Recommending Multidimensional Spatio-Temporal OLAP Queries

1st Olfa Layouni
BESTMOD Laboratory,
Université de Tunis,
Institut Supérieur de Gestion de Tunis,
Tunisia
layouni.olfa89@gmail.com

2nd Jalel Akaichi
College of Computer Science,
King Khalid University,
Abha, Saudi Arabia.
jalel.akaichi@kku.edu.sa

***Abstract*—Spatio-Temporal data warehouses store enormous amount of data. They are usually exploited by Spatio-Temporal OLAP systems to analyze and visualize data in order to extract relevant information. For extracting interesting information by exploiting spatio-temporal data warehouses, the current user launches Spatio-Temporal OLAP (ST-OLAP) queries to navigate within a geographic data cube (Geo-cube). Very often choosing which part of the Geo-cube to navigate further, and thus designing the forthcoming ST-OLAP query, is a difficult task. However, the problem of recommending ST-OLAP queries by exploiting a Geo-cube has not been studied so far. Therefore, we aim at filling this gap. In this paper, we propose an intensional and a collaborative approach for recommending Spatio-Temporal OLAP queries, expressed with the GeoMDX query language. It aims to help users in the process of exploiting spatio-temporal data warehouses by recommending ST-OLAP queries.**

***Index Terms*—Spatio-temporal data warehouse, ST-OLAP Query, GeoMDX, Intensional, Recommendation.**

I. Introduction

Spatio-Temporal data warehouse become an active research area. This is due to the explosive growth in the use of devices based on recent ubiquitous location technologies [1] such as GPS, smart phones, PDA, etc. The concept of a spatio-temporal data warehouse appeared in order to store moving data objects and temporal data information. Moving objects are geometries that change their position and shape continuously over time; the time could be an instant or a set of time intervals. In order to support spatio-temporal data, a data model and associated query language are needed for supporting moving objects.

Spatio-Temporal data warehouse stores large volumes of consolidation and historized multidimensional data, to be explored and analyzed by various users in order to make the best decision. To analyze and explore spatio-temporal data, a user needs a Spatio-Temporal OnLine Analytical Processing (ST-OLAP) system. To navigate in the spatio-temporal data cube (Geo-cube), the current user launches a sequence of ST-OLAP queries over a spatio-temporal data warehouse. Those queries are expressed with the GeoMDX [2] query language, in order to take into account spatial relationships: topological, direction, and metric distance relationships and temporal data.

Users interactively navigate a spatio-temporal data cube by launching sequences of ST-OLAP queries over a spatio-temporal data warehouse. The problem appeared when the current user may have no idea of what the forthcoming ST-OLAP queries should be and if it is relevant for him or not. As a solution and to help him in his navigation and his exploration, we propose an intensional and a collaborative recommendation system based on a query expression; expressed only with the GeoMDX query language. This system gives the possibility to recommend a ranking set of candidates for ST-OLAP queries.

This paper is organized as follows. Section 2 introduces backgrounds about the evolution from data warehouses to spatio-temporal data warehouses and the evolution from MDX to GeoMDX query languages. Section 3 introduces related works for recommending queries based on the query expression. Section 4 enlightens our approach for recommending ST-OLAP queries. Section 5 presents the set of experiments conducted to test the efficiency and the effectiveness of our proposal. We conclude and discuss future works in Section 6.

II. Related Works: Intensional Query Recommendations approaches

We presented the principals works proposed for recommending queries based on the exploration of a data cube and which are established by using the intension based approaches; based on the query expression. Indeed, this presentation allows as identifying the gaps of ST-OLAP queries recommendation system; in order to help the user in his exploration of Geo-cube. To the best of our knowledge, this is the first work dealing with the problem of recommending ST-OLAP queries (especially GeoMDX queries [2]). We present various approaches to recommend OLAP and Spatial OLAP (SOLAP) queries, expressed with MDX and Spatial MDX [3] languages.

The first proposed approach for recommending multidimensional OLAP queries expressed with MDX was proposed by [4]. The proposed approach exploits the log file which contains all the previous OLAP queries lunched in the data cube. It's a collaborative approach. The proposed approach computes a set of candidates' sessions and/or queries. Then, it recommends an order set of MDX queries for the current user. Adding to that, in the literature, we find that the proposed approach by [5], [6] exploits the profile. The proposed approach is based on the profile of the current user. This approach recommends a set of MDX queries after comparing between queries by

using a graphic model. This approach is a content based approach. Besides, the work proposed by [7] recommends OLAP sessions. It's a collaborative filtering context, and based on similarity measures between queries and sessions, expressed with MDX. Three phases compose this approach. The first phase aligns the log sessions with the current session, based on a modified version of Sequence Alignment. The second phase ranks each future by identifying densest areas of similar queries in the log sessions. The last phase adapts the future ranked first by modifying or adding fragments in its queries, using patterns extracted from the log and the current session, and recommends it. While recommendation has been widely explored in the context of OLAP systems, to the best of our knowledge, there is only the works proposed by [8], [9] developed recommendation approaches in the field of Spatial OLAP systems. Those approaches recommend SOLAP queries, expressed only with the Spatial MDX language, for supporting spatial data warehouse exploration. The proposed approach by [9] recommends a set of SOLAP queries but this approach doesn't take into account the specific characteristics of spatial data: topological, metric distance and direction relationships. The proposed approach is based on the query expression. To recommend a set of queries, it computes the distances between SOLAP queries by using the distance of Levenshtein [10] and the method of TF-IDF [11] in order to evaluate the importance of terms. However, the work proposed by [8] recommends only fives queries for the user after comparing between queries by applying a spatio-semantic similarity measure. This approach takes into account the specific characteristics of spatial data. However, the proposed approach for recommending Spatial OLAP queries has some disadvantages, the proposed algorithm eliminates all the old queries in the log and it takes into account only recent queries in the log.Table I reports a comparison of the above approaches.

III. Intensional ST-OLAP Queries Recommendation Approach

To help the user to go forward in his exploration of the geographic data cube, we propose an intensional and a collaborative approach for recommending a set of candidates of ST-OLAP queries; expressed especially with the GeoMDX query language. It uses both the sequences of ST-OLAP queries of the current session which are formerly launched on the Geo-cube and the historical of lunched sessions of ST-OLAP queries stored in the log. We introduce in this section the *ST-OLAPQReco system*, and more specifically the different phases to recommend a set of ST-OLAP queries for the current user. Figure 1 depicts the principle of *ST-OLAPQReco system*. A user conducts a ST-OLAP session for which *ST-OLAPQReco* will recommend an order set of candidates of ST-OLAP queries leveraging former sessions devised by previous users, saved in the log.

The *ST-OLAPQReco system* is organized in two phases:

1) A Pre-Processing phase allows partitioning the query log in order to compute all the generalized sessions of the log.
2) A Filtering phase allows predicting an ordering set of candidates ST-OLAP queries.

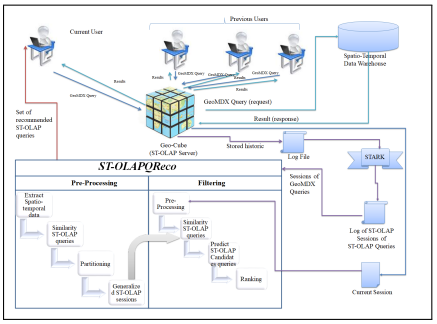

Fig. 1. Principal of *ST-OLAPQReco*.

The overall sequencing of the system are Pre-Processing and Filtering functions. The first step in this algorithm is to pre-process sessions of ST-OLAP 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 ST-OLAP queries. The results obtained in this step can be a set of candidates ST-OLAP 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 ST-OLAP queries, we sort this set in the order of the most similar to the query that represents the current session.

A. Pre-Processing Phase

Several users explore the geo-cube by launching sequences of ST-OLAP queries according to their intentions. These ST-OLAP queries are stored in the log file. However, the log contains all the previous ST-OLAP queries already launched in the geo-cube. It can be very large and voluminous because of the high number of queries and users. Therefore, we propose this phase in order to resolve this problem also the problem of the low density of the log. The main of this phase consists in partitioning the log of sessions of ST-OLAP queries $Geo\_session(ST\_C)$. In order to analyze the log file that contains a huge volumes of information such as user identification, number of queries, queries launched in each session, time of begin and end of each query execution time, set of queries launched by each user, errors, etc. We choose to user STARK framework [1] for analyzing and pre-processing the log file. STARK is an enhancement of SPARK [2] to support spatio-temporal data types. So, it takes into consideration data for record location or movement of user or object that periodically announce their current position. STARK framework includes spatial partitions, different mode for indexing as well as filter

[1]https://github.com/dbis-ilm/stark
[2]https://spark.apache.org/

TABLE I
COMPARING THE INTENSIONAL QUERY RECOMMENDATIONS APPROACHES BASED ON MDX AND SPATIAL MDX QUERY LANGUAGES.

Proposed methodProposed by		[4]	[5], [6]	[7]	[9]	[8]
Data Warehouse		*	*	*		
Spatial Data Warehouse					*	*
Data Cube	OLAP	*	*	*		
	SOLAP				*	*
Query Language	MDX	*	*	*		
	Spatial MDX				*	*
Input of the recommending approach	Log File	*		*	*	*
	Profile		*			
Results of the recommending approach	Query	*			*	
	Set of queries	*	*		*	*
	Set of sessions			*		
Content-based method			*			
Collaborative method		*		*	*	*

and join; k-nearest neighbor search and density clustering operators for data analysis [12]–[14]. By using STARK our log file will be more comprehensible for using in the next step in the pre-processing phase by accelerate the processing of data queries. He processes the log file and keeps track on session accessed by users after launching a set of queries in ST-OLAP server. The mainly step in STARK for our approach is data cleaning the entries that have status of 'error' or 'failed' have been removed. It consists of a minimization of the volumes of the log file by removing redundancy queries and sessions. In order to obtain a generalized log, we need especially the output of the log file after using STARK contains all the previous sessions of ST-OLAP queries and the schema of the spatio-temporal data warehouse. The first step is to extract the set of dimensions $STDim$ and the set measure SM_cube from the schema of spatio-temporal data warehouse by using the of TF-IDF (Term Frequency-Inverse Document Frequency) for evaluating the importance of terms [11] like measures SM_cube, classical dimensions Dim, spatial dimensions $SDim$ and temporal dimensions $TDim$. In this step, the Term Frequency-Inverse Document Frequency (TF-IDF) assigns a score to each term in the schema depending on not only the term's frequency in the document but also the term's value according to its appearance in the rest of the schema. After that, we need to extract spatial calculated measure in each query SMq by using the TF-IDF method but by replacing the schema by a query, in this case we calculate the score of each term in a query. So after that, we obtain a set of dimensions $STDim = SDim + TDim + Dim$ and a set of measures $SM = SM\_cube + SMq_{ia} + SMq_{ib}$. In order to determine the partitioning, we need to group similar queries in a class which represents a set of similar ST-OLAP queries. So, we propose to use the similarity measure distance proposed in the [15] for computing the distances between ST-OLAP queries (GeoMDX queries). This similarity measure of Spatio-Temporal OLAP queries takes into account not only the spatial data with specific characteristics such as topological, orientation and distance, but also the temporal data. In order to compare similarity of ST-OLAP queries, the authors proposed three new similarity measures: spatial similarity measure to compute spatial distance between ST-OLAP queries, temporal similarity measure to compute temporal distance between ST-OLAP queries, and spatio-temporal similarity measure to compute spatio-temporal aspects of the ST-OLAP queries. Furthermore, the spatio-temporal similarity measure is a combination of three components: one related to measure sets, one to the set selection and one to where set. The next step is the segmentation of ST-OLAP queries in order to obtained generalized log, we choose the implementation of the k-means algorithm for its simplicity and low computational cost compared with other clustering algorithms [16]–[18]. Besides, it's the most used in recommendation systems. In this step, the user should choose the numbers of initials clusters k, and for storing the distance results, we proposed to use the symmetric matrix $Matrix[a, b]$ that contains the results obtained of the similarity measure distance between queries. The result obtained, after this step, represents that each query should be belongs to a cluster of set of ST-OLAP queries. After that, we compute the set of the generalized sessions for obtained the generalized log. For each session in the log, we replace each ST-OLAP query in a session with the cluster which it belongs to.
The Algorithm 1 sketches the overall sequencing of this phase.

B. Filtering Phase

The filtering phase consists on recommending to the current user a ranking set of candidates ST-OLAP queries. This phase use the results obtained in the previous phase. The previous phase has compute the set of generalized sessions of ST-OLAP queries. By using this set and the current session of ST-OLAP

Algorithm 1 Pre-Processing Algorithm

Require: ℓ: The log of sessions of ST-OLAP queries.
$Schema.xml$: The geo-cube schema.
Ensure: Generalized set of sessions of ST-OLAP queries.
1: $Dim = Extract\_DimGeoCube(Schema.xml)$
2: $SDim = Extract\_SpatialDimGeoCube(Schema.xml)$
3: $TDim = Extract\_TemporalDimGeoCube(Schema.xml)$
4: $STDim = SDim + TDim + Dim$;
5: $SM\_cube = Extract\_Measures(Schema.xml)$
6: **For** All $q_{ia} \in Geo\_q\,(S_i) \in \ell$ **DO**
7: $SMq_{ia} = Extract\_SCMeasures(q_{ia})$
8: **For** All $q_{ib} \in Geo\_q\,(S_i) \in \ell$ **DO**
9: $SMq_{ib} = Extract\_SCMeasures(q_{ib})$
10: $SM = SM\_cube + SMq_{ia} + SMq_{ib}$
11: **If** $a = b$ **then**
12: $Matrix[a, b] = 0$
13: **else**
14: $Distance = STDQueries(q_{ia}, q_{ib}, STDim, SM)$
15: $Matrix[a, b] = Distance$
16: $Matrix[a, b] = Distance$
17: **End If**
18: **End For**
19: **End For**
20: **For** All $q_{ia} \in Geo\_q\,(S_i) \in \ell$ **DO**
21: **For** All $q_{ib} \in Geo\_q\,(S_i) \in \ell$ **DO**
22: $K\_Means(Matrix[a, b])$
23: **End For**
24: **End For**
25: **For** All $q_{ij} \in Geo\_q\,(S_i) \in \ell$ **DO**
26: $Generalized\ell = PreProcess\_Log(q_{ij}, Generalized\_Session(q_{ij}))$
27: **End For**

queries launched by the current user in the geo-cube, a set of candidates of ST-OLAP queries is computed by applying the algorithm 2. The Algorithm 2 sketches the overall sequencing of this phase. First, we propose to do the generalized current session. Then, the algorithm uses three functions: *Similarity*, *Prediction* and *Ranking*. The *Similarity* function is used to find a set of candidates of generalized sessions corresponding for each cluster in the generalized current session. The *Prediction* function is used to find a set of candidates of ST-OLAP queries that are relevant for the current user and which will be recommended to him. The *Ranking* function aims to order the set of candidates of ST-OLAP queries according to the preference of the current user.

Algorithm 2 Filtering Algorithm

Require: $Generalized\ell$: The generalized log.
$Geo\_q\,(S_c)$: The current session.
Ensure: A set of candidates ST-OLAP queries.
1: $PS_c = Pre - Processing(Geo\_q\,(S_c))$
2: $Candidates\_Sessions = Similarity(PS_c, Generalized\_Log)$
3: $Set\_Candidates\_ST - OLAP\_Queries = \emptyset$
4: **If** $Candidates\_Sessions = \emptyset$ **then**
5: **For** *each sessions* $S_i \in Candidates\_Sessions$ **DO**
6: $Set\_Candidates\_ST - OLAP\_Queries = Set\_Candidates\_ST - OLAP\_Queries \cup Prediction(S_i)$;
7: **End For**
8: $Ranking(Set\_Candidates\_ST - OLAP\_Queries)$
9: **End If**
10: $return\ (Set\_Candidates\_ST - OLAP\_Queries)$;

1) Similarity Function: In the similarity function, we propose to use the generalized current session and the generalized sessions of ST-OLAP queries stored in the log file. The Algorithm 3 describes all the instructions used in this function. For each cluster in the generalized current session, we propose to compute the number of occurrence for the corresponding cluster in each generalized session. The result obtained will be stored in a matrix with the number of clusters in the current session with the number of generalized sessions as parameters; and a cell in the matrix contains the number of occurrences. According to the obtained matrix, in each row of cluster, we propose to search for the generalized session that have the biggest number of occurrence of the corresponding cluster and to affect it to the set of candidates of generalized sessions.

Algorithm 3 Similarity_Generalized_Session Algorithm

Require: $Generalized\ell$: The generalized log.
PS_c: The generalized current session.
Ensure: A set of candidates of generalized session corresponding for clusters in the generalized current session.
1: **For** All $C[i] \in PS_c$ **DO**
2: $nb\_similar\_Cluster = 0$
3: **For** All $Generalized\_session[k] \in Generalized\ell$ **DO**
4: **If** $C[i]$ equals to $C[j] \in Generalized\_session[k]$ **then**
5: $nb\_similar\_Cluster++$
6: **End If**
7: **End For**
8: $Matrix[C[i], Generalized\_session[k]] = nb\_similar\_Cluster$
9: **End For**
10: **For** All $C[i] \in PS_c$ **DO**
11: k=1
12: **For** All $Generalized\_session[x = k + 1] \in Generalized\ell$ **DO**
13: **If** $Matrix[C[i], Generalized\_session[k]] = Matrix[C[i], Generalized\_session[x]]$ **then**
14: $Reco\_Generalized\_Session = \{Generalized\_session[x], C[i]\}$
15: **End If**
16: **End For**
17: **End For**

2) Prediction Function: The aim of the prediction function is to obtain a set of candidates ST-OLAP queries. The Algorithm 4 shows all the instructions used in this function.The algorithm use the set of candidates generalized sessions obtained by the *Similarity* function and the generalized current session. For each cluster in the generalized current session and for each candidates generalized sessions, we propose to replace each cluster with the corresponding query that is in the initial session in the log.

Algorithm 4 Prediction Algorithm

Require: $Reco\_Generalized\_Session$: A set of candidates of generalized session corresponding for each clusters in the generalized current session.
PS_c: The generalized current session.
Ensure: A set of candidates of ST-OLAP queries.
1: **For** All $C[i] \in PS_c$ **DO**
2: **For** All $Reco\_Generalized\_Session$ **DO**
3: $Reco\_Set\_Querie\_[C[i]\{q_{aj}\}] = Reco\_Generalized\_Session[C[i]\{q_{aj}\}]$
4: **End For**
5: **End For**

3) Ranking Phase: The obtained results represent a set of candidates ST-OLAP queries. The aim of this step is to order this set according to the queries in the current session. The Algorithm 5 describes all the instructions used in this step. First, we propose to compute the similarity measure between ST-OLAP queries in the current session with the set of candidates ST-OLAP queries. Then, we propose to rank the set of candidates ST-OLAP queries according to the similarity measure results by applying the Quick Sort methods because it has the fast average run time and have the best complicity by applying it in our case, it is equals to $O(nLogn)$ [19] with n represents the number of candidates ST-OLAP queries.

Algorithm 5 Ranking Algorithm

Require: $STDim$: Spatio-Temporal dimensions.
SM: Measures from the Geo-cube.
$Reco\_Set\_Queries\_C[i]$: A set of candidates of ST-OLAP queries.
$Geo\_q\,(S_c)$: The current session.
Ensure: A ranking set of candidates of ST-OLAP queries.
1: **For** All $q_{cj} \in Geo\_q\,(S_c)$ **DO**
2: $k = 1$
3: **For** All $q_{ij} \in Reco\_Set\_Queries\_C[i]$ **DO**
4: $Tab[k] = Spatio\_Temporal\_Distance\_Queries(q_{cj}, q_{ij}, STDim, SM)$
5: $k++$
6: **End For**
7: **End For**
8: $quicksort(Tab, 0, nb\_Candidates\_Queries - 1)$
9: **For** All $i = 1$ to $Size\_Table(Tab)$ **DO**
10: $Reco\_ST - OLAP\_Queries \longleftarrow Tab[i]$
11: **End For**

IV. Experimentations

In this section, we present the results of experiments that we have conducted in order to assess the capabilities of our proposal. First, we introduce the prototype *ST-OLAPQReco system* that implements the recommendation approach. Second, we present the set of experiments and their analysis.

A. Architecture of ST-OLAPQReco System

We present the architecture and the dataset used to evaluate and compare the prediction performance of the proposed recommender systems. The *ST-OLAPQReco system* implements all phases of the proposed approach to provide user with useful and relevant GeoMDX queries. First, to navigate in the geo-cube, the current user launch a sequence of ST-OLAP queries by using the ST-OLAP server GeoMondrian over a spatio-temporal data warehouse stored in the PostgreSQL integrating PostGIS in order to take into account spatial and temporal data types. All the previous sessions of GeoMDX queries are stored in the log file. Finally, our system recommends an order set of GeoMDX queries to the current user. The architecture of the system is presented in the figure 2.

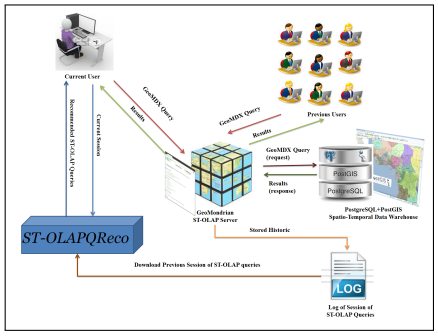

Fig. 2. The architecture of the *ST-OLAPQReco system*.

B. Data Set: NYC Taxi Trip Data

To realize our system, we need a log file contains the previous sessions of ST-OLAP queries launched by users in the ST-OLAP server. First, we need to connect between the GeoMondrian Server and the spatio-temporal data warehouse "NYC Taxi Trip Data". We use a real data set for taxi trips in New York city records from a freely available dataset. The official TLC trip [3] record data set contains data for over 1.1 billion taxi trips from January 2009 through June 2015, covering both yellow and green taxis. Each individual trip record contains precise location coordinates for where the trip started and ended (latitude and longitudes), timestamps for when the trip started and ended, plus a few other variables for detailing the trip such as fare amount, payment method, distance traveled, number of passengers and various taxes. For privacy and security reasons, it doesn't contain details about drivers or passengers.

We used PostgreSQL to store the data and PostGIS to perform geographic calculations, including the heavy lifting of mapping latitude/longitude coordinates to NYC taxi trip data. For more detailed information on the database schema and geographic calculations, we used the steps described at the GitHub repository [4].

C. Performance analysis

Our experiment evaluates the efficiency of our approach proposed to recommend ST-OLAP queries. The efficiency of our system assesses the time taken to generate the best recommendation for various log sizes. It's important because ST-OLAP queries should be resolved in a short time to enable an interactive analysis.

The performance is presented in Figure 3 according to various log sizes. These log sizes are obtained by playing with two different parameters that change over the time: the number of sessions store in the log, it ranges from 10 to 100; and the maximum number of queries that could be launched for per session, it ranges from 3 to 20. We thus obtain logs of size varying between 3 and 2000 queries. Note that what is measured is the execution time taken by the steps proposed: pre-processing the log, generating candidates ST-OLAP queries and ordering the candidates ST-OLAP queries. The goal of our experimentation is to measure the execution time taken by applying our proposed approach.

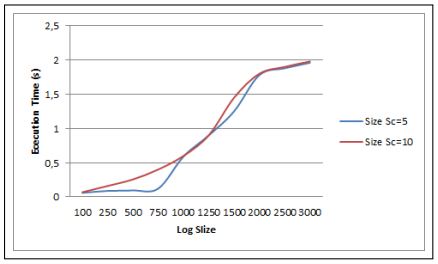

Fig. 3. Performance analysis: Average computation time for obtaining a recommendation.

Figure 3 shows the evolution of the average computation time to obtain recommendation with *ST-OLAPQReco system*. The goal of our experimentation is to measure the execution time taken by applying our proposed approach. So, we conclude that the time taken to recommend ST-OLAP

[3] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml
[4] https://github.com/toddwschneider/nyc-taxi-data

queries increases linearly with the log size but remains highly acceptable and is slightly influenced by the current session size as shown in Figure 3. As can be seen from Figure 3, it is obvious that the trend of execution time is upwards with the log size. The time taking for computing a recommendation is less than 1s for 25% of the log size and more than 3s for the complete log size.

As there is no approach for ST-OLAP queries recommendation system, we compare the performance of our system with related works in Spatial OLAP (SOLAP) queries recommendation systems. We notice that SOLAP systems may appear more performed than our ST-OLAP recommendation system. We should notice here that for ST-OLAP queries recommendation time is expected to be longer than one for SOLAP queries recommendation system. Since, our system handles not only spatial data with specific relationships: topological, direction and metric distance relationship; but also temporal data: movement data types.

V. Conclusions

In this paper, we have proposed a collaborative and intensional approach for recommending Spatio-Temporal OLAP queries with queries expressed only with GeoMDX query language; in order to help the current user in his exploration of the spatio-temporal data cube (Geo-cube). For that purpose, we have suggested an approach for generating recommendations of ST-OLAP queries to the current user. Our approach was organized in three different phases: A Pre-Processing the set of session in the log, Generalization of the candidate ST-OLAP queries and Ordering the candidates ST-OLAP queries. Then, we have developed our system ST-OLAPQReco by implementing all the phases of our proposed approach in order to recommend relevant ST-OLAP queries to the user. Adding to that, to validate our approach, we evaluated the efficiency and the effectiveness of our approach proposed to recommend ST-OLAP queries. To the best of our knowledge, our proposal is the first work proposing a Spatio-Temporal OLAP queries (GeoMDX queries) recommendation system.

References

[1] A. Vaisman and E. Zimányi, *Data Warehouse Systems: Design and Implementation*. Heidelberg: Springer, 2014.

[2] M. Tranchant, "Capacités des outils solap en termes de requêtes spatiales, temporelles et spatio-temporelles," CONSERVATOIRE NATIONAL DES ARTS ET METIERS CENTRE REGIONAL RHÔNE-ALPES CENTRE D'ENSEIGNEMENT DE GRENOBLE, Tech. Rep., 2011.

[3] T. Badard, "L'open source au service du géospatial et de l'intelligence d'affaires," *Geomatics Sciences Department*, avril 2011.

[4] E. Negre, "Exploration collaborative de cubes de données," Ph.D. dissertation, Université François Rabelais of Tours, France, 2009.

[5] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent, "A personalization framework for OLAP queries," in *DOLAP 2005, ACM 8th International Workshop on Data Warehousing and OLAP, Bremen, Germany, November 4-5, 2005, Proceedings*, 2005, pp. 9–18.

[6] L. Bellatreche, H. Mouloudi, A. Giacometti, and P. Marcel, "Personalization of MDX queries," in *22èmes Journées Bases de Données Avancées, BDA 2006, Lille, 17-20 octobre 2006, Actes (Informal Proceedings).*, 2006.

[7] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia, "Similarity measures for olap sessions," *Knowledge and Information Systems*, vol. 39, no. 2, pp. 463–489, 2014.

[8] S. Aissi, M. S. Gouider, T. Sboui, and L. B. Said, "A spatial data warehouse recommendation approach: conceptual framework and experimental evaluation," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, pp. 1–18, 2015.

[9] O. Layouni and J. Akaichi, "A novel approach for a collaborative exploration of a spatial data cube," *IJCCE: International Journal of Computer and Communication Engineering*, vol. 3, no. 1, pp. 63–68, Jan 2014.

[10] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[11] C. Brouard, "Comparaison du modèle vectoriel et de la pondération tf*idf associée avec une méthode de propagation d'activation," in *CORIA*, Neuchâtel, France, Apr. 2013, pp. 1–10.

[12] S. Hagedorn, P. Gotze, and K.-U. Sattler, "The stark framework for spatio-temporal data analytics on spark," *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, 2017.

[13] S. Hagedorn, P. Götze, and K.-U. Sattler, "Big spatial data processing frameworks: Feature and performance evaluation." in *EDBT*, 2017, pp. 490–493.

[14] S. Hagedorn and T. Räth, "Efficient spatio-temporal event processing with stark." in *EDBT*, 2017, pp. 570–573.

[15] O. Layouni and J. Akaichi, "New similarity measure for spatio-temporal OLAP queries," in *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery - 12th International Conference, BDAS 2016, Ustroń, Poland, May 31 - June 3, 2016*, 2016, pp. 328–337.

[16] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," 1999.

[18] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[19] R. Sedgewick, "Implementing quicksort programs," *Commun. ACM*, vol. 21, no. 10, pp. 847–857, Oct. 1978.