FIMIOQR: Frequent Itemsets Mining for Interactive OLAP Query Recommendation

Rym Khemiri, Fadila Bentayeb
ERIC Laboratory, University of Lyon, Lumière Lyon2
5 av. P. Mendès-France 69676 Bron Cedex Lyon, France
{Rym.Khemiri, Fadila.Bentayeb}@univ-lyon2.fr

***Abstract*—We propose in this paper an interactive query recommendation system, namely FIMIOQR. It is designed to help OLAP (On-line Analytical Processing) users in their decision query writing task based on both a set of selected measures and decision queries log file. Our FIMIOQR system is designed to discover associations from decision queries log file. For this end, we use association rules method to extract frequent itemsets from dimensions attributes according to user selected set of measures. This allows end users in OLAP systems to write relevant queries guided by an interactive recommending system and helps them to meet their analysis objectives. In addition, we propose a tool for the automatic implementation of FIMIOQR which provides a visual interface to OLAP users which helps them to write their queries step by step in an interactive way. We also carried out some experimental tests to evaluate our system. The experimental evaluation proves our FIMIOQR framework is efficient in term of recommendation quality.**

***Index Terms*—Interactive Recommendation; Data warehouse; Decision Query; Measure; Dimension attribute; Frequent Itemsets Mining; OLAP; Data warehouse**

I. Introduction

End users in OLAP systems tend to achieve the same goal for obtaining valuable and useful information out of data warehouse. However, OLAP is characterized by decision queries that are often very complex and involve a lot of aggregations. Due to the constantly and rapidly growth of data volumes, manipulation and analysis complexity also increases. In such situation, OLAP users would quite benefit from assistance for this task. This assistance may be accomplished through recommendation process.

Recommendation is a means of meeting user's needs more efficiently, making interactions faster and easier and, consequently, increasing user satisfaction. This assumption constitutes the starting point used in this paper to present a new recommendation approach over data warehouses.

In OLAP context, query recommendation approaches mostly focused on recommending alternative queries with close search intent to the original query. In this case, recommended queries are existing queries. However, the recommendation of only alternative existing queries may generate less interactivity with the user and does not create new decision queries.

Furthermore, there is a lack of a framework that assists the users while querying data warehouse. We believe that such a framework would be interactive during query writing and consequently increases user implication in the exploration task. It allows him/her to construct his/her decision query step by step (incrementally).

Query logs usually contain a sequence of SQL queries that show the action flows of users, their preference, their interests, and their behaviours during the action. In this paper, we aim to assist the user in formulating accurate queries to express his/her analysis needs. We propose an interactive real-time assistance process which aims to give users opportunities to refine their queries by suggesting queries completions.

Decision queries on which we are interested in this paper are in the form "SELECT ... FROM ... WHERE ... GROUP BY CUBE (ROLLUP)". Assuming that the relevance of a recommended query is strongly correlated to the usage frequency of the corresponding attributes within a given workload, the search for frequent itemsets [1] appeared well adapted to highlight this correlation and to facilitate query recommendation. Our tool parses the transaction log file (set of queries executed by the DBMS) and groups queries respecting the same measure(s) to build a context for mining frequent itemsets. This context connects queries from the input workload to the query attributes. The output frequent itemsets are sets of attributes forming a configuration of candidate recommendations. Finally, recommendation strategy can be applied to select the relevant attributes to effectively suggest from within this configuration.

Besides attributes (qualitative data) in a decision query, there is at least one numeric measure (quantitative data) that provide the object of analysis. Thereby, after asking user for his/her object of analysis (measure or combination of measures), recommendations during query writing will focus on query attributes. Queries from the transaction log constitute a workload that is treated by an SQL query analyzer. Thus, the SQL query analyzer extracts all the attributes. Then, we build a "query-attributes" matrix, the rows of which represent the workload queries, and the columns represent attributes. The role of this matrix is to link each attribute to the workload queries it appears in. This matrix represents the extraction context for frequent itemsets. To compute these frequent itemsets, we applied the Close algorithm [2].

Our approach goal is to assist OLAP user to accomplish his/her decision query writing by suggesting him/her possible candidate query attributes with respect to both their appearance in different *Clauses* (SELECT, WHERE, GROUP BY, etc.) and the selected measures. Consequently, the user can construct progressively his/her decision query by selecting attributes among different proposed suggestions by our approach. Thus, the new obtained query can be submitted by the user to the data warehouse. In order to validate our recommendation approach, we have implemented the recommendation framework called FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommendation) with Java using the integrated development environment Netbeans [3]. The experimental evaluation proves our framework is efficient in term of recommendation quality.

The remainder of this paper is structured as follows. Section II presents related works regarding query recommendation. In Section III, we present our query recommendation approach. In Section IV, we present an example to motivate our approach. Implementation features are described in Section V with some preliminary experimental results. Finally, conclusions are given in Section VI, together with a summary of our expected future work.

II. Related Works

Recently, recommendation systems have obtained the attention of research society in both database and data warehouse fields. In fact, items to recommend are queries. We find in literature two kinds of recommendations (1) alternative entire queries proposition and (2) assistance to query construction through query completions.

In the database field, based on results of each query, Stefanidis et al. recommend additional results called "You May Also Like" or YMAL results which might be of user's potential interests [4]. YMAL appraoch exploits the history of previously submitted queries to the database system, e.g. by using query logs. Recommendations that are generated can be either query-based YMAL results (similar to content-based recommendations), either user-based YMAL results (similar to collaborative recommendations).

Another approach for recommending queries in the database field creates automatically join query recommendations based on input and output specifications [5]. This approach analyses query log and extracts joins from previous submitted queries that are used to generate recommendations for the current user. After presenting the conceptual framework and its instantiation in [6], Chatzopoulou et al. came up with a system named *QueRIE* for personalized query recommendations (full SQL queries). To generate recommendations, *QueRIE* first constructs the summary for each user, then generates a "predicted" summary for the users and finally generates recommendation queries based on "predicted" summary. Finally, based on the analysis of query log, Khoussainova et al. introduced the *Snipsuggest* framework. *Snipsuggest* assists users in composing complex queries by recommending a set of additions to a specific clause in a partially stated SQL query [7].

In OLAP context, the first approach to be considered is based on providing query recommendations to the user by means of User Preference Analysis (RUPA) [8], [9]. In RUPA approach, OLAP analyses are represented using a graph-based model. In fact, both of user profile and current query are expressed by means of trees. Then, a Tree Matching Algorithm is applied to compare the two trees.

More recently, Golfarelli et al. propose an approach where preferences are used to annotate the query. They defined an algebra that allows formulating preferences on attributes, measures and hierarchies [10]. The used technique consists in personalizing a query by dealing with a sub-query of the current query. In this case, the recommended query is the sub-query which returns a non empty preferred result.

In this section, we reviewed the current approaches for query recommendation for databases and data warehouses. It seems that neither solution by now assist decision query construction. We have to come up with better, more meaningful solution to recommend query completions to the user instead of whole queries. We present a comparative table (table I) confronting the panoply of the proposed approaches. We define some criteria that we consider relevant to study exactly the recommendation approach.
Recommendation type: the recommendation system may be content-based or collaborative. Within the content-based context, it consists in considering the user individually with respect to his/her requirements. In the social or collaborative context, it consists in considering the context of other users who may have similar preoccupations.
Recommendation input data: this criterion presents recommendation source which can be a user profile, a query history (log file) or an external source(ontologies, web pages...).
Recommendation Time: this criterion presents time of recommendation: before querying, while querying or after querying.
Recommendation features: this criterion presents the recommendation object which can be a set of entire queries or a set of query fragments.
Research field: this criterion presents the domain of application of the recommendation approach. It can be database field (DB) or data warehouse field (DW).

To the best of our knowledge, only whole queries can be provided to users in OLAP recommendation context. However, we are more interested in the instant-response query recommendation. In fact, our work comes close to works that propose approaches of user query refinement in database field [7]. Even these works present some similarities with our approach (recommendation of query fragments), the challenges that need to be addressed and the techniques are very different to the ones we propose. In fact, we recommend decision queries based on a set of measures fixed by the user. Therefore, we use the Close algorithm based on minimal support which must be fixed also by the user.

III. Interactive query recommendation

It is important for query recommendations to identify the current user's purpose in order to make an accurate recommendation. However, previous queries may include too many features that could not provide the central themes of the analysis, while the current query has little information. To overcome this drawback, our framework introduces a new

TABLE I
SURVEY OF QUERY RECOMMENDATION APPROACHES

Research Works		Stefanidis et al. 2009	Chatzopoulou et al. 2009, 2011 Yang et al. 2009	Khoussainova et al. 2010	Jerbi et al. 2009	Golfarelli et al. 2011	Khemiri et al. 2012
Recommendation Type	Content-based	×			×		×
	Collaborative	×	×	×		×	
Recommendation Time (% querying)	Before				×		
	While			×			×
	After	×	×		×	×	
Recommendation Input Data	User profile				×	×	
	Log file	×	×	×		×	×
	External source	×					
Recommendation Features	Queries	×	×		×	×	
	Query fragments			×			×
Research Field	DB	×	×	×			
	DW				×	×	×

approach that asks the user for the analysis object (measure or combination of measures) and recommends attributes based on this analysis object. To extract the previous queries, the system can use the query log to summarize the querying behaviour from past users. Our recommendation framework (Figure 1) works on the OLAP system using decision queries. Every user's query is also recorded in the log with the *user ID* to store its queries. Our recommendation approach relies on current inputting query attributes and past user's queries which are stored in query log to generate a set of attributes recommendations to the user.

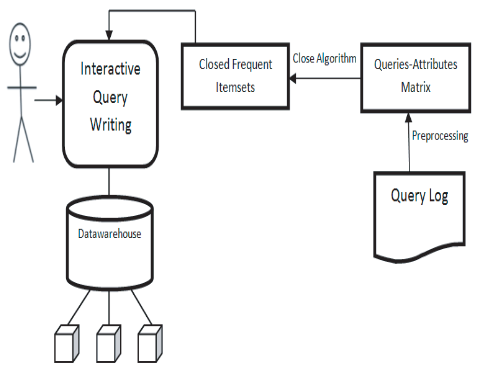

Fig. 1. Interactive query recommendation framework

A. Decision query

In data warehouse context, extracted queries are decision analysis oriented. Based on a star schema model, decision queries can be expressed on relational algebra as below:

$$q = \pi_{A,M}\sigma_P(F \bowtie D_1 \bowtie D_2 \bowtie ... \bowtie D_d)$$

where P is a conjunction of simple predicates on the attributes of dimension tables, A is a set of attributes of dimension tables D_i(attributes of Group by clause) and M is a set of measures, each measure is defined by applying an aggregation operator on the measure of fact table.

A major distinctive feature of data warehouse query is its aggregation of measures by one or more dimensions as one of the key operations; e.g., computing and ranking the total sales by each country (or by each year). Other popular operations include comparing two measures (e.g., sales and budget) aggregated by the same dimensions. Thus, the queries themselves return results computed over the measure attributes. Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. The dimensions together are assumed to uniquely determine the measure [11].

In our previous study [12], we have shown the potential of recommending OLAP queries through a data mining based-approach using Apriori method. Our first approach deals with the recommendation without taking into account the measures. It is why, in this paper, we focus on first the selected measures by the user after what we recommend to him/her a set of correlated attributes.

In this paper, a key assumption in our recommendation approach is that the measure attributes have an extreme importance in analytic queries. It represents the analysis object.

B. Query recommendation process

There are three steps in our approach. First, data preparation and pattern discovery phases. Second, extracting frequent itemsets by Close algorithm and then we focus on the details of our recommendation engine.

1) Preparing and Preprocessing data: The starting and critical point for successful recommendation based on usage data is data preprocessing. It is an offline component of our recommendation approach. It can be divided into three separate stages. The first stage is that of preprocessing and data preparation, it consists in queries extraction and attributes extraction. The second is the measures lattice construction and the final stage is the binary matrix construction. Each of these components is discussed below.

a) Query extraction: Each data warehouse system keeps logs whenever users input queries. The information displayed and generated depends on the attributes that a user adopts. Attribute information can be provided to help users consider business information from different angles. Therefore, we try to discover how users make use of attributes when querying a data warehouse. Such information can be provided to other users for reference to reduce their efforts when using business intelligence systems. Our recommendation mechanism provides such information based on existing query logs. In order to create the attribute-based query, we needed to preprocess the queries included in the query logs and decompose them. This process consists of two steps, namely query generalization and query parsing. In the first step, the queries are generalized based on a set of rules, in order to be analysed and matched more efficiently. Then, they are parsed in preparation for filling the binary matrix. In fact, a workload can be easily obtained from the DBMS transaction logs. The queries workload models decision-oriented queries involving common OLAP operations, such as cube, roll-up and drill down.

b) Measures lattice construction: We group queries using the same measure(s). We represent different possible combinations of these measures in the form of a lattice structure as shown in Figure 3. Each node in the measures lattice structure represents a combination of measures. The number of levels in a lattice is equal to the number of measures. The size of the lattice (i.e., the total number of nodes where each node represents a set of frequent itemsets) grows exponentially with the number of attributes that are considered. However, generally, in a data warehouse we have few measures. Let N be the number of nodes in a lattice, then $N = (2^A - 1)$ where A represents the number of measures.

c) Building the extraction context for the frequent closed itemsets: For every node (mesaure(s) workload), we build a matrix, the rows of which represent the workload queries, and the columns represent the set of all the attributes identified in the previous step. In each node, the "query-attributes" matrix links each query to the attributes within it. Attribute presence in a query is symbolized by 1, and absence by 0.

2) Frequent Closed Itemsets Mining: The Close algorithm scans in breadth first a lattice of closed itemsets in order to extract the frequent closed itemsets and their support. Its input is an extraction context.We selected the Close Algorithm because its output is the set of the frequent closed intemsets (closed regarding the Galois connection [2]), which is a generator for all the frequent itemsets and their support. In most cases, the number of frequent closed itemsets is much lower than the total number of frequent itemsets obtained by classical algorithms such as Apriori in our previous work [12]. In our context, using Close enables us to obtain a smaller (though still significant) configuration of candidate recommendations. For each query within the workload, we extract attributes in all the clauses prefixed by the name of the clause. Intuitivelty, a closed itemset is a maximal set of items (attributes) that are common to a set of transactions (queries).

A closed itemset is a maximal set of items (attributes) that are common to a set of transactions (queries). A maximal set of items is an independent set of items that is not a subset of any other independent set. In fact, an itemset is said frequent when its support is greater or equal to a threshold parameter named minsup (minimal support).

Eventually, the application of *Close* algorithm on the extraction context outputs the set of frequent itemsets (and their support) for a minimal support fixed by the user. We consider this set as our configuration of candidate recommendations.

3) Recommendation phase: The recommendation engine is the on line component of our recommendation system based on frequent itemsets mining. Our system restricts its processing to the measure combination selected by the user. Furthermore, when a user selects a node representing a measure combination, the underlying system can exploit the list of possible frequent itemsets that involve these measures to recommend appropriate attributes to the user. Then, the user may choose to further accept the proposed suggestions by selecting one or more of the frequent items that appear on the list. The role of the data mining process becomes, simply, to find the frequent itemsets that are of interest to the user and that satisfy a minimum support value if the user provides such minimums.

IV. Illustrative example

To illustrate our approach, we use as an example FoodMart data warehouse [13] which contains sales data of a supermarket chain that is specialized in food products. Sales are represented as a fact table namely *Sales* and the contexts of analysis are represented as dimension tables namely *Product*, *Promotion*, *Time*, *Store* and *Customer*.

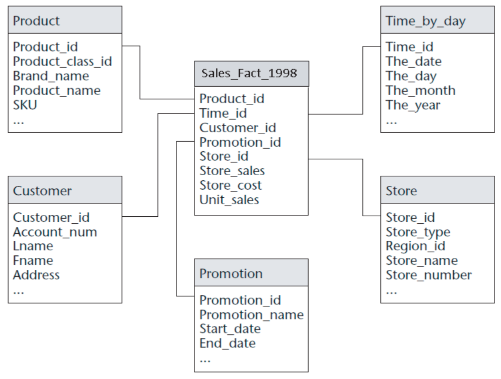

Fig. 2. Excerpt of FoodMart data warehouse

Based on this data warehouse and a related workload containing 100 queries, we show in this section how our FIMIOQR system works.

A. Preprocessing task

It is an offline task containing 3 steps.

a) Step 1: Lattice construction: On Foodmart data warehouse, we have 7 measures namely *Store_sales*, *Store_cost*, *Unit_sales*, *Amount*, *Warehouse_sales*, *Warehouse_cost* and *Store_invoice*. In this example, we are interested on sales cube (Figure 2, therefore, we use only three measures: *Store_sales*, *Store_cost* and *Unit_sales*. We represent different combinations of these measures in the above lattice in Figure 3.

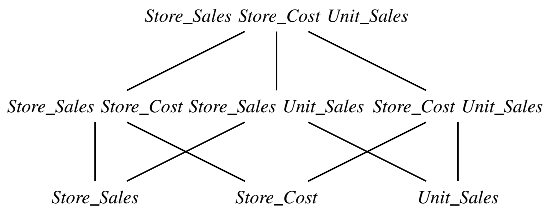

Fig. 3. Measures lattice

b) Step 2: Analysis context building: For every measure or a combination of measures in lattice nodes, we build the respective binary matrix, the rows of which represent the workload queries, and the columns represent the set of all the attributes identified in the previous step.

c) Step 3: Close algorithm application: The input of the Close algorithm consists in the binary matrix generated in step 2 and the output consists in closed frequent itemsets to each measure or a combination of measures. For instance, for the combination of measures *Store_sales*, *Store_cost* and *Unit_sales*, the output is: {*product_class_id, store_name, store_manager, product_name, product_id*}, {*customer_id, member_card, grocery_sqft, city, the_date*},...

B. Interactive query writing task

In this task, the user is assisted by our system. The basic principle that underlies our recommendation engine is the completions of user current query. In fact, after measures selection by the user, FIMIOQR will suggest to him/her dimension attributes while his/her query writing. Assume a user has selected *Store_sales*, *Store_cost* and *Unit_sales* measures. Therefore, our system will use the frequent itemsets of the corresponding lattice node. Assume a user enters the attribute *brand_name*, our system will search the closed frequent itemsets corresponding to the selected measures and containing this attribute. FIMIOQR will recommend to the user the other correlated attributes as possible completions. Our approach provides non-intrusive recommendations to the user for query composing. Thus, the user has always the choice to accept or not these suggestions. If the user accepts an attribute suggestion, FIMIOQR will search again closed frequent itemsets containing this attribute and so forth.

V. Implementation and Experimentation

In order to validate our approach, we implemented the recommendation framework called FIMIOQR (Frequent Itemsets Mining for Interactive OLAP Query Recommandation) using 'JAVA' on Netbeans environment on top of SQL Server 2005 DBMS [3]. We have applied it on a test workload of 100 queries related to FoodMart data warehouse used in our example.

To evaluate the performance of our system, several aspects of FIMIOQR can be used for answering these questions:

- Is FIMIOQR able to effectively recommend relevant attributes? (recommendation quality).
- Is the log size influence on recommendation time?
- Is the Close algorithm effective at maintaining recommendation quality, while changing the minimal support?

However, evaluating the quality of query recommendation is difficult, since there is usually no ground truth of recommendations and different annotators will have different judgements over the recommendation results. However, two metrics, precision and recall, are commonly used to measure the quality of a recommendation [14]. A recommendation system may suggest interesting or uninteresting objects. The recall measure indicates the effectiveness of a method for locating interesting objects, while the precision measure represents the extent to which the instances recommended by a method really are interesting to users. The formulas are as follows:

$$recall = \frac{\textit{number of correctly recommended objects}}{\textit{number of interesting objects}}$$

$$precision = \frac{\textit{number of correctly recommended objects}}{\textit{number of recommended objects}}$$

Whereas, in our study, the number of interesting objects is fixed since it is equal to the number of attributes used in a data warehouse. In our example in section IV, we have 95 attributes. Therefore, the range or scope of recommendations is limited; consequently, the recall metric is not applicable in our study. We can only use the precision metric to evaluate the quality of the proposed recommendations.

In our approach, the number of correctly recommended objects presents the number of accepted recommendations (attributes). The relevance of each attribute to the input query was judged by members of our laboratory (students and professors). They analysed the recommended itemsets and determined the items that are of interest to the input query. In fact, the recommendation precision (RP) is calculated as follows:

$$RP = \frac{\textit{number of accepted recommended attributes}}{\textit{number of recommended attributes}}$$

Our first experiment evaluates the accuracy of the proposed approach to make the recommendations. Figure 4 shows the performance recommendation time according to the size of query log.

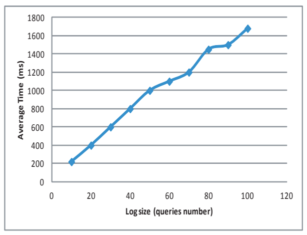

Fig. 4. Accuracy Analysis

As can be seen from Figure 4, it is obvious that the trend of execution time is upwards with the log size. The execution time is acceptable with the size of 50 queries in query log.

The Figure 5 (a) shows that the precision slightly increases with the log size. This figure demonstrates that most current queriess can obtain a successful recommendation by our approach with precision between 0.14 and 0.58 from the query log.

In addition, our recommendation framework has been executed for various values of the Close minsup (minimal support) parameter. In practice, this parameter helps us limiting the number of candidates to generate by selecting only those that are the most frequently used by the workload. Figure 5 (b) shows the precision of recommendation according to the value of minimal support. As can be seen from Figure 5 (a), it is obvious that the trend of recommendation precision is upwards with minsup parameter until 60% where it decreases.

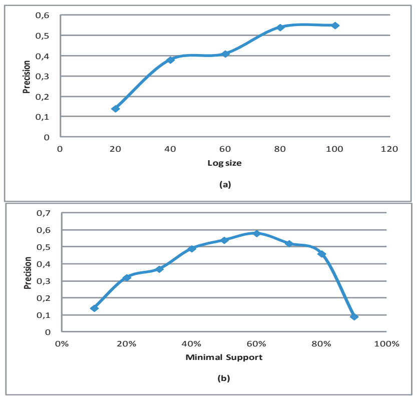

Fig. 5. (a) Recommendation precision for different values of minsup; (b) Average precision

VI. Conclusion and Future Works

In this paper, we have proposed a novel approach for interactive real time decision query recommendation. Our method allows to ease the user's burden of the query formulation based on decision query logs, namely FIMIOQR. The contribution of our paper consists in providing a tool to help users interact with data warehouse while formulating their queries. This tool is based on Close algorithm in order to extract frequent closed itemsets.

Unlike most previous works, the method we propose attempts to recommend dimension attributes rather than whole queries. Throughout the paper we tended to justify that mining log files can give us a concise set of information about the usage of the system and enables us efficient handling of that information. In the context of query recommendations, when a user submits a query, the system should be able to assist him/her in this task.

Our approach can be easily extended to collaborative recommendation by identifying user preferences in order to exploit query logs of all users and to recommend to the current user dimension attributes of similar users. Moreover, a data warehouse often contains vast amounts of information that is difficult for a new user to comprehend. What attributes should be used initially for a new user without any data warehouse experience? It is necessary to employ a recommendation mechanism to assist such users by suggesting him/her for example attributes based on the logs of other experienced users.

In terms of future work, there is much to be done. First and foremost, we intend on looking for most accepted recommendations by the user in order to rank them through skyline queries and to recommend them first to the user in his/her future sessions. Therefore, the recommendation engine matches past queries to an improved ranking, leading to recommendations of higher quality.

In addition, a data warehouse user follows a logical reasoning in a querying process. Data in an analysis session is often correlated. Therefore, we must think about generalizing our approach against user sessions.

Finally, a real data set should be used for the simulation. We assume that the query logs follow a normal distribution. However, the generated data has drawbacks that impact on the effectiveness of recommendations. Future work could be extended to make more solid recommendations by working with some organizations to obtain real data so that the analysis would be closer to real-world situations.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB*, 1994, pp. 487–499.
[2] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *ICDT*, 1999, pp. 398–416.
[3] http://eric.univ-lyon2.fr/~bentayeb/logiciels.html.
[4] K. Stefanidis, M. Drosou, and E. Pitoura, " "you may also like" results in relational databases," in *PersDB workshop*, 2009.
[5] X. Yang, C. M. Procopiuc, and D. Srivastava, "Recommending join queries via query log analysis," in *ICDE*, 2009, pp. 964–975.
[6] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query recommendations for interactive database exploration," in *SSDBM*, 2009, pp. 3–18.
[7] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "Snipsuggest: Context-aware autocompletion for sql," *PVLDB*, vol. 4, no. 1, pp. 22–33, 2010.
[8] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh, "Applying recommendation technology in olap systems," in *ICEIS*, 2009, pp. 220–233.
[9] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh, "Preference-based recommendations for olap analysis," in *DaWaK*, 2009, pp. 467–478.
[10] M. Golfarelli, S. Rizzi, and P. Biondi, "myolap: An approach to express and evaluate olap preferences," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 1050–1064, 2011.
[11] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *SIGMOD Record*, vol. 26, no. 1, pp. 65–74, 1997.
[12] R. Khemiri and F. Bentayeb, "Interactive query recommendation assistant," in *DEXA Workshops*, 2012, pp. 93–97.
[13] http://www.e-tservice.com/downloads.html.
[14] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.