On the Use of Hash Maps for Data Reconciliation Optimization over a Data Integration System

Abdelghani Bakhtouchi and M'hamed Mataoui

***Abstract*—The invention of the Internet and the emergence of the World Wide Web revolutionized people's access to digital data stored on electronic devices. But unlike traditional data management applications, the new services require the ability to share data among multiple applications and organizations, and to integrate data in a flexible and efficient fashion. Data integration systems enable building systems geared for flexible sharing and integration of data across multiple autonomous data providers. In this paper we propose the use of hash map structure to optimize the data reconciliation over our data integration system. We have noticed that the use of the array data structure to store the intermediate results of the reconciliation takes a considerable time. This led us to change it by the hash map that assures a reconciliation runtime much less than when using arrays. We validate our choice again theoretically and experimentally.**

***Index Terms*—Data integration, hash map, optimization, reconciliation.**

I. Introduction

In past years, Enterprise and Information Integration (EII) became an established business, with academic and commercial tools integrating data and XML sources more readily available. These tools offer final users a uniform and a transparent access to data. The spectacular development of this business has been motivated by the need companies have to be able to access data allocated over the internet and within their intranets [1]-[4].

The construction of a data integration system is a difficult task due to the following main factors: (a) the large number of data sources candidate for integration; (b) the lack of semantic sources explicitness; (c) the heterogeneity of sources; and (d) the autonomy of sources.

To deal with semantic problems and ensure an automatic data integration, a large number of research studies propose the use of ontologies. Several integration systems were proposed under this hypothesis. We can cite for instance: COIN [5], Observer [6], OntoDaWa [7], etc.

More and more sources explicit the semantic of their data using existing ontologies which are largely developed in several application domains: medicine (Unified Medical Language System), engineering (e.g., IEC [8]), biology[1], business intelligence applications, etc. The storage of ontologies in a database leads to the concept of Ontology-Based DataBases (OBDB). Several academic and industrial systems offer efficient solutions to store and manage ontologies and their associated data, for instance: Jena [9], Sesame [10], Oracle [11] and IBM Sor [12]. If we follow this trend, OBDB sources become then candidate for the integration process. Therefore, integration services should be developed for this type of sources.

Once source heterogeneity is solved by the use of ontologies, data integration designers have to propose solutions for data reconciliation when queries are answered over the integration system (following mediator architecture). In [13], we have proposed a complete integration methodology that incorporates these issues in a mediator architecture. It is mainly motivated by a conjunction of two main factors:

- The conceptual continuity offered by ontologies to generate conceptual models [14] and to ease the resolution of data heterogeneity
- The spectacular development of OBDB sources that may need to be integrated.

One might assume that query processing in a data integration system differs little from query processing in a traditional DBMS. After all, the query language (whether SQL, datalog or XQuery) is based on standard relational (or extended relational) operations. Its goal remains to find an efficient executable plan for the query. While data integration queries often process distributed data, even this problem has been studied in the context of distributed and federated database systems [1]. Therefore, the conventional optimization algorithms used in the databases can't be all applied in the case of heterogeneous data sources optimization. This led to poor (or no) knowledge of the properties of the manipulated data (index, distribution, patterns or cardinality). Despite these cursory similarities, data integration actually offers a number of challenges that require novel solutions.

The query execution method that we proposed involves five steps: (1) discovery of the query functional dependency; (2) concerned sources determining; (3) reconciliation key derivation; (4) queries evaluation; and finally (5) results reconciliation and fusion. It is clear that the runtime of the first three steps is negligible compared to steps (4) and (5). Although the step (4) is beyond the responsibility of the integration system, it relates to interrogated sources. We proposed a method to reduce the number of such sources to reduce the execution time of step (4) [15]. Our final margin of maneuver to optimize the queries response time is to reduce the runtime of the last step, namely results reconciliation and fusion [16].

Manuscript received in April 5, 2018; revised June 11, 2018.

Abdelghani Bakhtouchi is with Ecole nationale Supérieure d'Informatique (ESI) and Ecole Militaire Polytechnique (EMP), Algiers, Algeria (e-mail: a_bakhtouchi@esi.dz).

M'hamed Mataoui is with Ecole Militaire Polytechnique, Algiers, Algeria (e-mail: mataoui.mhamed@gmail.com).

[1] http://www.iplantcollaborative.org/

doi: 10.18178/ijmlc.2018.8.4.705

We have noticed that the use of the data structure array to store the intermediate results of the reconciliation takes a considerable time. This led us to change it by the hash map data structure that assures a reconciliation runtime much less than when using arrays. We justify our choice again theoretically and experimentally.

The remainder of this article is organized as follows: Section II summarizes state of the art related to data integration systems. Section III presents the reconciliation method of a result coming from a source and the global result. In Section IV, we give experimental results and Section V concludes the paper.

II. Related Works

A key challenge in developing an effective reconciliation solution is that some of the requirements are in conflict with others, for example, efficiency and effectiveness or genericity and facility of use. In one hand, using blocking methods improves efficiency by reducing the search space. However, this can eliminate some relevant entity pairs and thus reduce the effectiveness (recall) of the reconciliation. On the other hand, the combined use of various reconciliation algorithms can improve effectiveness, but increase computing time and thus reduce efficiency. Reconciling entities in different domains with a generic reconciliation solution is more difficult than for a single domain. A non-generic reconciliation solution may therefore require a reduced manual effort to provide the learning data or to find an appropriate combination and customization of the algorithms.

In large data integration, data sources tend to be heterogeneous in their structure. Also, many sources provide unstructured text data and data sources are dynamic and evolving. These characteristics make reconciliation of data particularly a difficult task [17]. When there are a large number of sources and a large volume of data, traditional reconciliation methods become ineffective in practice. To deal with the volume dimension, new techniques have been proposed to allow parallel reconciliation of data using MapReduce [18]. These include blocking techniques and techniques that dispense the charge between different nodes. When data sources are dynamic and ever-changing, applying reconciliation from the beginning for each update becomes unaffordable. To cope with the speed aspect, incremental clustering techniques [19] have been proposed.

Obviously, none of the strategies are perfect for resolving conflicts. They are all deprived of all or part of the following three aspects: the *accuracy* of the sources, the *freshness* of the sources and the *dependencies* between the sources [20].

First of all, the data sources are of different qualities and we often trust the data from the most accurate sources, but the precise sources may make mistakes as well. Therefore, neither the consideration that all sources are alike, nor the taking of all data from specific sources without verification, is appropriate. The work in [21], [22] and [23] propose to examine the accuracy of sources when deciding real values through probabilistic models that calculate the iterative accuracy of sources.

Second, the real world is dynamic and the real value often changes over time, but it is difficult to distinguish between incorrect values and out-of-date values. Thus, the most common value may be a value exceeded, while the most recent value may be a wrong value. In [24], authors propose a probabilistic model integrating the concept of freshness of the sources in order to solve the problem of finding the correct values.

Third, sources can integrate instances from other sources. As a result, errors can spread quickly. Thus, the negligence of possible dependencies between sources can lead to biased decisions because of the copied information. The proposal presented in [25] and [26] take into account dependencies between the sources during the discovery of the correct values. They use algorithms detecting iteratively dependency between sources.

III. Reconciliation of Query Result

Our architecture is composed of five components, namely: (1) a user interface; (2) an OBDB; (3) a cashing manager; (4) a query engine; and (5) a reconciliator of results [27], [28].

(1) The user interface allows the user to express his query and is responsible of displaying corresponding results.

(2) The used OBDB adopts the OntoDB model with extensions of the meta-schema as presented in [13] and [29].

Our proposal consists of adding a model of mapping between the mediator ontology and sources ontologies to the meta-schema of the OntoDB model. In the ontology part we store the mediator ontology, sources ontologies, the mapping and the functional dependencies between the classes and properties of the mediator ontology. We use the data part as cashing, in which we save the results of recently queries in order to a future re-use.

(3) The cashing manager allows three functionalities: (i) identifying a class instances present in the cashing after a request of the query engine; (ii) consulting the cashing to form the answer to a query after a request of the reconciliator; and (iii) updating the cashing after the execution of a query.

(4) The query engine identifies the instances present in the cashing in order to devise the user query into two queries, the first is executable on the cashing and the second will be sent to the sources. It rewrites the second query written in terms of the mediator ontology into a query written in terms of sources ontologies using the mapping. It generates then the query reconciliation plan. It sends each sub-query to the concerned source and sends the query reconciliation plan to the reconciliator.

(5) The reconciliator recomposes the results returned by the different wrappers after consultation of the cashing and the query reconciliation plan.

The reconciliation of result coming from a source and the global result can be performed by Algorithm 1. This algorithm takes each instance from the source result and checks if there is an instance which can be reconciled with it in the global result. If such instance exists, the algorithm merges the properties values of the two instances in a single instance in the global result; otherwise, the instance of the source is added to the global result as a new instance.

A. Reconciliation Using Arrays

To reconcile the result coming from a source with the

global result, we must look over the nS instances of the source result (line (1) of the Algorithm 1), implying a complexity of $O(n)$. For each source instance we must look over the nG instances of the global result looking for this instance (line (2) of the Algorithm 1), implying a complexity of $O(n)$. The reconciliation therefore requires a time equal to $nS \times nG$ iterations, a complexity of $O(n^2)$. The reconciliation of results coming from n sources requires a time equal to $nS \times nG \times n$ iterations, a complexity of $O(n^3)$.

Algorithm 1 – Reconciliation of a source result

Input: K_R: Reconciliation key;
ans(Q_i^{Sj}) : Source result;
R: Global result;
Output: R: Global result;
Begin
(1): **For each** $i_2 \in$ ans(Q_i^{Sj}) **do**
(2): **If** $\exists\, i_1 \in R$ *Reconcile*(i_1, i_2) **Then**
i_1 = *FusionOf* (i_1, i_2);
Else
Add i_2 to R;
End If
End For
End

B. Reconciliation Using Hash Maps

Using a hash map to store intermediate results can look over instances of global result in search of an instance in a time equal to one iteration instead of nG iterations, implying a complexity of $O(1)$. So, the reconciliation of result coming from n sources requires a time equal to $nS \times 1 \times n$ iterations, implying a complexity of $O(n^2)$.

As nG and nS are very large compared to n, so:

$$nS \times nG \times n >> nS \times n$$

This involves that the use of hash map saves a very considerable process time.

IV. Experiments

In this experiment we begin first with the presentation of the test data preparation. We present later, the response time of queries when we use arrays to store intermediate data. We then present, the response time of queries when hash maps are used. Finally, we make a comparison between these two response times to validate our choice.

A. Test Data Preparation

To measure the efficiency of our proposition, we conduct experiments using dataset of Lehigh University Benchmark (LUBM) and its 14 queries[2]. The used ontology of LUBM has 45 classes and 32 properties (including 25 object properties and 7 data type properties). Based on this ontology a set of ontology-based databases is generated. The experimental protocol used is described as follows:

1) Generation of 30 sources (OWL files) based on the LUBM benchmark ontology using the data generation tool *UBA 1.7*.
2) Recuperation of data sources (OWL files) in a database (*BaseTriplets*) as triplets (Subject, Predicate, object) using the knowledge base management system ontowiki[3].
3) Creation of a database schema from the *univ-bench.owl* ontology.
4) Transformation of the database schema into an ontology based database conforming to OntoDB model.
5) Insertion of the *univ-bench.owl* ontology in a starting ontology based database OBDB_Init.
6) Creation of 60 ontology-based databases (obdb1, ..., obdb60) from OBDB_Init.
7) Importation of data to sources from *BaseTriplets* containing triplets.
8) Creation of a mediator and importation of the *univ-bench.owl* ontology to it.
9) Integration of the 60 ontology-based databases in the mediator.
10) Execution of the 14 queries over 10, 20, 30, 30, 40, 50 and 60 databases.

All experiments have been carried out on an Intel platform, with 3.2 GHz processor clock frequency, under the Windows operating system.

B. Using Arrays

We implemented our first prototypes using the "array" data structure. We conducted a series of tests on the collection of generated data. Experiments are performed on a set of 60 ontology-based databases, each contains a table *Students*(*personId*, *name*, *address*, *age*). The number of instance of the table *Students* is varied using: 128, 1024, 16384 and 131072 instances.

The following query is executed over 10, 20, 30, 40, 50 and 60 databases:

SELECT name, address FROM Students WHERE name like "...%"

We recorded two different times: (1) The response time of the query before the results reconciliation and (2) the response time of the query after the results reconciliation.

The aim of these experiments is to compare the results reconciliation related to the query execution time.

We compared the response time of queries for tables containing 128, 1024, 16380 and 131072 instances. We varied the number of query result of 4, 64, 256 and all table instances.

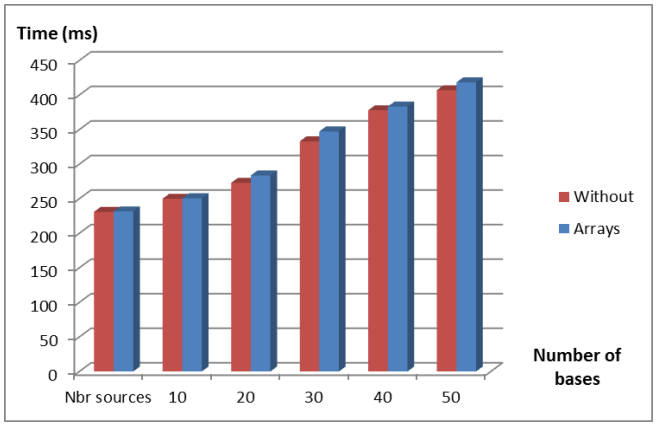

Fig. 1. Response time of a query whose the result contains 4 instances when using arrays.

Fig. 1, Fig. 2 and Fig. 3 respectively show the results for

[2] http://swat.cse.lehigh.edu/projects/lubm/

[3] www.ontowiki.net

queries whose response contains 4, 64 and 256 instances.

These figures show that the response time increases rapidly when the number of instances that contains the query increases. This implies that the choice of using arrays was an inappropriate choice.

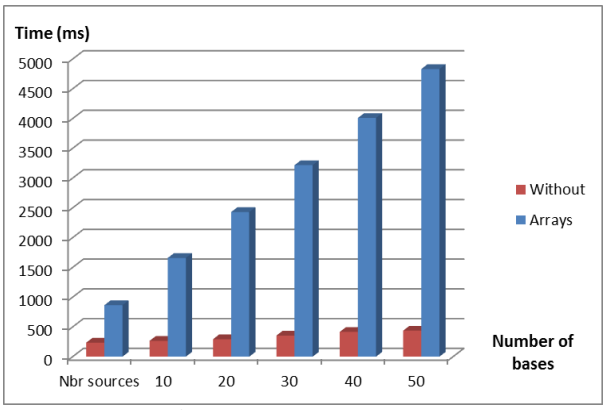

Fig. 2. Response time of a query whose the result contains 64 instances when using arrays.

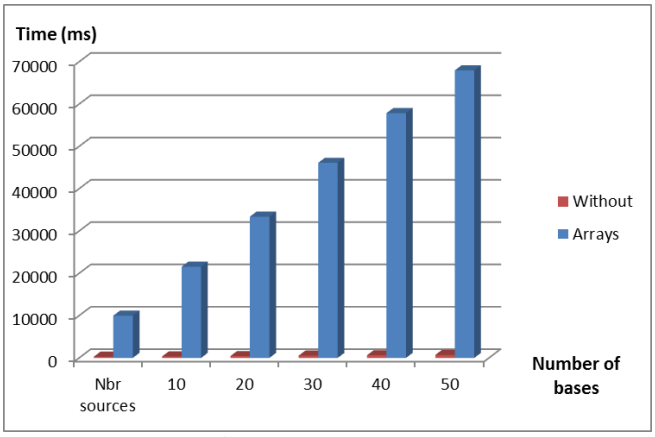

Fig. 3. Response time of a query whose the result contains 256 instances when using arrays.

C. Using Hash Maps

We performed the same tests as before using hash maps.

By using the hash maps structure, the response time after reconciliation has significantly improved to a point where the difference between the time without reconciliation and after reconciliation is negligible except when the query result from each source contains more than 1000 instances.

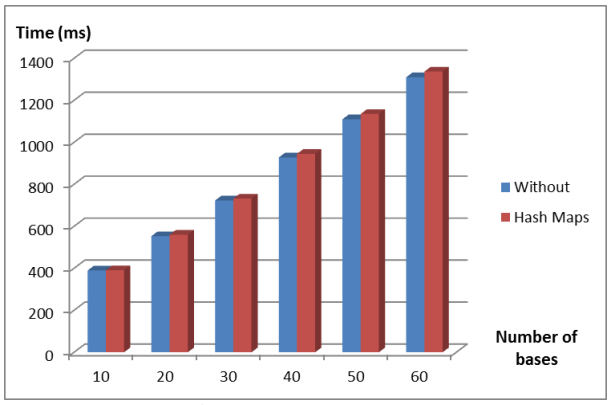

Fig. 4. Response time of a query whose the result contains 256 instances when using hash maps.

Fig. 4 (respectively Fig. 5) shows the results for queries whose response contains 256 instances (respectively 16380).

We notice that the response time increases very slowly when the number of instances containing the query increases. This implies that the choice of using hash maps can be considered as a good choice.

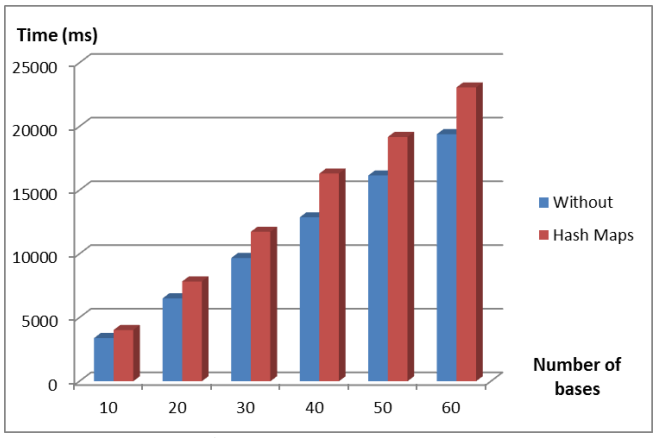

Fig. 5. Response time of a query whose the result contains 16380 instances when using hash maps.

D. Comparing Response Times by Using Arrays vs Hash Maps

Fig. 6 shows a comparison of a query response times related to the use of arrays and those related to hash maps. The two queries results contain 16380 instances.

This figure clearly shows a large divergence between the two curves, which confirms that hash maps structures are much better than using arrays in our context.

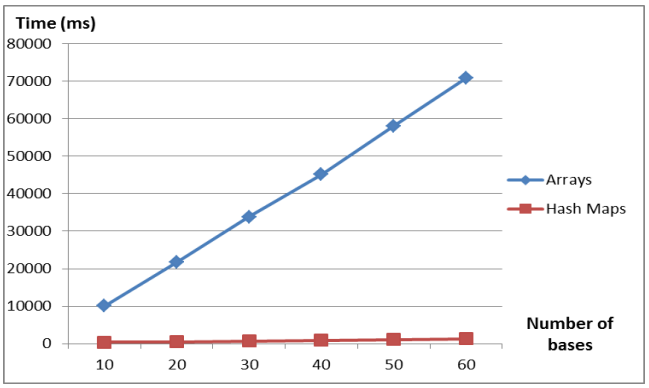

Fig. 6. Comparison between response times related to the use of arrays and hash maps for a query result containing 16380 instances.

V. Conclusion

The last and crucial phase in a data integration system is the result fusion. The needed time to achieve this phase defines the efficiency of such system. To improve the reconciliation of our integration system that used the "array" data structure to store the reconciliation intermediate results, we proposed to change the "array" data structure by the "hash map" data structure which provides a reconciliation runtime much less. We have demonstrated the effectiveness of this choice theoretically and through performed experiments comparison. An improvement can be done in the future is to implement more technical for data fusion.

References

[1] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*, 2012.

[2] A. Gusmini and M. Leida, "Data integration system," U.S. Patent 0,006,968 A1, January 3, 2013.
[3] A. D. Sarma, X. L. Dong, and A. Y. Halevy, "Data integration with dependent sources," in *Proc. 14th International Conference on Extending Database Technology*, Uppsala, Sweden, 2011, pp. 401–412.
[4] A. Y. Halevy, N. Ashish, D. Bitton, M. J. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka, "Entreprise information integration: successes, challenges and controversies," *SIGMOD*, pp. 778–787, 2005.
[5] C. Goh, S. Bressan, E. Madnick, and M. D. Siegel, "Context interchange: New features and formalisms for the intelligent integration of information," *ACM Transactions on Information Systems*, vol. 17, no. 3, pp. 270–293, 1999.
[6] E. Mena, V. Kashyap, A. P. Sheth, and A. Illarramendi, "Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies," *CoopIS* pp. 14–25, 1996.
[7] D. N. Xuan, L. Bellatreche, and G. Pierra, "A versioning management model for ontology-based data warehouses," in *Proc. the International Conference on Data Warehousing and Knowledge Discovery*, Krakow, Poland, 2006, pp. 195–206.
[8] ISO13584-25: Industrial automation systems and integration - parts library - part 25: Logical resource: Logical model of supplier library with aggregate values and explicit content. Technical report, International Standards Organization, Genève, 2004.
[9] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proc. the 13th international World Wide Web Conference*, New York, USA, 2004, pp. 74–83.
[10] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema," in *Proc. International Semantic Web Conference*, Springer, Sardinia, Italy, 2002, pp. 54–68.
[11] S. Das, E. I. Chong, G. Eadon, and J. Srinivasan, "Supporting ontology-based semantic matching in RDBMS," in *Proc. the International Conference on Very Large Databases*, Toronto, Canada, 2004, pp. 1054–1065.
[12] J. Lu, L. Ma, L. Zhang, J.-S. Brunner, C. Wang, Y. Pan, and Y. Yu, "Sor: a practical system for ontology storage, reasoning and search," in *Proc. the International Conference on Very Large Databases*, Vienna, Austria, 2007, pp. 1402–1405.
[13] A. Bakhtouchi, L. Bellatreche, S. Jean and Y. Aït-Ameur, "MIRSOFT: mediator for integrating and reconciling sources using ontological functional dependencies" *International Journal of Web and Grid Services*, vol. 8, no. 1, pp. 72-110, 2012.
[14] L. Bellatreche, Y. Aït-Ameur, and C. Chakroun, "A design methodology of ontology based database applications," *Logic Journal of the IGPL*, vol. 19, no. 5, pp. 648–665, 2011.
[15] A. Bakhtouchi, "Annotation des propriétés des ontologies: Une approche d'optimisation des requêtes sur un médiateur de sources de données à base ontologique,"*Technique et Science Informatiques, Revue des Sciences et Technologies de L'information*, vol. 33, no. 4, pp. 371-398, 2014.
[16] A. Bakhtouchi "Méthodes de réconciliation et de fusion des données: un survey," in *Proc. 6th International Conference on Software Engineering and New Technologies*, Hammamet, Tunisia, 2017.
[17] D. X. Luna and D. Srivastava, "Big data integration," *Synthesis Lectures on Data Management*, vol. 7, no. 1, pp. 1-198, 2015.
[18] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," 2008, *Communications of the ACM*, vol. 51, no. 1, pp. 107-113.
[19] G. Anja, X. L. Dong, and D. Srivastava, "Incremental record linkage," in *Proc. the VLDB Endowment*, vol. 7, no. 9, pp. 697-708, 2014.
[20] D. X. Luna and F. Naumann, "Data fusion: Resolving data conflicts for integration," in *Proc. the VLDB Endowment*, vol. 2, no. 2, 2009, pp. 1654-1655.
[21] X. Yin, J. Han, and S. Y. Philip, "Truth discovery with multiple conflicting information providers on the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796-808, 2008
[22] M. J. Wu and A. Marian, "Corroborating answers from multiple web sources," *WebDB*, 2007.
[23] S. A. Das, X. L. Dong, and A. Halevy, "Data integration with dependent sources," in *Proc. the 14th International Conference on Extending Database Technology*, ACM, 2011.
[24] X. L. Dong, B.-E. Laure, and D. Srivastava, "Truth discovery and copying detection in a dynamic world," *Proceedings of* the *VLDB Endowment*, vol. 2, no. 1, pp. 562-573, 2009.
[25] B.-E. Laure *et al*., "Sailing the information ocean with awareness of currents: Discovery and application of source dependence," *arXiv preprint arXiv:0909.1776*, 2009.
[26] X. L. Dong, B.-E. Laure, and D. Srivastava. "Integrating conflicting data: the role of source dependence,"*Proceedings of the VLDB Endowmen* , vol. 2, no. 1, pp. 550-561, 2009.
[27] A. Bakhtouchi, "Intégration et réconciliation des données hétérogènes: Une approche ontologique dans une architecture de médiation," Presses Académiques Francophones, 2013.
[28] A. Bakhtouchi, L. Bellatreche, S. Jean, and Y. Aït-Ameur, "Ontologies as a solution for simultaneously integrating and reconciliating data sources," in *Proc. Sixth International Conference on Research Challenges in Information Science (RCIS)*, 2012, pp. 1-12.
[29] A. Bakhtouchi, C. Chakroun, L. Bellatreche, and Y. Aït-Ameur, "Mediated data integration systems using functional dependencies embedded in ontologies," *Book Chapter by Springer Verlag in Recent Trends Information Reuse and Integration Book*, pp. 227-256, 2011.

Abdelghani Bakhtouchi was born in Algeria in 1977. He received a PhD degree in computer science from Ecole nationale Supérieure d'Informatique (ESI), Algiers, Algeria, in 2013. He received a master's degree in computer science from the same school in 2007. He received an engineer's degree in computer science from Ecole Militaire Polytechnique (EMP), Algiers, Algeria, in 2001. He has more than sixteen years of teaching, research and industry experience. He has various publications and worked for committees in national and international journals and conferences. His current interest includes data integration and sentiment analysis.

M'hamed Mataoui was born in Algeria in 1979. He received a PhD degree in computer science from Université M'Hamed Bougara de Boumerdès (UMBB), Algeria, in 2016. He received a master's degree in computer science from the same university in 2007. He received an engineer's degree in computer science from Ecole Militaire Polytechnique (EMP), Algiers, Algeria, in 2003. He has more than fifteen years of teaching, research and industry experience. He has various publications and worked for many committees in national and international conferences. His current interest includes information retrieval, natural language processing and sentiment analysis.