies* or *references*. An *Entity* (person or company) is an *author* of a document or a *mention* of a social-network account. The people interactions in social networks with copies, references or mentions, can be understood as a set of graph of influences. In other words, we can infer who influences who and about what. These *Influences* and the *Communities* are incrementally computed when a new tweet comes to the application and thus, these concepts are part of the application schema.

The specification of the main query the application uses is as follows: given a set of keywords k_1, k_2, k_3, find the 10 biggest communities and, for each community, find the 20 most influencers. For each of these influencers, the system must return the number of influenced entities inside the community, the influencer's id, name and account creation date and the last published document.

In order to implement this use case, we use a graph database (Sparksee) to store the graph of *Influences* and compute the *Communities*; a relational database (MonetDB) for all the basic information about *Entities* and *Documents* (only metadata); a document database (MongoDB) to store the *Document* contents; and a key-value data store (HBase) to index communities per keyword. Following the execution plan for the CloudMdSQL query, the query engine first invokes an HBase query to retrieve the communities preliminarily computed for a specific keyword; then, for each community, runs a Sparksee query using the Sparksee Python API to find the top 20 influencers, the number of influenced entities inside the community, and the maximum influence propagation depth. Finally, the basic information of each influencer (id, name, account creation date) and the last published document is retrieved by running queries to MonetDB and MongoDB.

For the execution plan, the query optimization plays an important role to assign the bind join method to all the join operations. The reason is that the selected communities relevant to the keywords k_1, k_2 *and* k_3 are always a few, and thus the Sparksee query is evaluated only for a few communities, which significantly reduces the number of executions of expensive graph computations.

BDA 2016, 15 au 18 Novembre, Poitiers, France.

1. REFERENCES

[1] B. Kolev, P. Valduriez, C. Bondiombouy, R. Jiménez-Peris, R. Pau, and J. Pereira. Cloudmdsql: querying heterogeneous cloud data stores with a common language. *Distributed and Parallel Databases*, 34(4):463–503, 2016.

FleDDi: Highlighting the Flexibility of Data Dissemination Systems

S. Dufromentel S. Cazalens F. Lesueur P. Lamarre
Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205
F-69621, France
<firstname.lastname@liris.cnrs.fr>

ABSTRACT

New generation data dissemination systems such as P2P-based Pub/Sub ones become more and more complex. Understanding how they dynamically organize their participants or how the results are disseminated is not easy. Even harder is comparing systems. Our FleDDi demonstrator enables a deeper understanding of their flexibility, in particular through a nice dynamic visualization of the organization building and data dissemination. The audience will be able to experience different scenarios: high variety of the queries issued, presence of very popular queries, arrival of a participant with high dissemination capacities and issuing queries of varying complexity. Comparison may be drawn between several systems ranging from Unicast and Multicast to more evolved solutions such as Delta or QTor.

1. INTRODUCTION

Data dissemination systems such as Pub/Sub ones enable users that express long-term possibly complex queries to get the desired results. While the traditional, old generation systems require some system nodes (brokers, mediators) to perform most of the work, more recent ones are deployed over peer-to-peer networks involving the users themselves as participants, asking them to perform some computations and to diffuse the resulting data streams. P2P-based systems avoid some major drawbacks of the (semi-)centralized approaches, such as bottlenecks due to a number of users growing faster than the number of system nodes to serve them; or such as the privacy issues due to a single participant having exhaustive and exclusive control of the system.

On the counterpart of their active participation, users should have some compensations such as getting the results with a reasonable load and a limited latency. Those constraints being opposed is one of the issues that P2P-based propositions have to address. Another one is the fact that such systems present important variabilities on several aspects. We say an organization is *flexible* if it copes with these problems by dynamically adapting itself to variations of i) the participants' capacities, that are limited and often heterogeneous; ii) the query popularity (some queries are expressed by a lot of participants while others by very few); iii) the complexity of the queries. Several approaches address the problem of data dissemination using different strategies to tackle the flexibility problem.

Efficiently working on data dissemination require to understand the details of the considered propositions. A good visualization is essential, as the exact behavior of a system depends on the structure of the organization graph, which is much harder to evaluate through statistics only. Flexibility-related aspects typically need some studies, as they may lead to major modifications of graphs. Thus it is of utmost importance to have visual tools that enable to dynamically compare the different organizations and highlight their strong and weak points. However, to the best of our knowledge, there is a lack of software to do this. So, we developed FleDDi (*FLExibility of Data DIssemination*), a demonstrator that provides both statistics and dynamic visualization to study the flexibility of several data dissemination systems.

In Section 2, we present the demonstrator itself. In Section 3, we briefly present the different propositions that are currently implemented and ready to be used in FleDDi, while Section 4 presents some example scenarios.

2. THE FLEDDI DEMONSTRATOR

FleDDi comes with a set of already implemented systems and several functionalities to study their flexibility and compare them. It enables to i) visualize the dynamic construction of the organization, ii) obtain statistics about the system organization such as the average depth of the organization graph, iii) visualize the data dissemination across the system once the organization is established, iv) visualize which queries participants contribute to, under the form of "computing units" for those solutions which have such possibility.

Launching FleDDi. In order to evaluate the flexibility of systems in various situations, FleDDi accepts any of the following events under the form of instructions:

- A new participant arrives in the system; it specifies both its dissemination[1] capacity (number of possible simultaneous connexions) and the initial query it aims to subscribe to.
- A previously inserted participant expresses another query.
- A participant withdraws a query (it entirely leaves the system when all its queries have been withdrawn).

- A participant (subscriber or publisher) has to change its dissemination[1] capacity, i.e. the number of other participants it is able to serve.
- A data source publishes some new item in its stream.

Launched in the graphics mode, the demonstrator randomly generates a set of those instructions at startup, or reads them from a previously generated run file (in .tsv format). At any time, the graphical interface enables to send some new instructions, such as changing the capacity of a given node or asking the data source to publish a new data item to visualize how the data is propagated.

Overview of the FleDDi architecture. Our demonstrator is fully written in Java 7, the code being split into several main packages.

Package **LANG** is about the query languages. In order to provide generic, language-independent implementations of the propositions, it contains interfaces for parsing queries, evaluate costs, check equivalences, and also compute rewritings, as most considered organizations rely on the concept of query rewriting using (materialized) views [4]. Currently a keyword-based filtering language is integrated, with logical and ($\wedge$) and or ($\vee$) operators and a custom rewriting system allowing unions and intersections of views. Although simple, such a language enables to compare the flexibility aspects of all the systems from the same perspective.

Package **ORGA** contains the different organization algorithms that can be used. In order to be generic, FleDDi relies on a tracker. It receives the participants' subscriptions and places them according to the different implemented algorithms. Participants are informed of (rewritten) queries they have to perform, and of the nodes they have to forward their results to. This enables to quickly and easily compare their situation regarding to the different organizations.

Package **NETW** contains the different ways to simulate network connections and transfers. It contains for now three implementations: a simple, direct communication between Java objects; a network simulator based on PeerSim [6] to evaluate the systems deployment in realistic conditions (e.g. simulations with a huge number of participants, in order to generate statistics); and a graphical-oriented implementation to visualize each step of the organization separately. A prototype for real deployments is currently in progress.

Package **INST** contains the instrumentation, i.e., the code responsible for measurements, analysis and statistics, and the graphical interface itself. It is based on GraphStream [3] and enables to dynamically visualize and interact with the system organization.

The whole[2] demonstrator is free software, released under the terms of the GNU GPL v3, and can freely be extended by adding new languages or approach implementations (which simply require to implement the related interfaces).

Extra documentation and download links can be found at `http://liris.cnrs.fr/~sdufrome/fleddidem`

3. IMPLEMENTED APPROACHES

The currently implemented systems in FleDDi are:

A simple **Unicast** system that connects all the participants directly to the data source, without taking care of its capacity limitations, and a basic **CDN** that uses several system nodes (if available) to share the load. Both of them can compute results before sending them, or send the original stream and let users compute their own queries.

An applicative **Multicast**, having all the participants connected in a common spanning tree, letting them i) receive and share the original stream unchanged, and then ii) locally compute their own queries to obtain the results, without sharing them. This highly takes care of the participants capacities and efficiently reduces the latency, but presents some obvious limitations regarding to the computation load.

Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks [1], called **SemPO**, is a containment-based organization: each rewriting involves a single view. The original paper is interesting regarding the computational aspects, but does not take care of the participants' dissemination limitations. Thus the participants may be overwhelmed. We developed an extension of this approach that avoids this issue and thus makes the system more flexible. Both the original organization and the extended one can be used in FleDDi.

Delta [5] is a P2P Pub/Sub system based on query rewriting using views, whose organization is obtained through several steps: computing the possible rewritings from the relevancy relations between the queries, then calling Integer Linear Programming (ILP)[2] to select the best rewritings considering the costs and diffusion aspects; and finally, reduce the general latency (possibly by re-increasing the costs). This is an interesting approach, but presents some limitations regarding flexibility. First, there's no taking care of query popularity (cycles suppression leads to lost a lot of relevancy relations between equivalent queries). Second, the latency-limitation algorithm follows the known relations in the rewriting graph, which are incomplete and may prevent some available dispositions. Third, the original paper describes an algorithm to run "every X steps", lacking information about what to do for the other steps of organization.

The approach we proposed [2], called **QTor** (for *Query Torrent*) is based on the query popularity. In order to deal with it, the approach relies on the notion of *communities*. Each community is dedicated to a given query, and involves the participants that contribute to the evaluation of this query (or any other expression known as strictly equivalent), and await the same results. Those communities are then linked to each others according to the rewriting relations, each of them being seen as a view to rewrite the others' queries. Participants initially join all the communities that correspond to the queries they express (by creating them if needed). Then, if they have enough capacities, they are allowed to contribute their resources to other communities (mostly those related to their original ones) in order to help them to work. The approach is incremental, avoiding cycles rather than suppressing them, and results in a huge flexibility regarding to both query popularity (the resource sharing between communities also gives some advantages for unpopular queries) and to heterogeneous capacities (the most powerful participants can easily help the more limited ones). The organization remains easy to deploy, the general problem being split into subproblems of more limited size.

[1]The computing capacity is also important in real situations, but the only effect in a simulated environment is to makes the resulting graphs less readable.

[2]Due to license limitation, the ILP solver used by Delta, Gurobi [7], cannot be included in a public release of the demonstrator, so this implementation will be shown, but not shared.

FleDDi also contains an **fQTor** implementation, which is a "flat" QTor, based only on equivalence, without any use of rewritings.

4. EXAMPLE SCENARIOS

We describe three scenarios that the attendance will experiment during the demonstration. For each of them, FleDDi enables the audience to first visualize the dynamic building of the system, and then the data dissemination. A scenario is described by: i) a number of participants with given diffusion capacities, that may be fully homogeneous or varying (e.g. following a Poisson law), ii) the total number of queries issued in the system and the repartition of keywords among them. In the aim to get human-readable graph visualizations, the scenarios are run with a low number of participants and queries.

In **Scenario 1**, 50 queries are inserted, each of them being expressed by 3 participants with homogeneous capacities. All participants, including the source, have a dissemination capacity limited to 10 connexions. Queries contain a single keyword, each different from the other, that prevent any possibility of rewriting. This corresponds to a case with a high heterogeneity of the queries within the system. This is thus a classical problematic case as the source is way too limited to serve all the network.

In such a case, query oriented systems (Delta, SemPO) have no other choice than to overwhelm the source (as the only possibility to discharge it is to follow the rewriting relations), but allow two thirds of the other participants to get their results without any computation. In the Multicast organization, there is no overwhelming, but all the participants have to compute themselves their queries. In QTor, resource sharing between communities allows to gain on both aspects, avoiding source overwhelming and enabling most of the participants to simply get the results without any computation.

During the second phase of the demonstration, the attendance will be able to compare the latency due to the source's overwhelming (it would not be able to serve all of its children at the same time) to the one due to distance from the source (a node can only serve it children once it had receive the tuple from its parents), by observing the successive highlights of the dissemination arcs between participants.

Scenario 2 involves 100 participants with heterogeneous capacities, all of them issuing equivalent queries. This is a "query with very high popularity" scenario.

Here, QTor and Multicast have similar network appearance. The main difference is that Multicast requires all the participants to compute the results by themselves, while QTor enables to directly share the results. Such case is problematic for the Delta approach, as a single equivalence class means a strongly connected embedding graph, which makes each step of the organization harder.

The attendance will be able to add some new participants, that express different but related queries, in order to see how they are inserted. In particular, adding a participant with limited capacities issuing a query that can be used by rewriting gives very different results in those three organizations.

Scenario 3 has 250 participants with heterogeneous capacities. The distribution of the keywords among the queries follows a Zipf law. Once the building of the whole organization has been shown, the experimentation is automatically paused, enabling the attendance to perform some changes in the system.

The aim is to evaluate how the systems react to an important capacity-related event, for instance the insertion of a powerful participant, able to serve the whole system by itself. Such a participant has to first express a new query: the attendance may choose queries of different complexity and see that their choice leads to very different results, particularly in Delta. Indeed, the possibility, for Delta, to take advantage of the participant huge capacity depends on the number of queries that can be rewritten using its one. In QTor, even if this query gives no rewriting possibility, the participant is able to join its parent communities to help them, and then the huge capacity is always exploited.

Another possibility for this scenario is to take an existing node, for instance the data source, and to highly reduce its capacity, to see how the organizations can adapt to the new situation. If the source is chosen, and as the aim of Delta is to discharge the source, regardless to its capacity, there would be no major change while it is possible for much of the nodes to find an existing rewriting (but some nodes are moved though, due to the non-deterministic aspects of the organization). In QTor, resource sharing between communities is simply redefined, without reconsidering the community relations, whatever the chosen node is.

5. CONCLUSION

The FleDDi demonstrator fills a gap in the data dissemination evaluation methods, by providing a simple way to evaluate and visualize the flexibility of an approach regarding to some aspects like the query popularity and the heterogeneous, variable capacities of the participants.

Unsurprisingly, the evaluations we ran using this demonstrator have confirmed that the QTor approach is, to the best of our knowledge, the most flexible approach, as none of the previous ones have explored all those aspects at the same time.

6. REFERENCES

[1] R. Chand and P. Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *EuroPar 2005 Parallel Processing*. Springer, 2005.

[2] S. Dufromentel, S. Cazalens, F. Lesueur, and P. Lamarre. QTor: A flexible publish/subscribe peer-to-peer organization based on query rewriting. In *DEXA*. Springer, 2015.

[3] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *ECCS*, 2007.

[4] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, December 2001.

[5] K. Karanasos, A. Katsifodimos, and I. Manolescu. Delta: Scalable data dissemination under capacity constraints. *PVLDB*, 2013.

[6] A. Montresor and M. Jelasity. PeerSim: A scalable P2P simulator. In *P2P'09*, Seattle, WA, 2009.

[7] Gurobi Optimization et al. Gurobi optimizer reference manual. *URL: http://www. gurobi. com*, 2012.

ACKNOWLEDGEMENTS

This work has been partially funded by the French ANR SocioPlug project under grant No. ANR-13-INFR-0003.

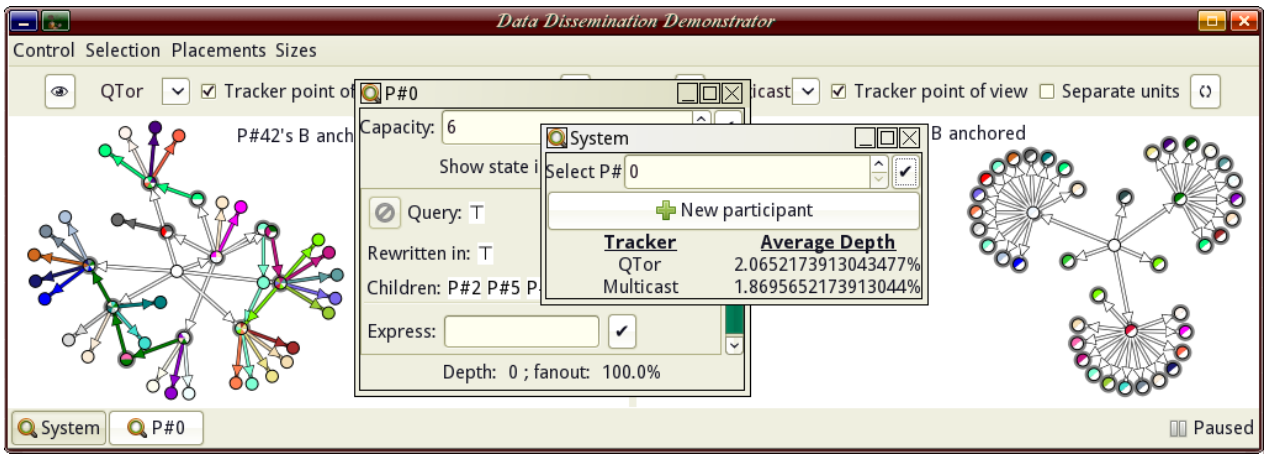


Figure 1: Graphical interface of FleDDi

APPENDIX: EXAMPLE SCREENSHOTS

Figure 1 shows an example of the graphical interface, comparing a QTor (on the left) and a Multicast system (on the right) with 42 participants. The source is able to diffuse results to 6 participants at the same time, which is more limited than the other participants. Therefore, in the Multicast system, the six first participants are linked to the source, and the other to one of them (there is here no need to have more than two links from a given participant to the source). In the QTor system, a few participants are involved in several communities and perform the related queries (the gray halos on some participants indicate that they have to compute some results, while the participants without halo have only to diffuse the stream they obtain unchanged).

The Control menu enables to perform some control operations, mainly suspend the processing of the startup-planned instructions to perform some manual modifications. The Selection menu helps to list the internal windows currently opened (the taskbar at the bottom of the figure is not always shown, as the Java graphical library (Swing) uses a plateform-specific appearance). It also provides an item to open the "System" internal window even if this one is currently closed. The two remaining menus enable to configure the way the nodes are shown: the placement algorithm (currently, as a tree, sources at bottom, or left to the GraphStream automatic layout, which is shown on the Figure), and the size of the nodes (homogeneous, or varying regarding to the (total or used) capacity).

As it is costly to display more than two graphs, only two of the tested organizations will be simultaneously displayed at the same time, but it is possible to switch the views at any time to compare all of them using the popdown menu. Notice that the interface also enables to display the computing units: a given node is split into as many units as there are requests which it contributes to. This is the case for Multicast and QTor visualizations, as the first one asks each participant to work on both its queries and the source's original stream diffusion; while the second has participants joining different communities to help.

The internal window called "System" provides information about the average depth in each ran system, and allow to select a given participant. This opens another internal window displaying its detailed situation. It is thus possible to read the currently expressed queries and the related tasks (rewritten query to perform, parents from which acquire the data, and children to send the results). Those windows also allow to express new queries, withdraw existing ones, or change capacities.

In the real application, system events are dynamically shown (which cannot appear in the figures). New participants appears once they have express a query to the tracker; and disappear when all their queries are withdrawn. During the data dissemination, links between the participants are successively highlighted, allowing to evaluate the latency.

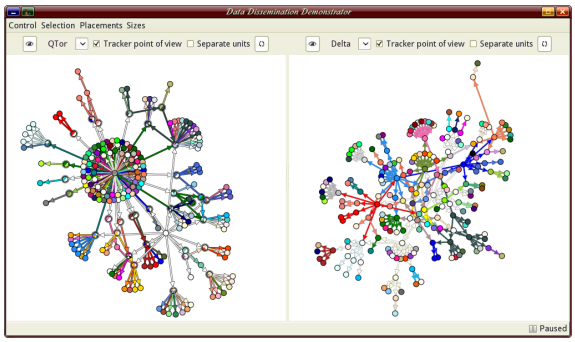


Figure 2: An example of scenario 3

Figure 2 presents an instance of the third scenario described above: a participant is inserted in the system with huge capacities, but with a complex query that makes it logical place far away from the source.

In the left panel, the QTor approach allows this participant to join several ancestor communities in order to perform a lot of processing and diffusion. There so is a lot of other participants (expressing distinct queries) that gets their input streams from it. In the right panel, the Delta approach cannot benefit from its strength, due to the lack of relevancy relations from its original query to the other participants' ones.

SPARQLGX : Une Solution Distribuée pour RDF Traduisant SPARQL vers Spark

Damien Graux
Inria, France
damien.graux@inria.fr

Louis Jachiet
Inria, France
louis.jachiet@inria.fr

Pierre Genevès
Cnrs, France
pierre.geneves@cnrs.fr

Nabil Layaïda
Inria, France
nabil.layaida@inria.fr

ABSTRACT

sparql est un langage de requête standardisé par le w3c permettant d'interroger des données exprimées au format rdf (*Resource Description Framework*). Avec l'augmentation des volumes de données rdf disponibles, de nombreux efforts de recherche ont été faits pour permettre l'évaluation distribuée et efficace de requêtes sparql. Dans ce contexte, nous proposons et partageons sparqlgx : notre solution de stockage rdf distribuée utilisant Apache Spark pour évaluer des requêtes sparql et stockant les données via des infrastructures Hadoop (hdfs). sparqlgx repose sur un traducteur de requêtes sparql vers une séquence d'instructions exécutables par Spark en adoptant des stratégies d'évaluation selon (1) le schéma de stockage des données utilisé et (2) des statistiques sur les données. Nous montrons que sparqlgx permet l'évaluation de requêtes sparql sur plusieurs milliards de triplets rdf répartis sur plusieurs nœuds. Nous comparons aussi sparqlgx à d'autres solutions issues de l'état-de-l'art. Nous démontrons ainsi les performances obtenues en permettant aux participants de reproduire par eux-mêmes les résultats présentés grâce à différents scénarios mettant directement en compétition plusieurs solutions de l'état-de-l'art. Nous montrons dans ce travail que tout en ayant une architecture relativement simple, sparqlgx représente une alternative intéressante dans de nombreux cas d'utilisation.

1. INTRODUCTION

sparql is the standard query language for retrieving and manipulating data represented in Resource Description Framework (rdf) [7]. sparql constitutes one key technology of the semantic web and has become very popular since it became an official w3c recommendation, first in 2008 [1] and then in 2013 as an improved version [2]. In this study, we investigate the problem of evaluating the conjunctive sparql queries *i.e.* the Basic Graph Pattern fragment.

The construction of efficient sparql query evaluators faces several challenges. First, rdf datasets are increasingly large, with some already containing more than a billion triples. To handle efficiently this growing amount of data, we need systems to be distributed and to scale. Furthermore, semantic data often have the characteristic of being dynamic (frequently updated). Thus being able to answer quickly after a change in the input data constitutes a very desirable property for a sparql evaluator.

In this context, we propose sparqlgx: an engine designed to evaluate sparql queries based on Spark [11]: a MapReduce-like data-parallel engine. sparqlgx relies on a compiler of sparql conjunctive queries which generates Scala code that is executed by the Spark infrastructure.

2. SPARQLGX: ARCHITECTURE

Data Storage Model. In order to process rdf datasets, we adopt the vertical partitioning approach introduced by Abadi *et al.* in [3] which stores a triple ($s\ p\ o$) in a file named p whose contents keeps only s and o entries. Converting rdf data into a vertically partitioned dataset is straightforward while (1) tending to minimize the memory footprint and the datasets size on disks and (2) reducing response time when queries have bounded predicates since searches are limited to the relevant files.

Compilation of SPARQL BGPs. Conjunctions of triple patterns (tps) are translated in terms of primitives of the Apache Spark framework expressed in Scala code. Each bgp is first translated in terms of a list of filters, which are then joined based on common variables.

SPARQL Fragment Extension. Moreover, we support the selection of distinguished variables in addition of the "SELECT *" form. We also support sparql solution modifiers *e.g.* removing duplicates with DISTINCT, sorting with ORDER BY, returning only few answers with LIMIT.

Furthermore, we also easily translate two additional sparql keywords: UNION and OPTIONAL, provided they are located at top-level in the WHERE clauses. Indeed, Spark allows to aggregate sets having similar structures with `union` and is also able to add data if possible with `leftOuterJoin`. Thus sparqlgx natively supports a slight extension (unions and optionals both at top level) of the extensively studied sparql fragment made of conjunctions of tps.

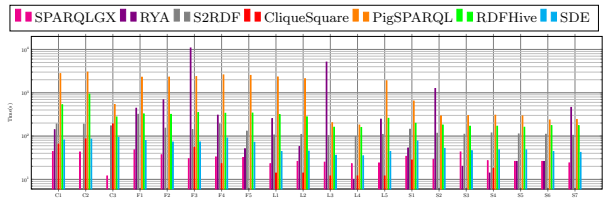


Figure 1: Query Response Times with WatDiv1k.

Optimized Joins With Statistics. The evaluation process (using Spark) first evaluates TPs and then joins these subsets according to their common variables; thus, minimizing the intermediate set sizes involved in the join process reduces evaluation time (since communication between workers is then faster). Thereby, statistics on data and information on intermediate results sizes provide useful information that we exploit for optimisation purposes.

Thereby, to rank each TP, we compute statistics on datasets counting all the distinct subjects, predicates and objects. This is implemented in a compile-time module that sorts TPs in ascending order of their selectivities before they are translated.

Finally, we also want to avoid cartesian products. Given an ordered list l of TPs we compute a new list l' by repeating the following procedure: remove from l and append to l' the first TP that shares a variable with a TP of l'. If no such TP exists, we take the first.

S-DE: SPARQLGX as a Direct Evaluator. In certain situations, queried data might be subject to updates; in others users might only need to evaluate a single query once (for data cleaning purposes for instance). In such cases, it is interesting to limit as much as possible both the preprocessing time and the query evaluation time. For this purpose, SPARQLGX provides a specific tool, called SDE, capable of directly evaluating SPARQL queries without preprocessing.

3. DEMONSTRATION DETAILS

We report on our experimental comparisons of SPARQLGX against other open source HDFS-based distributed RDF systems such as PigSPARQL [9], RYA [8], CliqueSquare [5], S2RDF [10] and RDFHive. Furthermore, we consider several datasets coming from two popular BGP benchmarks: LUBM [6] and WatDiv [4].

We present in Figure 1 the response times obtained with WatDiv1k (about 100 million triples). This illustrates that, for this dataset: (1) SDE always outperforms other tested "direct evaluators" (*i.e.* PigSPARQL and RDFHive); (2) SPARQLGX is able to answer all the queries unlike RYA and CliqueSquare; (3) it shares with CliqueSquare the same order of magnitude for linear queries; (4) it outperforms its competitors on complex queries. To further illustrate the performance of SPARQLGX, we provide an interactive GUI. Attendees can interact directly with our engines (*i.e.* SPARQLGX and SDE) and test them against state-of-the-art solutions.

4. CONCLUSION

SPARQLGX outperforms several related implementations in many cases, while implementing a simple architecture exclusively built on top of open source and publicly available technologies. The SPARQLGX implementation is available from:

`http://github.com/tyrex-team/sparqlgx`

Related Publications

- **SPARQLGX: Efficient Distributed Evaluation of SPARQL with Apache Spark.**
  D. Graux, L. Jachiet, P. Genevès, N. Layaïda. *The 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II, pages 80,87.*

- **SPARQLGX in action: Efficient Distributed Evaluation of SPARQL with Apache Spark.**
  D. Graux, L. Jachiet, P. Genevès, N. Layaïda. *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference, Kobe, Japan, October 19, 2016.*

5. REFERENCES

[1] SPARQL query language for RDF, January 2008. www.w3.org/TR/rdf-sparql-query/.

[2] SPARQL 1.1 overview, March 2013. http://www.w3.org/TR/sparql11-overview/.

[3] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, pages 411–422. VLDB Endowment, 2007.

[4] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of RDF data management systems. In *ISWC*, pages 197–212. Springer, 2014.

[5] F. Goasdoué, Z. Kaoudi, I. Manolescu, J.-A. Quiané-Ruiz, and S. Zampetakis. Cliquesquare: Flat plans for massively parallel RDF queries. In *ICDE*, pages 771–782. IEEE, 2015.

[6] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.

[7] P. Hayes and B. McBride. RDF semantics. *W3C Rec.*, 2004.

[8] R. Punnoose, A. Crainiceanu, and D. Rapp. Rya: a scalable RDF triple store for the clouds. In *International Workshop on Cloud Intelligence*, page 4. ACM, 2012.

[9] A. Schätzle, M. Przyjaciel-Zablocki, and G. Lausen. PigSPARQL: Mapping SPARQL to pig latin. In *Proceedings of the International Workshop on Semantic Web Information Management*, page 4. ACM, 2011.

[10] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, and G. Lausen. S2RDF: RDF querying with SPARQL on spark. *VLDB*, pages 804–815, 2016.

[11] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2. USENIX Association, 2012.

Exploring the evolution of science through interactive phylomemetic topic maps

Samuel Castillo
ISC-PIF, CNRS, UPS 3611
Sorbonne Universités, UPMC
Univ Paris 06, UMR 7606,
LIP6 Paris, France
samuel.castillo@iscpif.fr

Hubert Naacke
Sorbonne Universités, UPMC
Univ Paris 06, UMR 7606,
LIP6
Paris, France
hubert.naacke@lip6.fr

Bernd Amann
Sorbonne Universités, UPMC
Univ Paris 06, UMR 7606,
LIP6
Paris, France
bernd.amann@lip6.fr

David Chavalarias
ISC-PIF, CNRS, UPS 3611
EHESS, CAMS, UMR 8557
Paris, France
david.chavalarias@iscpif.fr

ABSTRACT

We present a web-platform[1] for building, customizing and exploring *phylomemies*, introduced by [1, 2], for the synthetic representation of the evolution of scientific topics appearing in large document corpora. Phylomemies are built using computationally expensive data-centric workflows composed of complex text and graph mining steps. For producing representative and high-quality phylomemies experts generally have to compute several instances using different parameters until they get an optimal result. Our prototype is focused on increasing interactivity and flexibility in the usage of these workflows by introducing parallelization and incremental computation. The implementation is based on a scalable parallel architecture and the demonstration will illustrate the interactive creation of phylomemies on a collection with over 20 million references to scientific publications.

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Database Management Systems [Textual databases]; H.3.6 [**Information Systems**]: Information Storage and retrieval Library Automation [Large text archives]; I.5.4 [**Computing Methodologies**]: Pattern Recognition Applications [Text processing]

General Terms

Large scale text analysis, graph data manipulation

Keywords

Incremental workflows, Pubmed, MeSH

[1] http://github.com/moma/Phylum

BDA 2016, 15 au 18 Novembre, Poitiers, France.

1. INTRODUCTION

The evolution of scientific knowledge is directly related to the history of humanity. Electronic archives and bibliographic resources like *Pubmed* [2] or *Web of Science* [3] are valuable sources for the analysis and reconstruction of this evolution. The reconstruction and exploration of the temporal evolution of scientific domains is important for many actors like (1) philosophers and historians of science who need to test their theories with data, (2) scientists who want to position themselves in their field, (3) policy makers who want to spot emergent fields, foster innovation and get key indicators to assist them in decision-making processes, (4) industry that have to find its ways through the scientific production and evaluate the potential for innovation and technological transfer and (5) librarians who need to propose classifications of documents which respect not only scientific topics hierarchy but also the evolution of ideas. The proposed demonstration builds on the concept of phylomemies (analogous to phylogenetic networks of natural species)[1, 2] for the synthetic representation of the evolution of scientific topics appearing in large document corpora. Figure 1 shows an example of a phylomemy concerning eleven scientific domain branches (*Hair cells, inner ear*, *Fatty Acids*, ..., *Carcinoma*). Each phylomemy covers the period between 1990 and 2010 partitioned into one year time-slices. The right part of the figure zooms into the phylmomemy of *Gap junctions*: each box (node) corresponds to a topic at a given time period and is potentially connected with one or more topics of the next time period illustrating some evolution in time.

The generation of phylomemies is a difficult task including complex and costly unsupervised text and graph mining algorithms (see Section 2). Producing a representative and high-quality phylomemy includes the interactive and iterative tuning of several mining parameters and data inputs where the expert computes a phylomemy, analyzes it, changes some parameters to compute a new phylomemy and so on. Current implementations are based on static batch-oriented mining workflows where tuning implies the reprocessing of the entire workflow after each modification. This makes the process of building high-quality phylomemies a cumbersome and time-consuming task, in particular over large document collections

[2] http://www.ncbi.nlm.nih.gov/pubmed
[3] https://webofknowledge.com/

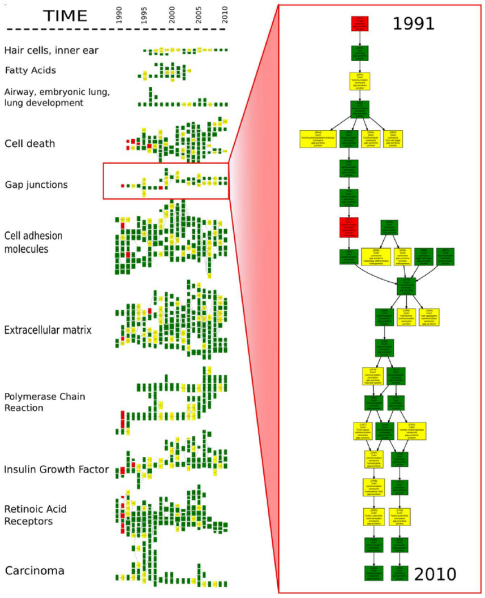


Figure 1: Phylomemy example (source : [1]

.

such as those mentioned before.

In this demonstration we will explore solutions for achieving more interactive phylomemy workflows over large data sets. These solutions apply two standard optimization techniques, *i.e.*, parallelization for exploiting modern distributed computation infrastructures and memoïzation for avoiding repeated computation. We will show that it is possible to achieve high interactivity by applying these techniques to the different workflow steps. More precisely, we will present the following contributions:

- A formal study of the cases and conditions that enable to isolate the necessary data fragments that may alter the existing phylomemy and to recompute only the fragments that are subject to change.
- A demonstrator implementation of two realistic science exploration scenarios over a large bibliographic dataset.

2. φ-WORKFLOWS

Let D be a document collection, L be a set of terms, t denotes a time period of granularity k (*e.g.*, $k = 1$ year and $t = 1996/01/01 - 1996/12/31$) and $D^t \subseteq D$ denotes the document collection corresponding to this time period. Observe that time periods might overlap. The indexing step generates a global inverted term-document index $I \subseteq L \times D$ in order to compute topics for each time-period t. The topic alignment step finally connects topics from different periods by alignment links (fusion/split). In the following, we denote by φ an existing phylomemy workflow execution and φ' a new variation of φ obtained by one or several customization steps.

The main phylomemy workflow, denoted φ-*workflow*, combines a set of tools based on scalable text and graph mining algorithms. As shown on Figure 2, the workflow is composed of four steps which can be performed in different ways by choosing different algorithms and parameters. Any combination of these parameters yields a different workflow instance where each instance is an interesting candidate for the end user, depending on his/her exploration goal.

1. The *Indexation* step transforms a corpora of text documents into an inverted term/document index. In this step, a list of terms produced by NLP techniques and/or human experts is used to index a selected document corpus in order to generate the term-document relations necessary for computing the term-term co-occurrence relations. We explore three alternatives for indexing documents by (1) keywords extracted from document titles and abstracts, (2) by keywords defined by the document authors and (3) by keywords from a controlled vocabulary (*e.g.*, MeSH thesaurus). These inverted indexes are divided into n sub-indexes corresponding to n (disjoint or overlapping) time periods of the input document corpora (this step is not shown in the figure).

2. The *Proximity computation* step computes for each time period an *undirected* term proximity graph. Following the available options for proximity in [1], the semantic distance between two terms i and j can be defined by the proximity index $d_p(i,j) = \frac{n_{ij}^2}{n_i * n_j}$, where $n_{\{x\}}$ is the number of publications mentioning $\{x\}$, where $\{x\}$ can be single term or a pair of terms. However, in this demo we do not use this proximity step because we don't consider dyadic relations between terms; instead we use the *Frequent Pattern Growth (FP-Growth)* approach for the *topic detection* step.

3. The *Topic detection* step consists in the detection of sets (clusters) of strongly semantically related terms. Given the goal of computing phylomemies in a flexible way, we consider the frequent itemset mining and specifically the *Frequent Pattern Growth (FP-Growth)*[5] algorithm, that helps in finding the frequent itemsets patterns across a bunch of transactions; for the phylomemy-context we can consider the terms of each publication as one *transaction* , where each item of the transaction represents a term. The FP-growth result will be a list of frequent itemsets ordered (*desc*) by their frequencies (the *term clusters*).

4. The *Topic alignment* step matches topics (term clusters) from different time intervals by using a set distance function to generate alignments representing the temporal evolution of topics. There are many possible configurations, starting from the simplest scenario comparing only topics of one period and the following period and generating an alignment between the most similar topics or all topics above a given threshold. More complex scenarios, might compare topics of more distant periods and generate complex alignments merging and splitting topics of different periods. This step is defined by:
   - The set distance function d_s between two topics (*e.g.*, Jaccard similarity coefficient).
   - The alignment similarity threshold δ_s for filtering all alignments with a lower similarity.
   - Whether the union of two topics detects merge or split alignments [4]. Two topics C_1 and C_2 at time t *merge* into topic C at time $t+1$ if the union of C_1 and C_2 is closer to C than any single topic. Inversely, a topic C at time t *splits* into topics C_1 and C_2 at time $t+1$ if the union of C_1 and C_2 is closer to C than any single topic.

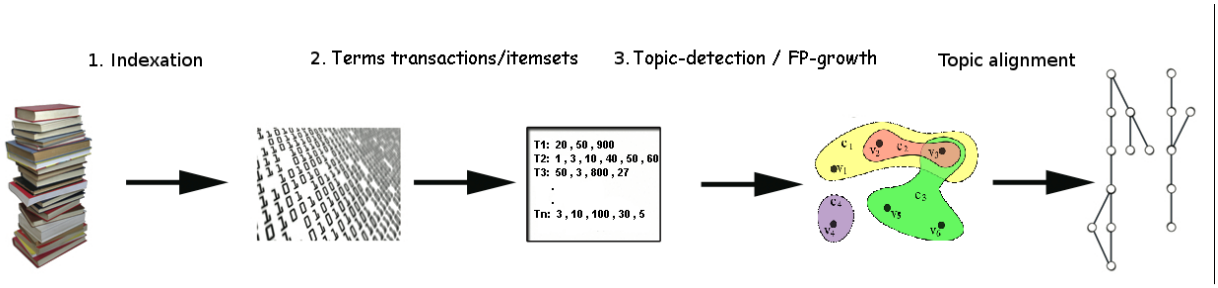


Figure 2: The φ-workflow which output is a phylomemy

- The aligment period threshold δ_t filtering all alignments between topics from two periods A and B where $B - A \leq \delta_t$. In our example in Section 1, $\delta_t = 1$.

Because of all these alternatives, the φ-workflow requires flexibility. End-users need to interactively customize the *φ-workflow* by changing data (*e.g.*, removing or adding a term in a topic) and parameters (*e.g.*, thresholds, time intervals, ...). Enabling interactivity is challenging and raises the problem of reusing existing intermediary data to save computation. In particular, we have to answer the following questions:

- Given a user demand for modifying some data or parameters of the current workflow, which steps of the workflow are impacted by that change?
- For the steps that require recomputation, which part of the current output remains unchanged, and which part requires computation?

For example, modifying the distance function of the Topic alignment (step 4 on Figure 2) implies to recompute only the last step, leaving unchanged the already detected topics. Removing a term from a topic in an existing phylomemy implies its removal from terms-clusters and the recomputation of steps 3 and 4 without recomputing step 1 and 2. On the other hand, adding new terms implies to append new occurrence information into the input of step 1, and consequently to recompute all four steps of the workflow. The entire workflow also requires reprocessing when the user asks for changing the time-window size. However, in the previous examples, each step might reuse intermediary results from previous computations and only recompute and update fragments of the previous result. For example, adding a term only will change a small fragment of the proximity graph and consequently simplify the topic detection and topic alignment step which have to consider only a small part of the initial data.

3. FORMAL MODEL AND ALGORITHMS

To achieve interactive φ-workflows means to be able to visualize in "real-time" the effect of modifications of any workflow parameters. However, as discussed in the previous section, such modifications might trigger a cascade of recomputations.

3.1 Initial phylomemy pre-computation φ

The main approach for achieving better performance is to precompute a first complete phylomemy φ that can be customized incrementally afterwards[4].

[4]Our computation model can be adapted to more complex scenarios reusing any final or intermediate results with known parameters.

We suppose that there exists a global inverted index I over the whole document corpus. In addition we define a minimal granularity k depending on the corpus and application; for example, for news, k might be one day, whereas for scientific articles, k might be one trimester or one year. The topic detection step then identifies for each time period all topics using the *FP-growth*, where each frequent itemset represents a topic. As explained before, the topic alignment includes a variety of parameters which allow to customize the final result. Initializing the alignment step consists in pre-computing a complete alignment lattice φ of all (equivalence, merge and split) alignments between topics using a fixed lower alignment similarity bound $\delta_s > 0$ and upper period bound $\delta_p > 1$. Let c represents the average number of clusters of a given period, then the computational complexity of an alignment step for one period rises to $O(c^2 * \delta_p^2)$. We use a parallel distributed computing framework for generating this alignment graph and store all intermediary and final results in a data store for reuse during the following customization steps.

3.2 Phylomemy customization

Customize thresholds δ_s and δ_p.

Increasing the lower topic similarity distance threshold $\delta'_s = \beta\delta_s$ where $\beta \geq 1$ mainly results in filtering out all edges in the alignment graph φ with weight $w < \beta\delta_s$. Decreasing the topic similarity ($\beta < 1$) needs the generation of new alignments by applying the alignment algorithm. The inverse observation holds for the period threshold $\delta'_p = \gamma\delta_p$ where alignments are filtered for $\gamma \leq 1$ and new alignments have to be generated for $\gamma > 1$. This generation can be done incrementally by comparing only clusters in periods at distance in interval $(\delta_p, \gamma\delta_p)$.

Customize terminology L.

Experts also can add and remove terms from the initial terminology L. This mainly means that the inverted index is updated correspondingly. Whereas term removal mainly consists in removing the corresponding graph nodes, adding new terms needs the local recomputation of terms clusters. Alignment links have to be regenerated for all new topics by applying the alignment method defined before. The removal of obsolete topics has to be considered more carefully since this needs reconsidering the alignments in the neighborhood of the removed topic depending on the alignment type (a detailed discussion is outside the scope of this article).

4. PROTOTYPE AND SCENARIOS

In this demonstration we focus on the Pubmed scientific corpus consisting of 20.539.068 meta data (titles, abstracts, authors, etc.) of biomedical papers, life science journals, and online books from

1982 to 2014. This corpus is open and downloadable via *Entrez API* [7]. This dataset is indexed by the following three vocabularies:

- **Author-keywords** correspond to keywords declared manually by the publication authors. More precisely, we extract this list of keywords (present in less than 50% of the total publications) and use it to re-index the complete Pubmed corpus.
- **N-grams** extracted from Pubmed via the Gargantext[5] [6] platform which focuses in automatic text-mining and semantic-network reconstruction.
- **MeSH subheadings**: Medical Subject Headings (MeSH) is a controlled hierarchical vocabulary or *thesaurus* used by the National Library of Medicine to index MEDLINE. In Pubmed, 93% of the MEDLINE publications have assigned one or more medical sub-headings and the term-document relation follows a power law distribution[3] where generic terms tend to appear a way more often than more specific terms. The hierarchical thesaurus structure enables the filtering of terms by their semantic preciseness reflected by their depth in the tree structure.

4.1 User interactions

The user can interact with the system through a web GUI using the following three main interface components.

Phylomemy Initialization.

This interface component allows to initialize all inputs and parameters for generating a new phylomemy:

- *Document selection:* The user is given the choice of selecting the initial documents of the corpus by writing one or more search terms. These terms define a full-text query evaluated by Lucene[6] over all titles and abstracts of Pubmed collection. Additionally, the interface displays the distribution chart of publications over time with a corresponding slider for selecting documents within a specific time-range.
- *Term selection:* User picks one of the three available vocabularies: *author-keywords*, *n-grams* and *MeSH*. If MeSH is selected, the query-mode changes and shows a tree with selectable / expandable items that represent the headings / subheadings of MeSH. Finally with the retrieved *{doc,terms}* relations, the user can set up the initial parameters of the φ-workflow.
- *Parametrization:* The user sets up the initial parameters (as shown in figure 3) which are minimal frequency of terms, threshold for semantic distances and threshold for temporal topic alignment. There are also two parameters for the frequent itemsets mining task: the minimum support specifies the minimum number of publications per terms-cluster (default: 0.0001) and the minimum itemset-size specifies the minimum number of items of each terms-cluster (default: 4).

Phylomemy exploration and topic highlighting.

This component displays a phylomemy and the user can interactively highlight all topics on a given branch (or sub-graph) or containing some user defined search terms. The user can also increase the alignment threshold and visualize a less dense phylomemy which only contains links between highly similar topics.

[5]http://iscpif.fr/projects/gargantext

[6]http://lucene.apache.org

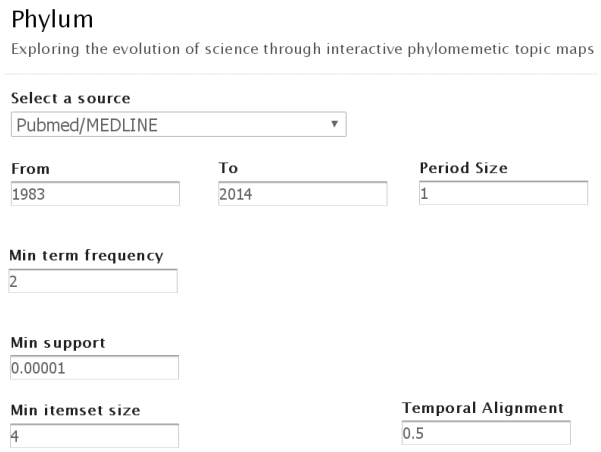


Figure 3: Main form and parameters - web application

.

Phylomemy customization.

In this component, the user interactively customizes the existing phylomemy by :

- *modifying the input vocabulary* by manually adding / removing some terms, by exploring the given topic phylomemy (add / remove all terms of a given set of highlighted topics) or by exploring one of the input vocabularies. In particular, she can exploit the hierarchical MeSH structure to automatically select more specific or more general terms of a given term etc.
- *changing the other input parameters* as described in section 3 and as shown in figure 3.

4.2 Scenarios

We illustrate what the audience will see during the demo session and the basic scenarios inspired from science exploration use-cases.

Scenario 1: The user sees a first precomputed phylomemy only based on *MeSH* $(term, doc)$ occurrences. Via frontend, the user can explore, zoom-in-out, select clusters, search for terms inside clusters, highlight branches and more important, to change the parameters shown in figure 3. This will allow to better understand certain complex (split/merge) alignments which will disappear by these modifications.

Scenario 2: The user can alter the phylomemy-thematic and completely change the initial query by selecting a MeSH sub-domain A, compute this phylomemy on-the-fly and then select a second sub-domain B that'll be added/overlayed to the existing phylo A. This scenario deals with the desired flexibility for phylomemies computation; in this case we use the Union operator applied to two different phylomemies computed separately, phylo A and B, conserving initial parameters and finally computing the jaccard proximity but now between the clusters of phylo A and B.

4.3 Implementation

Architecture.

We designed a three tiered web application, where each tier communicate with the others via HTTP protocol and REST messages. See Figure 4:

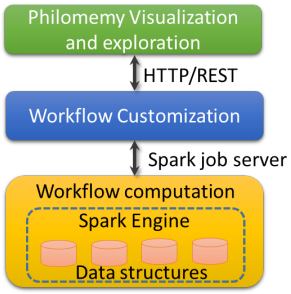


Figure 4: Architecture

.

- **Web User Interface**: The frontend (see Figure 5) provides visualization and exploration features for phylomemies, using mainly *TinawebJS* - a bipartite graph explorer written in Javascript (see figure 1). It uses Ajax to encapsulate the communication with the middle layer.

- **Application server**: It controls the workflow, on user demands. As shown in figure 3, it adapts the workflow when users change some parameters. It uses generic data transformation operators (*e.g.*, map, filter) provided by Apache Spark to drive the customization of the current workflow. We use Spark-jobserver[7] for communication between this layer and the bottom layer. It is a RESTful interface for creating and submitting Apache Spark jobs remotely.

- **Backend data server**: We use the Spark computing engine to store and manage the global data structures in main memory: $\{doc, date, terms\}$ and $\{term, label\}$ - approx. 6 million MeSH-terms. The intermediate data structures also reside in this layer: one $\{doc, terms\}$ collection per period of time, as well as the $\{term, frequency\}$, $\{cluster, terms\}$ and $\{cluster, support, cluster\_size\}$ datasets.

.

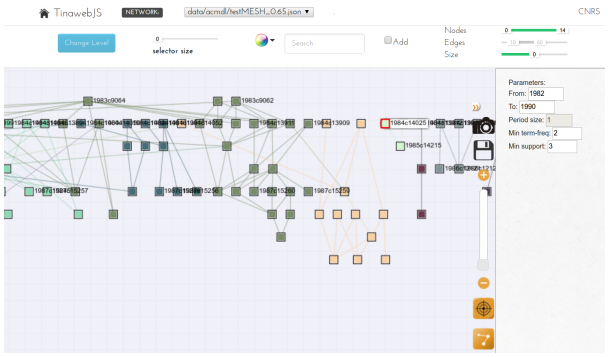


Figure 5: Screenshot web application

.

Enviroment.

For the three applications we use the same web server (8 cores, 16GB RAM and 1TB hardrive) . The web server configuration was setted up with *nginx*. In particular, the spark-jobserver-app interacts with the grid via Apache Spark 2.0 and over 220 cores on the LIP6 cluster (10 machines, 22 cores each) which allows sharing in-memory data without any conversion overhead, as well as higher parallelism.

The code, documentation (architecture and software design) and URL-application are located at :

http://github.com/moma/Phylum

[7] github.com/spark-jobserver/spark-jobserver

5. CONCLUSION

We have designed and implemented a flexible and interactive workflow for automatic reconstruction of science evolution based in phylomemetic modeling using a scalable framework for processing large-scale corpora such as PubMed (20 million publications). Our parallel and incremental strategies avoid a complete execution of φ-workflows after customization.

As future challenges we expect to transform this descriptive model of science-evolution to a predictive model, where we would precise additional information such as authors, publications-sources and citations, for answering research questions about phenomenology and socio-semantic patterns of scientific collaboration networks.

6. REFERENCES

[1] D. Chavalarias and J.-P. Cointet. The reconstruction of science phylogeny. *arXiv preprint arXiv:0904.3154*, 2009.

[2] D. Chavalarias and J.-P. Cointet. Phylomemetic patterns in science evolution—the rise and fall of scientific fields. *PloS one*, 8(2):e54847, 2013.

[3] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[4] A. Guichard. Analyse temporelle de l'évolution de domaines scientifiques par alignement de clusters sémantiques. Master's thesis, UPMC - LIP6, Paris, France, 7 2013.

[5] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

[6] C. Meadel. Les controverses comme apprentissage. *Hermès, La Revue*, (3):45–50, 2015.

[7] E. Sayers. Entrez programming utilities help. *URL http://www.ncbi.nlm.nih.gov/books/NBK25499*, 2013.