# A Novel Approach for a Collaborative Exploration of a Spatial Data Cube

Olfa Layouni and Jalel Akaichi

***Abstract*—Spatial data warehouses store large volumes of consolidated and historized multidimensional spatial data in order to be explored and analyzed by various users. The data exploration is a process *of searching relevant information in a data set. The data set t*o explore is a spatial data cube taken out from the spatial data warehouse that users interrogate by launching sequences of SOLAP (Spatial On-Line Analytical Processing) queries. However, this volume of information can be very large and diversified; it is thus necessary to help the user to face this problem by guiding him/her in his/her spatial data cube exploration in order to find relevant information.**

***Index Terms*—Spatial data, spatial data warehouse, spatial data cube, SOLAP (spatial on-line analytical processing) queries.**

## I. Introduction

Spatial data warehouse stores a huge quantity of spatial data. The work of [1] estimated that the percentage of storing this type of data in the future years will increase more and more. According to the authors in [2] a spatial data warehouse was defined as a collection of spatial and thematic data, integrated, no volatile and historical data to make the best decision. In addition, a spatial data warehouse integrates and stores large volumes of spatial and no spatial data from multiple sources. It is realized from a spatial multidimensional model which defines the concepts of spatial measures and dimensions to take account of the spatial component [3], [4]. A spatial data warehouse supports three types of spatial dimensions: the non-geometric spatial dimensions, the geometric spatial dimensions and the mixed spatial dimensions and also supports two types of spatial measures: the first type of spatial measures is a set of all the geometries representing the spatial objects corresponding to a particular combination of dimension members. The second type of spatial measures results from the computation of spatial metric or topological operators [5]–[8]. In order to analyze and explore a spatial data warehouse, we need a SOLAP server to help the user to make the best decisions.

According to the author in [9] OLAP tools offer no analytical instrument and exploration of spatial data that can help the user to make the best decision. Therefore, a solution has been developed under the term Spatial OLAP [8].

The Spatial OLAP has been identified as an effective means to explore the contents of a spatial data warehouse. The Spatial OLAP is the result obtained after the combination of Geographic Information Systems (GIS), with OLAP tools. To navigate in the spatial data cube the user launches a sequence of SOLAP queries over a spatial data warehouse. A spatial data cube can be queried by using the MDX (Multi-Dimensional eXpressions) with spatial extensions query language, named also the SOLAP queries [4].

SOLAP users interactively navigate a spatial data cube by launching sequences of SOLAP queries over a spatial data warehouse. The problem appeared when the user may have no idea of what the forthcoming SOLAP query should be. As a solution and to help the user in his navigation, we need a recommendation system. This system gives the possibility to recommend SOLAP queries based on the SOLAP server query log. In fact, a recommendation system is usually categorized into a content-based method, a collaborative method and a hybrid method [7], [10].

- Content-based method: The user is recommended elements similar to the ones the user preferred in the past.
- Collaborative method: The current user is recommended elements similar to the preferences of the previous users and the preferences of the current user.
- Hybrid method: This method combines both the content-based and the collaborative method.

In various studies [11]–[14], we find that the authors described the characteristics of the general algorithm of a recommender system for the exploration of data. These characteristics are the inputs, outputs and the recommendation steps. The inputs of the algorithm can be a log of sessions of queries, a schema, an instance of the relational or multidimensional database, a current session and a profile. The outputs of the algorithm can be a query, a set of ordered queries and a set of tuples. An algorithm of recommendation is decomposed into three steps [10], [11], [15].

The first step consists in choosing an approach for evaluating the used scores. In fact, in this step we can choose one of the categories of recommendation: a content-based, a collaborative and a hybrid method. The second step is the filter; this step consists in selecting the candidates' recommendations. The last step is the guide; this step consists in ordering the candidates' recommendations.

The problem we tackle in this paper is thus the following: How to help the user to design for forthcoming SOLAP queries, because when this user navigates a spatial data cube by launching a sequence of SOLAP queries he may have no idea of what the forthcoming SOLAP queries should be. As an answer, we propose to use what the SOLAP users did during their former exploration of the spatial data cube, and to use

Manuscript received November 3, 2013; revised January 24, 2014.

The authors are with the Computer science department, High Institute of Management, University of Tunis 41, rue de la Libert, 2000, Tunisia (e-mail: layouni.olfa89@gmail.com, j.akaichi@gmail.com).

this information as a basis for recommending what that forthcoming SOLAP queries could be.

Our contribution is to propose an approach for recommending SOLAP queries. For that, we have tree steps to make. The first step is the partition for the log of sessions of SOLAP queries to group the similar queries. The second step is the filter, for generating candidate SOLAP queries. The third step is the guide, for ordering the candidate SOLAP queries: the result obtained from the previous step. Adding to that, our approach has been implemented with the open sources GeoMondrian SOLAP server to recommend SOLAP queries (MDX with spatial extensions queries), in reality these SOLAP queries use the spatial functions (PostGIS functions when we use the GeoMondrian server).

This paper is organized as follows: Section II presents related works. Section III presents our approach for recommending SOLAP queries. Section IV presents our implementation. We discuss future work in Section V.

## II. Related Works

We introduce in this section the related works on the exploration of spatial data warehouse to provide recommendations to the user. For this reason, we present various methods that have been proposed to explore data.

### A. Existing Methods

In this section, we survey the methods that recommend queries for helping users to explore data. In fact, those methods can be classified into two categories, the first category exploits the profile and so does the second category with the log of queries.

Concerning the first category, we find a lot of research for recommending queries in the exploration of the data warehouse by exploiting the profile. So, we present the methods proposed in [12]-[14], [16]-[18]. The method proposed by the author in [13] gives the possibility to improve the current query by using the preferences of the current user, and recommends the best query for him, so as to guide him in his exploration of data. This method is based on the content and use a heuristic for the exploration of the scores. Besides, this method can be described as a method of recommendation or a personalization. In fact, we find that this method resolves many problems but it doesn't take into consideration the sequencing of queries launched by the current user, it takes only the last query launched.

The method proposed by the authors in [17], [18] is proposed for the personalization of OLAP queries. In fact, this method resolves the problem of OLAP query personalization by taking into account visualization constraints. On the other hand, we find that this method is based on the content, and it can be described as a method of recommendation or a personalization. In fact, we find that this method doesn't take into consideration the previous queries launched by the previous users and the sequencing of queries launched by the current user. In the work of the authors in [12]–[14], the authors proposed operators DIFF, EXCEP, RELAX and INFORM. These operators can return all sets of tuples corresponding to the explanatory cell anomalies. In fact, we observed that this method is based on the content and recommend to the user a query. Besides, we find that some proposed operators execute the results obtained after launching the current query and other operators execute the current query only.

Concerning the second category, we find a lot of research for recommending queries in the exploration of data by exploiting the log of queries. So, we present the proposed methods in [11], [19]-[21]. Moreover, the method proposed by the author in [19], [20] gives the possibility to predict the next OLAP query that the user can request for the rest of the current session, because the next queries proposed by the current user don't interest him. We find that this method is a collaborative method that uses a statistical model, the Markov model. Adding to that, we find that he uses the sequencing of queries and the previous queries in the log but this method doesn't take into consideration the MDX queries. In addition, the method proposed by the authors in [21], gives the possibility to take into account the previous queries and the current query launched by the user when he explores the OLAP cube. In fact, this method is based on the history of user queries. Besides, this method gives the possibility to use the history of user queries by using the Apriori algorithm to extract the most frequently used attributes or measures. We find that this method takes into account the previous queries but not the sequencing of queries. Furthermore, the method proposed by the author in [11] gives the possibility to recommend MDX queries in exploring an OLAP cube. We find that this method is a collaborative method. Besides, it uses the sequencing of queries in the current session and takes into consideration the previous queries in the log. Adding to that, this method uses only the MDX when we explore the OLAP cube and doesn't use the MDX with spatial data when we explore a spatial data cube.

## III. New Approach for Recommending SOLAP Queries

To help the user to go forward in his exploration of the spatial data cube, we propose an approach for recommending SOLAP queries. It uses both the sequences of SOLAP queries of the current session and the sessions of SOLAP queries stored in the log. In fact, the sequences of SOLAP queries formerly launched on the spatial data cube.

Our approach consists of the three following steps. The first step consists in computing all the generalized sessions of SOLAP queries of the log. The second step is the filter which consists in predicting the candidates SOLAP queries. The last step is the guide that consists in ordering the candidates SOLAP queries. The *RecoSOLAP* algorithm represents the global algorithm of our approach. The first step in this algorithm is to preprocess sessions of SOLAP queries saved in the log, to obtain a generalized log. The second step uses the result obtained in the first step to search the most similar sessions to the generalized current session. Then, with the similar sessions, we search the set of candidates SOLAP queries. The results obtained in this step can be a set of candidates SOLAP queries or an empty set. If we obtain an empty set, the recommendation of queries is done by the default function and if we obtain a set of candidates SOLAP queries, we sort this set in the order of the most similar to the query that represent the current session. (See Fig. 1)

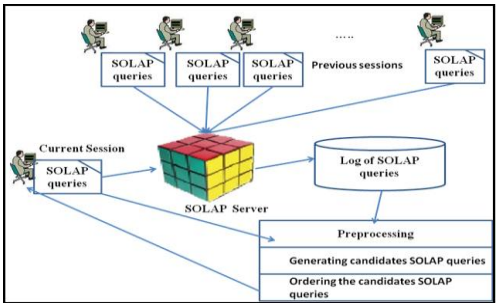

Fig. 1. Steps of the general algorithm.

**Algorithm 1** *RecoSOLAP (Sc, Log, Preprocessing, Sim, Rep, PSc, Ordering sort, Default)*

**Require:**
*Sc:* The current session,
*Log:* the log of sessions of SOLAP queries,
*Preprocessing:* A function used to generalize the log of sessions of SOLAP queries,
*PSc:* A function for generalized current session,
*Sim:* A function generating candidates sessions of SOLAP queries,
*Rep:* A function predicting recommendations queries from a set of candidates sessions,
*Ordering:* A function ordering the set of candidates SOLAP queries,
*sort:* Choosing order to sort queries: < or >,
*Default:* A function return default recommendations for user.
**Ensure:**
An ordered set of recommendations (one or more queries).

1: *Log_generalized* ← *Preprocessing (Log)*
2: *Set_Candidates_SOLAP_Queries* ← *Generating_ Candidates _ SOLAP _ Queries (Sc, Log_generalized, PSc, Sim, Rep)*
3: if *Set_Candidates_SOLAP_Queries* ≠ Ø then
4: return *Ordering (Set_Candidates_SOLAP_Queries, sort)*
5: else
6: return *Default (Log_generalized)*
7: end if

### A. Preprocessing the Log

This step consists in partitioning the log. The log contains all the previous sessions of SOLAP queries. In this step, we propose to use the distance of Levenshtein [22], [23] for computing the distances between SOLAP queries. Also, we propose to use the method of TF-IDF (Term Frequency-Inverse Document Frequency) [24] for evaluating the importance of terms like spatial measures, spatial dimensions... Beside, for doing the last classification of SOLAP queries, we choose the Hierarchical Ascendant Classification (HAC) [25].

The first thing to do is to extract all the spatial dimensions and measures from the schema of the spatial data cube. Then, we use them to classify queries in two different classifications C1 and C2. C1 contains the SOLAP queries without spatial data: OLAP queries and C2 contains the SOLAP queries. In fact, the queries classified in C2, can be classified in three sub- classes. To do those sub-classes, we use the TF-IDF method. The first subclass contains all the simple SOLAP queries without PostGIS functions and spatial calculate members. The second subclass contains all SOLAP queries with only PostGIS functions. And the last subclass contains all SOLAP queries with spatial calculate members.

The Fig. 2 illustrates these classifications.

Then, for C1 and the sub classes SOLAP, we calculate the distance between queries in the same class or subclass. Next, we use the hierarchical ascendant classification for getting a set of sets of SOALP queries. Finally, for each session of the log, we replace in the session each query with the class it belongs to. The generalized current session can be computed as well.

**Algorithm 2** *Preprocessing(Log)*

**Require:**
*Log*: the log of sessions of SOLAP queries.
**Ensure:**
A set of generalized sessions of SOLAP queries.

1: $i \leftarrow 1$
2: $j \leftarrow 1$
3: $k \leftarrow 1$
4: $D_s, M_s$ *Extract_spatial_ dimensions_measures (schema_ spatial _ data _ cube.xml)*
5: for *each query* $q_i \in Log$ do
6: $x \leftarrow$ *TF_IDF* ($q_i$,*Ds,Ms, Total_nb_queries, Log)*
7: if $x == 0$ then
8: *C1[j]* ←$q_i$ // clustering contains the set of OLAP queries
9: $j \leftarrow j + 1$
10: else
11: *C2[k]* ← $q_i$ // clustering contains the set of SOLAP queries
12: $k \leftarrow k + 1$
13: end if
14: end for
15 : *Distance_Matrix (C1;M)*
16: *Classify_queries (C1;M)*
17: $i \leftarrow 1$
18: $j \leftarrow 1$
19: $k \leftarrow 1$
20: $l \leftarrow 1$
21: for *each query* $q_i \in C2$ do
22: $x \leftarrow$*TF_IDF* ($q_i$, *functions_PostGIS, nb_queries_C2, C2)*
23: if $x == 0$ then
24: *C2.1[j]* ←$q_i$ // clustering contains the set of simple SOLAP queries
25: $j \leftarrow j + 1$
26: else
27: $y \leftarrow$*TF_IDF*($q_i$,*with_member, nb_queries_C2,C2)*
28: if $y! = 0$ then
29: *C2.2[k]* ← $q_i$ // clustering contains the set of SOLAP queries with spatial calculate
30: $k \leftarrow k + 1$
31: else
32: *C2.3[l]* ←$q_i$ // clustering contains the set of SOLAP queries with PostGIS functions
33: $l \leftarrow l + 1$
34: end if
35: end if
36: end for
37:*Distance_Matrix(C2.1,M1)*
38: *Classify_queries(C2.1,M1)*
39: *Distance_Matrix(C2.2,M2)*
40: *Classify_queries(C2.2,M2)*
41 : *Distance_Matrix(C2.3,M3)*
42: *Classify_queries(C2.3,M3)*
43: for *each query* $q_i \in S \in Log$ do
44: *Log_generalized* ← *Replace* ($q_i$, *classification_*$q_i$)
45: end for
46: return *Log_generalized*

**Algorithm 3** *Distance_Matrix(Class, Matrix)*

**Require:**
*Class*: A class contains queries that are of the same type,
*Matrix:* A matrix of distances, between queries in the same class.
**Ensure:**
A matrix contains distances between queries in the same class.

1: for *each query* $q_i \in Class$ do
2: for *each query* $q_j \in Class$ do
3: if $i == j$ then
4: *Matrix[i, j]* ←*0*
5: else

6: $d \leftarrow DistanceOfLevenshtein(q_i , q_j)$
7: $Matrix[i, j] \leftarrow d$
8: $Matrix[j, i] \leftarrow d$
9: end if
10: end for
11: end for

**Algorithm 4** Classify_queries(Class,Matrix)

**Require:**
*Class:* A class contains queries have the same type,
*Matrix:* A matrix of distances, between queries in the same class.
**Ensure:**
A table contains the final classification of each query.

1: for *each query* $q_i \in Class$ do
2: $j \leftarrow i$
3: for *each queries* $q_j \in Class$ do
4: *HAC (Matrix[i, j])*
5: end for
6: end for

### B. Generating Candidates Solap Queries

The previous step has generalized sessions of SOLAP queries. First, in this step, we propose to do the generalized current session. Then, we present a function *Sim* used to search among the set of generalized sessions of SOLAP queries the ones that are the most similar to the generalized current session. This function output is a set of pairs indicating which generalized sessions match with the generalized current session and the position of the matching. Then, we propose another function *Rep* which is used to obtain the query representing its session. This function gives the possibility to move from a candidate session to a candidate query. This set of queries is returned as an answer, it can be an empty set or a set of SOLAP queries or a set of OLAP queries or a set of SOLAP and OLAP queries.

**Algorithm 5** *Generating_Candidates_SOLAP_Queries (Sc, Log_ generalized, PSc, Sim, Rep)*

**Require:**
*Sc*: The current session,
*Log_generalized*: The generalized log of sessions of SOLAP queries obtained in the previous step,
*PSc*: A function for generalized current session,
*Sim*: A function generating candidates sessions of SOLAP queries,
*Rep*: A function predicting recommendations queries from a set of candidates sessions.
**Ensure:**
A set of candidates SOLAP queries:a set of SOLAP queries or a set of OLAP queries or a set of SOLAP and OLAP queries or an empty set.

1: $S \leftarrow$ *Set of sessions of SOLAP queries in the generalized log*
2: $PS \leftarrow PSc\ (Sc, Log\_generalized)$
2: $Candidates\_Sessions \leftarrow Sim\ (PS, S)$
3: $Set\_Candidates\_SOLAP\_Queries \leftarrow \emptyset$
4: if $Candidates\_Sessions \neq \emptyset$ then
5: for *each session* $\in Candidates\_Sessions$ do
6: $Set\_Candidates\_SOLAP\_Queries \leftarrow Set\_Candidates\_SOLAP\_Queries \cup Rep(s)$
7: end for
8: end if
10: return *(Set_Candidates_SOLAP_Queries)*

### C. Ordering the Candidates SOLAP Queries

In the previous step, a set of candidates SOLAP queries is returned. So, we order this set by calculating the distance by using the distance of Levenshtein between candidates SOLAP queries and the query representing the current session, and the order selected by the user. In this step we choose to order queries by using the quick sort.

### D. Default Recommendation

As previously noted, the set of candidates SOLAP queries can be empty. In that case, it could be useful to still be able to provide the user with default recommendations. Various default recommendations can be proposed to the user.

So, we choose to use the idea proposed by [26], we can propose as a default recommendation the representative of the authority class or the hub class.

The authority class has the highest number of successors and the hub class has the highest number of predecessors.

## IV. Implementation and Experimentation

### A. Implementation

In this section, we present the architecture of our system *RecoSOLAP*. This system applies the algorithm proposed in the Section III. Fig. 2 presents the architecture.

This system gives the possibility to recommend an ordered set of SOLAP queries for the user, after launching the current session. First, to navigate in the spatial data cube the current user launched a sequence of SOLAP queries by using the SOLAP server GeoMondrian over a spatial data warehouse Simple geofoodmart, stored in PostgreSQL integrating PostGIS 1.4. All the previous sessions of SOLAP queries are stored in the log. Finally, our *RecoSOLAP* system recommends an ordered set of SOLAP queries to the current user. (See Fig. 3 and Fig. 4).

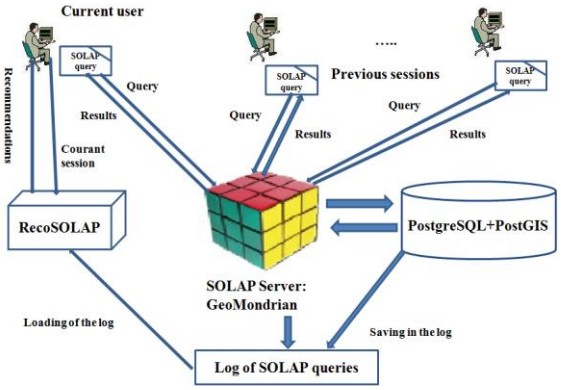


Fig. 2. The architecture of the*RecoSOLAP* system.

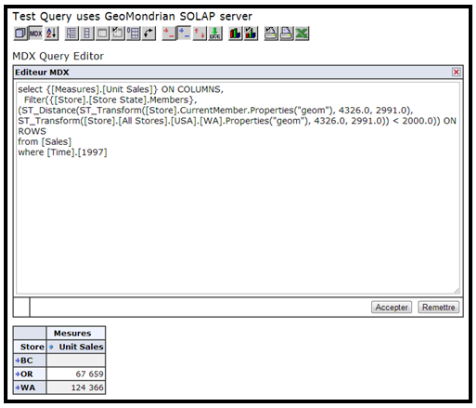


Fig. 3. An example of a SOLAP query launched by the current user via the SOLAP server GeoMondrian.

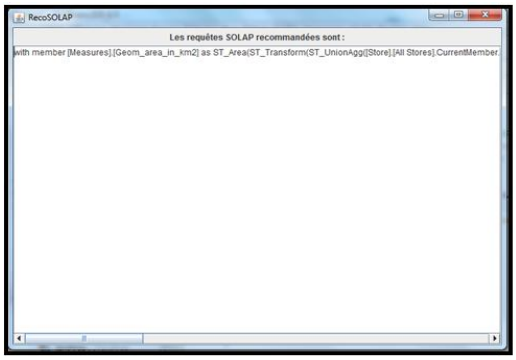


Fig. 4. Recommended SOLAP queries for the current user by using the *RecoSOLAP* system

### B. Experimentation

Our experiment evaluates the efficiency of our approach proposed to recommend SOLAP queries.

The performance is presented in Fig. 5 according to various log sizes. These log sizes are obtained by playing with parameters X (number of sessions) and Y (maximum number of queries per session). X ranges from 10 to 100 and Y ranges from 5 to 50. We thus obtain logs of size varying between 50 and 5000 queries. Note that what is measured is the execution time taken by the steps proposed: preprocessing the log, generating candidates SOLAP queries and ordering the candidates SOLAP queries.

So, the Fig. 5 shows that the time taken to recommend queries increases with the log size but remains highly acceptable.

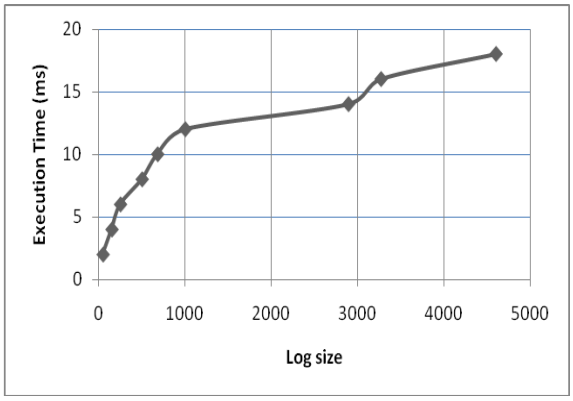


Fig. 5. Performance analysis.

## V. Conclusion

In this paper, we proposed a recommendation system to help users in their exploration of a spatial data cube. For that purpose, we suggested an approach for generating recommendations SOLAP queries in the context of the collaborative exploration of spatial data cubes. Adding to that, we developed a prototype *RecoSOLAP* system integrating and validating our approach. Future work consists on going further in the recommendations. In fact, the work presented in this paper is based on the exploration of a spatial data cube of a spatial data warehouse. We would like to improve our proposed system by recommending queries based on data exploration from a trajectory data warehouse which gathers data from pervasive systems involving mobility data.

## References

[1] C. Franklin, "An introduction to geographic information systems: linking maps to databases," *Database*, pp. 13-21, April 1992.
[2] N. Stefanovic, J. Han, and K. Koperski, "Object-based selective materialization for efficient implementation of spatial data cubes," *IEEE Transactions on knowledge and Data Engineering and Data Engineering*, vol. 12, no. 6, pp. 938-958, 2000.
[3] T. Badard, "L'Open Source au service du géospatial et de l'intelligence d'affaires," presented at the Conférence Midi-Innovation TI Organisée par l'ITIS et la Chambre de commerce et d'industrie de Québec, Université Laval, 29 avril 2011.
[4] T. Badard and E. Dubé, "Enabling geospatial business intelligence," Geomatics Sciences Department, Laval University, 2009.
[5] G. Marketos, "Data warehousing & mining techniques for moving object databases," Ph.D. dissertation, Department of Informatics, University of Piraeus, 2009.
[6] S. Rivest, Y. Bédard, M. J. Proulx, and M. Nadeau, "SOLAP: a new type of user interface to support spatio-temporal multidimensional data exploration and analysis," in *Proc. the ISPRS Joint Workshop on Spatial, Temporal and Multi Dimensional Data Modelling and Analysis*, Quebec, Canada, 2–3 October 2003.
[7] Y. Bédard, T. Merrett, and J. Han, *Geographic Data Mining and Knowledge Discovery*, 2nd ed., 2008, ch.3, pp. 45-68.
[8] Y. Bédard, M. J. Proulx, and S. Rivest, *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, London, UK: IRM Press, 2007, ch. 13, pp. 298–319.
[9] P. Y. Caron, "Étude du potentiel de OLAP pour supporter l'analyse spatio-temporelle," Mémoire de M.Sc., Département des sciences géomatiques, Faculté de foresterie et géomatique, Université Laval, pp. 132, 1998.
[10] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2005.
[11] E. Negre, "Exploration collaborative de cube de données," Ph.D. dissertation, Université François Rabelais of Tours, France, 2009.
[12] G. Sathe and S. Sarawagi, "Intelligent rollups in multidimensional OLAP data," in *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, 2001, pp. 531-540.
[13] S. Sarawagi, "Explaining differences in multidimensional aggregates," in *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, 1999, pp. 42-53.
[14] S. Sarawagi, "User-adaptive exploration of multidimensional data," in *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, 2000, pp. 307-316.
[15] F. H. del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach," *ScienceDirecte, Expert Systems with Applications*, vol. 35, no. 3, pp. 790-804, 2008.
[16] H. Jerbi, "Personnalisation d'analyses décisionnelles sur des données multidimensionnelles," Ph.D. dissertation, Institut de Recherche en Informatique de Toulouse – UMR 5505, France, 2012.
[17] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent, "A personalization framework for OLAP queries," in *Proc. the 8th ACM International Workshop on Data Warehousing and OLAP*, New York, NY: ACM, 2005, pp. 9-18.
[18] L. Bellatreche, A. Giacometti, P. Marcel, and H. Mouloudi, "Personalization of MDX Queries," *Journées Bases de Données Avancées* (*BDA*), 2006.
[19] C. Sapia, "On modeling and predicting query behavior in OLAP systems," in *Proc. DMDW*, 1999, pp. 2.1-2.10.
[20] C. Sapia, "PROMISE: predicting query behavior to enable predictive caching strategies for OLAP systems," in *DaWaK*, pp. 224-233, 2000.
[21] F. Bentayeb, R. Khemiri, and O. Boussaid, "Recommandation interavtive de requêtes décisionnelles," Actes des Ateliers d'EGC 2012, 2012.
[22] D. Jurafsky, "Minimum edit distance. stanford university," 2012.
[23] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 707-710, February, 1966.
[24] C. Brouard, "Comparaison du modèle vectoriel et de la pondération tf*idf associée avec une méthode de propagation d'activation," 2013.
[25] G. Gasso and P. Leray, *Clustering*, 2010.
[26] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604-632, 1999.

**Olfa Layouni** is a PhD student at the Higher Institute of Management of Tunis, received the master degree in business intelligence from the Higher institute of Management of Tunis in 2013 and has received the fundamental license of computing science of management in 2011 from the Higher Institute of Management of Tunis.

**Jalel Akaichi** received his PhD in computer science from the University of Sciences and Technologies of Lille in France and then his habilitation degree from the University of Tunis, Tunisia, where he is currently an associate professor in the Computer Science Department. Jalel Akaichi has published in international journals and conferences, and has served on the program committees of several international conferences and journals. He is currently the chair of the Master Science in Business Intelligence. Jalel Akaichi visited and taught in many institutions such as the State University of New York, Worcester Polytechnic Institute, INSA-Lyon, University of Blaise Pascal, University of Lille 1, etc.