Actes de la conférence BDA 2014 : Gestion de données - principes, technologies et applications

David Gross-Amblard, Christine Collet, Christophe Bobineau, Fabrice Jouanot

BDA 2014

Gestion de Données – Principes, Technologies et Applications

30e anniversaire
14-17 octobre 2014, Grenoble-Autrans

Actes de la conférence BDA 2014
Conférence soutenue par l'Université Joseph Fourier, Grenoble INP, le CNRS et le laboratoire LIG

Site de la conférence : `http://bda2014.imag.fr`
Actes en ligne : `https://hal.inria.fr/BDA2014`

Message des organisateurs

Cette année notre conférence a fêté ses 30 ans d'existence à Autrans dans le massif du Vercors, à quelques kilomètres de son lieu de naissance, le massif de Chartreuse (BDA 1985).

Cette conférence a illustré les changements radicaux à l'œuvre dans notre domaine : hyper-massification des données, diversification des supports de calcul (de la carte à puce jusqu'au nuage), renouveau des usages (Web, réseaux sociaux, Internet des objets, capteurs personnels, ...). En particulier, cette édition a été l'occasion de réfléchir sur l'évolution de notre activité de recherche.

Reprenant les éléments innovants de l'édition précédente, le programme a comporté trois conférences invitées, deux ateliers, des articles originaux, des présentations d'articles internationaux, des démonstrations de prototypes, des présentations de doctorant-e-s et une séance "réseautage" pour monter des partenariats, s'attaquer à un problème difficile, ou présenter des offres/recherches de postes.

À l'occasion de cet anniversaire, les conditions de soumission ont été modifiées : retour des actes avec indexation, afin de donner plus de visibilité à des travaux originaux ; possibilité de présenter des papiers déjà acceptés dans une grande conférence internationale, pour en faire profiter la communauté (cette catégorie de papiers a été limitée en nombre et ne figure naturellement pas dans les actes).

Les organisateurs souhaitent remercier chaleureusement nos différents soutiens, nos conférenciers invités, les auteurs, ainsi que les nombreux participants de cette édition anniversaire.

David Gross-Amblard (président du comité de programme)
Christine Collet (présidente des journées)
Christophe Bobineau (co-président du comité d'organisation)
Fabrice Jouanot (co-président du comité d'organisation)

Table des matières

1 Comités de BDA 2014

Présidente des Journées
Christine Collet, LIG/Institut Polytechnique de Grenoble

Président du comité de Programme
David Gross-Amblard, IRISA, Université de Rennes 1

Président du comité des démonstrations
Romuald Thion, LIRIS/Université Lyon 1

Présidents du comité d'organisation
Christophe Bobineau, LIG/Institut Polytechnique de Grenoble
Fabrice Jouanot, LIG, Université Joseph Fourier Grenoble 1

Comité d'organisation
Front Agnès, LIG/Université Pierre Mendès-France Grenoble 2
Labbé Cyril, LIG/Université Joseph Fourier Grenoble 1

Comité de Programme
Akbarinia Reza,INRIA
Amann Bernd, LIP6/UPMC
Aubourg Eric, Soleb
Bellatreche Ladjel, LIAS/ENSMA
Bidoit Nicole, LRI/Université Paris 11
Bonifati Angela, INRIA/Université de Lille 1
Bouganim Luc, INRIA/UVSQ
Boulicaut Jean-François, INSA de Lyon
Cautis Bogdan, Telecom ParisTech
Constantin Camelia, LIP6/UPMC
Defude Bruno, Telecom Sud'Paris
Defude Bruno, Telecom Sud'Paris
Delot Thierry, LAMIH/INRIA/Université de Valenciennes
Gallinari Patrick, LIP6/UPMC
Gheerbrant Amélie, LIAFA/CNRS/Université Paris Diderot
Goasdoué François, Université de Rennes 1
Jaudoin Hélène, IRISA/ENSSAT
Labbé Cyril, LIG/Université de Grenoble Alpes
Lesueur François, LIRIS/INSA de Lyon
Lumineau Nicolas, LIRIS/Université de Lyon 1
Maabout Sofiane, LaBRI/Université de Bordeaux
Manolescu Ioana, INRIA/LRI/Université Paris Sud
Miklos Zoltan, IRISA/Université de Rennes 1
Molli Pascal, LINA/Université de Nantes
Nguyen Kim, LRI/Université Paris Sud
Petit Jean-Marc, LIRIS/INSA de Lyon
Pucheral Philippe, INRIA/UVSQ
Rashia Guillaume, LINA/Université de Nantes
Rigaux Philippe, CNAM/Internet Memory Research
Senellart Pierre, LTCI/Institut Mines-Telecom/Telecom ParisTech
Serrano-Alvarado Patricia, LINA/Université de Nantes

Termier Alexandre, LIG/Université Joseph Fourier Grenoble 1
Teste Olivier, IRIT/Université Toulouse 2 Le Mirail
Toumani Farouk, LIMOS/Université Blaise Pascal Clermont-Ferrand
Vargas-Solar Genoveva, LIG/CNRS
Vodislav Dan ETIS/CNRS/Université de Cergy-Pontoise
Zeitouni Karine, PRISM/UVSQ
Zurfluh Gilles, IRIT/Université de Toulouse 1 Capitole

Comité des démonstrations

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

Vérification des plagiats

Pierre Senellart, Telecom Paris'Tech

Edition des actes

Camélia Constantin, LIP6, UPMC
David Gross-Amblard, IRISA/Université de Rennes 1
Anne Jaigu, IRISA

2 Conférenciers invités

2.1 Big Data Integration
Divesh Srivastava

AT&T Labs, ACM Fellow, Board of trustees VLDB endowment
http://www2.research.att.com/~divesh/
Divesh Srivastava is the head of Database Research at AT&T Labs-Research. He is an ACM fellow, on the board of trustees of the VLDB Endowment, the managing editor of the Proceedings of the VLDB Endowment (PVLDB) and an associate editor of the ACM Transactions on Database Systems (TODS). His research interests and publications span a variety of topics in data management.

Abstract : The Big Data era is upon us : data is being generated, collected and analyzed at an unprecedented scale, and data-driven decision making is sweeping through all aspects of society. Since the value of data explodes when it can be linked and fused with other data, addressing the big data integration (BDI) challenge is critical to realizing the promise of Big Data. BDI differs from traditional data integration in many dimensions : (i) the number of data sources, even for a single domain, has grown to be in the tens of thousands, (ii) many of the data sources are very dynamic, as a huge amount of newly collected data are continuously made available, (iii) the data sources are extremely heterogeneous in their structure, with considerable variety even for substantially similar entities, and (iv) the data sources are of widely differing qualities, with significant differences in the coverage, accuracy and timeliness of data provided. This talk explores the progress that has been made by the data integration community in addressing these novel challenges faced by big data integration, and identifies a range of open problems for the community.

2.2 Big Data : Hype and Reality
C. Mohan

IBM Almaden Research Center
http://bit.ly/CMohan
Dr. C. Mohan has been an IBM researcher for 32 years in the information management area, impacting numerous IBM and non-IBM products, the research and academic communities, and standards, especially with his invention of the ARIES family of locking and recovery algorithms, and the Presumed Abort commit protocol. This IBM, ACM and IEEE Fellow has also served as the IBM India Chief Scientist. In addition to receiving the ACM SIGMOD Innovation Award, the VLDB 10 Year Best Paper Award and numerous IBM awards, he has been elected to the US and Indian National Academies of Engineering, and has been named an IBM Master Inventor. This distinguished alumnus of IIT Madras received his PhD at the University of Texas at Austin. He is an inventor of 40 patents. He has served on the advisory board of IEEE Spectrum and on the IBM Software Group Architecture Board's Council. Mohan is a frequent speaker in North America, Western Europe and India, and has given talks in 40 countries. More information can be found in his home page at http ://bit.ly/CMohan

Abstract : Big Data has become a hot topic in the last few years in both industry and the research community. For the most part, these developments were initially triggered by the requirements of Web 2.0 companies. Both technical and non-technical issues have continued to fuel the rapid pace of developments in the Big Data space. Open source and non-traditional software entities have played key roles in the latter. As it always happens with any emerging technology, there is a fair amount of hype that accompanies the work being done in the name of Big Data. The set of clear-cut distinctions that were made initially between Big Data systems and traditional database management systems are being blurred as the needs of the broader set of ("real world") users and developers have come into sharper focus in the last couple of years. In this talk, I will survey the developments in Big Data and try to distill reality from the hype !

2.3 Declarative Modeling for Machine Learning and Data Mining
Luc De Raedt

Katholieke Universiteit Leuven, ECCAI fellow, http://people.cs.kuleuven.be/~luc.deraedt/

Abstract : Today, it remains a challenge to develop applications and software that incorporates data mining. One reason is that the field has focussed on developing high-performance algorithms for solving particular tasks rather than on developing general principles and techniques. I propose to alleviate these problems by applying the constraint programming methodology to machine learning and data mining and to specify data mining tasks as constraint satisfaction and optimization problems. What is essential is that the user be provided with a way to declaratively specify what the data mining problem is rather than having to outline how that solution needs to be computed. This corresponds to a model + solver- based approach to data mining, in which the user specifies the problem in a high level modeling language and the system automatically transforms such models into a format that can be used by a solver to efficiently generate a solution. This should be much easier for the user than having to implement or adapt an algorithm that computes a particular solution to a specific problem. I shall illustrate this perspective by presenting our work on developing models as well as modeling languages for several data mining tasks. I shall include our recent results on the MiningZinc language and system, an extension of the MiniZinc framework for constraint programming.

3 Ateliers

3.1 Comment publier : l'aventure d'Ike Antkare
Cyril Labbé

LIG Lab, Université Joseph Fourier, http://membres-lig.imag.fr/labbe/

Cyril Labbé est maître de conférences (HDR) de l'Université Joseph Fourier. Il effectue ses recherches au laboratoire d'Informatique de grenoble (LIG), celles-ci portent sur la gestion de données dans les grands systèmes d'information ainsi que sur la fouille de données textuelles. Il a effectué une thèse d'informatique dans le domaine de la simulation et de l'évaluation de performances pour les réseaux de télécommunication (1999). Il est diplômé en mathématiques appliquées (1995). Il a été chercheur à Orange Labs de 2000 à 2001 et chercheur invité de 2009 à 2010 à la "Faculty of Information Technology", Monash University, Melbourne, Australie.

3.2 Faire sa recherche aujourd'hui : un avis personnel
Serge Abiteboul

INRIA & ENS Cachan, Conseil national du numérique, http://abiteboul.com, http://abiteboul.blogspot.fr

Serge Abiteboul obtained his Ph.D. from the University of Southern California, and a State Doctoral Thesis from the University of Paris-Sud. He has been a researcher at the Institut National de Recherche en Informatique et Automatique since 1982 and is now Distinguished Affiliated Professor at Ecole Normale Supérieure de Cachan . He was a Lecturer at the École Polytechnique and Visiting Professor at Stanford and Oxford University. He has been Chair Professor at Collège de France in 2011-12 and Francqui Chair Professor at Namur University in 2012-2013. He co-founded the company Xyleme in 2000. Serge Abiteboul has received the ACM SIGMOD Innovation Award in 1998, the EADS Award from the French Academy of Sciences in 2007 ; the Milner Award from the Royal Society in 2013 ; and a European Research Council Fellowship (2008-2013). He became a member of the French Academy of Sciences in 2008, and a member the Academy of Europe in 2011. He is a member of the Conseil National du Numérique and Chairman of the Scientific Board of the Société d'Informatique de France. His research work focuses mainly on data, information and knowledge management, particularly on the Web. He founded and is an editor of the blog binaire.blogs.lemonde.fr.

Une algèbre floue pour l'interrogation flexible de bases de données graphes

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

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

Hélène Jaudoin
Université Rennes 1, IRISA
Lannion, France
Helene.Jaudoin@irisa.fr

Grégory Smits
Université Rennes 1, IRISA
Lannion, France
Gregory.Smits@irisa.fr

ABSTRACT

Cet article décrit une algèbre de requête floue adaptée à l'interrogation flexible de bases de données graphes. Cette algèbre, fondée sur la théorie des ensembles flous et sur la notion de graphe flou, se compose d'un ensemble d'opérateurs permettant de formuler des requêtes à préférences sur des objets de type graphe, flous ou non. Les préférences exprimables dans ce cadre peuvent concerner i) le contenu des nœuds du graphe et/ou ii) la structure du graphe (qui peut inclure des arcs pondérés quand le graphe est flou). De même que l'algèbre relationnelle constitue la base du langage SQL utilisé dans les systèmes commerciaux, l'algèbre floue proposée ici est destinée à servir de fondement à l'extension d'outils plus orientés utilisateur tels que le langage Cypher implanté dans le système Neo4j.

Keywords

Bases de données graphes, requêtes floues, algèbre, graphes flous

De nombreux travaux ont été consacrés à l'interrogation floue de bases de données *relationnelles*, voir par exemple [DP96, ZDDK08, PB12], qui ont débouché notamment sur une extension floue du langage SQL, nommée SQLf [BP95]. Cependant, bien que les bases de données relationnelles soient encore largement utilisées, en particulier pour les applications classiques de gestion dans les entreprises, le besoin de gérer des données plus complexes s'est fait sentir depuis déjà plusieurs décennies, et a conduit à l'émergence d'autres modèles de données. Ainsi, un concept a fait son apparition ces dernières années, et a attiré l'attention de nombreux chercheurs de la communauté des bases de données, celui de *base de données graphe* [GS02, HS08, DSUBGV$^+$10, VMZ$^+$10, Ang12, CAH12, BT12], dont la finalité première est de permettre une gestion efficace de réseaux d'entités où chaque nœud est décrit par un ensemble de caractéristiques (par exemple un ensemble d'attributs décrivant l'entité, mais une structure de données plus complexe peut aussi être associée à un nœud) et les arcs représentent les liens de différentes natures existant entre les entités. Un tel modèle de données a de nombreuses applications potentielles, et peut servir par exemple à représenter des réseaux sociaux, des données RDF [AG05], des bases de données cartographiques, des bases de données bibliographiques, etc.

Le schéma des données associé à une base de données graphe est généralement complexe à appréhender par les utilisateurs, ce qui les amène à écrire des requêtes "en aveugle", dont les réponses peuvent souvent s'avérer vides ou pléthoriques. Dans ce contexte, il est nécessaire de proposer à l'utilisateur des moyens d'interroger la base de données de façon flexible.

Les bases de données graphes offrent de nombreuses possibilités en termes d'interrogation flexible puisque deux aspects peuvent intervenir dans les préférences exprimables par un utilisateur : i) le contenu des nœuds et ii) la structure du graphe lui-même. Dans cet article, nous nous attachons à définir une algèbre de requête floue adaptée à ce type de base de données. Nos points de départ sont, d'une part, l'article [Yag13] de R.R. Yager dans lequel ce dernier recense un certain nombre de critères de recherche flexibles pertinents dans un tel contexte, sans toutefois entrer dans les détails de leur expression à l'aide d'un langage de requête formel, et l'article [HS08] de H. He et A.K. Singh dans lequel les auteurs proposent une algèbre permettant d'interroger (sans préférences ni gradualité d'aucune sorte) des bases de données graphes.

La suite de l'article est organisée de la façon suivante. La section 1 présente des notions de base sur les graphes flous et les requêtes à préférences floues. La section 2 décrit les éléments principaux pouvant intervenir dans une requête floue dans un contexte de base de données graphe (critères sur les nœuds, conditions graduelles sur la structure du graphe, connecteurs flous). La section 3 présente une algèbre de requête floue permettant d'exprimer des préférences sur une base de données représentant un graphe (ou un ensemble de graphes) classique(s) ou flou(s). Les travaux connexes les plus pertinents sont discutés en section 4. Finalement, la section 5 rappelle les contributions principales et esquisse quelques perspectives.

BDA 14 octobre 2014, Grenoble-Autrans, France.

1. NOTIONS DE BASE

1.1 Bases de données graphes

Un système de gestion de bases de données graphes (souvent appelé simplement et abusivement *base de données graphe* dans la littérature) permet de gérer des données telles que la structure du schéma et les instances sont modélisées par des graphes (ou des généralisations de ce concept) et la manipulation des données se fait au moyen d'opérations orientées graphe et de constructeurs de type [AG08]. Parmi les systèmes de ce genre, on peut citer AllegroGraph [All], InfiniteGraph [Inf], Neo4j [Neo] et Sparksee [Spa]. Il existe différent modèles de bases de données graphes (voir [AG08] pour une vue d'ensemble), notamment le *graphe d'attributs* (appelé aussi *graphe de propriétés*) permettant de modéliser un réseau d'entités encapsulant des données. Dans ce modèle, les nœuds représentent les entités et les arcs correspondent à des relations entre entités. La modélisation des nœuds et des arcs peut faire intervenir des attributs décrivant leurs propriétés. La figure 1 est un exemple de base de données graphe contenant des données provenant de DBLP[1].

1.2 Ensembles flous et graphes flous

1.2.1 Rappels sur les ensembles flous

La théorie des ensembles flous a été définie par L.A. Zadeh [Zad65] pour modéliser des classes d'objets aux frontières vagues. Pour de tels ensembles, la transition entre l'appartenance complète et la non-appartenance est graduelle plutôt que tranchée. Des exemples typiques de classes floues sont celles associées à des adjectifs du langage naturel tels que *jeune, bon marché, rapide*, etc. Formellement, un ensemble flou F défini sur un référentiel U est caractérisé par une fonction d'appartenance $\mu_F : U \to [0,1]$ où $\mu_F(u)$ désigne le degré d'appartenance de u à F. En particulier, $\mu_F(u) = 1$ reflète l'appartenance totale de u à F, tandis que $\mu_F(u) = 0$ exprime la non-appartenance absolue. Lorsque $0 < \mu_F(u) < 1$, on parle d'appartenance partielle.

Deux ensembles classiques présentent un intérêt particulier lorsque l'on définit un ensemble flou F:

- le noyau $C(F) = \{u \in U \mid \mu_F(u) = 1\}$, constitué des *prototypes* de F,
- le support $S(F) = \{u \in U \mid \mu_F(u) > 0\}$.

La notion de coupe de niveau, ou α-coupe englobe ces deux concepts. L'α-coupe (resp. α-coupe stricte) F_α (resp. $F_{\overline{\alpha}}$) d'un ensemble flou F correspond à l'ensemble des éléments du référentiel qui possèdent un degré d'appartenance à F au moins égal à (resp. strictement plus grand que) α :

$$F_\alpha = \{u \in U \mid \mu_F(u) \geq \alpha\} \text{ et } F_{\overline{\alpha}} = \{u \in U \mid \mu_F(u) > \alpha\}.$$

De façon directe, on a : $C(F) = F_1$ et $S(F) = F_{\overline{0}}$.

En pratique, la fonction d'appartenance associée à un ensemble flou F est souvent choisie de forme trapézoïdale. Ainsi, F peut se représenter par le quadruplet (A, B, a, b) où $C(F) = [A, B]$ et $S(F) = [A-a, B+b]$, voir la figure 2.

Soit F et G deux ensembles flous définis sur l'univers U. On dit que $F \subseteq G$ ssi $\mu_F(u) \leq \mu_G(u)$, $\forall u \in U$. Le complément de F, noté F^c, est défini par $\mu_{F^c}(u) = 1 - \mu_F(u)$. De plus, $F \cap G$ (resp. $F \cup G$) est défini de la façon suivante : $\mu_{F\cap G} = min(\mu_F(u), \mu_G(u))$ (resp. $\mu_{F\cup G} = max(\mu_F(u), \mu_G(u))$).

[1] `http://www.informatik.uni-trier.de/~ley/db/`

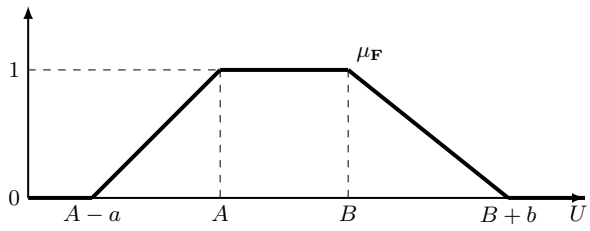


Figure 2: Fonction d'appartenance trapézoïdale

Comme dans le cas booléen, les contreparties logiques des opérateurs ensemblistes $\cap$, $\cup$ et la complémentation sont respectivement la conjonction $\wedge$, la disjonction $\vee$ et la négation $\neg$. Voir [DP00] pour de plus amples détails.

1.2.2 Graphes flous

Un graphe est un couple (V, R), où V est un ensemble et R est une relation sur V. Les éléments de V (resp. R) sont les nœuds (resp. arcs) du graphe. Une relation floue ρ sur un ensemble V peut être vue comme définissant un graphe pondéré, ou graphe flou [Ros75, MN00], où un arc $(x, y) \in V \times V$ a un *poids* ou *force* de valeur $\rho(x, y) \in [0, 1]$. Ce *degré* peut exprimer l'intensité de n'importe quelle relation graduelle entre deux nœuds.

REMARQUE 1. *Le graphe peut être flou au départ — c-à-d que la relation ρ est connue au départ — ou peut être issu d'un traitement le rendant flou. Un graphe flou peut modéliser des relations statiques ou dynamiques. Si un graphe flou modélise un réseau social de style Twitter et notamment les relations entre utilisateurs de ce réseau, $\rho(x, y)$ peut par exemple être défini en fonction du nombre d'articles de y que x a retransmis.*

La relation floue ρ peut être vue comme un sous-ensemble flou sur $V \times V$, ceci permettant d'utiliser les formalismes des ensembles flous [Yag13]. On peut ainsi dire que $\rho_1 \subseteq \rho_2$ si $\forall(x, y)$, $\rho_1(x, y) \leq \rho_2(x, y)$.

Quelques propriétés d'intérêt peuvent êtres associées aux relations floues, comme la réflexivité ($\rho(x, x) = 1$, $\forall x$), la symétrie ($\rho(x, y) = \rho(y, x)$), ou encore la transitivité ($\rho(x, z) \geq \max_y \min(\rho(x, y), \rho(y, z))$).

La composition est une importante opération associée aux relations floues. Supposons que ρ_1 et ρ_2 soient deux relations floues sur V. Alors, la composition $\rho = \rho_1 \circ \rho_2$ est une relation floue sur V tq. $\rho(x, z) = \max_y \min(\rho_1(x, y), \rho_2(y, z))$. La composition est associative : $(\rho_1 \circ \rho_2) \circ \rho_3 = \rho_1 \circ (\rho_2 \circ \rho_3)$. Cette propriété d'associativité permet d'utiliser la notation $\rho^k = \rho \circ \rho \circ \ldots \circ \rho$ pour représenter la composition de ρ avec elle-même $k-1$ fois. En complément, on définit ρ^0 comme étant $\rho^0(x, y) = 0$, $\forall(x, y)$ [Yag13]. Si ρ est réflexive alors $\rho^{k_2} \supseteq \rho^{k_1}$ pour $k_2 > k_1$. Si ρ est transitive, on peut montrer que $\rho^{k_2} \subseteq \rho^{k_1}$ si $k_2 > k_1$. On peut montrer que si ρ est réflexive et transitive alors $\rho^{k_2} = \rho^{k_1}$ pour tous k_1 et $k_2 \neq 0$.

REMARQUE 2. *Les graphes flous comme définis ci-dessus peuvent être généralisés au cas des graphes contenant des nœuds flous. En notant toujours V l'ensemble des nœuds et F le sous ensemble flou de V, un graphe flou contenant des*

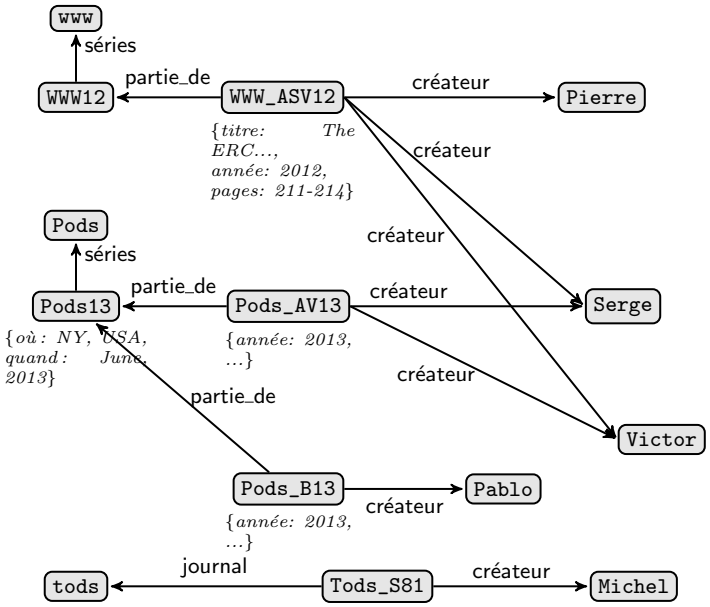


Figure 1: Base de données graphe inspirée d'une extraction de données de DBLP

nœuds flous est un triplet (V, F, ρ_F) *où* ρ_F *est une relation sur* V *définie par* $\rho_F(x, y) = \min(\rho(x, y), \mu_F(x), \mu_F(y))$ *où* μ_F *est la fonction d'appartenance associée à* F*. Dans la suite, on ne considère que le cas de nœuds non flous.*

Si ρ est symétrique alors on peut dire que (V, ρ) est un graphe non orienté. Si ρ n'est pas symétrique alors (V, ρ) est un graphe orienté. Sans perte de généralité, on ne considère que des graphes orientés dans la suite.

2. PRÉFÉRENCES FLOUES

Nous décrivons dans cette section les principaux éléments qui devraient pouvoir être exprimés dans une requête floue adressée à une base de données graphe. Deux types de préférences doivent pouvoir être considérés : des préférences concernant le contenu (attributs) du graphe et des préférences concernant la structure du graphe.

2.1 Concernant le contenu du graphe

L'idée est d'exprimer des conditions flexibles concernant les attributs associés aux nœuds (et éventuellement aux arcs) du graphe. Un exemple de requête flexible de ce type est "trouver les personnes *jeunes*, *hautement diplômées*, vivant en *Europe de l'Est*" ; en supposant qu'un nœud décrivant une personne contienne des informations concernant son âge, son niveau d'étude, son adresse, etc. Des conditions composées plus complexes peuvent être exprimées en utilisant des connecteurs flous (dont une large gamme est disponible). Nous ne détaillons pas cet aspect déjà intensivement étudié dans le cadre de l'interrogation floue des bases de données relationnelles [PB12].

2.2 Concernant la structure du graphe

Nous décrivons maintenant les concepts issus de la théorie des graphes flous qui nous semblent les plus utiles dans la perspective d'interrogation d'une base de données graphe. Soit un graphe flou $G = (V, \rho)$.
Un chemin p dans G est une suite $x_0 \to x_1 \to \ldots \to x_n$ $(n \geq 0)$ telle que $\rho(x_{i-1}, x_i) > 0$, $1 \leq i \leq n$. Le nombre de pas dans le chemin est n.

Force d'un chemin. — La *force* d'un chemin p de G est définie par :

$$ST(p) = \min_{i=1..n} \rho(x_{i-1}, x_i). \quad (1)$$

En d'autres termes, la force d'un chemin est le degré du plus faible des arcs entrant dans la composition du chemin. Deux nœuds pouvant être reliés par un chemin p tel que $ST(p) > 0$ sont dits *connectés*. Un chemin p est un cycle si $n \geq 1$ et $x_0 = x_n$. Il est possible de montrer que $\rho^k(x, y)$ est la force du plus fort chemin allant de x à y et contenant au plus k pas. Ainsi, la force du plus fort chemin reliant deux nœuds x et y, quel que soit le nombre de pas considéré, peut être écrite $\rho^\infty(x, y)$. Un algorithme permettant de calculer ρ^∞ en $O(|V|^4)$ est proposé dans [BS91].

Longueur et *distance*. — La *longueur* d'un chemin $p = x_0 \to x_1 \to \ldots \to x_n$ (au sens de ρ) est un concept défini par Rosenfeld [Ros75] par :

$$Length(p) = \sum_{i=1}^{n} \frac{1}{\rho(x_{i-1}, x_i)}. \quad (2)$$

En toute généralité, on a $Length(p) \geq n$. Dans le cas particulier où G est non flou (c-à-d dans le cas où ρ est booléenne), on a $Length(p) = n$. On peut définir la *distance* entre deux nœuds x et y dans G par :

$$\delta(x, y) = \min_{\text{tous les chemins allant de } x \text{ à } y} Length(p). \quad (3)$$

Il s'agit de la longueur du plus court chemin allant de x à y. On peut monter que δ est une métrique [Ros75] c-à-d que

$\delta(x, x) = 0$, $\delta(x, y) = \delta(y, x)$, et $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$.

α-coupe d'une relation. — Elle est définie par : $\rho^{\alpha} = \{(x, y) \mid \rho(x, y) \geq \alpha\}$ où $\alpha \in [0^+, 1]$. On peut noter que ρ^{α} est une relation non floue.

2.3 Combinaison de préférences

La théorie des ensembles flous fournit une large gamme d'opérateurs permettant de connecter des conditions flexibles. Dans le cadre considéré, ces connecteurs permettent de combiner des prédicats sur le contenu et/ou la structure du graphe. Outre le minimum et le maximum qui généralisent la conjonction et la disjonction usuelles, on peut utiliser différents opérateurs de compromis tels que les moyennes ou leurs variantes pondérées, les opérateurs de conjonction et de disjonction pondérées proposées dans [DP86], l'opérateur de moyenne pondérée dynamique (OWA) introduit dans [Yag88], des quantificateurs flous [LK98], ou des opérateurs hiérarchiques permettant d'affecter des priorités aux différentes conditions que l'on combine [Yag08, BP12]. Nous invitons le lecteur à consulter [FY00] pour une présentation détaillée des divers connecteurs flous.

3. BASE DE DONNÉES GRAPHE FLOUE ET ALGÈBRE FLOUE

Dans cette section, nous définissons une algèbre permettant une interrogation flexible de bases de données graphes, dans lesquelles les graphes peuvent être flous ou non. Nous introduisons le modèle de données, puis les opérateurs de l'algèbre. Un exemple illustre les notions définies au fur et à mesure de leur introduction.

3.1 Modèle de données

Nous nous intéressons dans la suite à la manipulation de bases de données graphes floues dans lesquelles les nœuds et les arcs peuvent être porteurs de données (pe. des paires clef-valeur dans les graphes d'attributs évoqués dans la section 1.1). Nous commençons donc par proposer une extension de la notion de graphe flou à celle de *graphe de données flou*.

Définition 1 (Graphe de données flou). *Soit E un ensemble d'étiquettes (destinées à étiqueter les arcs du graphe). Un* graphe de données flou *$\mathcal{G}$ est un quadruplet (V, R, κ, ζ), où V est un ensemble fini de nœuds pour lequel chaque nœud n a un identifiant id, aussi noté n.id, $R = \bigcup_{e \in E} \{\rho_e : V \times V \to [0, 1]\}$ est un ensemble d'arcs flous reliant les nœuds de V, et κ (resp. ζ) est une fonction associant des données, par exemple un ensemble d'éléments de type clef-valeur, aux nœuds (resp. arcs) de $\mathcal{G}$.*

Remarque 3. (Arcs et graphe de données non flous) *Un graphe de données flou peut contenir à la fois des arcs flous et des arcs non flous puisqu'un arc flou ayant un degré de 0 ou 1 peut être considéré non flou. Par extension, un graphe de données non flou est simplement un cas particulier de graphe de données flou pour lequel $\rho_e : V \times V \to \{0, 1\}$ pour tout $e \in E$. Nous ne considérerons donc que des arcs et graphes de données flous, dont le cas non flou n'est qu'une spécialisation.*

Dans la suite, le terme *base de données graphe* est utilisé pour désigner un graphe de données flou. L'exemple suivant illustre cette notion.

Exemple 1 (Graphe de données flou). *La figure 3 est un exemple de graphe de données flou, contenant à la fois des arcs flous (dont le degré est par convention indiqué entre parenthèses à côté de l'étiquette) et des arcs non flous (équivalents à des arcs ayant un degré associé de valeur `1`). Dans cet exemple, le degré associé à un arc de la forme `A -contributeur-> B` correspond à la proportion de papiers de journaux écrits par `B` qui ont été co-écrits par `A`. Cette notion de degré est fondée sur une notion statistique simple, qui peut être rendue plus fine par l'application d'opérations floues ou par l'intégration de connaissances expertes.* ⋄

Cette fonctionnalité étant offerte par de nombreux systèmes, les nœuds sont supposés typés. Dans la figure 3, les nœuds `WWW12` et `Pods13` sont de type *Conférence*, les nœuds `Pods_AV13`, `Pods_B13`, `Tods_S81`, et `WWW_ASV12` sont de type *Article*, les nœuds `Pods`, `Tods`, et `WWW` sont de type *Série* et les autres nœuds sont de type *Auteur*. Si n est un nœud de V, alors $Type(n)$ représente son type.

3.2 Algèbre

Nous passons maintenant à la définition de l'algèbre floue permettant l'interrogation de graphes de données éventuellement flous. Notre algèbre est en partie inspirée de la notion de requête (non floue) à base de patron de graphe de [FLM$^+$12] et de l'algèbre non floue proposée dans [HS08]. L'unité d'information de base de l'algèbre est le graphe.

L'opérateur de sélection est fondé sur le concept de *patron flou de graphe*, une extension de la notion de patron de graphe non flou proposé dans [FLM$^+$12]. Nous commençons donc par introduire la notion de *patron flou de graphe*, s'appuyant elle-même sur la notion d'*expression régulière floue*.

Définition 2. *(*(Expression régulière floue) *Une* expression régulière floue *est une expression de la forme:*

$$F ::= e \mid F{\cdot}F \mid F \cup F \mid F^* \mid F^{Cond}$$

où

(i) $e \in E \cup \{\_\}$ *représente un arc étiqueté par e, le caractère souligné (_) représentant n'importe quelle étiquette de E;*
(ii) $F{\cdot}F$ *est une concaténation d'expressions;*
(iii) $F \cup F$ *représente des expressions alternatives;*
(iv) F^* *représente la répétition d'expression;*
(v) F^{Cond} *représente un chemin p satisfaisant F et la condition Cond, où Cond est une combinaison booléenne de formules atomiques de la forme `Property is Fterm` où `Property` est une propriété définie sur p et `Fterm` est un terme flou pré-défini ou défini par un utilisateur comme le terme flou `short` dont une représentation de la fonction d'appartenance est proposée en figure 4.*

Dans la suite, nous limitons les propriétés des chemins à l'ensemble {`ST`, `Length`} représentant respectivement $ST(p)$ (voir équation 1) et $Length(p)$ (voir équation 2). Ainsi, des exemples de conditions sur la forme d'un chemin sont `Length is short` et `ST is strong`. Observons qu'une condition booléenne de la forme `Property op a` où `Property` est une propriété sur p, `a` est une constante et `op` est un opérateur de comparaison (<,=,...) est un cas particulier de condition floue.

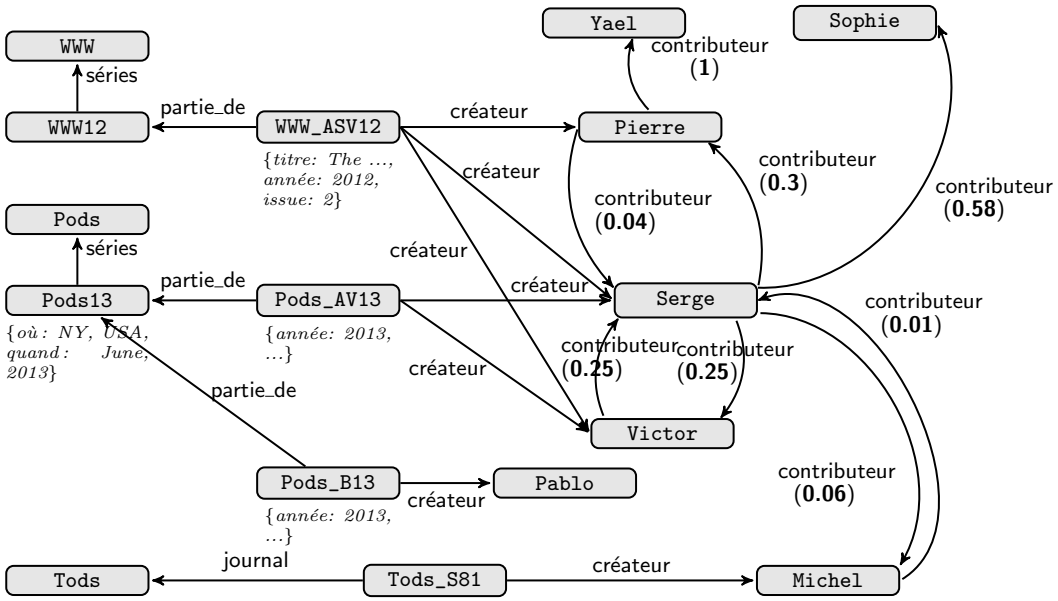


Figure 3: Graphe de données flou $\mathcal{BD}$ inspiré d'un extrait de DBLP data

Nous utilisons les notations suivantes : étant donnée une expression régulière floue f, f^+ est une notation raccourcie pour $f{\cdot}f^*$, f^k est une notation raccourcie pour $f{\cdot}f{\cdot}\cdots{\cdot}f$ ayant k occurrences de f et $f^{n,m}$ est une notation raccourcie pour $\bigcup_{i=n}^{m} f^i$.

Une expression régulière floue est dite *simple* si elle est de la forme e où $e \in E \cup \{\_\}$, c'est-à-dire qu'elle fait explicitement référence à un arc seul.

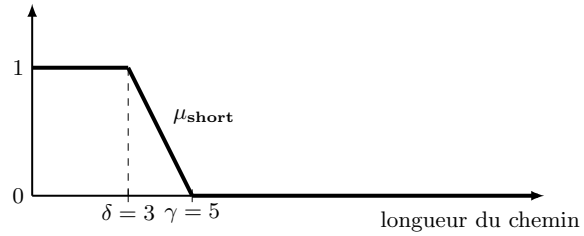


Figure 4: Représentation du terme flou *short*

Remarque 4. *Même si on ne désire considérer que des graphes classiques non flous, les concepts présentés ci-dessus peuvent toujours être utilisés en arguments des conditions floues (par exemple une longueur courte (short)) puisqu'ils restent valides sur des graphes non flous (avec $\rho(x) \in \{0,1\}$).*

Définition 3. (Satisfaction d'une expression régulière floue) *Soient un chemin p et une expression régulière floue exp; p* satisfait *exp* avec un degré de satisfaction $\mu_{exp}(p)$ *defini comme suit, en fonction de la forme de exp (dans la suite, f, f_1 et f_2 sont des expressions régulières floues):*

exp **est de la forme** *e* **avec** *$e \in E$* **(resp. "_").** *Si p est de la forme $v_1 \xrightarrow{e'} v_1'$ où $e' = e$ (resp. où $e' \in E$) alors $\mu_{exp}(p) = 1$ sinon $\mu_{exp}(p) = 0$.*

exp **est de la forme** *$f_1{\cdot}f_2$.* *Soit P l'ensemble des couples de chemins (p_1, p_2) tq p est de la forme p_1p_2. On a alors $\mu_{exp}(p) = \max_P(min(\mu_{f_1}(p_1), \mu_{f_2}(p_2)))$.*

exp **est de la forme** *$f_1 \cup f_2$.* *Dans ce cas, on a $\mu_{exp}(p) = \max(\mu_{f_1}(p), \mu_{f_2}(p))$.*

exp **est de la forme** *f^*.* *Si p est un chemin vide alors $\mu_{exp}(p) = 1$. Sinon, on note P l'ensemble des listes de chemins $(p_1, \cdots, p_n)$ $(n > 0)$ telle que p est de la forme $p_1 \cdots p_n$. On a alors $\mu_{exp}(p) = \max_P(\min_{i\in[1..n]}(\mu_f(p_i)))$.*

exp **est de la forme** *f^{Cond} où Cond est une condition floue telle que définie dans la définition 2. Dans ce cas, on a $\mu_{exp}(p) = \min(\mu_f(p), \mu_{Cond}(p))$ où $\mu_{Cond}(p)$ est le degré de satisfaction de Cond par p.*

Dans la suite, on dira qu'une expression régulière est satisfaite si elle l'est à un degré strictement supérieur à 0.

Exemple 2. *Les chemins représentés dans la figure 5 sont extraits de la base de données graphe de la figure 3.*

- *L'expression e_1 = créateur $\cdot$ contributeur$^+$ est une expression régulière floue. Tous les chemins p_i ($i \in [1..4]$) de la figure 5 satisfont e_1 avec un degré de égal à $\mu_{e_1}(p_i) = 1$.*
- *L'expression $e_2 = (\text{créateur}\cdot\text{contributeur}^+)^{\texttt{ST>0.4}}$ est une expression régulière floue. Le chemin p_4 est le seul chemin de la figure 5 satisfaisant e_2 (car $ST(p_1) = 0.3$, $ST(p_2) = 0.3$, $ST(p_3) = 0.01$ et $ST(p_4) = 0.58$), avec $\mu_{e_2}(p_4) = 1$.*
- *L'expression $e_3 = \text{créateur}\cdot(\text{contributeur}^+)^{\texttt{Length is short}}$ où short est le terme flou représenté en figure 4, est une expression régulière floue. Les chemins p_1, p_2 et p_4 de la figure 5 satisfont e_3 avec $\mu_{e_3}(p_1) = 0.83$ puisque $\mu_{short}(1/0.3) = 0.83$ (où $1/0.3$ est la longueur*

du chemin allant de `Serge` *à* `Pierre`*),* $\mu_{e_3}(p_2) = 0.67$ *puisque* $\mu_{short}(1/0.3 + 1) = 0.67$ *(où* $1/0.67$ *est le longueur du chemin allant de* `Serge` *à* `Yael`*) et* $\mu_{e_3}(p_4) = 1$ *puisque* $\mu_{short}(1/0.58) = 1$*. Le chemin* p_3 *ne satisfait pas* e_3 *puisque* $\mu_{short}(1/0.01) = 0$*.* $\diamond$

Nous introduisons maintenant la notion de *patron flou de graphe*, correspondant à un graphe (classique non flou) pour lequel les arcs sont étiquetés avec des expressions régulières floues, d'éventuelles conditions peuvent être posées sur les nœuds et les arcs, et un type peut être associé à un nœud.

Définition 4. (Patron flou de graphe) *Soit* $\mathcal{F}$ *un ensemble de termes flous. Un* patron flou de graphe *est défini par un sextuplet* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, f_e^{path}, f_n^{cond}, f_e^{cond}, f_n^{type})$ *où*

(i) $V_\mathcal{P}$ *est un ensemble fini de nœuds ;*
(ii) $E_\mathcal{P} \subseteq V_\mathcal{P} \times V_\mathcal{P}$ *est un ensemble fini d'arcs* (u, u')*;*
(iii) f_e^{path} *est une fonction définie sur* $E_\mathcal{P}$ *telle que pour tout* (u, u') *dans* $E_\mathcal{P}$*,* $f_e^{path}(u, u')$ *est une expression régulière floue ;*
(iv) f_n^{cond} *est une fonction définie sur* $V_\mathcal{P}$ *telle que pour tout nœud* u*,* $f_n^{cond}(u)$ *est une condition sur les attributs de* u*, définie par une combinaison de formules atomiques de la forme* `A is Fterm` *où* `A` *est un attribut et* `Fterm` *est un terme flou (pe.* `année is recent`*). De nouveau, un prédicat booléen de la forme* `A op a` *(où* `A` *est un attribut,* `a` *est une constante et* `op` *est un opérateur de comparaison, comme pe.* `année` $>$ `2012` *est un cas particulier de prédicat flou.*
(v) f_e^{cond} *est la contrepartie de* f_n^{cond} *pour les arcs. Pour tout* (u, u') *dans* $E_\mathcal{P}$ *pour lequel* $f_e^{path}(u, u')$ *est simple,* f_e^{cond} *est une condition sur les attributs de* (u, u') *; et*
(vi) f_n^{type} *est une fonction définie sur* $V_\mathcal{P}$ *tq pour tout nœud* u*,* $f_n^{type}(u)$ *est le type de* u*.*

Dans la suite, nous choisissons d'adopter une syntaxe *à la Cypher* pour la représentation des patrons. Deux raisons motivent ce choix : 1) cette syntaxe inspirée de l'art ASCII pour la représentation de graphes est intuitive, et 2) adopter cette syntaxe constitue un premier pas vers la définition d'un langage orienté utilisateur fondé sur l'algèbre, qui constitue une perspective naturelle à court terme. Un patron flou de graphe exprimé *à la Cypher* consiste en un ensemble d'expressions de la forme

```
(n1:Type1)-[exp]->(n2:Type2)
ou (n1:Type1)-[e:label]->(n2:Type2)
```

où `n1`, `n2` sont des variables de nœuds, `e` est une variable d'arc, `label` est une étiquette de E, `exp` est une expression régulière floue, et `Type1` et `Type2` sont des types de nœuds. Une telle expression représente un chemin satisfaisant une expression régulière floue (qui s'avère être *simple* dans la seconde forme) allant d'un nœud de type `Type1` à un nœud de type `Type2`. Tous les arguments de l'expression sont individuellement optionnels, ce qui signifie que la forme la plus dépouillée d'une telle expression est `()-[]->()` représentant un chemin constitué d'un arc quelconque reliant deux nœuds quelconques.

Les conditions sur les attributs sont exprimées sur les variables de nœuds et d'arcs dans une clause `where`.

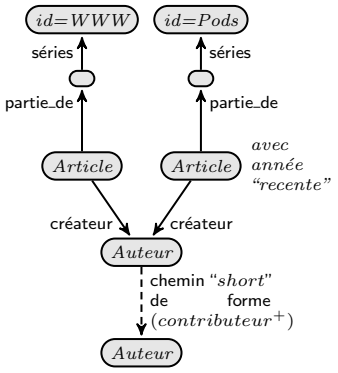


Figure 6: Patron $\mathcal{P}$

Exemple 3. *On note* $\mathcal{P}$ *le patron flou de graphe suivant :*

```
1 (art1:Article)-[partie_de.séries]->(s1),
2 (art2:Article)-[partie_de.séries]->(s2),
3 (art1)-[:créateur]->(auth1:Auteur),
4 (art2)-[:créateur]->(auth1:Auteur),
5 (auth1)-[(contributeur+)|Length is short]->(auth2:Auteur),
6 where
7 s1.id=WWW, s2.id=Pods,
8 art2.année is recent.
```

Le graphe de la figure 6 est une représentation graphique du patron $\mathcal{P}$ *dans laquelle l'arc en pointillés représente un chemin, les informations en italique représentent des conditions sur les attributs des nœuds ou arcs et le type associé à un nœud est indiqué en italique dans celui-ci. Ce patron capture des informations concernant des auteurs (*`auth2`*) qui ont parmi leurs contributeurs proches un auteur (*`auth1`*) ayant publié un papier (*`art1`*) à* `WWW` *et un autre papier (*`art2`*) à* `Pods` *récemment (*`art2.année is recent`*).* $\diamond$

Nous passons maintenant à la notion de la satisfaction d'un patron flou de graphe.

Définition 5. (Satisfaction d'un patron flou de graphe) *Un graphe de données flou* $G = (V, R, \kappa, \zeta)$ satisfait *un patron flou de graphe* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, f_e^{path}, f_n^{cond}, f_e^{cond}, f_n^{type})$ *avec un degré de satisfaction noté* $\mu_\mathcal{P}(G)$ *s'il existe une relation binaire* $S \subseteq V_\mathcal{P} \times V$ *représentant une fonction injective de* $V_\mathcal{P}$ *dans* V *telle que :*

1. *(correspondance des nœuds) pour tout nœud* $u \in V_\mathcal{P}$*, il existe un nœud* $v \in V$ *tel que* $(u, v) \in S$ *;*
2. *(correspondance des arcs) pour tout arc* $(u, u') \in E_\mathcal{P}$*, il existe deux nœuds* v *et* v' *de* V *tels que* $\{(u, v), (u', v')\} \subseteq S$ *et il existe un chemin* p *dans* G *de* v *à* v' *tel que* p *satisfait* $f_e^{path}(u, u')$ *(on rappelle que, d'après la définition 3, un degré de satisfaction de valeur strictement supérieure à zéro est associé en cas de satisfaction) ;*
3. *(vérification des conditions sur les attributs et des types des nœuds) pour tout tuple* $(u, v) \in S$*,* $\kappa(v) \vdash f_n^{cond}(u)$ *(la sémantique de* $\vdash$ *découle trivialement du contexte ici) et* $f_n^{type}(u) = Type(v)$*.*
4. *(vérification des conditions sur les attributs des arcs) le même raisonnement peut être appliqué en ce qui concerne les conditions sur les attributs des arcs étiquetés par une expression régulière floue simple dans* $E_\mathcal{P}$*, c-à-d* $\zeta(v, v') \vdash f_e^{cond}(u, u')$*.*

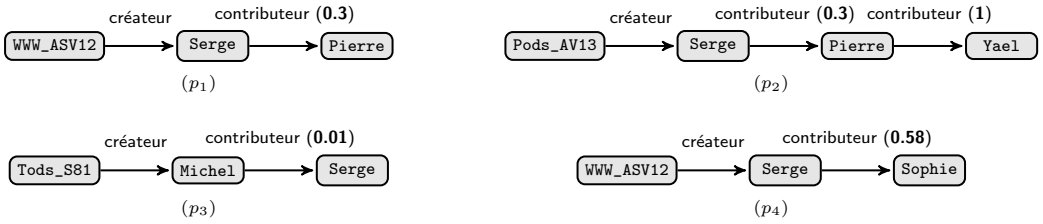


Figure 5: Satisfaction d'une expression régulière floue

$\mu_{\mathcal{P}}(G)$ est le degré de satisfaction minimal induit des correspondances et vérifications de 2., 3. et 4. Si aucune relation S satisfaisant ces conditions n'existe alors $\mu_{\mathcal{P}}(G) = 0$, autrement dit, G ne satisfait pas $\mathcal{P}$.

Exemple 4. *La figure 7 donne l'ensemble des sous-graphes de $\mathcal{BD}$ satisfaisant le patron $\mathcal{P}$ de l'exemple 3 avec le degré de satisfaction associé (par manque de place, les données véhiculées par les nœuds ne sont pas reportées).* `auth1` *est nécessairement* `Serge` *ou* `Victor` *qui sont les seuls auteurs de $\mathcal{BD}$ ayant écrit à la fois un papier à* `WWW` *et un papier récent à* `Pods`*. De façon à pouvoir traiter la ligne 8, on suppose que $\mu_{recent}(2013) = 0.7$. On note ici que le degré de satisfaction d'un graphe satisfaisant $\mathcal{P}$ est le minimum des degrés de satisfaction induits par les lignes 5 et 8.* $\diamond$

Nous donnons maintenant la définition des opérateurs flous de l'algèbre.

Il convient de noter que même si une base de données graphe contient un seul graphe, une requête de l'algèbre peut retourner un ensemble de graphes puisque plusieurs sous-graphes peuvent satisfaire un patron comme le montre l'exemple 4. Un degré de satisfaction est associé à chaque graphe. Un ensemble de couples $\langle$ *graphe*, *degré* $\rangle$ est tout simplement un ensemble flou de graphes. Chaque opérateur de l'algèbre prend ainsi en entrée un ou plusieurs (selon l'arité de l'opérateur) ensemble(s) flou(s) de graphes et génère un ensemble flou de graphes. L'algèbre est close. En cas de production de graphes doublons (un même graphe apparaissant avec différents degrés de satisfaction), seul le plus haut degré de satisfaction est conservé. Appliquer un opérateur à la base de données initiale $\mathcal{BD}$ revient à appliquer l'opérateur au singleton $\{\langle \mathcal{BD}, 1 \rangle\}$. Il est à noter que les graphes d'un ensemble ne partagent pas nécessairement la même structure. Les *expressions de l'algèbre* sont récursivement définies par: (i) une base de données graphe $\mathcal{BD}$ est une expression de l'algèbre, et (ii) si $e_1, \cdots, e_n$ sont des expressions et O est un opérateur d'arité n alors $O(e_1, \cdots, e_n)$ est une opération de l'algèbre.

Définition 6. (Opérateur de sélection) *L'opérateur de sélection σ prend en entrée un patron flou de graphe $\mathcal{P}$ et un ensemble flou $\mathcal{G}$ de graphes. Il renvoit un ensemble flou constitué des sous-graphes de $\mathcal{G}$ satisfaisant $\mathcal{P}$:*

$$\sigma_{\mathcal{P}}(\mathcal{G}) = \{\langle s, \min(d, \mu_{\mathcal{P}}(s)) \rangle \mid \mu_{\mathcal{P}}(s) > 0\}$$

où s est un sous-graphe de g tel que $\langle g, d \rangle \in \mathcal{G}$.

Exemple 5. *La figure 7 présente l'ensemble des réponses de $\sigma_{\mathcal{P}}(\mathcal{BD})$ où $\mathcal{P}$ est le patron de l'exemple 3 et $\mathcal{BD}$ est la base de données graphe de la figure 3.*

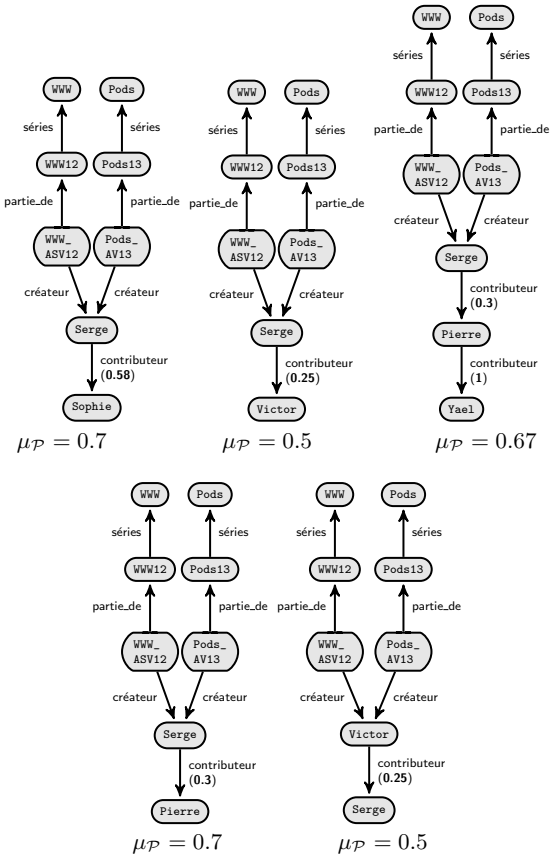


Figure 7: Sous-graphes de $\mathcal{BD}$ satisfaisant $\mathcal{P}$

Définition 7. (Opérateurs de projection) *L'opérateur de projection "sur les arcs" noté Π^{arcs} permet de supprimer des relations d'un graphe (sans supprimer de nœud du graphe). Cet opérateur prend en entrée un ensemble flou $\mathcal{G}$ de graphes, un ensemble d'étiquettes $\mathcal{L} \subseteq E$, et renvoie un ensemble flou de graphes défini comme suit:*

$$\Pi^{arcs}_{\mathcal{L}}(\mathcal{G}) = \{\langle (V, R', \kappa, \zeta), d \rangle \mid \langle (V, R, \kappa, \zeta), d \rangle \in \mathcal{G}\}$$

où $R' = \{\rho_e \mid \rho_e \in R \text{ et } e \in \mathcal{L}\}$.

L'opérateur de projection "sur les nœuds" noté $\Pi^{nœuds}$ permet de supprimer des nœuds d'un graphe. Cet opérateur prend en entrée un ensemble flou $\mathcal{G}$ de graphes, et un ensemble de types $\mathcal{T}$, et renvoie un ensemble flou de graphes

défini comme suit:

$$\Pi_{\mathcal{T}}^{nœuds}(\mathcal{G}) = \{\langle (V',\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \mid \langle (V,\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \in \mathcal{G}\}$$

où $V' = \{v \mid v \in V$ *et* $Type(v) \in \mathcal{T}\}$ *et* R' *est la restriction de* R *sur* $V' \times V'$.

EXEMPLE 6. *La figure 8 contient les réponses de* $\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD}))$ *où* $\sigma_{\mathcal{P}}(\mathcal{BD})$ *est l'expression de l'exemple 5 (c-à-d que* $\mathcal{P}$ *est le patron de l'exemple 3 et* $\mathcal{BD}$ *est la base de données graphe de la figure 3).*

DÉFINITION 8. (OPÉRATEUR D'ALPHA-COUPE) *Cette opération effectue une* α*-coupe sur une relation floue du graphe (voir section 2.2). L'opérateur d'alpha-coupe* Cut *prend en entrée un ensemble flou de graphes* $\mathcal{G}$*, une étiquette* $e \in E$ *et un nombre réel* $\alpha \in [0^+, 1]$*. Il produit un ensemble flou de graphes constitué des graphes de* $\mathcal{G}$ *où une alpha-coupe a été effectuée sur la relation* ρ_e *:*

$$Cut_{e,\alpha}(\mathcal{G}) = \{\langle (V,\ R',\ \kappa,\ \zeta)\ ,\ d\rangle \mid \langle (V,\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \in \mathcal{G}\}$$

où $R' = \{\rho_l \mid \rho_l \in R$ *et* $l \neq e\} \cup \{\rho_e^{\alpha}\}$.

L'opérateur d'alpha-coupe met à jour la relation ρ_e en lui appliquant une α-coupe, et donc rend ρ_e non floue. Cette opération n'a pas d'impact sur les données véhiculées par les arcs étiquetés par e.

EXEMPLE 7. *La figure 9 donne les réponses renvoyées par l'expression* $Cut_{contributeur,0.3}(\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD})))$ *où* $\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD}))$ *est l'expression de l'exemple 6 (c-à-d que* $\mathcal{P}$ *est le patron de l'exemple 3 et* $\mathcal{BD}$ *est la base de données graphe de la figure 3).*

Si le graphe comportait des nœuds flous, l'opérateur Cut pourrait être étendu pour permettre une α-coupe sur les nœuds.

DÉFINITION 9 (OPÉRATEURS ENSEMBLISTES). *Les opérateurs classiques d'union, intersection et différence sont définis à partir des opérations classiques des ensembles flous (voir section 1.2.1 pour les références).*

Les auteurs de [HS08] définissent des opérateurs permettant de combiner, fusionner et restructurer des graphes, qui pourraient compléter l'algèbre.

4. TRAVAUX CONNEXES

Comme de nombreux modèles ont été proposés pour représenter des données ayant implicitement ou explicitement une structure de graphe [AG08], il existe de nombreux langages pour l'interrogation de graphes. Les auteurs de [AG08], [Woo12] et [BB13] proposent des états de l'art complémentaires des langages de requête pour les graphes proposés au cours de ces vingt-cinq dernières années incluant des langages pour l'interrogation de bases de données objet, de données semi-structurées, de réseaux sociaux, ou de données du web sémantique. Les travaux de [BB13] abordent les langages pour les bases de données graphes sous un angle théorique, en soulignant le fait que les systèmes de gestion de bases de données graphes actuels reposent sur des langages pour lesquels une syntaxe et une sémantique précise doivent encore être définies.

Certains langages de requête pour des bases de données graphes sont fondés sur une algèbre. Dans le cadre des graphes de données RDF, les auteurs de [AP11] proposent une formalisation algébrique dédiée à SPARQL, avec extension navigationnelle du langage. Dans [SEH14], les auteurs proposent un langage *à la SPARQL* permettant d'interroger des graphes d'attributs, en y associant un mécanisme d'évaluation fondé sur une algèbre. Dans [AG05], les auteurs proposent de nouvelles primitives inspirées du monde des bases de données graphes pour l'interrogation de données RDF. Une algèbre étendant l'algèbre relationnelle pour l'interrogation de bases de données graphes est proposée dans [HS08]. Comme mentionné précédemment, notre travail est partiellement inspiré de cette contribution. Les langages de requête proposés dans tous ces travaux concernent des bases de données graphes ou RDF non floues, sans interrogation flexible des données.

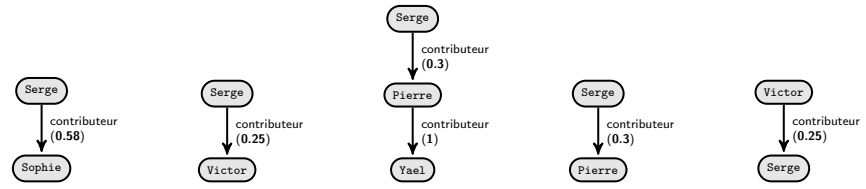
Concernant l'interrogation flexible de la topologie des bases de données graphes, il existe trois grandes catégories d'approches : (i) les approches du type *interrogation par mot-clef* ignorant le schéma des données (voir par exemple [HWYY07]) souffrant d'un manque d'expressivité dans une grande partie des cas d'utilisation d'une base de données graphe [MMVP09] ; (ii) les approches consistant à proposer à l'utilisateur des *réponses approximatives* à une requête (non floue), par exemple par la mise en œuvre d'un mécanisme d'extension ou de relâchement de la requête ou par un mécanisme calculant des réponses correspondant approximativement à la requête (voir par exemple [KS01], [BDBH08], ou [MMVP09]) ; (iii) les approches consistant à permettre à un utilisateur d'*introduire une forme de flexibilité dans les requêtes*, famille d'approches dans laquelle ce présent travail se situe.

Parmi le dernier type d'approche, de nombreuses contributions concernent l'extension flexible de XPath [DMP07, CDG+09, AJLM11] pour l'interrogation de données semi-structurées (arbre). Même si les langages navigationnels *à la XPath* sont jugés adéquats pour l'interrogation de données graphes (voir [LMV13]), aucune extension flexible de ce type de langage n'a encore été proposée pour le modèle des bases de données graphes.

Dans [CMY10], les auteurs proposent une extension de la syntaxe du langage SPARQL permettant d'introduire des termes et des relations floues dans une requête SPARQL, en focalisant les travaux sur l'aspect d'enrichissement syntaxique du langage SPARQL et la faisabilité d'une implantation. Enfin, les auteurs de [CnC11] proposent une extension de SPARQL permettant d'interroger des graphes contenant des arcs pondérés, dans l'objectif final de pouvoir ordonner les réponses à une requête. La contribution est axée sur la mise en œuvre de l'extension via un moteur SPARQL. De notre côté, nous nous plaçons dans un cadre plus général permettant l'interrogation de bases de données graphes éventuellement floues en proposant une algèbre (qui pourrait servir de fondement à d'autres langages) fondée sur la théorie des ensembles flous et la théorie des graphes flous.

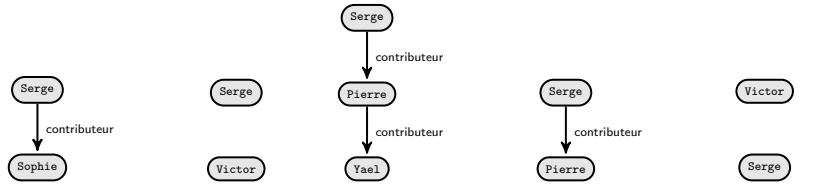
5. CONCLUSION ET PERSPECTIVES

Nous présentons dans ce papier une algèbre permettant l'interrogation flexible de bases de données graphes, dans lesquelles un graphe peut être flou ou non. Cette algèbre,



$d = 0.7$ $d = 0.5$ $d = 0.67$ $d = 0.7$ $d = 0.5$

Figure 8: Réponse de $\Pi^{nœuds}_{Auteur}(\sigma_{\mathcal{P}}(\mathcal{BD}))$



$d = 0.7$ $d = 0.5$ $d = 0.67$ $d = 0.7$ $d = 0.5$

Figure 9: Réponse de $Cut_{contributor,0.3}(\Pi^{nœuds}_{Auteur}(\sigma_{\mathcal{P}}(\mathcal{BD})))$

fondée sur la théorie des ensembles flous et la théorie des graphes flous, propose un modèle de données et une famille d'opérateurs permettant d'exprimer des préférences flexibles sur 1) les données contenues dans le graphe, et 2) la structure du graphe.

De nombreuses perspectives théoriques et appliquées sont ouvertes par ce travail. De même que l'algèbre relationnelle constitue la base du langage SQL utilisé dans les systèmes commerciaux, l'algèbre floue proposée ici est destinée à servir de fondement à l'extension d'outils plus orientés utilisateur tels que le langage Cypher [Neo13] implanté dans le système Neo4j [Neo] (permettant actuellement la gestion de graphes d'attributs non flous). Comme dans le cadre relationnel, des fonctionnalités d'ordonnancement et d'agrégation seraient offertes par le langage, ouvrant la porte à la prise en compte de nouvelles notions des graphes flous telles que la *distance* ou le *diamètre* des chemins entre les nœuds, et également les *degrés entrant et sortant* ou la *centralité* des nœuds [Yag13], qui sont des notions régulièrement utilisées pe. dans le cadre de l'analyse structurelle de réseaux sociaux. Ces fonctionnalités permettraient également de pourvoir supprimer les nœuds faiblement connectés des réponses.

La mise en œuvre soulève (au moins) deux problèmes. Le premier problème, a priori facile à résoudre, concerne l'éventuelle modélisation d'une base de données floue dans un système gérant des bases de données graphes non floues. Une relation $\rho_e(x, y)$ d'une base de données floue peut être représentée dans une base de donnée non floue en considérant que l'étiquette reliant deux nœuds x et y n'est pas juste e mais un couple e, v où v est la valeur de $\rho_e(x, y)$. Ainsi, un graphe non flou être utilisé pour représenter un graphe flou. Dans le cas des graphes d'attributs, ce mécanisme peut être très simplement implanté en ajoutant à chaque arc étiqueté par e un attribut *fdegree* portant la valeur de v (en supposant ensuite que *fdegree* devient un mot clef du langage).

Un second problème concerne l'optimisation de l'évaluation des requêtes. À l'image du cadre relationnel, des règles d'optimisation permettant d'optimiser les requêtes en exploitant au mieux les index existants et la sélectivité des opérateurs seront à définir.

6. REFERENCES

[AG05] Renzo Angles and Claudio Gutiérrez. Querying RDF Data from a Graph Database Perspective. In *Proc. of European Semantic Web Conf. (ESWC)*, pages 346–360, 2005.

[AG08] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008.

[AJLM11] Jesús M. Almendros-Jiménez, Alejandro Luna, and Ginés Moreno. A Flexible XPath-based Query Language Implemented with Fuzzy Logic Programming. In *Proc. of the Intl. Conf. on Rule-based Reasoning, Programming, and Applications (RuleML)*, pages 186–193. Springer-Verlag, 2011.

[All] AllegroGraph web site. http://franz.com/agraph/allegrograph.

[Ang12] Renzo Angles. A comparison of current graph database models. In *Proc. of IEEE Intl. Conf. on Data Engineering Workshops*, pages 171–177, 2012.

[AP11] Marcelo Arenas and Jorge Pérez. Querying semantic web data with SPARQL. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 305–316, 2011.

[BB13] Pablo Barceló Baeza. Querying graph databases. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 175–188, 2013.

[BDBH08] Patrice Buche, Juliette Dibie-Barthélemy, and Gaëlle Hignette. Flexible Querying of Fuzzy RDF Annotations Using Fuzzy Conceptual Graphs. In *Proc. of Intl. Conf. on Conceptual Structures (ICCS)*, pages 133–146, 2008.

[BP95] Patrick Bosc and Olivier Pivert. SQLf: a relational database language for fuzzy querying.

IEEE Trans. on Fuzzy Systems, 3:1–17, 1995.

[BP12] Patrick Bosc and Olivier Pivert. On four noncommutative fuzzy connectives and their axiomatization. *Fuzzy Sets and Systems*, 202:42–60, 2012.

[BS91] Prabir Bhattacharya and Francis Suraweera. An algorithm to compute the supremum of max-min powers and a property of fuzzy graphs. *Pattern Recognition Letters*, 12(7):413–420, 1991.

[BT12] Shalini Batra and Charu Tyagi. Comparative analysis of relational and graph databases. *Intl. Journal of Soft Computing and Engineering*, 2(2), 2012.

[CAH12] Marek Ciglan, Alex Averbuch, and Ladislav Hluchý. Benchmarking traversal operations over graph databases. In *IEEE ICDE Workshops*, pages 186–189, 2012.

[CDG+09] Alessandro Campi, Ernesto Damiani, Sam Guinea, Stefania Marrara, Gabriella Pasi, and Paola Spoletini. A Fuzzy Extension of the XPath Query Language. *J. Intell. Inf. Syst.*, 33(3):285–305, 2009.

[CMY10] Jingwei Cheng, Z. M. Ma, and Li Yan. f-SPARQL: A flexible extension of SPARQL. In *Proc. of the Intl. Conf. on Database and Expert Systems Applications*, pages 487–494, 2010.

[CnC11] Juan P. Cedeño and K. Selçuk Candan. R2DF Framework for Ranked Path Queries over Weighted RDF Graphs. In *Proc. of the Intl. Conf. on Web Intelligence, Mining and Semantics*, pages 40:1–40:12, 2011.

[DMP07] Ernesto Damiani, Stefania Marrara, and Gabriella Pasi. FuzzyXPath: Using Fuzzy Logic an IR Features to Approximately Query XML Documents. In *Foundations of Fuzzy Logic and Soft Computing*, LNCS, pages 199–208. Springer, 2007.

[DP86] Didier Dubois and Henri Prade. Weighted minimum and maximum operations in fuzzy set theory. *Information Sciences*, 39:205–210, 1986.

[DP96] Didier Dubois and Henri Prade. Using fuzzy sets in database systems: Why and how? In *Proc. of FQAS'96*, pages 89–103, 1996.

[DP00] Didier Dubois and Henri Prade. *Fundamentals of fuzzy sets*, volume 7 of *The Handbooks of Fuzzy Sets*. Kluwer Academic Pub, 2000.

[DSUBGV+10] David Dominguez-Sal, P. Urbón-Bayes, Aleix Giménez-Vañó, Sergio Gómez-Villamor, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark. In *Proc. of WAIM'10 Workshops*, pages 37–48, 2010.

[FLM+12] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Yinghui Wu. Adding regular expressions to graph reachability and pattern queries. *Frontiers of Computer Science*, 6(3):313–338, 2012.

[FY00] János Fodor and Ronald R. Yager. Fuzzy-set theoretic operators and quantifiers. In *The Handbooks of Fuzzy Sets Series, vol. 1: Fundamentals of Fuzzy Sets*, pages 125–193. Kluwer Academic Publishers, 2000.

[GS02] Rosalba Giugno and Dennis Shasha. Graphgrep: A fast and universal method for querying graphs. In *ICPR (2)*, pages 112–115, 2002.

[HS08] Huahai He and Ambuj K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proc. of SIGMOD'08*, pages 405–418, 2008.

[HWYY07] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. Blinks: Ranked keyword searches on graphs. In *Proc. of the ACM SIGMOD*, pages 305–316, 2007.

[Inf] InfiniteGraph web site. http://www.objectivity.com/infinitegraph.

[KS01] Yaron Kanza and Yehoshua Sagiv. Flexible queries over semistructured data. In *Proc. of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 40–51, 2001.

[LK98] Yaxin Liu and Etienne E. Kerre. An overview of fuzzy quantifiers. (i). interpretations. *Fuzzy Sets and Systems*, 95(1):1–21, 1998.

[LMV13] Leonid Libkin, Wim Martens, and Domagoj Vrgoč. Querying Graph Databases with XPath. In *Proc. of ICDT'13*, pages 129–140, 2013.

[MMVP09] Federica Mandreoli, Riccardo Martoglia, Giorgio Villani, and Wilma Penzo. Flexible query answering on graph-modeled data. In *Proc. of the Intl. Conf. on Extending Database Technology: Advances in Database Technology (EDBT)*, pages 216–227, 2009.

[MN00] John N. Mordeson and Premchand S. Nair. *Fuzzy Graphs and Fuzzy Hypergraphs*, volume 46 of *Studies in Fuzziness and Soft Computing*. Springer, 2000.

[Neo] Neo4j web site. http://www.neo4j.org.

[Neo13] Neo4j Team of Neo Technology. The Neo4j Manual v2.0.0, 2013. Part III.

[PB12] Olivier Pivert and Patrick Bosc. *Fuzzy Preference Queries to Relational Databases*. Imperial College Press, London, UK, 2012.

[Ros75] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, pages 77–97. Academic Press, 1975.

[SEH14] Sherif Sakr, Sameh Elnikety, and Yuxiong He. Hybrid query execution engine for large attributed graphs. *Inf. Syst.*, 41:45–73, 2014.

[Spa] Sparksee (formerly known as DEX) web site. http://sparsity-technologies.com.

[VMZ+10] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *ACM Southeast Regional Conf.*, page 42, 2010.

[Woo12] Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, 2012.

[Yag88] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[Yag08] Ronald R. Yager. Prioritized aggregation operators. *Int. J. Approx. Reasoning*, 48(1):263–274, 2008.

[Yag13] Ronald R. Yager. Social network database querying based on computing with words. In *Flexible Approaches in Data, Information and Knowledge Management*, Studies in Computational Intelligence. Springer, 2013.

[Zad65] Lotfi A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[ZDDK08] Slawomir Zadrozny, Guy De Tré, Rita M. De Caluwe, and Janusz Kacprzyk. An overview of fuzzy approaches to flexible database querying. In José Galindo, editor, *Handbook of Research on Fuzzy Information Processing in Databases*, pages 34–54. IGI Global, 2008.

Une approche holistique combinant flux temps-réel et données archivées pour la gestion et le traitement d'objets mobiles

Loic Salmon [*]
Institut de Recherche de l'Ecole Navale
29240 BREST Cedex 9 - FRANCE
loic.salmon@ecole-navale.fr

Cyril Ray [†]
Institut de Recherche de l'Ecole Navale
29240 BREST Cedex 9 - FRANCE
cyril.ray@ecole-navale.fr

Christophe Claramunt
Institut de Recherche de l'Ecole Navale
29240 BREST Cedex 9 - FRANCE
christophe.claramunt@ecole-navale.fr

ABSTRACT

La numérisation de nos espaces de vie et de mobilité s'est largement accentuée durant la dernière décennie. La multiplication des capteurs de toute nature permettant de percevoir et de mesurer notre espace physique en est le levier principal. L'ensemble de ces systèmes produit aujourd'hui de grands volumes de données hétérogènes sans cesse croissants ce qui soulève de nombreux enjeux scientifiques et d'ingénierie en termes de stockage et de traitement pour la gestion et l'analyse de mobilités. Les travaux dans le domaine d'analyse des données spatio-temporelles ont largement été orientés soit vers la fouille de données historiques archivées, soit vers le traitement continu. Afin d'éviter les écueils de plus en plus prégnants dûs à l'augmentation des volumes de données et de leur vélocité (temps de traitement trop long, modèles conceptuellement plus adaptés, analyse des données approximative), nous proposons la conception d'une approche hybride distribuée permettant le traitement combiné de flux temps-réel et de données archivées. L'objectif de cette thèse est donc de développer un système nouveau de gestion et de traitement distribué pour l'analyse des mobilités maritimes.

Keywords

Base de données spatio-temporelles, objets mobiles, traitement temps-réel, système distribué

1. INTRODUCTION

L'analyse de mobilités intervient dans de nombreux domaines tels que l'aménagement urbain, la surveillance du trafic, la climatologie, l'étude des phénomènes sociaux ou la zoologie. L'émergence et la multiplication de systèmes mobiles et des capteurs véhiculant des informations provoquent une explosion du volume de données spatiales et temporelles. Ce gisement de données qui n'a évidemment pas encore atteint sa pleine mesure devient de plus en plus difficile à traiter et soulève de nombreux enjeux scientifiques et d'ingénierie en termes de stockage et de traitement des objets mobiles.

L'analyse de mobilités est un domaine spécifique qui met en difficulté les systèmes de bases de données relationnelles dans la mesure où les objets mobiles reportent leur position en continu (**V**élocité) ce qui produit rapidement une masse de données conséquente (**V**olume) ce qui nécessitera de mettre en place une solution dite *Big Data*. Enfin, bien que moins déterminant par rapport aux deux facteurs précédents, il faut prendre en compte le fait que les objets mobiles peuvent être de toute sortes : points, polylignes, surfaces dont la taille et la forme peuvent fortement varier (**V**ariété) dans l'espace et le temps nécessitant l'usage d'index particuliers.

Aux trois "V" traditionnels s'ajoutent d'autres problèmes plus spécifiques concernant les données spatio-temporelles. Une distribution équilibrée des données sur l'ensemble des nœuds du système par rapport à leur couverture spatiale, spatio-temporelle ou sémantique est plus difficile à mettre en œuvre car les phénomènes et déplacements observés se répartissent dans l'espace et le temps à différents niveaux de densité. Ceci a également une incidence sur l'échelle de représentation choisie et le volume de données manipulées. En effet, si on restreint trop le volume temporel ou spatial de données à analyser l'information extraite peut être biaisée ou erronée. A contrario s'il est trop grand, l'information obtenue peut être lissée et peu représentative car certaines particularités locales (spatiales, temporelles ou spatio-temporelles), auront affecté les résultats observés, sans avoir été détectées. Enfin, les traitements et opérateurs spatiaux font interagir des objets de nature et de taille différentes ce qui peut faire intervenir de nombreuses jointures et des calculs plus complexes que pour des données usuelles (opérateurs topologiques, comparaison de trajectoires ...).

2. TRAITEMENT DISTRIBUÉ DE DONNÉES SPATIO-TEMPORELLES

L'objectif de cette thèse sera donc de développer un sys-

[*]L. Salmon, corresponding author
[†]

BDA 14 octobre 2014, Grenoble-Autrans, France.

tème nouveau de traitement distribué et parallélisé, tenant compte de ces spécificités, afin de favoriser la gestion et le traitement de données spatio-temporelles dans un contexte *Big Data*.

2.1 Traitement on-line vs. off-line

Les travaux dans le domaine de l'analyses de mobilité ont largement été orientés soit vers la fouille de données historiques archivées, soit vers le traitement temps-réel.

La fouille de données historiques ou traitement *off-line* se caractérise par le stockage de la totalité de l'historique des mouvements des entités mobiles pour pouvoir étudier à posteriori les phénomènes du passé et éventuellement inférer le comportement futur d'un objet donné. Au vu des forts volumes de données à manipuler, le temps de réponse est important et certains mécanismes sont nécessaires pour accéder plus vite aux données (index, partitionnement) empêchant des mises à jour en continu. Les techniques actuelles de collecte, de stockage et d'interrogation des mobilités sont issues des travaux sur les bases de données pour objets mobiles (Moving Object Database ; MOD) [3]. Ces dernières sont presque exclusivement basées sur un modèle relationnel et intègrent ou exploitent des extensions pour la gestion de ces mobiles (types et opérateurs spatiaux, notion de temps intégrée, index associés aux objets mobiles) comme Hermes [9] ou Secondo [2]. Ces données d'objets mobiles stockées et archivées peuvent être exploitées à l'aide de différentes techniques de fouille de données : extraction, agrégation, *clustering*, fusion et permettre notamment l'identification de comportements type et d'anomalies. Seulement ces techniques de fouilles nécessitent la distribution des données et des traitements lorsque le volume de données augmente considérablement [5].

Le traitement temps-réel ou approche *on-line* s'intéresse au maintien continu des informations sur la position actuelle de l'entité pour pouvoir détecter des événements se produisant en temps-réel et éventuellement prédire une future position proche. Divers travaux ont été réalisés concernant ce type de traitement qui se caractérise par un temps de réponse rapide car effectué en mémoire. Par exemple, dans [8] les auteurs tentent de répondre à la problématique d'analyse de mobilité temps-réel en étendant un système de gestion des flux temps-réel au contexte spatio-temporel. Cependant cette approche peut fournir une réponse de moins bonne qualité à cause du traitement mémoire imposant de supprimer des données, de faire de l'échantillonnage, d'utiliser des fenêtres temporelles ou d'agréger certaines données et résultats intermédiaires par un traitement incrémental des flux [4]. L'analyse se fait alors en même temps que l'objet mobile évolue et les requêtes sur les données ne s'exécutent plus une seule fois comme en *off-line* mais en continu au gré du flux de données entrant [7].

L'évaluation des requêtes est un compromis entre temps d'exécution et précision ou qualité de la réponse. L'approche base de données historiques a donc pour précepte de préférer la qualité au temps de calcul et inversement en ce qui concerne les systèmes temps-réels.

2.2 Proposition d'une architecture hybride

Afin d'éviter les écueils de plus en plus prégnants dûs à l'augmentation des volumes de données et de leur vélocité (temps de traitement trop long, modèles conceptuellement plus adaptés, analyse des données approximative), nous proposons une approche hybride distribuée permettant le traitement combiné de flux temps-réel et de données archivées qui permettra de fournir une réponse satisfaisante en un temps acceptable (Figure 1).

Cette architecture est inspirée de l'approche hybride non distribuée de [1] dans laquelle trois types de requêtes sont distinguées : celles portant sur les données archivées, celles portant sur les données reçues en temps-réel et enfin les requêtes dites "hybrides" nécessitant de combiner les données arrivant en temps-réel et les informations extraites des données historiques. Plus récemment, Nathan Marz propose avec son architecture lambda un système de gestion de données prenant en compte aussi bien les aspects vélocité, volumétrie que la contrainte de faible latence [6]. L'architecture se compose de trois couches, une couche qui correspond aux données archivées dans une base de données NOSQL et pré-calcule des vues relatives à des requêtes souvent posées, une couche qui correspond au traitement temps-réel et une couche intermédiaire qui permet de fusionner facilement les résultats obtenus des deux couches précédentes.

Dans notre système, les reports de position s'effectuent via différents flux de données qui seront gérés sur un système temps-réel distribué. Au niveau de la gestion des données, on distingue le composant relatif au traitement *off-line* et celui relatif au traitement *on-line*.

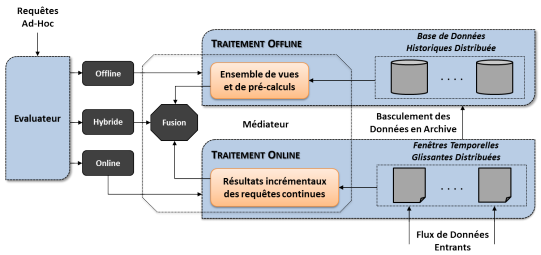


Figure 1: Principe architectural

La gestion des traitements en mémoire est faite sur une fenêtre glissante distribuée dont la taille pourra être modifiée selon le nombre de données collectées en temps-réel sur la zone de couverture concernée. Des vues *on-line* sur les requêtes continues sont mises à jour et incrémentées au gré du flux entrant de données. Si l'utilisateur exprime une requête portant sur des données n'étant pas synthétisées par le traitement continu, les données nécessaires sont accessibles via la fenêtre glissante. Une fois, que la période temporelle dédiée à la fenêtre glissante est dépassée, les données sont déplacées vers la base de données historiques distribuée pour effectuer les traitements *off-line*. Afin d'avoir un système réactif, des pré-calculs sont effectués sur les données historiques et mis à jour au fur et à mesure des arrivées en base de données.

Au niveau des requêtes deux entités sont utilisées pour identifier les données à extraire et traiter, ainsi que pour gérer les interactions entre la base de données historiques et le système de traitement temps-réel. Une de ces entités est le médiateur dont le rôle est de gérer les flux entre les composants *on-line* et *off-line*, de conserver et stocker les vues associées et de pouvoir les fusionner pour permettre de répondre aux requêtes hybrides. L'évaluateur analyse la

requête en entrée et essaie d'inférer le type de la requête, à savoir *on-line*, *off-line* ou hybride pour orienter, en fonction du type de requête identifiée, la récupération des données et des informations nécessaires dans notre architecture. Il transmet au médiateur les données désirées à traiter et ce dernier se charge de prendre, combiner ou d'effectuer des traitements sur la fenêtre temporelle glissante ou l'archive suivant la demande de l'évaluateur.

3. CONCLUSIONS

L'objectif principal de ce travail concerne la mise en place d'une architecture hybride pour la gestion et le traitement d'objets mobiles. Nous nous concentrerons en premier lieu sur la gestion des mécanismes de médiation ainsi que la distribution des données et des traitements. Le cas d'application de cette thèse, débutée en novembre 2013 (encadrée par Cyril Ray et dirigée par Christophe Claramunt), sera l'étude des positions et trajectoires de navires issues du système de positionnement AIS (Automatic Identification System). Le but final étant de traiter, stocker et analyser les positions de navires qui permettront d'obtenir des vues analytiques (multidimensionnelles) du trafic maritime et l'identification de comportements types (eg. trajectoire anormale) en temps-réel.

4. REFERENCES

[1] S. Chandrasekaran and M. Franklin. Remembrance of streams past : Overload-sensitive management of archived streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, VLDB '04, pages 348–359, 2004.

[2] V. T. de Almeida, R. H. Guting, and T. Behr. Querying moving objects in secondo. In *Proceedings of the 7th International Conference on Mobile Data Management*, MDM '06, pages 47–52. IEEE Computer Society, 2006.

[3] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. pages 319–330, 1999.

[4] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Rec.*, pages 5–14, 2003.

[5] Q. Ma, B. Y. 0002, W. Qian, and A. Zhou. Query processing of massive trajectory data based on mapreduce. In X. Meng, H. Wang, and Y. Chen, editors, *CloudDb*, pages 9–16. ACM, 2009.

[6] N. Marz. *Big data : principles and best practices of scalable realtime data systems*. O'Reilly Media, [S.l.], 2013.

[7] M. F. Mokbel, X. Xiong, M. A. Hammad, and W. G. Aref. Continuous query processing of spatio-temporal data streams in place. *Geoinformatica*, pages 343–365, 2005.

[8] K. Patroumpas. Multi-scale window specification over streaming trajectories. *J. Spatial Information Science*, pages 45–75, 2013.

[9] N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. Hermes - a framework for location-based data management. In *In Proceedings of EDBT*, pages 1130–1134, 2006.

Intentional Data Placement Policy for Improving *OLAP* Cube Construction on Hadoop Clusters

Billel ARRES
2nd year PhD student
Universite Lumiere Lyon 2
b.arres@univ-lyon2.fr

Nadia KABACHI
Advisor
Universite Lyon 1
n.kabachi@univ-lyon1.fr

Omar BOUSSAID
Director
Universite Lumiere Lyon 2
o.boussaid@univ-lyon2.fr

ABSTRACT

In the recent past, we have witnessed dramatic increases in the volume of data literally in every area: business, science, and daily life to name a few. The Hadoop framework - an open source project based on the MapReduce paradigm - is a popular choice for big data analytics. However, the performance gained from Hadoop's features is currently limited by its default block placement policy, which does not take any data characteristics into account. Indeed, the efficiency of many operations can be improved by a careful data placement, including indexing, grouping, aggregation and joins. In our work we propose a data warehouse partitioning strategy to improve query gain performances. We investigate the performance gain for OLAP cube construction with and without data organization on a Hadoop cluster. And this, by varying the number of nodes and data warehouse size. Our experiments suggest that a good data placement on a cluster during the implementation of the data warehouse can significantly increase the OLAP cube construction and querying performances. In the next step, we will extend the experiments to study the effects of other configuration parameters on collocation data in the context of parallel data warehousing, such as partitions size, replication factor and OLAP query complexity. We plan also to study an intelligent system for warehouses data placement on clusters by integrating Multi-Agent System (MAS) and Intelligent Agents to the process.

Keywords

MapReduce, HDFS, Data warehouses, Block Placement

1. RELATED WORK

In this section, we describe the MapReduce paradigm in the context of OLAP. We then discuss the Hadoop data organization enhancement techniques in brief.

1.1 MapReduce and OLAP

BDA 14 octobre 2014, Grenoble-Autrans, France.

MapReduce [9] is a framework for parallel processing of massive data sets. A job to be performed using the MapReduce framework has to be specified as two phases: the map phase as specified by a Map function, takes key/value pairs as input, performs some computation on this input, and produces intermediate results as key/value pairs; and the reduce phase which processes these results as specified by a Reduce function. The data from the map phase are shuffled, i.e., exchanged and merge-sorted, to the machines performing the reduce phase. It should be noted that the shuffle phase can itself be more time-consuming than the two others depending on network bandwidth availability and other resources.

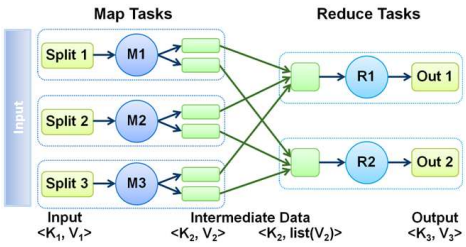


Figure 1: The MapReduce model.

MapReduce runs in cluster of nodes; one node acts as a master node (called Namenode) and other nodes act as workers (called Datanodes). It efficiently uses network bandwidth by moving computation to data. The input data is managed by a distributed file system[11] which divides input data into a set of blocks; the block size can be specified by users. In addition, it replicate each block, the default replication number is three, and puts one replica in the same rack and puts the other replica in other rack. The strategy of replica distribution helps in restoring data in case of node or rack failures. The MapReduce programming model has many advantages, such as high throughput/performance, use of commodity machines, and fault tolerance. Hence, MapReduce is used in not only index construction for search engines [9] but also data analysis of both homogeneous and heterogeneous sets [7]. Data join processing, which is very important for complex analysis in data warehouses, is addressed in [7] and [10] using MapReduce. Other works like [3] and [6] used Hadoop-based implementations, such as Hive [2] and CloudBase [6], as alternatives to relational DBMSs benchmarking and comparing different approaches to retrieve OLAP data cubes with MapReduce.

In fact, MapReduce can be useful for OLAP processing in large data warehouses. For example, Facebook has implemented a large data warehouse system using MapReduce instead of DBMS's [5]. According to [8], a data warehouse is an online repository for data from operational systems of an enterprise. It is usually maintained using a star schema that is composed of a single fact table and a number of dimension tables. A fact table contains atomic data or records for business areas such as sales and production. Dimension tables have a large number of attributes that describe records of the fact table. In the context of MapReduce all data in data warehouses are stored as a form of chunks in distributed file system [9]. The data warehouse is splited into *blocks* and distributed over the cluster nodes. However, the MapReduce data file system tries to balance the load by placing blocks randomly, independently of the intended use of the data. In this paper, we focus on the performance gain by careful data organization for star schema data warehouses.

1.2 The Hadoop Framework

Hadoop [12] is a MapReduce based framework designed for scalable and distributed computing. It consists of two main parts: the Hadoop distributed file system (HDFS) and MapReduce for distributed processing. Files in HDFS are split into a number of large blocks (usually a multiple of 64 MB) which are stored on Datanodes. A file is typically distributed over a number of Datanodes in order to facilitate high bandwidth and parallel processing. In fact, efficient access to data is an essential step for achieving improved performance during query processing which is a very important step in data warehousing context. Indeed, in contrast to many positive features of MapReduce and it's open-source implementation Hadoop, such as scalability and fault tolerance, it has some limitations, especially in data access. According to literature [1, 4], three subcategories of data access improvement exists, namely indexing, data layouts, and intentional data placement.

2. PROBLEM STATEMENT

A data warehouse is an online repository for data from operational systems of an enterprise. It is usually maintained using a star schema that is composed of a single fact table and a number of dimension tables. In the context of MapReduce all data in data warehouses are stored as a form of a chunk in distributed file system. The data warehouse is split into *blocks* and distributed over the cluster nodes. However, the Hadoop Data File System tries to balance the load by placing blocks randomly, independently of the intended use of the data. Our study focuses on the performance gained by careful data organization for star schema data warehouses. Assuming a star schema data warehouse with a fact table FF and four dimensions $D1$, $D2$, $D3$, $D4$. With the Hadoop distributed file system HDFS, all the data in the data warehouse are split into blocks of fixed size and stored on Datanodes. The block size is configurable and defaults to 64MB. So there will be FF_i, $D1_j$, $D2_k$, $D3_m$ and $D4_n$ blocks in the file system. The integers $\{i, j, k, m, n\}$ depends on the size of each table.

By default, the data placement policy of HDFS tries to balance load by placing blocks randomly on the Datanodes (Fig. 1). This default data placement policy of HDFS arbitrarily places partitions across the cluster so that mappers often have to read the corresponding partitions from remote nodes. In this case, the network overhead can be eliminated or at least reduced by colocating the corresponding partitions, i.e., storing them on the same set of Datanodes as shown in Figure 2. Furthermore, it improves the efficiency of many operations such as joins and grouping.

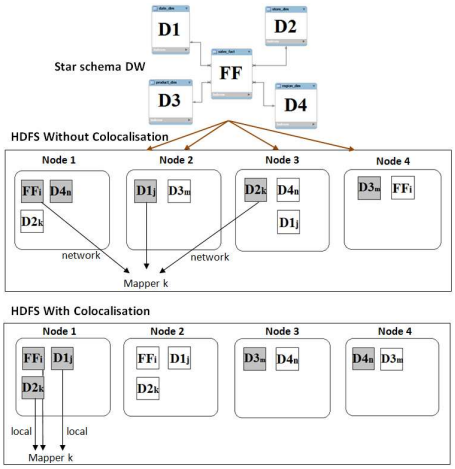


Figure 2: DW blocks processing without colocation vs. with colocation.

3. SYSTEM IMPLEMENTATION

The main idea of our work suggests that to improve data warehouse query performances, particularly OLAP queries, we must first define a good strategy for data distribution. In this case, we propose to colocate on the same set of Datanodes the fact table partitions, related attributes and dimensions partitions which are involved in the user query. We implement our data warehouse approach by using the HDFS-385 API (Version 1.2.0 released on May 2013) which is an expert-level interface for developers who want to try out custom placement algorithms to override the HDFS default placement policy. We used also a data placement mechanism called locator (M.Eltabakh, et Al. 2011). During the loading phase, each data warehouse table file is assigned to at most one locator and many tables files can be assigned to the same locator. Tables data files and partitions with the same locator are placed on the same set of Datanodes, whereas others with no locator are placed via Hadoop's default strategy. The table locator is set with default values according to the policy location initially defined.

Figure 3 shows the locator table corresponding to four tables files colocation on a cluster. Our approach was evaluated with well-known, large-scale data analysis benchmark: SSBM (Star Schema Benchmark). It is a data warehousing benchmark derived from TPC-H. The star-join query, in which the fact table is joined with one or more dimension tables, is one of the well-known queries in OLAP. In a context of parallelization and data colocation we have chosen to evaluate our approach with a star-join OLAP cube construction query that involve only two dimension tables.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

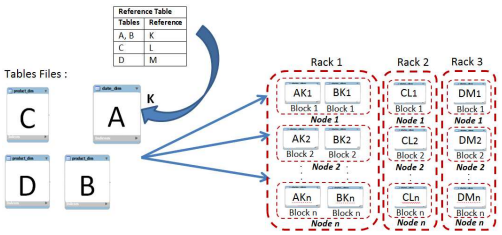


Figure 3: Example of four tables files colocated using locator on three nodes cluster.

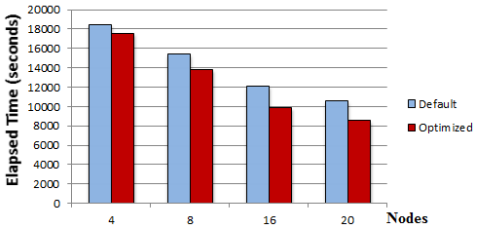


Figure 5: OLAP cube construction time by varying number of nodes. (DW size is 920GB)

In the experiments, we used a total of 20 PCs in a cluster. The operating system is Ubuntu 12.04 LTS, and the MapReduce framework is Hadoop 1.0.3. In the experiments, we compare the performances (Execution time) of the OLAP cube construction query presented previously, first, without optimization (Default) by using the Hadoop.1.0.3 version, then with optimization (Optimized) by using our HDFS extension with the same Hadoop version. We used Hive.0.10.0 for the execution of the query for both platforms.

4.2 Experimental Results

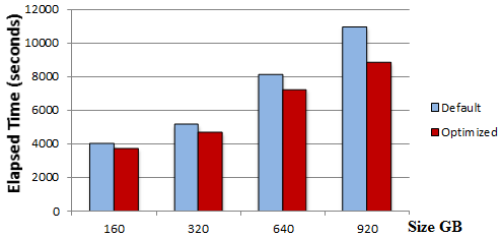


Figure 4: OLAP cube construction time by varying data warehouse size. (20 nodes cluster)

As shown in Figure 4, cube computation execution time increases significantly as the data size increases and the benefits of the proposed data warehousing colocation approach are appreciable with the increasing size compared to default HDFS data distribution. The Figure shows that tables files colocation improves the query performance from 7% for 160GB to 19% for a 920GB data warehouse size. This behaviour is expected since colocation of data, in the context of data warehousing, avoids network overhead, besides, it reduces the expensive data shuffling operation compared to default data placement policy.

In Figure 5 we observe that the execution time of cube computation decreases as the number of nodes increases. In contrast, colocating data improves query performance as the cluster getting larger. We also note that intentional data warehouse placement is more suitable for large clusters, this is because the map and reduce jobs avoids the data shuffling/sorting phase, moreover, reading the data locally is much faster than reading data over the network.

5. CONCLUSION

In this first step of work, we present a data warehouse partitioning strategy to improve query performances on a Hadoop cluster. By adopting existing colocation mechanisms, we empirically tested the performance gain for OLAP cube queries with and without data colocation.

6. REFERENCES

[1] A.Elsayed, O.Ismail, and M.E.El-Sharkawin. Mapreduce: State-of-the-art and research directions. *International Journal of Computer and Electrical Engineering*, 6(1):34–39, 2014.

[2] A.Thusoo, J.S.Sarma, N.Jain, Z.Shao, P.Chakka, S.Anthony, H.Liu, P.Wyckoff, and R.Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[3] B.Arres, N.Kabbachi, and O.Boussaid. Building olap cubes on a cloud computing environment with mapreduce. In *ACS International Conference on Computer Systems and Applications*, pages 1–5, 2013.

[4] C.Doulkeridis and K.Norvagl. A survey of large-scale analytical query processing in mapreduce. *The VLDB Journal*, 23(3):355–380, 2014.

[5] C.Monash. Cloudera presents the mapreduce bull case. *http://www.dbms2.com/2009/04/15/cloudera-presents-the-mapreduce-bull-case*, 2009.

[6] H.Han, Y.C.Lee, S.Choi, H.Y.Yeom, and A.Y.Zomaya. Cloud-aware processing of mapreduce-based olap applications. *Proceedings of the Eleventh Australasian Symposium on Parallel and Distributed Computing*, 140:31–38, 2013.

[7] H.Yang, A.Dasdan, R.-L.Hsiao, and D.S.Parker. Map-reduce-merge: simplified relational data processing on large clusters. *SIGMOD '07*, 2007.

[8] W. H. Inmon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49–50, 1996.

[9] J.Dean and S.Ghemawat. Mapreduce: simplified data processing on large clusters. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2004.

[10] R.Pike, S.Dorward, R.Griesemer, and S.Quinlan. Interpreting the data: Parallel analysis with sawzall. *Scientific Programming - Dynamic Grids and Worldwide Computing*, 13(4):277–298, 2005.

[11] S.Ghemawat, H.Gobioff, and S.T.Leung. The google file system. *the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[12] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009.

Optimisation de requêtes dans les systèmes d'intégration de données

BELGHOUL Abdeslem
Université Blaise Pascal, LIMOS, CNRS
Clermont-Ferrand, France
belghoul@isima.fr

ABSTRACT

Le concept d'intégration de données est conçu initialement pour fédérer et unifier l'accès aux diverses sources de données. La grande partie des travaux menés dans ce domaine a porté sur le traitement des problèmes d'architecture des systèmes d'intégration de données et de réécriture de requêtes (GAV, LAV, GLAV, etc.). Une réécriture de requêtes peut être vue comme un plan logique de requête auquel plusieurs plans physiques d'exécution peuvent correspondre. Si le problème de réécriture de requêtes a été intensivement étudié dans la littérature, la génération de plans physiques à partir des réécritures est encore mal maîtrisée. Nous nous intéressons, dans notre travail de recherche, au problème d'optimisation pour l'évaluation de requêtes dans les systèmes d'intégration de larges volumes de données.

1. INTRODUCTION

Le concept d'intégration de données est conçu pour fédérer et unifier l'accès à des sources de données hétérogènes. Les différents travaux de recherche menés dans ce domaine [11, 8, 4, 10] ont proposé les principaux composants de l'architecture des systèmes d'intégration de données I. Ces systèmes sont constitués d'un schéma global G, d'un ensemble de schémas de sources de données S et d'un ensemble de liens (mapping) M reliant les relations du schéma G aux relations des schémas de S.

Pour la définition des liens M, deux principales approches ont été proposées [8, 4]. La première, appelée *Global As View (GAV)*, consiste à décrire chaque élément du schéma global G à travers les schémas des sources de données de S. Quant à la deuxième approche, appelée *Local As View (LAV)*, consiste a définir les éléments des sources de données de S à travers les relations du schéma global G. Le mécanisme le plus utilisé pour matérialiser l'ensemble des liens M est celui des vues *(views)*.

La difficulté dans les systèmes d'intégration de données réside dans l'évaluation de requêtes en utilisant les liens de M. Cette difficulté se trouve dans l'identification des vues pertinentes et la génération de l'ensemble des plans logiques possibles contenant le maximum de réponse [5, 6, 9].

BDA 14 octobre 2014, Grenoble-Autrans, France.

2. PROBLÉMATIQUE

La problématique générale abordée a trait à l'optimisation de requêtes dans les systèmes d'intégration de larges volumes de données. Il est constaté que la grande partie des travaux menés dans ce domaine a traité les problèmes liés à l'architecture des systèmes (GAV, LAV, GLAV, etc.) et la réécriture de requêtes [11, 8, 4, 10, 5]. Cette réécriture, peut être perçue comme un plan logique de requête, pouvant être exécutée concrètement de plusieurs manières correspondant à différents plans physiques possibles. Ces derniers dépendent de plusieurs paramètres, tels que, l'optimiseur, les méthodes d'accès aux données, les agorithmes de jointures implémentés, etc.

Notre travail de recherche vise, d'une part, la définition d'une approche permettant de générer les différents plans physiques possibles, et d'autre part, l'élaboration d'un modèle de coût permettant de choisir le plan physique efficient. Les paramètres importants dudit modèle devraient être identifiés dans le cadre de ce travail de thèse.

3. ANALYSE DES PREMIERS RÉSULTATS

Dans l'objectif d'effectuer une première analyse sur le fonctionnement de système d'intégration de données et l'identification des paramètres impactant les performances d'exécution de requêtes, une plate-forme expérimentale a été mise en place, selon l'approche *(GAV)*. Elle est composée de trois machines virtuelles munies de système de gestion de base de données *(SGBD)* dont deux hébergeant les données et la troisième jouant le rôle de médiateur.

Notre travail est inscrit dans le cadre du projet intitulé *"Petasky" (http://com.isima.fr/Petasky)*, qui est défini et soutenu dans le cadre du défi *"Grandes masses de données scientifiques"* de la mission interdisciplinarité du *CNRS-France*. Le jeu de données utilisé provient du projet *"Large Synoptic Survey Telescope" (LSST : http://www.lsst.org/lsst/)*. Ce jeu de données est d'une taille estimée à *90 Go* et d'un nombre d'enregistrements estimé à *170 millions* répartie sur deux tables dans deux serveurs. Un protocole de tests a été élaboré afin d'étudier le fonctionnement du système d'intégration et d'évaluer les performances d'exécution de requêtes de jointure.

L'analyse des premiers résultats fait ressortir les éléments suivants :

- Le médiateur décompose la requête en un ensemble de

sous-requêtes qui sont envoyées aux sources de données concernées. Aussi, il pousse la sélection et la projection au niveau des sources distantes. Il collecte les réponses et applique un algorithme de jointure.

- La variation des paramètres liés à l'espace mémoire du SGBD, utilisé par le médiateur, a permis d'observer des changements dans les performances d'exécution sans le changement de l'algorithme de jointure appliqué. Aussi, il a été observé que le médiateur n'utilise pas le cache mémoire du SGBD.

- La variation des paramètres liés à la couche réseaux du SGBD, au niveau du médiateur, a permis de constater des changements dans les performances d'exécution.

- Le médiateur est sensible à la façon de définir les liens entre les relations du schéma global et les relations des sources distantes. Les cas suivants ont été observés :

  – Dans une liaison de type une vue (relation globale) pour une et une seule relation de source distante, le médiateur interroge la deuxième source distante autant de fois que le nombre d'enregistrements de la première source. Dans cas, l'algorithme jointure appliqué est appelé bind-join qui affiche une exécution non optimale de la requête.

  – Dans une liaison de type une vue (relation globale) composée au moins deux relations distantes et l'autre composé d'une seule relation distante, le médiateur interroge la deuxième source distante autant de fois que le nombre d'enregistrements de l'autre source. Ce plan d'exécution est le même que le plan précédent.

  – Dans une liaison de type une vue reliée à au moins à deux relations distantes, le médiateur matérialise les données de la première relation globale sur l'espace mémoire de travail et commence à lire et faire la jointure des enregistrements de la deuxième relation globale en appliquant le Hash-Join algorithme.

  – Dans une liaison de type une vue pour une et une seule relation et les vues résidant dans le même site, le médiateur pousse la jointure vers le site contenant les deux relations qui aura le choix de définir le plan physique d'exécution de la requête.

4. CONCLUSIONS ET PERSPECTIVES :

Des premiers résultats obtenus, il convient de dire que les systèmes d'intégration actuels ne sont pas adaptés à la gestion de large volumes de données. Le mode d'exécution d'une requête de jointure dans un environnement distribué est plus complexe que l'exécution d'une requête de jointure en local. En effet, l'optimisation de l'exécution d'une requête de jointure en local repose sur le choix de l'ordonnancement des relations, la méthode d'accès et l'algorithme de jointure à appliquer [1]. Quant à l'exécution dans un système d'intégration de données, en plus des éléments d'optimisation d'une requête de jointure en local, elle dépend notamment de l'environnement distribué, à titre d'exemple :

- A quel moment faut-il réaliser la jointure au niveau du médiateur ou au niveau du site distant ?

- Quel type d'algorithme de jointure et de méthode d'accès faut-il appliquer ?

- Quel type de fonction objective faut-il appliquer ?

Dans ce contexte, notre travail de recherche vise l'étude des différentes techniques d'optimisation de requêtes distribuées [3, 7]. Les auteurs de [2] considèrent que la gestion et l'évaluation de requêtes dans les systèmes d'intégration de larges volumes de données est un défi.

Pour notre travail qui s'appuie sur [3, 7, 2] , il s'agit : (i) d'identifier les différents opérateurs physiques utiles pour la construction de l'espace des solutions, (ii) d'identifier les paramètres impactant les performances d'exécution d'un plan physique de requête et l'élaboration d'un modèle de coût adapté, et (iii) la définition de stratégie d'optimisation qui s'appuient sur les éléments précédents pour générer des plans physiques avec des coûts d'exécution raisonnables.

5. REFERENCES

[1] S. Chaudhuri. An overview of query optimization in relational systems. In *In PODS*, pages 34–43, 1998.

[2] X. L. Dong and D. Srivastava. Big data integration. *Proc. VLDB Endow.*, 6(11):1188–1189, Aug. 2013.

[3] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 276–285, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[4] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 9–16. VLDB Endowment, 2006.

[5] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, Dec. 2001.

[6] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *In Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91.

[7] D. Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, Dec. 2000.

[8] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233–246, New York, NY, USA, 2002. ACM.

[9] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 251–262, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[10] J. D. Ullman. Information integration using logical views. In *Proceedings of the 6th International Conference on Database Theory*, ICDT '97, pages 19–40, London, UK, UK, 1997. Springer-Verlag.

[11] G. Wiederhold. Mediation in information systems. *ACM Comput. Surv.*, 27(2):265–267, June 1995.

Optimisation sémantique des requêtes continues Application aux bâtiments intelligents

Manel Charfi*#
*Laboratoire LIRIS, INSA-Lyon, Université de Lyon - CNRS (F)
#Laboratoire LCIS, IUT de Valence, Université de Grenoble (F)
manel.charfi@liris.cnrs.fr
Bourse financée par la région Rhône-Alpes
Début : Octobre 2013

1. CONTEXTE

Les bâtiments dits "intelligents" apparaissent aujourd'hui comme une réponse prometteuse aux enjeux sociétaux liés au bien-être des occupants et à la consommation énergétique, et plus globalement au développement durable, dans le secteur du bâtiment. Ceci explique l'apparition de nombreux projets dans ce domaine, dont plusieurs se sont intéressés à la réduction de la consommation des bâtiments (énergie primaire, fluides...). Un des enjeux actuels est de pouvoir pleinement exploiter les données issues des systèmes de contrôle des bâtiments intelligents afin de fournir des services de qualité aux occupants et aux gestionnaires des bâtiments.

Une maison intelligente peut posséder diverses fonctions selon les habitants et leurs besoins. Une telle maison contient des capteurs (température, luminosité, présence, ...) et des actionneurs (alarme, mise en marche/arrêt chauffage, ...) qui lui permettent respectivement de récupérer les données (par exemple au moyen de requêtes continues à partir des flux de données définis dans le système) et d'agir selon la situation détectée afin de réaliser un objectif donné, par exemple assurer que la température moyenne du logement d'un locataire donné ne descend pas au dessous d'un seuil qu'il aura fixé. Nous nous intéressons à l'optimisation énergétiques d'un bâtiment collectif, avec par exemple 50 locataires. Puisque chaque locataire peut définir ses préférences, nous voulons aborder le problème d'optimisation tout en satisfaisant les préférences des utilisateurs. Nous nous intéressons à un type particulier de requêtes que nous appelons requêtes de monitoring : il s'agit des requêtes continues qui durent très longtemps, aussi longtemps que dure le contrat entre un résident et le gestionnaire du bâtiment (par exemple 3 ans). Ces requêtes, qui n'ont que quelques tuples mais de nombreux effets de bord sur le bâtiment, doivent respecter un contrat donné défini par l'utilisateur qui pourrait à la fois définir ses seuils (température maximale, humidité tolérée, pièce favorite...) et les approximations qui lui conviennent (fréquence d'acquisition de la température, temps de passivité du système...)

Considérons un scénario jouant autour du locataire Bob. Par exemple, pour Bob, il faut que la température à l'intérieur de la maison ne soit pas inférieure à 15°C tout au long de son contrat de location qui dure 3 ans. Une telle contrainte n'est pas une demande qui nécessite une notification en temps réel : une notification par an pourrait être suffisante de la part du gestionnaire du bâtiment. On voit bien que, le système doit interagir avec les actionneurs pour respecter le contrat défini par l'utilisateur.

Ainsi, en cas diminution de la chaleur, le système peut fermer les fenêtres, mettre en marche le système de chauffage, ouvrir les stores... Il est aussi possible de notifier l'utilisateur afin de demander son intervention manuelle pour l'ouverture/la fermeture des stores ou des fenêtres, partant de l'idée que la notification est moins coûteuse que l'activation du levier de fermeture de la fenêtre et que Bob accepte de fermer les fenêtres.

Bob peut aussi décider que la température de chaque chambre à une seconde donnée est la même que sa température lors de la dernière heure. Grâce à une telle approximation, il sera possible d'éviter certaines pratiques courantes dans les bâtiments intelligents qui surchargent le réseau et consomment de l'énergie. En fait, il pourrait être inutile d'user les batteries des capteurs avec une telle fréquence si le service demandé par Bob ne nécessite pas une valeur du capteur chaque seconde.

2. SOLUTIONS EXISTANTES

Revenons à l'approximation définie par Bob : la température de chaque chambre à une seconde donnée est la même que sa température lors de la dernière heure. Cette approximation, qui ne tient évidemment pas dans la pratique, peut soulager la charge de travail du réseau et préserver la batterie de capteurs. La question est : comment arriver à exprimer une telle approximation afin de pouvoir l'utiliser en tant qu'une optimisation exploitable par notre système ?

Cela nous rappelle la technique de délestage ("load shedding") [7] dans les systèmes de gestion de flux, qui interfère généralement dans le plan physique de la requête. Dans notre cas, nous voulons réécrire un ensemble de requêtes continues au niveau logique à l'aide des contraintes exprimées par chaque habitant.

Dans le cadre relationnel classique, le problème est connu comme l'*optimisation sémantique des requêtes* et date des années 80 [4]. L'idée est d'exploiter une certaine information sémantique exprimée par l'utilisateur pour rendre la

requête plus efficace. A notre connaissance, l'optimisation sémantique pour les requêtes continues n'a pas été encore étudiée. Les méthodes déployées pour optimiser les requêtes sémantiquement avaient recours aux contraintes d'intégrité utilisées lors de la conception et la normalisation des schémas de la base de données. L'utilisation de ces contraintes servait à détecter les anomalies présentes dans les requêtes [3] et de mettre en place des techniques de réécriture, telle que l'ajout/la suppression des jointures/sélections [6, 5], des requêtes en question.

3. SOLUTION À APPORTER

Nous voulons intégrer les approximations de l'utilisateur dans les requêtes continues existantes dans notre bâtiment intelligent. Ces approximations seront vues comme des nouvelles contraintes définies par l'utilisateur que le système doit comprendre et intégrer dans son processus d'optimisation des requêtes continues. Il s'agit donc d'utiliser de nouvelles contraintes qui peuvent varier selon l'utilisateur et le temps contrairement aux techniques utilisées dans l'optimisation sémantique des requêtes classiques où on réutilise les contraintes statiques qui ont servi à la conception et la modélisation de la base de données. Il est alors nécessaire de trouver le bon modèle permettant de les définir et facilitant leur intégration dans le processus de l'optimisation.

La première idée simple consiste à exprimer les approximations de l'utilisateur en utilisant les dépendances fonctionnelles (DF) [2], voire les dépendances fonctionnelles conditionnelles (DFC) [1] ou plus souvent les dépendances temporelles (DFT) [8, 9]. Par exemple l'approximation de Bob pourrait être représentée par la DFT $room \rightarrow_{hour} temperature$: selon cette dépendance, la température d'une pièce donnée pendant une heure ne change pas.

La définition de notre problème devient : Étant donné un ensemble M de requêtes de monitoring et un ensemble F de dépendances (DF, DFC, DFT ou autres) : comment réécrire M en un ensemble de requêtes de monitoring M' tel que M' est équivalente à M par rapport à F.

Notre travail en cours consiste à :

1. Modéliser les approximations de l'utilisateur, les flux de données et les requêtes continues,
2. Proposer des règles de réécritures logiques pour les requêtes continues en présence de dépendances,
3. Choisir un système de gestion de flux de données (et les requêtes continues associées),
4. Implémenter un algorithme de réécriture,
5. Réaliser des tests.

4. REFERENCES

[1] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 746–755. IEEE, 2007.

[2] R. Fagin. Functional dependencies in a relational database and propositional logic. *IBM Journal of Research and Development*, 21(6) :534–544, Nov 1977.

[3] Bryan Howard Genet. *Is Semantic Query Optimization Worthwhile ?* PhD thesis, The University of Waikato, 2007.

[4] Michael Hammer and Stanley B Zdonik. Knowledge-based query processing. In *Proceedings of the sixth international conference on Very Large Data Bases-Volume 6*, pages 137–147. VLDB Endowment, 1980.

[5] Barry GT Lowden and Jerome Robinson. Constructing inter-relational rules for semantic query optimisation. In *Database and Expert Systems Applications*, pages 587–596. Springer, 2002.

[6] Sreekumar T. Shenoy and Z. Meral Ozsoyoglu. A system for semantic query optimization. *SIGMOD Rec.*, 16(3) :181–195, December 1987.

[7] Nesime Tatbul, Uğur Çetintemel, Stan Zdonik, Mitch Cherniack, and Michael Stonebraker. Load shedding in a data stream manager. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 309–320. VLDB Endowment, 2003.

[8] X Sean Wang, Claudio Bettini, Alexander Brodsky, and Sushil Jajodia. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems (TODS)*, 22(2) :115–170, 1997.

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

Multisite Management of Data-intensive Scientific Workflows in the Cloud

Ji Liu
ji.liu@inria.fr
Thesis Start Date: 01/10/2013
MSR-INRIA Joint Centre, Paris, France
LIRMM, University Monpellier 2, Montpellier, France

1. INTRODUCTION

Scientific experiments generally contain multiple computational activities to process experimental data and these activities are related by data or control dependencies. Scientific workflows enable scientists to model the data processing of these experiments as a graph, in which vertexes represent data processing activities and edges represent dependencies between them. Since scientific workflow activities may process big data, one scientific workflow activity can correspond to several executable tasks for different parts of input data during scientific workflow execution. As big amounts of data are handled or produced during the scientific workflow execution, data-intensive scientific workflows become an important issue.

A Scientific Workflow Management System (SWfMS) is an efficient tool to execute scientific workflows. In order to execute a data-intensive scientific workflow within reasonable time, SWfMSs generally exploit High Performance Computing (HPC) resources in a cluster, grid or cloud environment. Because of virtually infinite resources, diverse scalable services, stable service quality and flexible payment policies, cloud become a primary solution for workflow execution.

Due to the geographic distribution of scientists, data and computing resources, multisite cloud is appealing for data-intensive scientific workflow execution. A multisite cloud contains multiple data centers in multiple cloud sites and each data center is explicitly accessible to cloud users. Explicitly accessible has two meanings. The first meaning is that each cloud site is separately visible and accessible to cloud users. The other meaning is that cloud providers will not change the data location, i.e. the data of users will not be moved from one cloud site to another cloud site without the permission of users. In this case, SWfMSs need to meet the challenges of scheduling data and executable tasks to computing nodes across multiple cloud sites. In addition, SWfMSs have to efficiently transfer data within one site or across different sites at this situation. The main objective of this thesis is to propose efficient approaches to execute data-intensive scientific workflows in a multisite cloud.

BDA 14 octobre 2014, Grenoble-Autrans, France.

The current solutions for the parallel execution of scientific workflows are appropriate for static computing and storage resources in a grid environment. They have been extended to deal with more elastic resources in a cloud, but with only one site. Our analysis [1] of the current techniques of scientific workflow parallelization and scientific workflow execution has shown that there is a lot of room for improvement in the following directions:

1. Data staging: existing techniques mainly focus on the mechanism that starts scientific workflow execution after gathering all the related data in a shared-disk file system at one data center, which is time consuming.

2. Architecture: the structure of SWfMSs is generally centralized, with a master node, which is a single point of failure and performance bottleneck, managing all the optimization and scheduling processes.

3. Task scheduling and data location: most SWfMSs do not take data location into account during task scheduling, which makes it inefficient to read or write data.

4. Multisite: novel task and data scheduling approaches are required for utilizing resources in a multisite cloud.

In the rest of this paper, we define more precisely the problem and introduce our approach to address it.

2. PROBLEM DEFINITION

We consider the problem of executing scientific workflows in a multisite cloud. A scientific workflow representation file is stored in a Virtual Machine (VM) of a cloud site while workflow data may be distributed in different cloud sites because of the geographical distribution of scientists. The workflow data includes the data to be processed by a scientific workflow and the instruction data, i.e. the instructions of the programs in the scientific workflow. Some data can be transferred between different sites while some other data is fixed at a specific site and cannot be moved to another site because of big data or security restrictions.

Generally, the execution of scientific workflows is realized by the VMs at each site. The VMs in each site are able to communicate with each other with appropriate configuration, e.g. public IP address, port number etc. The data transfer rate, however, is different in different situations, e.g. intersite communication, intrasite communication, the VMs that share storage resources and so on. Moreover, the VMs and required storage resources can be created before

workflow execution or dynamically created during workflow execution.

We formulate the problems we address in this thesis as follows. Given a scientific workflow W, which is composed of activities V and dependencies E, and a multisite cloud MS, in which each site stores its own data (input data for W or instruction data of W) and there are available computing and storage resources to execute the scientific workflow, how to optimize and schedule W in MS in a way that achieves required objectives while respecting additional limitations, e.g. security restrictions, budget limitations etc. This problem can be divided into two sub-problems:

Workflow Parallelization Given a scientific workflow $W = \{V,E\}$ and a multisite cloud $MS = \{S_1, S_2, ..., S_n\}$, how to efficiently parallelize scientific workflow execution and schedule each part of W into available VMs at each site with consideration of activity dependencies in E and special features of each site to achieve required objectives, e.g. minimizing execution time, reducing monetary cost etc.

Data Management Given the data in a scientific workflow, how to manage it in a multisite management environment to achieve required objectives, e.g. minimizing the time to get data or reducing data storage cost etc.

The data can be stored in the local disk of a VM, a global file system for all the VMs at one site or a global file system for all the VMs at multiple sites. Thus, it is important to generate appropriate strategies to transfer and store data among different cloud sites during scientific workflow execution. In addition, since there are big data produced during workflow execution, it is also helpful to support dynamic elastic storage resource provisioning, i.e. data storage resources can be inserted to or removed from VMs during workflow execution.

3. PROPOSED APPROACH

Workflow parallelization is the process of transforming and optimizing a (sequential) workflow into a parallel WEP, which consists of workflow parallelism and workflow scheduling. The WEP is a program that captures optimization decisions and execution directives, typically the result of compiling and optimizing a workflow. Similar to the concept of Query Execution Plan (QEP) in distributed database systems [6], it allows the SWfMS to execute scientific workflows in parallel in multiple computing nodes.

A SWfMS can exploit activity parallelism or data parallelism to execute a scientific workflow. Activity parallelism makes the execution of different activities run on different computing nodes at the same time. Activity parallelism includes independent parallelism and pipeline parallelism. Independent parallelism achieves parallel execution for the activities that have few data or control dependencies between them. Pipeline parallelism is for executing dependent activities, where one may process the output data of another, in different computing nodes at the same time. Data parallelism is obtained by having multiple tasks performing the same activity, each on a different data chunks while the tasks can be simultaneously executed in different computing nodes.

A SWfMS can exploit all the possible types of parallelism, i.e. independent parallelism, pipeline parallelism and data parallelism to achieve efficient parallel execution of scientific workflows. First, a SWfMS can partition a scientific workflow into workflow fragments and distribute each workflow fragment into appropriate sites, at which input data is stored. By minimizing data transfer volumes between different workflow fragment, workflow partitioning can yield good performance [2]. Scientific workflow partitioning partially realizes the independent parallelism across multiple cloud sites. Second, the identification of independent activities achieves complete independent parallelism within one cloud site. Third, a SWfMS can achieve pipeline parallelism by allocating dependent activities in different computing nodes and by making them process different parts of data at the same time. Finally, the parallel execution of one activity can be achieved by distributing executable tasks of this activity into different VMs. To achieve these different kinds of parallelism, an algebraic approach is a good solution because of its powerful operators and possible optimization for the entire algebraic workflow expression [4]. Similar to multisite query processing in distributed and parallel database systems [6, 7], we intend to adapt the algebraic approach to a multisite cloud.

After devising a parallel execution strategy for a scientific workflow, a SWfMS should schedule executable tasks to available computing nodes through a static, dynamic or hybrid method, i.e. the combination of static and dynamic methods. We argue that SWfMSs can make a static WEP to specify a static scheduling strategy before execution. During workflow execution, if the execution environment varies a lot and the execution cannot achieve load balancing through the original WEP, the SWfMS can exploit dynamic scheduling according to the run-time environment features. Furthermore, SWfMSs can dynamically insert or remove VMs to achieve multiple objectives, e.g. budget limit, time limit etc [5].

For managing data, most existing SWfMSs exploit a shared-disk file system and perform small optimization to store data. In order to achieve better performance, we propose to schedule tasks according to data location while using local storage resources across the VMs in multiple cloud sites for data-intensive scientific workflow execution. Since the activities in a data-intensive scientific workflow may process massive data sets, we believe this co-scheduling method can accelerate scientific workflow execution and avoid useless data transfer cost among different VMs located at different sites.

Furthermore, the data transfer between different sites can be directly achieved through multiple computing nodes at each site while data streams do not need to pass through a centralized computing node during execution. This solution can avoid the communication bottleneck caused by a centralized architecture.

Finally, we plan to validate our solutions with BlobSeer [3], a distributed file system, on Microsoft Azure. We will adapt BlobSeer to multisite cloud and optimize the communication methods between different sites and the cooperation between BlobSeer and SWfMSs.

4. ACKNOWLEDGMENT

The thesis is advised by Esther Pacitti (University of Montpellier), Patrick Valduriez (INRIA) at LIRMM and Marta Mattoso (UFRJ).

The work is partially funded by Microsoft (Zcloudflow project) and CNPq, FAPERJ, INRIA (HOSCAR and MUSIC projects, and performed in the context of the Computational Biology Institute (`www.ibc-montpellier.fr`).

5. REFERENCES

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

[2] J. Liu, V. Silva, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow partitioning in multi-site clouds. In *Euro-Par 2014: Parallel Processing Workshops*, pages 1–12, 2014.

[3] B. Nicolae, G. Antoniu, L. Bougé, D. Moise, and A. Carpen-Amarie. Blobseer: Next-generation data management for large scale infrastructures. *Journal of Parallel and Distributed Computing*, 71(2):169–184, 2011.

[4] E. Ogasawara, D. Oliveira, P. Valduriez, J. Dias, F. Porto, and M. Mattoso. An algebraic approach for data-centric scientific workflows. *Proceedings of the VLDB Endowment (PVLDB)*, 4(12):1328–1339, 2011.

[5] D. Oliveira, K.A.C.S. Ocaña, E. Ogasawara, J. Dias, J. Gonçalves, F. Baião, and M. Mattoso. Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows. *Future Generation Computer Systems*, 29(7):1816–1825, 2013.

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

[7] E. Pacitti, R. Akbarinia, and M.E. Dick. *P2P Techniques for Decentralized Applications*. Morgan & Claypool Publishers, 2012.

An Upper Bound on the Number of Accesses for $Datalog^{\alpha_{Last}}$ Queries

John Samuel [*]
LIMOS, CNRS Université Blaise Pascal
Aubière, France
samuel@isima.fr

Benjamin Momège [†]
LIMOS, CNRS Université Blaise Pascal
Aubière, France
momege@isima.fr

ABSTRACT

In the mediation approach for data integration, domain rules [2, 3] were previously proposed to deal with access limitations (aka access patterns). For data integration systems (e.g., DaWeS [6, 5]) that use domain rules, we study an upper bound on the possible number of accesses implied by the evaluation of an executable query (expressed with relations having access patterns). Indeed it allows to compare various evaluation algorithms, to schedule API operation calls and meet the service level agreements (SLA) of the service providers.

1. BOUNDING THE NUMBER OF ACCESSES FOR $DATALOG^{\alpha_{LAST}}$ QUERIES

Web service providers introduce access patterns in a manner that a complete query response isn't obtained in a single operation call, but rather it mandates multiple operation calls (accesses). For our previous work DaWeS (Data Warehouse fed with Web Services) ([6, 5]), it is important to optimize (i.e. reduce) the number of accesses prior to actually making them. In our work, we introduce $Datalog^{\alpha_{Last}}$ (datalog query with access pattern on the last atom in bodies of rules) similar to the one found in [2, 3], study its operational semantics and compute the upper bound on the number of accesses. Classical inverse query rewriting algorithm doesn't take into consideration data dependencies existing among the web service API operations (relations with access patterns). In DaWeS, we explore such dependencies and study the upper bound defined in terms of source relations to compare various optimization heuristics.

Consider for example, a $Datalog^{\alpha_{Last}}$ query Φ (q is the query predicate):

$\phi_1 : dom_{X_1}(X_1) \leftarrow r_2^o(X_1)$
$\phi_2 : dom_{X_2}(X_2) \leftarrow r_1^o(X_2)$
$\phi_3 : dom_{X_3}(X_3) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_4 : dom_{X_4}(X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_5 : dom_{X_5}(X_5) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_6 : dom_{X_1}(X_1) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_7 : rr_1(X_2) \leftarrow r_1^o(X_2)$
$\phi_8 : rr_2(X_1) \leftarrow r_2^o(X_1)$
$\phi_9 : rr_3(X_1, X_2, X_3, X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_{10} : rr_4(X_4, X_5, X_1) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_{11} : q(X_1, X_3, X_5) \leftarrow rr_1(X_2), rr_2(X_1), rr_3(X_1, X_2, X_3, X_4), rr_4(X_4, X_5, X_1)$

A naive datalog query evaluation of the above program Φ will iterate through every $\phi_j, 1 \leq j \leq 11$ until a fixpoint is obtained. $\mathcal{D}_0$ is the initial database and $\mathcal{D}_i, 1 \leq i \leq n_0$ is the new database obtained after every iteration. For web services API, $|r_3^{iioo}(X_1, X_2, X_3, X_4)|$ or $|r_3^{iioo}(\mathcal{D}_0)|$ can be obtained (e.g., total number of search results) but $|r_3^{iioo}[X_4]|$, (i.e., the number of X_4 attributes in r_3) cannot be obtained. We compute the total number of accesses for all iterations and all queries ϕ_j as:

$$|Accesses(\Phi, \mathcal{D}_0)| = \sum_{i=0}^{n_0} \sum_{j=1}^{11} |Accesses(\phi_j, \mathcal{D}_i)|$$
$$\leq 4n_0 + 3n_0^3 \times (|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + 3 \times n_0^2 \times |r_3^{iioo}(\mathcal{D}_0)|$$

where n_0 is the number of the last iteration (during which the fixpoint is generated). If we consider n_0 as a parameter, we can conclude that:

$|Accesses(\Phi, \mathcal{D}_0)| =$
$\mathcal{O}\left((|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + |r_3^{iioo}(\mathcal{D}_0)|\right)$ ■

Discussion: Web service API operation calls are expensive and must be reduced. An upper bound as shown above is useful to compare various optimization [3, 1] techniques on a $Datalog^{\alpha_{Last}}$ program. For example, with cache rules [3] optimization, same accesses are not repeated again, as seen in the following excerpt (after transforming the above program):

...
$\phi_9 : rr_3(X_1, X_2, X_3, X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_3 : dom_{X_3}(X_3) \leftarrow rr_3(X_1, X_2, X_3, X_4)$

[*]Current Affiliation: Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR-CNRS 5205, Labex IMU, Laboratoire d'Informatique en Image et Systèmes d'Information
Email: john.samuel@liris.cnrs.fr

[†]Current Affiliation: Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
INRIA, France
Email: benjamin.momege@inria.fr

BDA 14 octobre 2014, Grenoble-Autrans, France.

$\phi_4 : dom_{X_4}(X_4) \leftarrow rr_3(X_1, X_2, X_3, X_4)$

...

The total number of accesses in this case is given by:

$|Accesses(\Phi, \mathcal{D}_0)| \leq 2n_0 + n_0^3 \times (|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + n_0^2 \times |r_3^{iioo}(\mathcal{D}_0)|$.

n_0 as a parameter is a safe assumption for two reasons. Firstly the overall goal is to recognize the products in the bound since they are usually responsible for the majority of accesses. Secondly in our tests, for different relation sizes and randomly generated data, n_0 generally stayed constant (3 for above e.g.).

2. ACKNOWLEDGMENTS

We acknowledge Christophe Rey for his direction. We would like to thank Conseil General of the Region of Auvergne (France), CNRS and FEDER for funding our respective PhD Theses (*Feeding a Data Warehouse with Data coming from Web Services [4]* and *Paths in graphs with forbidden transitions*).

3. REFERENCES

[1] Andrea Calì and Davide Martinenghi. Querying data under access limitations. In *ICDE*, pages 50–59, 2008.

[2] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy. Recursive query plans for data integration. *J. Log. Program.*, 43(1):49–73, 2000.

[3] Chen Li and Edward Y. Chang. Query planning with limited source capabilities. In *ICDE*, pages 401–412, 2000.

[4] John Samuel. *Feeding a data warehouse with data coming from web services. A mediation approach for the DaWeS prototype*. PhD thesis, Université Blaise Pascal, 2014. Thèse de doctorat dirigée par Toumani, Farouk et Rey, Christophe Informatique Clermont-Ferrand 2 2014.

[5] John Samuel. Towards a data warehouse fed with web services. In *ESWC PhD Symposium*, 2014.

[6] John Samuel and Christophe Rey. Dawes: Data warehouse fed with web services. In *INFORSID*, 2014.

APPENDIX

Une Borne Supérieure sur le Nombre d'accès pour les Requêtes $Datalog^{\alpha_{Last}}$

A. RÉSUMÉ

Dans l'approche médiation pour l'intégration de données les règles de domaine ont été proposées [2, 3] pour prendre en compte ces sources. Pour les systèmes d'intégration qui utilisent les règles de domaine (par exemple DaWeS [6, 5]), il est utile de pouvoir calculer une borne supérieure du nombre d'accès impliqués par l'évaluation d'une requête exécutable (exprimée avec des relations limitées en accès). Ainsi cette borne supérieure permet de comparer les différents algorithmes d'évaluation de requêtes exécutables. Ceci permet une meilleure planification des appels de service et donc une plus grande facilité pour gérer les contraintes de qualité de service (service level agreements, SLA) imposées par leurs fournisseurs.

RQL : un langage “à la SQL” pour découvrir des règles à partir des données

Brice Chardin
LIAS, ISAE-ENSMA
France

Emmanuel Coquery
Université Lyon 1
CNRS, LIRIS, UMR 5205
France

Marie Pailloux
Université Blaise Pascal
CNRS, LIMOS, UMR 6158
France

Jean-Marc Petit
INSA Lyon
CNRS, LIRIS, UMR 5205
France

ABSTRACT

RQL (pour *Rule Query Language*) est un langage de requêtes “à la SQL” qui étend et généralise les dépendances fonctionnelles [1] à de nouvelles catégories de règles. RQL apporte aux analystes de données un outil pratique pour découvrir les implications logiques entre attributs d'une base de données. Ces implications peuvent mettre en évidence des problèmes de qualité de données ou de nouvelles corrélations inattendues entre les attributs. Le traitement de ces requêtes RQL est basé sur une technique de réécriture qui délègue un maximum de calculs au SGBD sous-jacent. Cette contribution vise à renforcer le lien entre la fouille de données et les bases de données et de faciliter l'utilisation de techniques de fouille par des analystes ou des étudiants habitués au SQL.

1. INTRODUCTION

La fouille de motifs peut être vue comme une partie automatisée de l'exploration de données. Par exemple, les dépendances fonctionnelles ou les dépendances fonctionnelles conditionnelles sont particulièrement utiles pour comprendre les données et identifier des problèmes de qualité [5]. Cependant, les techniques de fouille de motifs sont rarement utilisables directement par les analystes. La plupart du temps, ils ont à réaliser du pré-traitement entre différents systèmes et formats. Les codes de fouille eux-mêmes nécessitent souvent d'être compilés à partir de langages de programmation spécifiques. Toutes ces étapes sont hors de portées pour de nombreux analystes, tournant le procédé de bout en bout en cauchemar. La génération automatique de règles peut également submerger l'analyste par une quantité énorme de motifs, et rendre l'extraction d'information utile difficile. D'autres techniques ont été proposées pour interagir avec les données et fournir des retours utiles à l'analyste.

1. Nous utiliserons indifféremment les termes *règles*, *implications*, *dépendances* et *motifs* par la suite.

BDA 14 octobre 2014, Grenoble-Autrans, France.

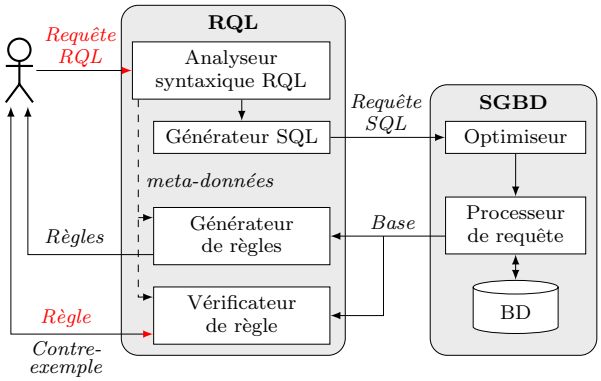


Figure 2: Traitement des requêtes RQL

Contribution de la démonstration Pour faciliter l'utilisation des techniques de fouille de motifs pour l'exploration de données, nous introduisons un langage de requête pour les règles, RQL (pour *Rule Query Language*), qui permet aux analystes habitués au langage SQL d'utiliser les techniques de fouille de motifs à l'aide une interface interactive et simple à utiliser. Dans cette démonstration, nous montrons l'utilisation de cette interface Web du point de vue de ces analystes. Nous nous focalisons sur l'expressivité de RQL à travers divers exemples, montrant ainsi la facilité de concevoir de nouvelles règles avec un langage très simple dérivé de SQL. Nous introduisons également comment les analystes peuvent interagir avec le système par l'intermédiaire des requêtes RQL et des contre-exemples issus de la base de données. Au cours de la démonstration, les participants sont invités à formuler leurs propres requêtes sur des bases de données prédéfinies pour découvrir les relations entre les attributs à l'aide des règles générées et des contre-exemples.

La figure 1 donne un aperçu de l'interface Web [2] pour RQL, disponible pour la recherche et l'éducation. Cette interface fournit un accès unifié aux données de l'utilisateur et aux techniques de fouille de motifs en utilisant deux langages déclaratifs : SQL et RQL.

À partir des travaux précédents [1, 2, 6], RQL vérifie les

2. `http://rql.insa-lyon.fr`

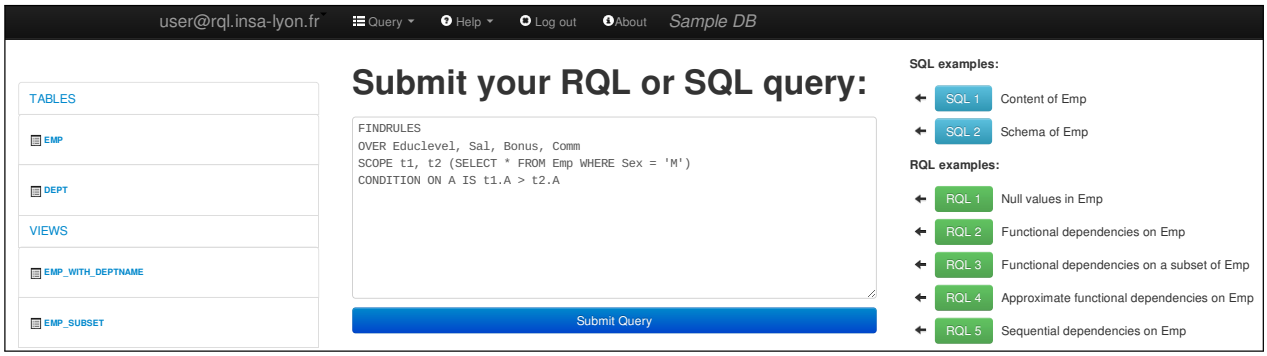


Figure 1: Interface Web pour RQL

axiomes d'Armstrong, i.e. le langage généralise les dépendances fonctionnelles à une nouvelle classe de dépendances basée sur les implications logiques (expressions de type : si ... alors). Nous avons prouvé que ces dépendances peuvent être calculées efficacement à partir de la base de données à l'aide d'un traitement en deux étapes. Premièrement, une requête SQL non triviale est générée pour calculer la *base* du système de fermeture associé aux implications recherchées. Cette base est également appelée un *contexte* dans la terminologie de l'analyse de concepts formels [8]. Ensuite, un algorithme tiré de l'état de l'art [12] est utilisé pour générer une couverture canonique des règles à partir de cette base. Cette approche permet à RQL de bénéficier de l'optimisation des requêtes du SGBD pour accéder aux données. Outre l'efficacité de l'optimisation, cette approche évite l'écueil de changer de système pour les analystes contribuant à leur simplifier leur tâche de découverte.

La figure 2 donne un aperçu de cette architecture vis-à-vis du traitement des requêtes RQL. Pour les requêtes SQL, l'application se contente de les faire suivre au SGBD sous-jacent, ce qui rend la transition entre SQL et RQL transparente pour l'utilisateur. L'objectif final de ce travail est d'intégrer les techniques de fouille de motifs au cœur des SGBDs [14].

Travaux connexes Définir des langages dédiés à la fouille de motifs est un objectif poursuivi de longue date [3], par exemple en utilisant des techniques de programmation par contrainte [10]. Cependant, nous soutenons que ces langages devraient bénéficier d'extensions directes du SQL, car les données sont souvent gérées par des SGBDs. D'autres approches pratiques, aussi proches que possible des SGBDs, ont été proposées pour interagir plus directement avec leurs moteurs de requête [7, 15, 4].

2. LE LANGAGE RQL

Pour illustrer le langage RQL, nous considérons l'exemple donné en figure 3 avec la relation *EMP*. L'attribut *Educlevel* représente le nombre d'années d'éducation formelle, *Sal* le salaire annuel, *Bonus* le bonus annuel et *Comm* la commission annuelle. La signification des autres attributs est immédiate.

Pour commencer, nous souhaitons extraire les dépendances fonctionnelles (DF) de la relation *Emp*. Pour mémoire, une DF $X \rightarrow Y$ est vérifiée dans r si pour tout tuple $t1, t2 \in r$, et pour tout attribut $A \in X$ tel que $t1[A] = t2[A]$ alors pour tout $A \in Y$, $t1[A] = t2[A]$. Avec RQL, les DFs sont exprimées d'une manière similaire.

Exemple 1. Q_1 découvre les DFs de *Emp* sur un sous-ensemble d'attributs.

```
Q1 : FINDRULES
     OVER Empno, Lastname, Workdept, Job,
          Sex, Bonus, Mgrno
     SCOPE t1, t2 Emp
     CONDITION ON $A IS t1.$A = t2.$A
```

À noter comment la clause `CONDITION` correspond à l'implication logique précédente. Cet exemple restreint aussi la découverte de DFs sur un sous-ensemble de sept attributs dans la clause `OVER`. Dans cet exemple, une couverture canonique des DFs vérifiés dans *Emp* est générée (composée de vingt-quatre DFs), dont les DFs *Empno* $\rightarrow$ *Lastname* ou *Workdept* $\rightarrow$ *Job*.

Plus précisément, une requête RQL possède la forme suivante :

```
FINDRULES
OVER [ensemble d'attributs : A1, ..., An]
SCOPE [variable de tuple : t1, ..., tn]
WHERE [condition sur (t1, ..., tn)]
CONDITION ON [variable d'attribut : $A]
  IS [condition sur ($A, t1, ..., tn)]
```

Le mot-clé `FINDRULES` identifie une requête RQL, qui génère des règles de la forme $X \rightarrow Y$ avec X et Y des ensembles disjoints d'attributs issus de la clause `OVER`. La clause `SCOPE` définit des variables de tuples sur des relations obtenues par des requêtes SQL classiques. Une clause optionnelle `WHERE` définit les relations entre variables de tuples, de manière similaire à la clause SQL `WHERE`. La clause `CONDITION ON $A` définit le prédicat devant être vérifié pour chaque attribut `$A` apparaissant dans la partie gauche et la partie droite des règles.

Pour illustrer l'expressivité des requêtes RQL, nous fournissons maintenant plusieurs exemples.

Exemple 2. Considérons les valeurs nulles (*null*), fréquentes dans les bases de données. Avec RQL, l'analyste a l'opportunité de découvrir des relations d'implication entre attributs par rapport aux valeurs nulles, en utilisant par exemple la requête Q_2.

EMP	Empno	Lastname	Workdept	Job	Educlevel	Sex	Sal	Bonus	Comm	Mgrno
	10	SPEN	C01	FINANCE	18	F	52750	500	4220	20
	20	THOMP	-	MANAGER	18	M	41250	800	3300	-
	30	KWAN	-	FINANCE	20	F	38250	500	3060	10
	50	GEYER	-	MANAGER	16	M	40175	700	3214	20
	60	STERN	D21	SALE	14	M	32250	500	2580	30
	70	PULASKI	D21	SALE	16	F	36170	700	2893	100
	90	HENDER	D21	SALE	17	F	29750	500	2380	10
	100	SPEN	C01	FINANCE	18	M	26150	800	2092	20

Figure 3: Exemple de relation

Q_2 :
```
FINDRULES
  OVER Empno, Lastname, Workdept, Job,
       Sex, Bonus, Mgrno
  SCOPE t1 Emp
  CONDITION ON $A IS t1.$A IS NULL
```

La règle *Mgrno* → *Workdept* est vérifié dans *Emp* car à chaque fois que l'attribut *Mgrno* possède une valeur nulle, alors *Workdept* est également nul pour le même tuple (seul l'employé No. 20 dans cet exemple).

À noter que la différence entre Q_1 et Q_2 provient naturellement du prédicat à évaluer, mais également du nombre de variables de tuples nécéssaires. Le prédicat de Q_1 est évalué sur des paires de tuples, tandis que Q_2 considère chaque tuple individuellement.

Exemple 3. La requête Q_1' restreint la portée de Q_1, amenant à la notion de dépendance fonctionnelle conditionnelle [5] en ne considérant que les employés avec un niveau d'éducation supérieur à 16.

Q_1' :
```
FINDRULES
  OVER Empno, Lastname, Workdept, Job,
       Sex, Bonus
  SCOPE t1, t2 (SELECT * FROM Emp
        WHERE Educlevel > 16)
  CONDITION ON $A IS t1.$A = t2.$A
```

Ici, *Sex* → *Bonus* est vérifiée avec cette restriction, ce qui signifie qu'à partir d'un certain niveau d'éducation (16), le sexe détermine le bonus.

Exemple 4. La requête Q_1'' est une approximation de Q_1 pour les valeurs numériques, de manière similaire aux dépendances fonctionnelles métriques [11], où l'égalité stricte est assouplie pour prendre en compte des variations inférieures à 10%. Par exemple, les salaires 41250 et 38250 sont considérés proches (avec une différence de 7.5%), mais pas les salaires 41250 et 36170 (différence de 13.1%).

Q_1'' :
```
FINDRULES
  OVER Educlevel, Sal, Bonus, Comm
  SCOPE t1, t2 Emp
  CONDITION ON $A IS
    2*ABS(t1.$A-t2.$A)/(t1.$A+t2.$A)<0.1
```

Dans ce cas, *Sal* → *Comm* est vérifié, ce qui signifie que les employés percevant un salaire similaire reçoivent des commissions équivalentes.

Les exemples montrés jusqu'à présent sont inspirés des implications (en FCA) et des dépendances fonctionnelles (en BD). Cependant, RQL n'est pas limité à cet type de requête et peut servir à exprimer de nombreuses autres règles.

Exemple 5. supposons que nous soyons intéressés par un type de dépendance séquentielle [9], i.e. des dépendances montrant un comportement similaire entre les attributs. Q_3 découvre des attributs numériques qui varient conjointement (i.e., $X \rightarrow Y$ signifie que si X croit, alors Y croit également).

Q_3 :
```
FINDRULES
  OVER Educlevel, Sal, Bonus, Comm
  SCOPE t1, t2 Emp
  CONDITION ON $A IS t1.$A > t2.$A
```

Sal → *Comm* et *Comm* → *Sal* sont vérifiées dans *Emp*, ce qui signifie qu'un salaire plus important correspond toujours à une commission plus élevée.

Exemple 6. En continuant l'exemple précédent, on suppose que l'analyste souhaite se concentrer sur les employés de sexe masculin.

Q_3' :
```
FINDRULES
  OVER Educlevel, Sal, Bonus, Comm
  SCOPE t1, t2 (SELECT * FROM Emp
        WHERE Sex='M')
  CONDITION ON $A IS t1.$A > t2.$A
```

Dans ce cas, *Educlevel* → *Bonus* est vérifiée, ce qui signifie que les hommes avec un niveau d'éducation plus élevé reçoivent des bonus plus importants.

Exemple 7. Au lieu de réduire la portée d'une requête, les conditions définies par l'utilisateur peuvent lier les variables de tuple entre elles par une relation ad-hoc définie dans la clause `WHERE` de la requête RQL. Par exemple, Q_4 découvre des disparités entre les managers et leurs employés, i.e. des règles sur les attributs pour lesquels les managers possèdent des valeurs supérieures ou égales à leurs employés.

Q_4 :
```
FINDRULES
  OVER Educlevel, Sal, Bonus, Comm
  SCOPE t1, t2 Emp
  WHERE t1.Empno = t2.Mgrno
  CONDITION ON $A IS t1.$A >= t2.$A
```

Dans cet exemple, $\emptyset \rightarrow$ *Bonus* est vérifiée dans *Emp*, ce qui signifie que les managers reçoivent toujours un bonus supérieur ou égal à leurs employés.

3. RETOURS AVEC CONTRE-EXEMPLES

Étant donné une requête RQL, un analyste peut – en plus de générer une couverture canonique – interagir avec le système pour vérifier si une règle est valide ou non. Il peut fournir une règle au système et deux cas sont envisageables : soit une règle est vérifiée et l'analyste est notifié que cette

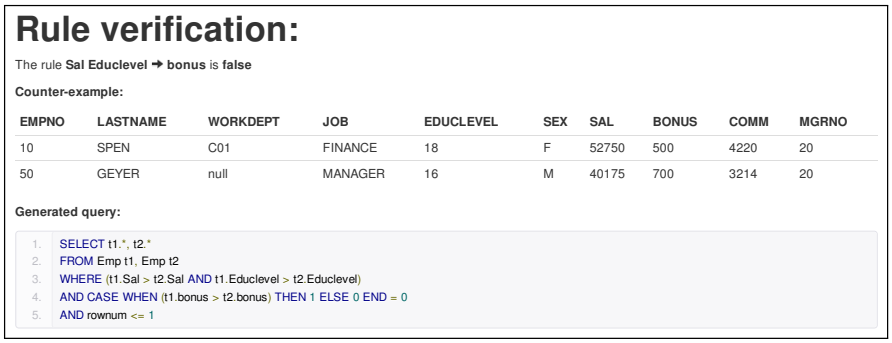


Figure 4: Génération de contre-exemples avec RQL

règle est valide, soit la règle n'est pas vérifiée ce qui signifie qu'au moins un contre-exemple existe et l'un d'eux est fourni par le système. Cette notion de contre-exemple est notamment utilisée depuis longtemps pour les dépendances fonctionnelles et les relations d'Armstrong. Ces relations exemples donnent un retour très intéressant pour l'analyste sur ses propres données (comme ce fut le cas pour la conception par l'exemple des bases de données dans les années 80). Avec RQL, le nombre de tuples requis pour fournir un contre-exemple dépend du nombre de variables de tuple de la requête. Pour Q_3, il faut au moins deux tuples, alors que pour Q_2, un tuple (par exemple l'employé No. 30) suffit pour prouver par exemple que la règle $Workdept \rightarrow Mgrno$ est fausse.

Pour illustrer les contre-exemples, supposons que l'analyste souhaite explorer son hypothèse comme quoi un salaire et un niveau d'éducation plus élevés amènent à des bonus plus importants ($Salary$, $Educlevel \rightarrow Bonus$), en utilisant Q_3. La figure 4 donne un aperçu du retour fourni par RQL comme contre-exemple à cette règle, i.e. deux tuples pour lesquels l'un (employé No. 10) possède un salaire et un niveau d'éducation plus élevé que l'autre (employé No. 50), mais pas un bonus plus important.

Avec ce contre-exemple comme point de départ, et en particulier la requête SQL générée pour l'extraire de la base de donnée, l'analyste pour rapidement passer au langage SQL afin de comprendre pourquoi cette règle n'est pas vérifiée. Dans cet exemple, les employés qui sont soit de sexe féminin soit travaillant dans la finance sont rapidement mis en évidence comme ayant un meilleur salaire et un meilleur niveau d'éducation, mais pourtant un bonus plus bas que les autres. L'analyste peut ensuite affiner sa requête RQL, par exemple en réduisant la portée des données, comme pour Q'_3 où un meilleur niveau d'éducation suffit à entraîner un bonus plus important.

Nous sommes convaincus que les contre-exemples constituent un outil indispensable pour aider l'analyste à comprendre pourquoi une règle n'est pas vérifiée, et affiner son analyse si nécessaire en suivant un processus itératif.

4. IMPLÉMENTATION ET APPLICATION

L'application Web RQL a été implémentée en Java à l'aide du Framework Play [16]. Des outils externes ont été utilisés pour la partie la plus coûteuse du processus de génération de règles : l'énumération des transversaux minimaux d'un hypergraphe [13]. Le SGBD choisi est Oracle 11g Release 2.

L'interface fournit les requêtes SQL générées par le système aussi souvent que possible, en particulier pour identifier le (ou les) contre-exemple(s), pour que l'analyste puisse concevoir sa propre requête et en identifier davantage.

RQL peut être utilisé de deux manières : (i) une BD jouet pré-remplie est fournie avec un ensemble de requêtes pour donner un aperçu rapide du langage (ii) un mode bac à sable autorise l'utilisateur à mettre en ligne ses propres données (limité à 3 tables et 200 ko) et à formuler ses propres requêtes.

A titre expérimental, RQL a été utilisé par des étudiants (niveau L3) en marge d'un cours de bases de données à l'INSA de Lyon. L'idée était de fournir aux étudiants la possibilité de manipuler les dépendances fonctionnelles avec RQL. Sans surprise, RQL a été facilement appréhendé par les étudiants et la notion de contre-exemple a été largement mise en pratique. RQL a été apprécié par sa capacité à faire le lien entre le langage SQL et les dépendances fonctionnelles pour la conception de bases de données, et pour proposer une interface unifiée pour les requêtes SQL et RQL.

Des travaux précédent [6] ont mis en évidence l'efficacité de RQL en tant que processus à deux étapes, même sur des bases de données volumineuses.

5. CONCLUSION

RQL est introduit comme une interface Web pour découvrir des règles sur des bases de données relationnelles. RQL englobe les expressions SQL en fournissant un moyen de spécifier et d'obtenir des résultats sous la forme d'un ensemble de règles et de contre-exemples. Le problème de la fouille de motifs est vu comme un problème d'optimisation de requêtes, pour lequel nous avons proposé une technique de réécriture permettant de déléguer au maximum le traitement aux SGBD sous-jacent [6]. RQL permet aux développeurs habitués au SQL d'extraire des informations précises sans requérir des connaissances préalables en fouille de données.

6. REFERENCES

[1] M. Agier, C. Froidevaux, J.-M. Petit, Y. Renaud, and J. Wijsen. On Armstrong-compliant logical query languages. In *Proceedings of the 4th International Workshop on Logic in Databases*, LID '11, pages 33–40, 2011.

[2] M. Agier, J.-M. Petit, and E. Suzuki. Unifying framework for rule semantics : Application to gene

expression data. *Fundamenta Informaticae*, 78(4) :543–559, 2007.

[3] H. Blockeel, T. Calders, E. Fromont, B. Goethals, A. Prado, , and C. Robardet. A practical comparative study of data mining query languages. In Springer, editor, *Inductive Databases and Constraint-Based Data Mining*, pages 59–77. 2010.

[4] H. Blockeel, T. Calders, É. Fromont, B. Goethals, A. Prado, and C. Robardet. An inductive database system based on virtual mining views. *Data Min. Knowl. Discov.*, 24(1) :247–287, 2012.

[5] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Proceedings of the 23rd International Conference on Data Engineering*, ICDE '07, pages 746–755, 2007.

[6] B. Chardin, E. Coquery, B. Gouriou, M. Pailloux, and J.-M. Petit. Query Rewriting for Rule Mining in Databases. In *Languages for Data Mining and Machine Learning, in conjunction with ECML/PKDD*, pages 35–49, 2013.

[7] L. Fang and K. LeFevre. Splash : ad-hoc querying of data and statistical models. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 275–286, 2010.

[8] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.

[9] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1) :574–585, 2009.

[10] T. Guns, S. Nijssen, and L. D. Raedt. Itemset mining : A constraint programming perspective. *Artif. Intell.*, 175(12-13) :1951–1983, 2011.

[11] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 1275–1278, Washington, DC, USA, 2009. IEEE Computer Society.

[12] S. Lopes, J. Petit, and L. Lakhal. Efficient discovery of functional dependencies and armstrong relations. In *Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*, pages 350–364, 2000.

[13] K. Murakami and T. Uno. Efficient algorithms for dualizing large-scale hypergraphs. *CoRR*, 1102.3813, 2011.

[14] A. Netz, S. Chaudhuri, J. Bernhardt, and U. M. Fayyad. Integration of data mining with database technology. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 719–722, 2000.

[15] C. Ordonez and S. K. Pitchaimalai. One-pass data mining algorithms in a DBMS with UDFs. In *SIGMOD Conference*, pages 1217–1220, 2011.

[16] Play framework. http ://www.playframework.com/.

Inferring multimodal itineraries from rich smartphone data

David Montoya
Cofely Ineo, GDF Suez & INRIA Saclay
david.montoya@cofelyineo-gdfsuez.com
david.montoya@inria.fr

Serge Abiteboul
INRIA Saclay & ENS Cachan
serge.abiteboul@inria.fr

ABSTRACT

We designed a system to infer the multimodal itineraries traveled by a user from a combination of smartphone sensor data (e.g., GPS, Wi-Fi, inertial sensors), personal information, and knowledge of the transport network topology (e.g., maps, transportation timetables). The system operates with a *Multimodal Transport Network* that captures the set of *admissible multimodal itineraries*, i.e., paths of this network with weights providing the statistics (expected time and variance) of the paths. The network takes into account public transportation schedules. Our Multimodal Transport Network is constructed from publicly available transport data of Paris and its neighbourhoods published by different transport agencies and map organizations. The system models sensor uncertainty with probabilities, and the likelihood that a *multimodal itinerary* was taken by the user is captured in a Dynamic Bayesian Network. For this demonstration, we captured data from users travelling over the Paris region who were asked to record data for different trips via an Android application. After uploading their data into our system, a set of most likely itineraries is computed for each trip. For each trip, the system displays recognized multimodal itineraries and their estimated likelihood over an interactive map.

Introduction

The democratisation of connected and sensor-rich personal mobile devices has increased the demand for context-aware commercial applications taking advantage of the information they are able to generate. Most prominent examples of such applications are the smartphones' navigational applications. Modern smartphones can track the position of their carrier using three independent radio networks: Satellite Navigaton (GPS, GLONASS), Cellular and Wi-Fi networks. Embedded inertial sensors (accelerometer and gyroscope) can be used to enrich positional information via *dead reckoning*. And more recently, inertial sensors have also been successfully used in different *activity recognition* applications. Current technology is far from taking full advantage of positional and inertial data to understand the users' daily movements in urban environments. The present system demonstrates sophisticated techniques towards this goal.

We designed a system to infer the multimodal itineraries traveled by a user from a combination of smartphone sensor data, personal information and knowledge of the transport network topology. The sensor data is recorded by users over the course of a journey. The network topology can be constructed from available geographic data and public transport timetables. The system ultimately aims at two modes of operations, each with distinct goals:

Offline mode After the data has been acquired, the system determines the user's location, mode of transport and different transportation routes and lines taken over the time of a travel. The expected result is a set of candidate itineraries ranked by their likeliness.

Online mode In real time, the system determines the current user's location, mode of transportation and current transportation route if applicable. It predicts the user's future movements, both in the short term (Is the user going to grab a bicycle at the next bicycle-sharing station?) and in terms of distant goals (Is the user going to the office?).

This demonstration will most likely only feature the offline mode. The work on the online mode will be probably too preliminary to be demonstrated.

In such a setting, a key aspect is that of the available knowledge. First, sensor data can be classified based on the kind of knowledge they deliver (e.g. location, movement) and its characteristics (e.g. frequency, accuracy). Then the system has access to geographic data (roads, points of interest), public transportation data (routes and schedules), and finally historical traffic data (traffic jams, schedule delays). These allow creating a *Multimodal Transport Network*. The key notion is that of *admissible multimodal itineraries* that are paths of this network with weights that give the statistics (expected time and variance) of the paths. Finally, the system may rely on the user's personal knowledge such as a history of past itineraries to bias the probabilities of the different routes. Difficulties arise from lack of data (e.g., lack of positioning inside the metro) and from too much data (e.g., combination of possibly conflicting localisation data, overlapping public transportation lines).

For our demonstration, the Multimodal Transport Network is constructed from publicly available transport data of Paris and its neighbourhoods published by different transport agencies and mapping organizations. In terms of transport

BDA 14 octobre 2014, Grenoble-Autrans, France.

network heterogeneity, the region features a multitude of modes for public transport: sub-urban train, metro, bus, tram. These are spread over different transport agencies. The region is also notable for its car-sharing and bike-sharing systems. Smartphone data is recorded via an Android application by multiple users travelling over the Paris region. We demonstrate our system by displaying recognized multimodal itineraries and their estimated likelihood for a single trip of a single-user over an interactive map.

One contribution of our approach is an algorithm that computes a set of most likely itineraries using a combination of particle filtering [9, 10, 28] and a special path sampling algorithm over a Dynamic Bayesian Network [20]. Figure 1 summarizes the different components of our system.

Related work

Activity recognition, that is concerned with determining the actions and goal of one or several agents given a series of observations on their actions and the environment, has gained increased attention over the years in the fields of articial intelligence, robotics, and ubiquitous computing. Our work follows pioneering works published over the last ten years on sensor-based activity recognition, for which accelerometer based approaches are notable [3, 7, 15, 19, 22, 24]. More recently, several researchers have been particularly concerned with *transport mode* identification [14, 18, 25, 29, 30], for which more general approaches can be devised. In these approaches, we are only interested in determining whether the user is currently *stationary, walking, running, cycling* or in a *motor vehicle*. This technology is now available, via their respective APIs, to developers of the two most widespread smartphone platforms [1, 8].

In our setting, the requirement is to determine both transport mode and traveled routes. To serve this purpose, and provide context awareness to travel assistant technologies, location-based activity tracking were developed, that use positional information from an embedded GPS chip to track the user's most frequent locations, travel routines and routes taken [2, 6, 16]. Location-based vehicle tracking has been successfully used to generate accurate schedules and provide information about traffic delays to the rest of the community [4, 26, 27]. On the other hand, our interest is to provide a single user with real-time itinerary-aware applications and an accurate summary of one's routines. To this respect, our work is closely related to the literature in [6, 16], which uses a combination of online activity recognition techniques and location-based map-matching. However, these publications do not distinguish overlapping public transportation routes with different schedules and do not handle long periods of missing GPS observations. Map matching, for which a survey can be found in [23] with more recent results in [17, 21], is concerned with matching a set of positional observations to a path in a given road network. In our case, we cover a more general problem that considers a multimodal transport network. Our multimodal network adds two extra difficulties: timetable management and the fact that two geographic routes may belong to several transport modes and lines.

In itinerary detection, the use of a dynamic Bayesian network is not new. It had to our knowledge never been stressed as far as we do by considering richer transportation data (multimodal distinguishing lines and timetables), with a richer combination of user data: both positional sensors and accelerometer. Another novelty is in the processing of zones with long periods of time with missing GPS observations, which happen frequently in high density urban areas, e.g. with an underground transit system.

Sensor Data

We collected sensor measurements from multiple users equipped with a smartphone and running a logging application. In this section, we quickly overview the different types of sensor measurements recorded in our experiments.

Absolute Positioning. Three different technologies are able to pinpoint the smartphone's position. They come with varying degrees of accuracy, from the most to the least accurate: Satellite Navigation (GPS, GLONASS), Wi-Fi Networks and Cellular Network. These sensors will raise positional events the moment they get a new reading of where the smartphone might be. We use the term *location fix* to refer to such events. Metadata (signal quality, visible satellite count) is recorded as well. We constrain geographic locations to a longitude and latitude pair with no elevation information. Satellite Navigation, besides being the most accurate, is the only one capable of providing speed and bearing as well. Absolute positioning technologies are all characterized by being dependent on wireless transmission of electromagnetic signals. Also, for them to work, they require some kind of static almanac or algorithm that accurately maps signal emitters to a position on Earth or in space. We note that signal unavailability or failure to determine one's position is a very important piece of information. In some cases, we could get a hint of where the user might be, and in many, we will know where the user *cannot be*.

Inertial. Most smartphones have an embedded accelerometer, a gyroscope and a magnetometer sensor. Altogether they provide information regarding the device's movement with respect to an earth's bound frame of reference. Given an initial starting point, orientation and velocity, and sufficient sensor accuracy, the position of the smartphone can be tracked over time. For transport mode detection, we are mostly concerned by accelerometer measurements [14, 18, 25, 29]. However, since the accelerometer measures acceleration within the device's inertial frame, it is measured with respect to the device's orientation. Thus, ideally, one needs to combine information from other sensors to derive a geo-centric orientation. We will not further discuss gyroscope/magnetometer data. Mainly, we will consider that accelerometer data has been augmented, when possible, with gyroscope/magnetometer data. A system that performs this augmentation (sensor-fusion) is found in [13].

Frequency of measurements. Location sensors do not necessarily provide a *location fix* at the requested rate. For example, Cellular/Wi-Fi Network Position sensors do not necessarily raise *fixes* if they deem that the position of the smartphone has not sufficiently changed provided the accuracy of their measurements. Satellite Navigation, which can acquire fixes at 1 Hz irrespective of a position change, can temporarily stop raising *fixes* if the received satellite signals are not strong enough for a sufficient lapse of time.

Accelerometer and other low-level sensors are able to acquire data at an almost fixed-rate, and are not subject to interruptions under a wide array of circumstances. We can

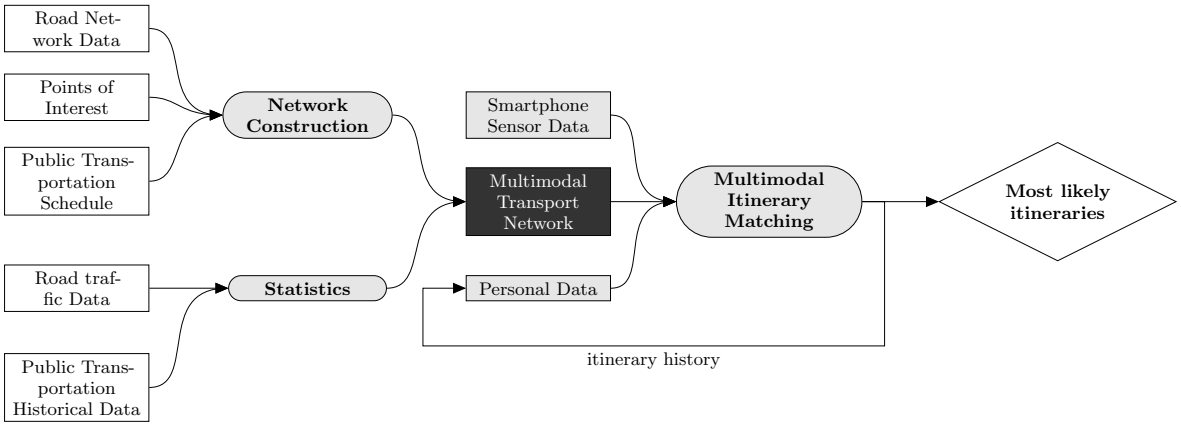


Figure 1: Overall picture: white rectangles are input geographic data, grey ones are user data, dark ones are derived models. Rounded rectangles represent algorithms, and the diamond is the output.

expect such readings at a rate between 50 and 100 Hz. Sensor-fused accelerometer in the geo-centric frame data is thus expected at a rate of at least 50 Hz.

Recording of measurements. Measurements are retrieved from multiple users equipped with an Android smartphone. For privacy considerations, users are asked to press a "Start logging" button at the start each trip, and they are asked to press "Stop logging" at the end of it. During the trip, the application runs in the background, listening to incoming sensor data, and records it in tabular form. Each row represents a single measurement, and is associated a timestamp. The application produces different tabular files for each sensor and type of measurement.

Parisian Multimodal Transport Network

Figure 2: Glimpse of the Parisian public transportation lines, Paris 1er arrondissement

The *Multimodal Transport Network* is constructed with data from various Web sources, including OpenStreetMap (road and infrastructure), RATP & SNCF Open Data (public transportation), as well as Velib' and Autolib'. We distinguish private transport modes (*walking/running, cycling, car, motorcycle*) from public (*bus, metro, tram, train*). For public transport modes, we distinguish different transportation lines (e.g. metro Line 1, Line 2, etc.). Our system is able to construct "admissible multimodal itineraries" that are paths of this network with weights that give the statistics (expected time and variance) of the paths. Similar to [30], mode transitions are only allowed in and out of *human* transport modes (*walking/running, cycling*). Waiting times, e.g. when waiting for a bus at the station, are considered as well. Multimodal Transport Network takes into account the time variability of traffic. As of now, this variability only includes public transportation schedules. More information will be introduced when it is available. For instance, the multimodal transport network will tell (at a particular time of a given day) the average time it would take (and variance) to pick up a bike at the Marguerite de Navarre bike-sharing station, drop it at Grands Boulevards and get on Line 9 of metro.

Type of data	Count
Roads	$> 10{,}000$
Public transportation lines	375
Bike-sharing stations	1,227
Car-sharing stations	905

Table 1: Multimodal transport network: Paris and neighborhoods

Time variability of traffic is our first step towards modelling uncertainty of complex events in the Multimodal Transport Network. Link uncertainty, such as missing or modified roads and public transportation lines, although frequent, is not considered in this work. Table 1 shows some key figures of our Multimodal Transport Network. Figure 2 overlays a view of different public transportation lines over a map of Paris at the time of the submission.

Multimodal Itinerary Matching

The system is able to compute the probability that a multimodal itinerary was taken. We model the probability that

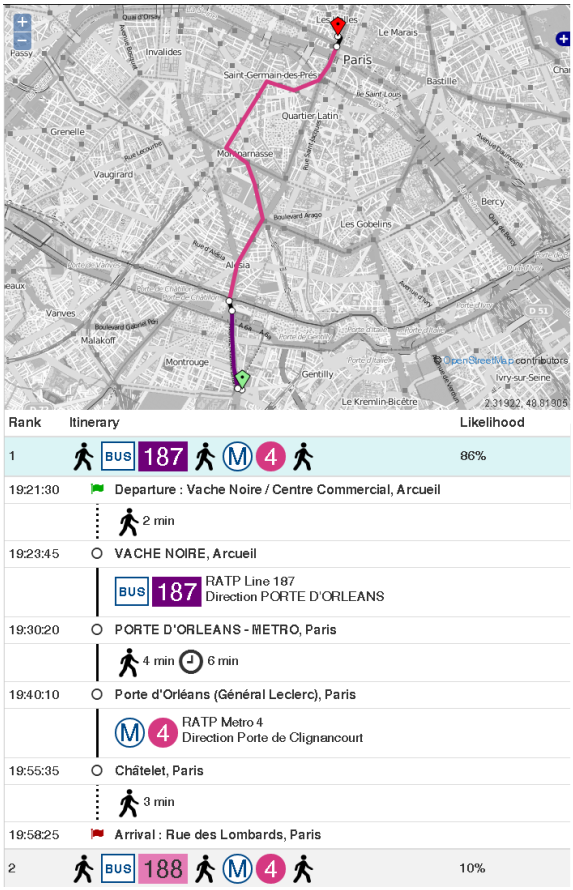


Figure 3: Displaying the most likely itineraries. The itinerary displayed on the map is highlighted in blue in the list.

a *multimodal itinerary* was taken via a Dynamic Bayesian Network [20]. Based on probabilistic observational variables and state transitions, one's current itinerary is represented in the Dynamic Bayesian Network through a state variable that keeps tracks of one's current location in the Multimodal Transport Network. The model integrates a certain number of priors, gathered from the training of local models, verified in practice and extracted from knowledge in previous work [2, 6, 14, 16, 25, 29, 30]. The system performs approximate inference by keeping track of a belief state via particle filtering [5, 9, 12, 16] with a special path sampling algorithm [6, 11].

Demonstration setting

For the demo, user data will be displayed in the form of a list of trips.

For each trip, multiple inferred multimodal itineraries will be displayed in a list, ranked by their likelihood with respect to trip measurements. Selecting an itinerary will overlay it on a map (Fig. 3). When the algorithm does not succeed to infer precise portions of an itinerary, they will be displayed as "grey zones". Raw positional measurements will be overlaid on the map as a baseline. When available, a manually annotated ground truth itinerary may be overlaid for comparison. To demonstrate that the system takes public transportation schedules into account, we will show how an artificially added time offset on raw data produces different results.

Finally, we will display various per-user statistics: frequent itineraries, average departure times and duration of travel. When a user uses multiple itineraries for the same start and endpoints, their statistics may be compared.

References

[1] Android Developers Guide. *Recognizing the User's Current Activity*. 2013. URL: `http://developer.android.com/training/location/activity-recognition.html` (visited on 05/16/2014).

[2] D. Ashbrook and T. Starner. "Using GPS to learn significant locations and predict movement across multiple users". English. In: *Personal and Ubiquitous Computing* 7.5 (2003), pp. 275–286.

[3] L. Bao and S. Intille. "Activity Recognition from User-Annotated Acceleration Data". In: *Pervasive Computing*. Ed. by A. Ferscha and F. Mattern. Vol. 3001. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 1–17.

[4] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. "EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones". In: *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. SenSys '11. Seattle, Washington: ACM, 2011, pp. 68–81.

[5] H. H. Bui, S. Venkatesh, and G. West. "Policy Recognition in the Abstract Hidden Markov Model". In: *J. Artif. Int. Res.* 17.1 (Dec. 2002), pp. 451–499.

[6] J. Chen and M. Bierlaire. "Probabilistic multimodal map-matching with rich smartphone data". In: *Journal of Intelligent Transportation Systems: Technology, Planning and Operations* (2013).

[7] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. Lamarca, L. Legrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, P. Klasnja, K. Koscher, J. Landay, J. Lester, D. Wyatt, and D. Haehnel. "The Mobile Sensing Platform: An Embedded Activity Recognition System". In: *Pervasive Computing, IEEE* 7.2 (Apr. 2008), pp. 32–41.

[8] iOS Developer Library. *CMMotionActivity Class Reference*. 2013. URL: `https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CMMotionActivity_class/Reference/Reference.html` (visited on 05/16/2014).

[9] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. "Rao-blackwellised Particle Filtering for Dynamic Bayesian Networks". In: *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. UAI'00. Stanford, California: Morgan Kaufmann Publishers Inc., 2000, pp. 176–183.

[10] A. Doucet, N. Freitas, and N. Gordon. "An Introduction to Sequential Monte Carlo Methods". English. In: *Sequential Monte Carlo Methods in Practice*. Ed. by A. Doucet, N. Freitas, and N. Gordon. Statistics for Engineering and Information Science. Springer New York, 2001, pp. 3–14.

[11] G. Flötteröd and M. Bierlaire. "Metropolis–Hastings sampling of paths". In: *Transportation Research Part B: Methodological* 48 (2013), pp. 53–66.

[12] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. "Bayesian Filtering for Location Estimation". In: *IEEE Pervasive Computing* 2.3 (2003), pp. 24–33.

[13] E. Foxlin. "Pedestrian tracking with shoe-mounted inertial sensors". In: *Computer Graphics and Applications, IEEE* 25.6 (Nov. 2005), pp. 38–46.

[14] S. Hemminki, P. Nurmi, and S. Tarkoma. "Accelerometer-based Transportation Mode Detection on Smartphones". In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. SenSys '13. Roma, Italy: ACM, 2013, 13:1–13:14.

[15] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. "Activity recognition using cell phone accelerometers". In: *ACM SigKDD Explorations Newsletter* 12.2 (2011), pp. 74–82.

[16] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. "Learning and inferring transportation routines". In: *Artificial Intelligence* 171 (2007), pp. 311–331.

[17] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. "Map-matching for Low-sampling-rate GPS Trajectories". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. Seattle, Washington: ACM, 2009, pp. 352–361.

[18] V. Manzoni, D. Maniloff, K. Kloeckl, and C. Ratti. *Transportation mode identification and real-time CO2 emission estimation using smartphones*. Tech. rep. Technical report, Massachusetts Institute of Technology, Cambridge, 2010.

[19] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. "Activity recognition and monitoring using multiple sensors on different body positions". In: *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*. IEEE. 2006, 4–pp.

[20] K. P. Murphy. "Dynamic bayesian networks: representation, inference and learning". PhD thesis. University of California, 2002.

[21] P. Newson and J. Krumm. "Hidden Markov Map Matching Through Noise and Sparseness". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. Seattle, Washington: ACM, 2009, pp. 336–343.

[22] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. "Activity classification using realistic data from wearable sensors". In: *Information Technology in Biomedicine, IEEE Transactions on* 10.1 (Jan. 2006), pp. 119–128.

[23] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. "Current map-matching algorithms for transport applications: State-of-the art and future research directions". In: *Transportation Research Part C: Emerging Technologies* 15.5 (2007), pp. 312–328.

[24] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. "Activity recognition from accelerometer data". In: *AAAI*. Vol. 5. 2005, pp. 1541–1546.

[25] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. "Using mobile phones to determine transportation modes". In: *ACM Transactions on Sensor Networks (TOSN)* 6.2 (2010), p. 13.

[26] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. "Cooperative transit tracking using smart-phones". In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. SenSys '10. Zürich, Switzerland: ACM, 2010, pp. 85–98.

[27] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones". In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2009, pp. 85–98.

[28] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[29] S. Wang, C. Chen, and J. Ma. "Accelerometer Based Transportation Mode Recognition on Mobile Phones". In: *Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on*. Apr. 2010, pp. 44–46.

[30] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. "Understanding transportation modes based on GPS data for web applications". In: *ACM Transactions on the Web (TWEB)* 4.1 (2010), p. 1.

Le web sémantique en aide à l'analyste de traces d'exécution

Léon Constantion Fopa
LIG, Université de Grenoble
Grenoble, France
leon-constantin.fopa@imag.fr

Fabrice Jouanot
LIG, Université de Grenoble
Grenoble, France
fabrice.jouanot@imag.fr

Alexandre Termier
INRIA - IRISA
Université Rennes 1
Rennes, France
alexandre.termier@irisa.fr

Maurice Tchuenté
IRD-UMMISCO et LIRIMA
Cameroun
maurice.tchuente@lirima.org

ABSTRACT

L'analyse de traces d'exécution est devenue l'outil privilégié pour débugger et optimiser le code des applications sur les systèmes embarqués. Ces systèmes ont des architectures complexes basées sur des composants intégrés appelés SoC (System-on-Chip). Le travail de l'analyste (souvent, un développeur d'application) devient un véritable challenge car les traces produites par ces systèmes sont de très grande taille et les évènements qu'ils contiennent sont de bas niveau. Nous proposons d'aider ce travail d'analyse en utilisant des outils de gestion des connaissances pour faciliter l'exploration de la trace. Nous proposons une ontologie du domaine qui décrit les principaux concepts et contraintes pour l'analyse de traces issues de SoC. Cette ontologie reprend les paradigmes d'ontologie légère pour supporter le passage à l'échelle de la gestion des connaissances. Elle utilise des technologies de "triple store" RDF pour son exploitation à l'aide de requêtes déclaratives SPARQL. Nous illustrons notre approche en offrant une analyse de meilleure qualité des traces d'un cas d'utilisation réel.

Keywords

ontologie, RDF, traces, web sémantique

1. INTRODUCTION ET CONTEXTE

Les systèmes embarqués sont basés sur des System-on-Chip (SoC) qui intègrent sur une même puce plusieurs composants : processeurs, mémoires, port d'entrée/sortie. Ces SoCs sont par exemple utilisés pour exécuter les applications multimédia embaquées dans nos smartphones, tablettes, téléviseurs ou set-top boxes. Les applications exploitant les SoCs sont complexes et peuvent difficilement être déboguées ou optimisées avec les méthodes d'analyse de codes. Les développeurs ont donc recourt à la capture et à l'analyse post-mortem des traces d'exécution [3, 4, 8]. Une trace d'exécution contient des évènements de bas niveau tels que les interruptions, les changements de contexte ou les appels de fonctions. Identifier des problèmes à partir de ces évènements présente deux difficultés. Premièrement la taille de la trace peut atteindre plusieurs millions d'évènements pour seulement quelques minutes d'exécution, ce qui pose un problème de passage à l'échelle des méthodes d'analyse. Deuxièmement l'interprétation à un niveau métier des évènements est difficile car elle requiert l'expérience et la connaissance métier de l'analyste, cependant ce niveau d'abstraction n'est pas explicite dans la trace. Nous proposons d'aider l'analyste en apportant des outils de gestion des connaissances pour naviguer dans la trace en utilisant des concepts métier de haut niveau.

La suite de cet article est organisée comme suit. Dans la section 2 nous proposons une ontologie du domaine de l'analyse de traces d'applications multimédia sur SoCs. Dans la section 3 nous présentons l'adaptation de l'ontologie à un cas d'utilisation réel. Dans la section 4 nous présentons l'analyse des traces du cas d'utilisation à l'aide de requêtes SPARQL. La section 5 conclut ce papier et présente nos travaux en cours.

BDA 14 octobre 2014, Grenoble-Autrans, France.

2. VIDECOM, UNE ONTOLOGIE POUR L'ANALYSE DE TRACES D'EXÉCUTION

L'analyste cherche à retrouver dans la trace les concepts métier correspondants à ses connaissances de l'application et de l'architecture du SoC. Cependant la relation entre les évènements de trace et ces concepts métier n'est pas explicite dans la trace. Nous présentons dans cette section notre approche pour enrichir les évènements de la trace. L'approche consiste à connecter les évènements de trace, à l'aide éventuellement des règles de déduction, à des classes et propriétés d'une ontologie représentant les concepts et contraintes pour l'analyse de traces issues de SoC.

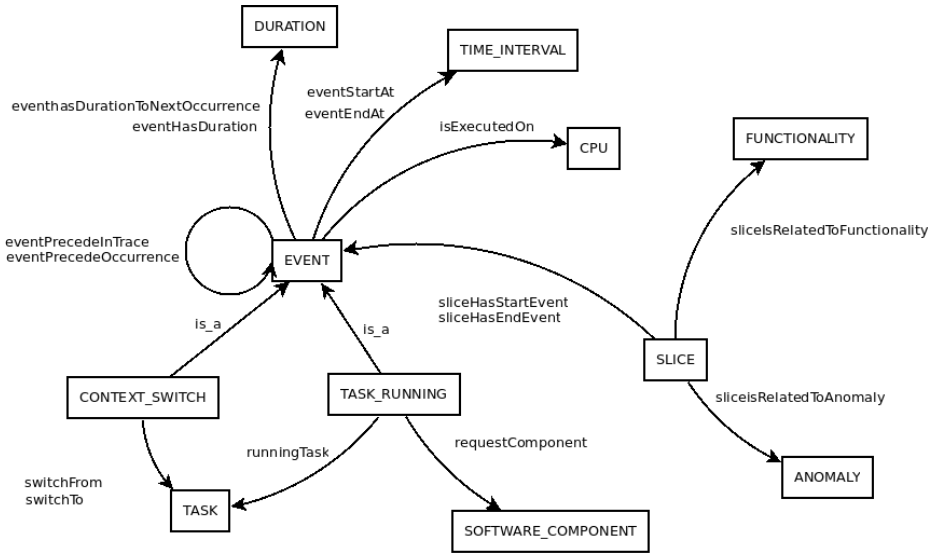


Figure 1: Extrait des classes et propriétés principales de Videcom.

2.1 Modèle de VIDECOM

VIDECOM est une ontologie légère, basée sur la sémantique de RDFS. Elle contient 608 classes et 238 propriétés représentant des concepts du domaine de l'analyse de trace d'application multimédia sur SoC.

La Figure 1 présente un extrait des classes et propriétés de VIDECOM. La classe principale Event représente les différents types d'évènements, on distingue TASK_RUNNING pour les évènements d'exécution de tâche et CONTEXT_SWITCH pour les évènements de changement de contexte entre deux tâches. Les classes FUNCTIONALITY et ANOMALY représentent respectivement les fonctionnalités et les anomalies de l'application. TIME_INTERVAL permet de représenter la temporalité des évènements et DURATION les différents types de durées. La classe SLICE sert à représenter une zone de trace. La propriété *isExecutedOn* capture la connaissance du processeur où se produit un évènement, *runningTask* représente la tâche exécutée par l'évènement et *requestComponent* permet de représenter les composants logiciels (interruptions, appels système ou appel fonction) sollicités par l'évènement. L'ordre entre les évènements est capturé par plusieurs propriétés : *eventPrecedeInTrace* indique l'évènement précèdent dans la trace tandis que *eventPrecedeOccurrence* indique l'occurrence précédente de l'évènement dans la trace. Ces classes et propriétés sont utilisées pour construire des règles métier qui permettent à l'analyste de décrire la sémantique des concepts et contraintes métier qu'il souhaite identifier dans la trace.

2.2 Implémentation efficace du triple Store

Pour exploiter VIDECOM dans l'analyse de trace, ses classes et propriétés sont instanciées à partir des évènements de la trace, puis saturées, stockées et requêtées dans un Triple Store [6]. Nous avons fait le choix d'une saturation à priori du Triple Store pour assurer des résultats complets aux requêtes. La saturation matérialise de nouvelles instances à partir des règles d'inférence RDFS et métier. A cause de la grande taille de la trace, le Triple Store résultant est de l'ordre de plusieurs dizaines de millions de triplets. Une étude comparative du passage à l'échelle de l'exploitation de ces triplets sur 7 systèmes de bases de connaissances (Jena, Sesame, Sesame-native, TDB, SDB, rdf-3x, vertical-mdb), a montré que la méthode du partitionnement vertical, introduit par Abadi [1], permet d'avoir des temps acceptable de chargement et de réponse aux requêtes sur 95 millions de triplets [7]. Cette méthode stocke les triplets dans des tables ayant le modèle relationnel suivant : $propertyP(subject, property, object)$. Chaque table représente une propriété de VIDECOM et contient tous les triplets correspondants à ses instances. Les tables sont physiquement stockées sous forme de collections de colonnes par des gestionnaires de bases de données orientés colonnes spécialement conçus pour le partitionnement vertical [2, 13].

2.3 Description d'un cas d'utilisation réel

Pour expérimenter notre approche, nous nous intéresserons au cas réel d'usage de l'analyse des traces d'une application d'enregistrement en streaming provenant de STMicroelectronics. L'application, nommée *ts_record*, exécute plusieurs fonctionnalités en parallèle. Tout d'abord, les données streaming sont collectées et stockées dans des buffers IP par une tâche t1, une seconde tâche t2 les copie ensuite des buffers IP vers la mémoire centrale. La dernière tâche t3 se charge enfin de les copier de la mémoire centrale vers le disque USB. Les tâches t1, t2 et t3 respectent des contraintes temporelles pour éviter de lire trop tard ou trop tôt des données. Par exemple la tâche t2 doit lire les données des buffers IP et les écrire en mémoire toutes les 100 millisecondes à l'aide de l'appel système *sys_write*. Le non respect de cette contrainte temporelle peut conduire à des anomalies dans l'application *ts_record* résultant en l'enregistrement d'un fichier corrompu sur le disque USB. Les traces de *ts_record* contiennent non seulement les évènements produits par l'exécution en parallèle des tâches t1, t2 et t3, mais aussi les évènements liés aux différentes activités du système d'exploitation.

IF	THEN
$(?e1, runningTask, ts\_record0)$ $(?e1, requestComponent, sys\_write0)$ $(?e1, eventPrecedeOccurrence, ?e2)$ $(?e1, eventHasDurationToNextOccurrence, ?period)$ $(?period == 100)$	$(\_ : s, sliceHasStartEvent, ?e1)$ $(\_ : s, sliceHasEndEvent, ?e2)$ $(\_ : s, sliceIsRelatedToFunctionality, sysWriteNormal);$

Table 1: Règle d'inférence d'une instance de zone de trace rattachée à la fonctionnalité sysWriteNormal

IF	THEN
$(?e1, runningTask, ts\_record0)$ $(?e1, requestComponent, sys\_write0)$ $(?e1, eventPrecedeOccurrence, ?e2)$ $(?e1, eventHasDurationToNextOccurrence, ?period)$ $(?period > 100)$	$(\_ : s, sliceHasStartEvent, ?e1)$ $(\_ : s, sliceHasEndEvent, ?e2)$ $(\_ : s, sliceIsRelatedToAnomaly, sysWriteBlocked);$

Table 2: Règle d'inférence d'une instance de zone de trace rattachée à l'anomalie sysWriteBlocked

2.4 Extension de VIDECOM au cas d'utilisation ts_record

L'analyste peut ajouter des sous-classes aux classes de base de VIDECOM pour l'enrichir avec les connaissances métier correspondant aux cas d'utilisation à analyser. Dans le cas de *ts_record* nous avons ajouté les sous-classes de FUNCTIONALITY suivantes : *dataFromEthernet2IP*, *dataFromIP2Memory, dataFromMemory2UB* et *sysWriteNormal*. Les trois premières correspondent aux fonctionnements respectifs des tâches t1, t2 et t3 de *ts_record* et la dernière correspond au respect de la contrainte temporelle. De même nous avons ajouté une sous-classe *sysWriteBlocked* à la classe ANOMALY qui correspond au non respect de la contrainte temporelle.

L'analyste utilise les règles d'inférence métier pour enrichir VIDECOM avec la sémantique des nouveaux concepts. Ces règles décrivent comment sont créées les instances de ces nouveaux concepts. Les règles d'inférence, que nous illustrons sous la forme conditionnelle IF_THEN, ajoutent les triplets de la clause THEN si les triplets ou les expressions de la clause IF sont présents ou vérifiés dans le Triple Store. La règle d'inférence du Tableau 1 illustre la règle d'inférence qui permet de rattacher une zone de trace à la fonctionnalité *sysWriteNormal* si elle correspond à une zone où la contrainte temporelle a été respectée. La règle du Tableau 2 quant à elle permet de rattacher une zone de trace à l'anomalie *sysWriteBlocked* si au contraire la zone ne respecte pas la contrainte temporelle. Les zones sont rattachées aux fonctionnalités et aux anomalies par les propriétés *sliceIsRelatedToFunctionality* et *sliceIsRelatedToAnomaly*.

3. EXPLOITATION DE VIDECOM

L'exploitation du Triple Store pour l'analyse de trace se fait à l'aide de requêtes SPARQL. Dans cette section nous allons l'illustrer dans l'analyser la trace de *ts_record*. La Figure 2 représente une interface pour directement exécuter des requêtes SPARQL sur le Triple Store.

3.1 L'exploration de la trace à l'aide de requêtes SPARQL

Nous allons commencer l'analyse de la trace *ts_record* en visualisant les différentes fonctionnalités ou anomalies de la trace. La requête R1 recherche toutes les zones de trace rattachées à des fonctionnalités ou à des anomalies dans la trace. Leurs temps de début et de fin respectifs sont également retournés pour les identifier dans la trace.

```
R1 :
SELECT ?slice, ?start, ?end, ?func, ?anomaly
WHERE {
    ?slice sliceHasStartEvent ?e1.
    ?slice sliceHasEndEvent ?e2.
    ?e1 eventHasStart ?start.
    ?e2 eventHasEnd ?end.
      OPTIONAL
{?slice sliceIsRelatedToFunctionality ?func.}
      OPTIONAL
{?slice sliceIsRelatedToAnomaly ?anomaly.}
}
```

La Figure 3 représente l'illustration des résultats de la requête R1. Cette illustration permet de rapidement identifier des zones intéressantes pour une analyse plus fine. La requête R2 sert à identifier les tâches exécutées sur le processeur cpu0 dans l'intervalle de 152537756 à 194882669 millisecondes, correspondant à la zone intéressante sélectionnée dans la Figure 3.

```
R2 :
SELECT ?task
WHERE {
  ?event eventStartAt ?start.
  ?event eventEndAt ?end.
  ?event runningTask ?task.
  ?event isExecutedOn cup0
FILTER (?start >= 152537756 AND ?end  < 19482669)}
```

R2 permet d'isoler de façon déclarative les tâches exécutées sur un processeur précis dans un intervalle de temps. La requête R3 est un exemple de requête complexe permettant d'analyser un problème identifié dans la trace. Elle recherche dans tous les cas où une anomalie est déclarée sur un processeur, toutes les tâches qui s'exécutent sur les autres pro-

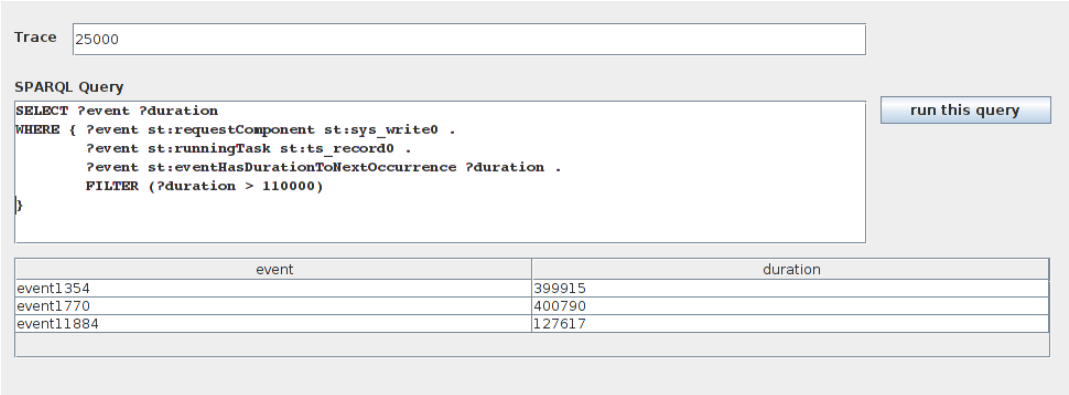


Figure 2: Interface pour exécuter des requêtes SPARQL sur le Triple Store.

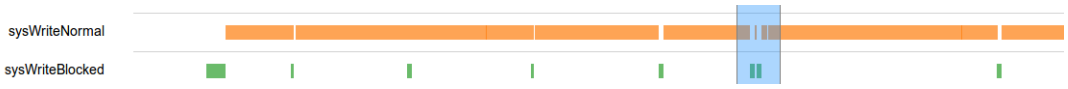


Figure 3: Time line représentant les fonctionnalités sysWriteNormal et les anomalies sysWriteBlocked.

cesseurs. Les résultats de cette requête ont permis de détecter que la tâche t3 s'exécute toujours en cas d'anomalie *sysWriteBlocked*. Une cause de l'anomalie s'est révélée être le fait que l'interruption levée par la tâche t3 pour lire les données en mémoire pouvait bloquer la tâche t2 si la même zone mémoire était sollicitée par les deux tâches. Ce qui occasionnait alors la perte des données des buffers qui n'étaient pas lu à temps.

```
R3 :
SELECT ?task ?cpu1
WHERE {
      ?slice sliceIsRelatedToAnomaly sysWriteBlocked.
      ?slice sliceHasStartEvent ?event1.
      ?slice sliceHasEndEvent ?event2.
      ?event1 eventStartAt   ?sstart.
      ?event2 eventEndAt   ?send.
      ?event1 isExecutedOn ?cpu0.
      ?event eventStartAt ?start.
      ?event eventEndAt   ?end.
      ?event runningTask  ?task.
      ?event isExecutedOn ?cpu1
      FILTER (?start >= ?sstart and ?end  <= ?send)
      FILTER (?cpu0 != ?cpu1)
}
```

3.2 Les performances

Nous avons utilisé MonetDB [12] comme gestionnaire de base de données orienté colonnes. Dans le but de rendre cette implémentation transparente à l'analyste, nous transformons toutes les requêtes SPARQL de l'analyste en requêtes SQL [5, 9] applicables au modèle relationnel du partitionnement vertical. De façon générale une conjonction dans la requête SPARQL est transformée en une jointure entre deux tables dans le partitionnement vertical. Nous avons effectué nos expériences sur une machine ayant un processeur de 2.27 GHz et 64 Go de RAM. Dans ces conditions nous avons traité une trace de 5 000 000 d'évènements correspondant à 25 minutes d'exécution de *ts_record*. Le Triple Store obtenu contient 95 309 610 triplets qui ont été saturés en 2 j 11 h 35 m 08 s à l'aide d'une implémentation SQL de l'opération de saturation. L'algorithme de saturation RETE implémenté dans Jena [10] est plus efficace mais nous n'avons pas pu l'utiliser car il nécessitait largement plus que les 64 Go de mémoire centrale. Les temps d'exécution des requêtes SPARQL varient entre 5 millisecondes pour les requêtes de type R2 où l'intervalle temporel est précisé et 3 minutes et 33 secondes pour des requêtes de type R3 qui s'exécutent sur toute la trace.

4. CONCLUSION

L'analyse des traces pour le débogage et l'optimisation des applications sur SoC est une tâche difficile, à cause de la taille importante des traces et du bas niveau des évènements. Nous avons proposé une approche pour enrichir la sémantique des évènements de traces en utilisant des concepts métier de plus haut niveau, représenté dans une ontologie du domaine de l'analyse de traces issues des SoC. Nous avons considéré un cas réel d'utilisation pour montrer l'intérêt de l'approche dans l'analyse de trace. Et nous avons montré qu'à l'aide de requêtes SPARQL, l'approche permet d'identifier rapidement des zones intéressantes et d'explorer plus finement la trace ce qui améliore l'analyse de la trace.

Bien que le temps de réponse aux requêtes reste abordable pour de grandes traces, la saturation à priori des triplets est une opération très coûteuse. Nous travaillons actuellement sur des techniques d'optimisation de la saturation. La première idée est de maintenir la saturation à priori dans ce cas nous avons expérimenté la distribution du Triple Store suivie de la saturation parallèle des portions. Cependant ces portions de Triple Store saturées n'assurent pas des réponses complètes aux requêtes à cause de la dépendance sémantique

des évènements. La deuxième idée est de réduire le nombre des évènements considérés dans la trace soit en effectuant un échantillonnage soit en considérant une première abstraction de ces évènements sous forme de patterns fréquents [11].

5. ACKNOWLEDGMENTS

Ce travail est financé par le projet FUI SocTrace. L'ontologie VIDECOM, des requêtes métiers ainsi que des triplets obtenus à partir de différentes de traces sont disponibles à l'adresse http ://videcom.imag.fr

6. REFERENCES

[1] 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.

[2] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Sw-store : a vertically partitioned dbms for semantic web data management. *The VLDB Journal The International Journal on Very Large Data Bases*, 18(2) :385–406, 2009.

[3] T. Ball. The concept of dynamic analysis. In *Software Engineering ESEC/FSE '99*, pages 216–234. Springer, 1999.

[4] T. Ball and J. R. Larus. Optimally profiling and tracing programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(4) :1319–1360, 1994.

[5] R. Cyganiak. A relational algebra for sparql. *Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170*, page 35, 2005.

[6] C. David, C. Olivier, and B. Guillaume. A survey of rdf storage approaches. *ARIMA Journal*, 15 :11–35, 2012.

[7] L. C. Fopa, J. Fabrice, A. Termier, T. Maurice, and I. Oleg. Benchmarking of triple stores scalability for mpsoc trace analysis. In *2nd International Workshop on Benchmarking RDF Systems*, 2014.

[8] A. Hamou-Lhadj and T. C. Lethbridge. A survey of trace exploration tools and techniques. In *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 42–55. IBM Press, 2004.

[9] S. Harris and N. Shadbolt. Sparql query processing with conventional relational database systems. In *Web Information Systems Engineering–WISE 2005 Workshops*, pages 235–244. Springer, 2005.

[10] A. Jena. A free and open source Java framework for building Semantic Web and Linked Data applications. `https://jena.apache.org/`, 2011. [Online ; accessed 10-Febuary-2015].

[11] C. Kamdem Kengne, L. C. Fopa, A. Termier, N. Ibrahim, M.-C. Rousset, T. Washio, and M. Santana. Efficiently rewriting large multimedia application execution traces with few event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1348–1356. ACM, 2013.

[12] MonetDB. Monetdb colums-store pioneers. `https://www.monetdb.org/`, 2008. [Online ; accessed 10-Febuary-2015].

[13] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, et al. C-store : a column-oriented dbms. In *Proceedings of the 31st international conference on Very large data bases*, pages 553–564. VLDB Endowment, 2005.

PlantRT : a Distributed Recommendation Tool for Citizen Science

PlantRT : Outil de recommandation distribué pour les sciences citoyennes

Maximilien Servajean
INRIA & LIRMM
University of Montpellier
France
servajean@lirmm.fr

Esther Pacitti
INRIA & LIRMM
University of Montpellier
France
pacitti@lirmm.fr

Miguel Liroz-Gistau
INRIA & LIRMM
Miguel.Liroz_Gistau@inria.fr

Alexis Joly
INRIA & LIRMM
alexis.joly@inria.fr

Julien Champ
INRIA & LIRMM
julien.champ@lirmm.fr

ABSTRACT

Les utilisateurs du Web 2.0 sont de gros producteurs de données diverses qu'ils stockent dans une grande variété de systèmes. Dans ce travail, nous nous concentrons sur le cas particulier des botanistes. En effet, établir une connaissance précise de l'identité, de la distribution géographique et de l'évolution des espèces vivantes est essentiel pour la pérennité de cette biodiversité, tout autant que pour l'espèce humaine. L'émergence des sciences citoyennes et des réseaux sociaux sont des outils supplémentaires favorisant la création de grandes communautés d'observateurs de la nature, qui ont commencé à produire d'énormes collections de données multimédias. Cependant, la complexité inhérente à la réalisation de ces collections provoque une certaine méfiance des utilisateurs, ces dernier ne souhaitant pas stocker leurs données sur un serveur central. Dans ce travail, nous avons réalisé un prototype multi-sites, où chaque site, peut représenter 1 à n utilisateurs permettant la recherche et la recommandation d'observations de plantes diversifiées à grande échelle.

Categories and Subject Descriptors

H.4 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Search process

Keywords

Multi-sites, top-k, search and recommendation

BDA 14 octobre 2014, Grenoble-Autrans, France.

1. INTRODUCTION

Web 2.0 users are massive producers of diverse data (*e.g.* photos, videos, scientific data). Additionally, while users are often willing to share their data with each other in a community of interest, they do not want to lose control over them using a central site. However, the distribution of the users' data in many different devices (*e.g.* own computer, servers) makes data sharing especially difficult.

In this work, we focus on the particular case of botanists. Building an accurate knowledge of the identity, the geographic distribution and the evolution of living species is essential for a sustainable development of humanity as well as for biodiversity conservation. The emergence of citizen sciences and social networking tools has fostered the creation of large and structured communities of nature observers (*e.g.* e-bird, xeno-canto, Tela Botanica, Pl@ntNet [3]) who started to produce outstanding collections of multimedia records. Scaling up such collaborative approaches to real-world ecological surveillance systems involving millions of contributors is however still challenging. In this context, users, or botanists, make observations of plants. An observation is composed of the plant's picture and its associated metadata, namely, the plant family, genus and species, the observation's geographic position (*i.e.* GPS) and time, and a description. The effort required to build a high-quality collection of observations is the reason why some botanists want to keep their data on their own computers or small servers. However, they still want to share observations, though in a controlled manner, so that they can search and be recommended from the whole community's observations.

Problem Definition: the goal of our work is to propose a large scale distributed platform that enables searching and recommending relevant and diversified plants observations taking into account both items' content and users' profile.

2. ARCHITECTURE'S OVERVIEW

In our distributed search and recommendation approach, users are represented by virtual nodes. Each site is composed of 1 to p virtual nodes and a virtual node can be composed of 1 to m users with a similar profile. Users can select the site to which they are connected, while they are

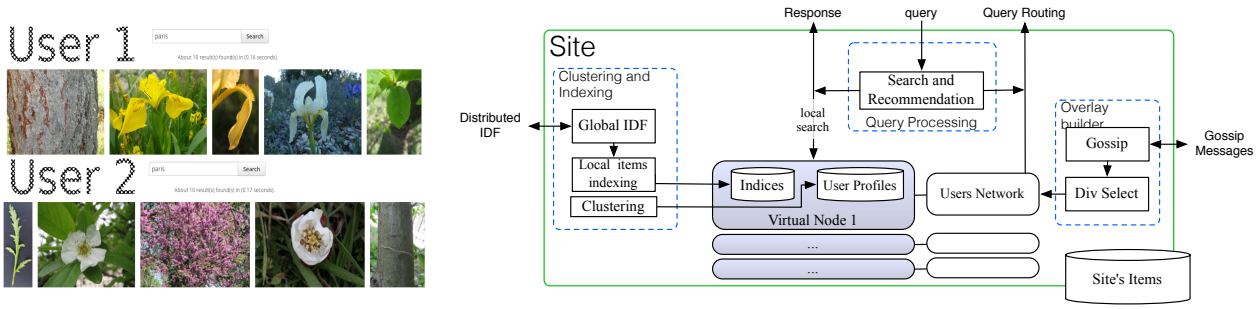


Figure 1: PlantRT's search and recommendation example and architecture.

clustered to virtual nodes depending on their profile. Each virtual node is associated to an index containing all items stored in the site. Items are indexed with respect to the profiles of the users associated to the current virtual node, so within a site all indexes will point to the same set of elements but will rank them in a different order. Virtual nodes are connected between them through a *User Network* overlay.

Whenever a user u submits a query q, the system sends it to this subset of virtual nodes (*i.e.* User Network), that will return their relevant results to u and will also recursively forward the query to the virtual nodes in their respective *User Network* until the *TTL* is reached. To build *User Network*, we use a two step approach. First, based on *random gossiping*, each site s is aware of other virtual nodes available on the network. Second, by means of a *diversified clustering* algorithm, u's virtual node chooses among these virtual nodes the best ones to answer u's queries and keep them in *User Network*.

Thus, PlantRT uses three components: (1) PlantRT Clustering and Indexing which is in charge of managing the virtual nodes and data indexing, (2) PlantRT User Network which is in charge of establishing the overlay between virtual nodes among sites and (3) PlantRT Query Processing, which is in charge of propagating queries submitted by users through a *User Network* overlay, to a subset of relevant virtual nodes and processing them. PlantRT's architecture is presented in Figure 1.

2.1 PlantRT Clustering and Indexing

The first component, deals with three tasks: a) maintaining the virtual nodes with similar users, b) maintaining the indices employed to retrieve items from keywords and geoposition in each virtual node, and c) generating the users profile from the locally shared items.

Based on [1], similar users are clustered together, each cluster corresponding to a virtual node. This is done by executing periodically *k-means* clustering algorithm. An index of all items stored in the site is built taking into account the profiles of the users in the cluster. The maximum number of virtual nodes is system defined and depends on the storage and memory capacity of the site [1].

Items (*i.e.* observations) are associated to both their GPS position and to a keywords vector [5] where each of them is associated to a $tf \times idf$ (*i.e.* term frequency $\times$ inverse document frequency) score representing its importance in the item with respect to the whole corpus I.

However, unlike centralized solutions, since the global corpus is not available at each site, PlantRT uses a gossip-based protocol that progressively produces the score of each keyword in the set of items shared in the site s with respect to the distributed global corpus of items I.

The intuition behind our distributed $tf \times idf$ protocol is that statistics about the global corpus can be estimated using average computing which can be quickly computed using gossip protocols [4].

Then, a profile is only computed as the average of its shared observations' $tf \times idf$ vector. This information is used during index generation. Notice that this index is later used during query processing to *efficiently* recommend relevant items with respect to a query at each involved virtual node.

2.2 PlantRT User Network

The second component aims at establishing an overlay between virtual nodes. Based on random gossiping [2], each site s maintains a set of random view entries corresponding to the virtual nodes profile s is aware of. Periodically, sites gossip, and exchange a random subset of virtual nodes views entries. After the random gossip merging phase, a diversified clustering algorithm is triggered at each virtual node. In fact, taking into account the previous gossip exchange, the algorithm selects the most relevant nodes – using a similarity measure – from the random view considering the relevant nodes previously selected in the *User Network* of each virtual node. Indeed, the *User Network* should diversified to increase coverage and therefore the probability to answer any query [citation globe].

2.3 PlantRT Query Processing

Finally, the third component deals with the execution of queries. Whenever u submits a query q, the query is redirected to all virtual nodes in the participating nodes' *User Network* recursively, until a predefined upper threshold, *TTL* (*i.e. Time-To-Live*). Whenever a node v receives a query, it computes its *top-k* most relevant and diversified items, taking into account both items' content and users' profile [6], among the locally indexed elements with respect to the query using a specific similarity measure (*e.g.* jaccard). Then, v returns its set of recommended items to u. An item recommended by a user v_i is defined by its identifier, its score, the site's identifier and v_i's profile. Once u receives the set of recommended items from all users $v_1, ..., v_n$ that received the query q, it ranks all received recommendations based on their score and on the similarity of v_i with respect to u.

3. PLANTRT DEMONSTRATION

Figure 1 presents the results of a search executed on our prototype. It shows a use case where two users are searching plants around Paris. However, since they are not interested in the same kind of plants both results lists are different. Also, the results are diversified in the sense that they only contain plants from different families.

Our prototype can be deployed on several nodes. In our experiments we have simulated up to $6,000$ nodes, reaching a recall (*i.e.* the proportion of results answering a query retrieved) of $99,9\%$ [citation globe].

4. CONCLUSION

This work presents the implementation of a diversified and distributed recommendation tool for citizen science, and more precisely, for botanists. Our platform integrates a complete set of features (*e.g.* subscribe, share, search, recommendation) and the still evolving source code is available online [1]

1. `http://www2.lirmm.fr/~servajean/prototypes/plant-sharing/plant-rt.html`.

5. REFERENCES

[1] S. Amer-Yahia and M. Benedikt. Efficient network aware search in collaborative tagging sites. *VLDB Endowment '08*, 1(1):710–721, 2008.

[2] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. In *ESOA*, volume 3910 of *Lecture Notes in Computer Science*, pages 1–15, Berlin, Heidelberg, 2005.

[3] A. Joly, H. Goëau, P. Bonnet, B. Vera, J. Barbe, S. Souheil, Y. Itheri, J. Carré, E. Mouysset, J.F. Molino, N. Boujemaa, and D. Barthélémy. Interactive plant identification based on social image data. In *Ecological Informatics*, 2013.

[4] W. Kowalczyk, M. Jelasity, and AE. Eiben. Towards data mining in large and fully distributed peer-to-peer overlay networks. In *BNAIC*, pages 203–210, 2003.

[5] G. Salton. *Automatic Information Organization and Retrieval*. McGraw Hill Text, 1968.

[6] M. Servajean, E. Pacitti, S. Amer-Yahia, and P. Neveu. Profile diversity in search and recommendation. In *WWW Companion*, pages 973–980, 2013.

CROWD, a platform for the crowdsourcing of complex tasks

Ahmad Chettih
University of Rennes 1

David Gross-Amblard
University of Rennes 1 / IRISA
dga@irisa.fr

David Guyon
University of Rennes 1

Erwann Legeay
University of Rennes 1

Zoltán Miklós
University of Rennes 1 / IRISA
zoltan.miklos@irisa.fr

ABSTRACT

Crowdsourcing is an emerging technique that enables to involve humans into information gathering or computational tasks. With the help of crowdsourcing platforms a group of participants (workers) can solve otherwise difficult problems. While the existing, generic crowdsourcing platforms such as Amazon Mechanical Turk have been used to address various challenges, they only support simple questions that we consider as basic tasks. In this demo we present CROWD, a platform where one can submit a workflow, which is a composition of such basic tasks. CROWD also supports a simple skill-management mechanism: each basic task is annotated with expertise tags. Upon the validation of completed tasks, the worker's expertise is updated according to these tags. The worker's expertise can later be used for better task selection. CROWD is implemented in Python/Django, and can be used both on the Web or on mobile devices.

1. INTRODUCTION

Crowdsourcing is a recent technique that allows so called taskers to rely on an unknown crowd on the Internet to solve a difficult task. Many ad hoc crowdsourcing platforms are devoted to the resolution of specialized tasks: see for example the FoldIt project[1] for the discovery of new protein foldings, or Transifex[2] for the translation of technical documents. Besides, generic crowdsourcing platforms have emmerged, such as Amazon Mechanical Turk (AMT[3]), CrowdFlower[4] or CloudFactory[5] , to name a few. These generic systems support mainly simple data acquisition tasks presented as a sequence of questions (forms). They are however not suited for complex tasks that require repetitions, alternatives or conditional execution. A typical example of such complex tasks is cooperative relief–support during disasters[6] [7], where Internet-connected people assist rescue teams by providing outside information (based on e.g. satellite pictures). We use the following complex task T as a running example:

> T: A storm has just hit Miami. Please help the coordination rescue operations, by providing information on available hospitals and passable roads to reach them. If you are trained in emergency management or you have experience with road traffic control, your are very welcome to join our online support team.

This task can be submitted to existing platforms, but as a unique text block. One could imagine a more structured and and more precise formulation of the same task:

> T (rewritten): A storm has just hit Miami. Please help the coordination of rescue operations, by first (T_1) locating nearby hospitals on this map (link given). Then, (T_2) obtain their phone numbers, and then (T_3) try to contact them to obtain their availability. Meanwhile, (T_4) list the passable roads reaching these hospitals. If you are trained in emergency management or you have experience with road traffic control, your are very welcome to join our online rescue teams.

This latter formulation shows opportunity for task parallelization and expertise tagging, however such queries are not supported by the existing platforms. Even if the API of the crowdsourcing platforms enables to realize the combination of basic tasks, this would require expert programming skills and considerable efforts for each task. Instead we propose a platform where one can submit such complex workflows directly.

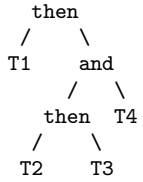
Contribution. In this demo we present CROWD, a platform where **one can submit a whole *workflow*, that is a composition of basic tasks**. The workflow is expressed as an execution plan with various composition operators.

[1] http://fold.it
[2] https://www.transifex.com/
[3] https://www.mturk.com/
[4] http://www.crowdflower.com/
[5] http://www.cloudfactory.com

[6] http://www.usnews.com/opinion/articles/2012/11/23/how-to-make-crowdsourcing-disaster-relief-work-better
[7] http://www.nature.com/news/crowdsourcing-goes-mainstream-in-typhoon-response-1.14186

BDA 14 octobre 2014, Grenoble-Autrans, France.

The workflow associated with our previous example is shown below. Its interpretation is, informally, "perform T1, then perform (in parallel) T4 and the sequence T2 and T3".



In this example, basic tasks (T_1 to T_4) can be submitted to existing platforms, and parallelism between tasks can be expressed. In our model the available operators are sequences (then), alternatives (or), conjunction (and), tests (if) and loops (while).

CROWD also supports a **simple skill-management mechanism**. Each basic task can be annotated by a list of expertise tags (e.g. "emergency, hospital" for T_1 to T_3 and "road, traffic" for T_4). Upon the validation or rejection of completed tasks, the workers expertise score is updated according to the relevant tags. The worker's expertise can later be used for better task assignment and the evaluation of the available expertise on a completed execution plan.

CROWD is implemented in Python/Django, and can be used both on the Web or on mobile devices. Participation is free (but extensible with fee facilities for participants if required) and symmetric (any user can participate in or create a task, contrary to existing platforms).

The rest of this demo is organized as follows: after giving the related work in Section 2, we expose the model underlying CROWD in Section 3. Then we detail our system in Section 4 and conclude with our demo scenario in Section 5.

2. RELATED WORK

The idea of composing smaller tasks and offering a composite task execution service is on the agenda of several startup companies. For example, Ville Miettinen, the CEO of MICROTASK[8], mentioned in a recent interview: "In the long term, all human processes that can be standardized will be available as a cloud service..." [9] MICROTASK already offers some composite services, but these services are predefined.

While basic tasks can be easily deployed on Amazon mechanical turk or similar platforms, sequences of tasks have to be modeled in a form. Alternatives can be hacked in Crowdflower using a specific $only-if$ construct in their CML language. Up to our knowledge, AMT nor CrowdFlower can model for example parallel tasks within a sequence.

Recent academic works propose some form of programming languages that enable a more flexible task composition, including [1, 2]. These procedural approaches offer a fine control of human-based computation, but the required skill for the task developer is high. Several declarative approaches that ease data acquisition campaigns have also been constructed recently [5, 8, 7, 6], but they do not reason on task composition. A generic, data oriented crowd system is presented in [3]. This active rules-based system is able to implement workflows that are similar to ours, but this requires complex skills for defining appropriate triggers. Sophisticated expertise mining models have also been proposed in the context of crowdsourcing (e.g. [4]), but independently from the workflow where these tasks belong to.

[8] http://www.microtask.com/

[9] http://venturebeat.com/2011/03/22/crowdsourcing-startup-microtask-gets-gamers-to-do-some-real-work/

3. MODEL

We present in this section the model underlying CROWD. This model encompasses the workflow execution, workflows intermediate results, result provenance, user and task expertise and task validation.

Participants and Expertise. Each participant in a CROWD process is uniquely identified, with identifiers denoted as *uid* in the sequel.

We suppose given a vocabulary of expertise tags, such as "hospital" or "traffic" (the vocabulary of these tags is free. A systematic building of these tags, for example as a taxonomy, is out of the scope of this demo). The expertise E_{uid} of participant *uid* is a function mapping a expertise tag e to an expertise integer score $E_{uid}(e)$ (where 0 means no expertise at all). The initial expertise of a new participant maps all expertise tag to 0.

Basic and Complex Tasks. A user can submit tasks to the system (we then call it a tasker, but any participant can become a tasker). The concrete syntax of a *basic task* such as T_1 is for example

```
ask(2,"Give the location of nearby hospitals",
      [hospital, emergency])
```

It denotes the triggering of 2 questions on hospital locations on the CROWD platform for available users. The task will be tagged by the hospital and emergency expertise. Once 2 distinct participants have answered this question, the task is finished. The result of this task is a list of (key,value) pairs, where the key is formed by the running task identifier denoted by *tid*, the participant's *uid* and the answer number. It is noteworthy that tasks results in CROWD are not typed, but are nested lists of (key,value) pairs (for the sake of simplicity we do not elaborate here on constraints on participant answers, like requiring the answer to be an integer of to respect a regular expression. This is a classical topic for all crowdsourcing platform). Hence, a possible result for T_1, denoted $res(T_1)$ could be

```
{(tid1-uid1-1,"Miami Children hospital"),
 (tid1-uid5-2,"University of Miami hospital")}
```

In turn, a *complex task* is either a basic task or a composition of complex tasks. When several tasks are involved, the result of one task can be the input of another task (in practice, the participant of the second task can see this input). Given two complex tasks t and t', we use a prefix syntax to connect them: $then(t, t')$ (sequence),$and(t, t')$ (conjunction), $or(t, t')$ (alternative), $while(t, t')$ (loop). The conditional structure $if(t, t', t'')$ requires a third task t''.

The evaluation of $then(t, t')$ launches t, and upon termination, launches t'. The result of t is passed as input to t'. The final result is the tuple $(then, res(t), res(t'))$ (the result of t is preserved in the trace of the overall execution). The evaluation of $and(t, t')$ launches both tasks in parallel, and both must terminate for the *and* task to finish. The result is the tuple $(and, res(t), res(t'))$. The evaluation of

$or(t, t')$ launches both tasks in parallel, the first to finish, say t, interrupts the other. The result is the tuple $(or, res(t))$. The evaluation of $if(t, t', t'')$ launches the evaluation of t. Upon termination, if the first tuple of the result matches $(\_, true)$, task t' is launched. Otherwise t'' is launched. The if result is the tuple $(if, res(t))$ or $(if, res(t'))$ according to the condition output. Finally, the evaluation of $while(t, t')$ launches t and t' each time the result of t matches the tuple $(\_, true)$, and otherwise finishes. The result is the tuple $(while, res(t^*))$ where t^* is the last execution of t' in the loop.

Based on this syntax, the translation of our running example would be the following, if we expect at least 2 answers of each kind:

```
then(
    ask(2,"Give the location of nearby hospitals",
    [emergency,hospital]),
    and(
      then(
        ask(2,"Obtain phone numbers of the previous hospitals",
          [emergency,hospital]),
        ask(2*"Using the previous hospital phone numbers,
         obtain their availability",
         [emergency,hospital])
      )
      ask(2,"List passable roads reaching the
         previous hospitals",
         [road, traffic]
      )
    )
)
```

While this syntax is sufficient, we also provide an infix notation for quick task design for *ask*, *and*, *or* and *then*, using shortcuts $\{*\}$, &&, || and ;; respectively. The corresponding notation of the previous example is the following:

```
{ 2*"Give the location of nearby hospitals"
    [emergency,hospital]
}
;;
( (
    { 2*"Obtain phone numbers of the previous hospitals"
        [emergency,hospital]
    }
    ;;
    { 2*"Using the previous hospital phone numbers,
         obtain their availability"
         [emergency,hospital]
    }
  )
  &&
    { 2*"List passable roads reaching the
         previous hospitals"
         [road, traffic]
    }
)
```

Expertise Update. Once a complex task T is finished, the tasker can see the result and all task intermediate results and provenance. Then the tasker can validate the overall task or reject it. In case of validation (resp. rejection), we set a *score* value to 1 (resp. -1). We consider a participant *uid* who contributed to T. We update this participant's expertise according to this score, for each expertise tag he contributed. More precisely, let $E_T = \{e_1, \ldots, e_n\}$ be the set of expertise tags associated with the basic tasks of T that were answered by participant *uid*. We modify the expertise of *uid* such that

$$E_{uid}(e_i) := E_{uid}(e_i) + score, \text{ for each } i = 1 \ldots, n.$$

4. SYSTEM OVERVIEW

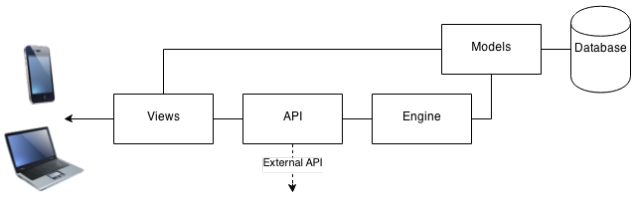


Figure 1: The CROWD Architecture

Figure 1 shows the overall architecture of the CROWD system. It is composed by several components: a web-based graphical user interface, an internal API, a database and an engine for task scheduling.

The GUI is based on the Django framework. Users can log using a registered account or anonymously. A panel shows all the available tasks ranked by expertise requirements. Another panel allows for task launching. Tasks can be entered either using the concrete syntax described in Section 3, or using a workflow design interface shown in Figure 2.

Tasks are modeled as python objects and mapped to a Postgresql database using the Django ORM module with concurrency control enabled. The classical execution cycle of the system is as follows. Each complex task is decomposed as a set of task nodes. Each root node begins in the *start* state. Then the tree of tasks is visited top-down: the left child of a sequence (*then*) node is started, and all child of a started *and* or started *or* node, and the conditions of a started if or *while* node are started. Any available basic task in the *start* state is presented on the user interface. Each time a basic task is answered by a participant, a counter is updated until the total number of awaited answers is reached. Then, the basic task turns to the $finished$ state. The termination of a child of a *or* node terminates the node. The termination of all the child node of an *and* node terminates the node. The termination of the left child of a *then* node launches the right node (similarly for f and *while* nodes).

When a complex task is fully answered, its state changes and becomes $finished$. At this point the tasker can get his answers thanks to the related web page. Results can be exported in a CSV document. The tasker can validate or reject the task, and users experience is updated accordingly.

5. DEMONSTRATION

In the demonstration, we will illustrate the following aspects of the CROWD platform: logging and task answering, basic and complex task creation and advanced XHTML features. A video of a preliminary version of our demo is available here:

`https://www.youtube.com/watch?v=3Zd6QpHpYhQ&feature=youtu.be`

Logging and Task Answering. In the first part of this demo, users are invited to log on the platform, either by defining a user account, or by logging anonymously in one click (anonymous logging is an incentive for the free participation to interesting tasks, without leaving personal traces on the system). Logging can be performed either on the website or on the user's personal mobile phone.

Users will be able to browse the available tasks, to select one (for example, "How many persons is now attending this demo ?"), and to answer it on the website or on their phones.

Basic and Complex Task Creation. In this part, users will be able to define a new task, either using the concrete syntax in an editor, or by using a workflow editor that allows to describe a tree in graphical mode (this last feature is not available on phones). After defining basic tasks, we will demonstrate the use of connectors to build a complex task. Finally, we will show how to express the required expertise for these tasks.

Advanced Features. In the last part of the demo, the Miami storm scenario is launched, allowing the audience to participate (virtually) in the relief. Participants are invited to locate hospitals on an interactive Google Map, launched from the task board. It is noteworthy that any XHTML code can be inserted in task descriptions, which allows for rich interaction with external platforms (Figure 3).

Monitoring Tasks. All along its lifetime, users can watch the progression of a complex task and of its related basic tasks using the "My Tasks" panel. By clicking on the answer link, detailed answers of each basic tasks composing the complex task are shown. An export as a csv file is available.

Expertise Ranking. We will demonstrate how the validation of a task impacts on the user's expertise, and how this expertise affects the ranking of the next proposed task.

Acknowledgements

We would like to thank the following persons for their help with the development of CROWD during their undergraduation project: Pierre-Luc Blay, Thomas Daniellou, Archibald Jego, Guillaume Landurein, Sylvain Medard, Houssam Ouazzani and Victor Petit.

6. REFERENCES

[1] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. The jabberwocky programming environment for structured social computing. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 53–64, 2011.

[2] Daniel W. Barowy, Charlie Curtsinger, Emery D. Berger, and Andrew McGregor. Automan: a platform for integrating human-based and digital computation. *SIGPLAN Not.*, 47(10):639–654, October 2012.

[3] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Andrea Mauri. Reactive crowdsourcing. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 153–164, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[4] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. Choosing the right crowd: Expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 637–648, New York, NY, USA, 2013. ACM.

[5] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 61–72, 2011.

[6] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proc. VLDB Endow.*, 5(1):13–24, September 2011.

[7] Atsuyuki Morishima, Norihide Shinagawa, Tomomi Mitsuishi, Hideto Aoki, and Shun Fukusumi. Cylog/crowd4u: a declarative platform for complex data-centric crowdsourcing. *Proc. VLDB Endow.*, 5(12):1918–1921, August 2012.

[8] Hyunjung Park, Richard Pang, Aditya Parameswaran, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. An overview of the deco system: data model and query language; query processing and optimization. *SIGMOD Rec.*, 41(4):22–27, January 2013.

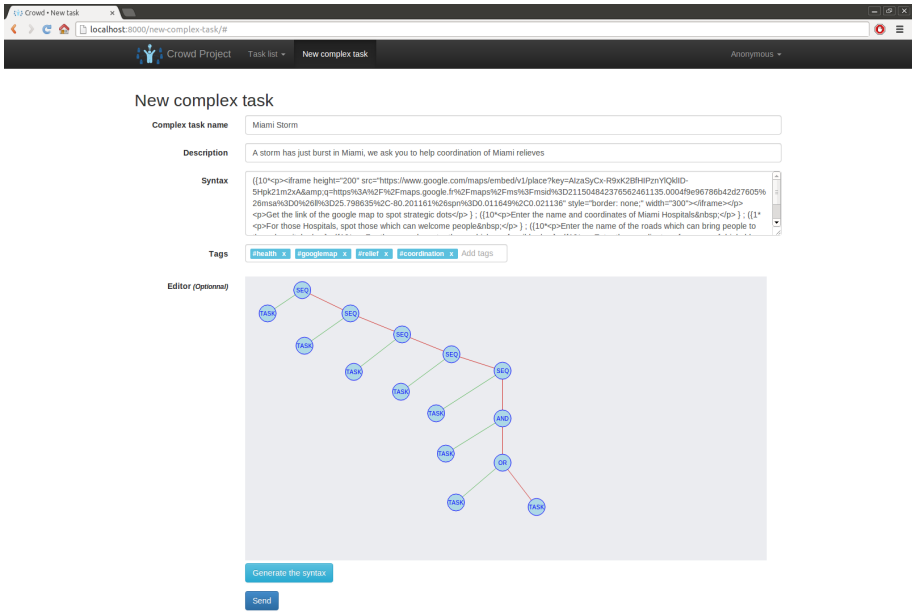


Figure 2: Screenshot of a complex task creation

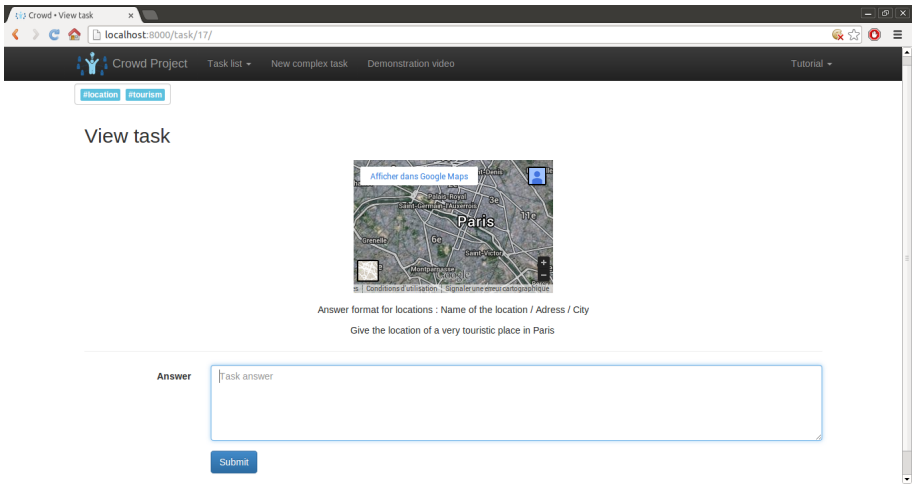


Figure 3: Advanced XHTML features