A Framework for Summarizing a Log of OLAP Queries

Julien Aligon
Université François Rabelais Tours
Laboratoire d'Informatique
Email: julien.aligon@etu.univ-tours.fr

Patrick Marcel
Université François Rabelais Tours
Laboratoire d'Informatique
Email: patrick.marcel@univ-tours.fr

Elsa Negre
Université François Rabelais Tours
Laboratoire d'Informatique
Email: elsa.negre@univ-tours.fr

***Abstract*—Leveraging query logs benefits the users analyzing large data warehouses. But so far nothing exists to allow the user to have concise and usable representation of what is in the log. In this paper, we propose a framework for summarizing OLAP query logs. This framework is based on the idea that a query can summarize another query and that a log can summarize another log. It includes a simple language to declaratively specify a summary, a measure to assess the quality of a summary and an algorithm for automatically computing a good quality summary of a query log.**

I. Introduction

It is becoming accepted that leveraging query log would help the user analyzing large databases or data warehouses [1]. This is particularly relevant in a collaborative context for instance to issue recommendations [2], [3], [4], [5]. But to the best of our knowledge, even the simple problem of providing the user with a concise representation of what is inside a large log has rarely been addressed [1]. Using such a summary, that avoids overwhelming the user, would have many advantages, including:

- helping an administrator to manage and tune the OLAP server if the summary indicates the frequently accessed members,
- allowing a user to have a rough idea of the queries launched by other users,
- assisting the user to perform new analysis sessions, for instance by not redoing what has been done, or to write new queries, by looking for what has been done.

In this paper, we propose to summarize automatically an OLAP query log. In this context, we work with the following assumptions:

1) A log is a (often very large) sequence of queries that must be summarized with a concise representation,
2) The way summaries are obtained should be very flexible (i.e., various relevant summaries may be computed from one query log),
3) In addition, the quality of the summary must be evaluated, in terms of how faithful it is to the log.

To answer point 1, we consider that a log will be summarized by a sequence of queries, i.e., by another (much shorter) log. Our answer to point 1 entails that a query will summarize other queries. For point 2, we propose that summaries are specified in a declarative fashion, i.e., summaries are the results expressions constructed with a dedicated manipulation language. This language allows to transform one or more queries into another query for which its quality as a summary is measured (point 3).

Our contributions include:

- A query manipulation language (called QML) tailored for OLAP queries. This language is composed by binary and unary operators that allow to summarize queries,
- A quality measure based on the classical notions of precision and recall that allows to measure to which extent a query is a good summary of another query,
- A greedy algorithm for automatically constructing, using QML, a summary of a query log w.r.t. a quality threshold given by the user.

This paper is organized as follows. The next section motivates the approach with a toy example. In Section III, we propose a method to summarize queries by introducing the language for manipulating queries and in Section IV we introduce a quality measure for the summarized queries. In Section V we present the algorithm that automatically computes a summary of a query log. Section VI briefly discusses related work. We conclude and draw perspectives in Section VII.

II. Motivation

In this section, we illustrate with a toy example our approach for summarizing a log of OLAP queries. The context of this example is that of a user navigating a data warehouse. In our example, the data warehouse records sales of beverages in different locations at different time. The dimensions of this data warehouse are given Figure 1. Consider a sequence of queries $s = \langle q_1, q_2, q_3 \rangle$ where q_1 is the first query launched, q_2 the second one and q_3 the last one. Suppose these queries are logged in a log L and ask respectively for:

1) The sales of Pepsi and Coke for July 2008, in cities Paris or Marseille,
2) The sales of Coke for July 2008, in regions North or South,
3) The sales of Orangina for the second semester 2008, in regions North or South.

Assume we want to summarize these queries by another query. Various solutions are possible. First, we can summarize

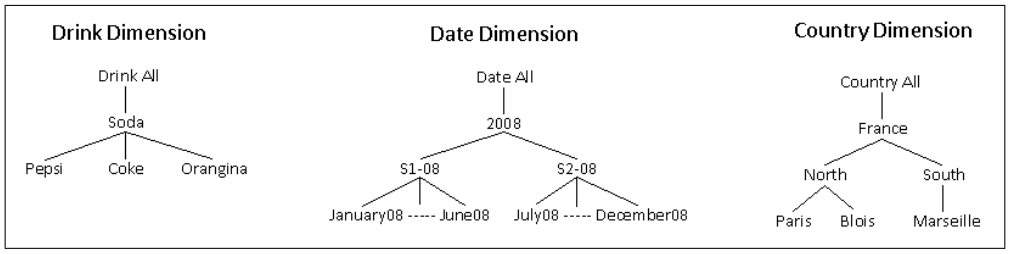

Fig. 1. Dimensions of our toy example

the queries by retaining for each dimension the most frequent members. In that case, the resulting query would ask for sales of Coke in regions North or South during July 2008.

A second alternative would be to summarize the queries with another query having for each dimension the members that cover all members present in the initial queries. For example, note that 'Pepsi', 'Coke' and 'Orangina' are sodas, cities 'Paris' and 'Marseille' and regions 'North' and 'South' are in France and all three queries concern year '2008'. The query summarizing the log L would then ask for the sales of soda in France in 2008.

Finally, note that we can have a compromise by summarizing q_1 and q_2 first, say with the second alternative, and then summarizing the resulting summary with q_3, say with the first alternative. In that case, we would obtain the query asking for the sales of Soda and Orangina in France, region North and region South, for year 2008 and the second semester of 2008.

These examples show the need for flexibility in how the summary is computed. That is why in this paper we propose to have a query manipulation language that is used to specify summaries. In addition, as various summaries can be computed from one log, the quality of these summaries should be evaluated. For instance, for our first alternative, the quality measure should take into account the fact that 'Orangina' is present in the log but not in the summary. In our second alternative, this measure should take into account that indeed 'North' and 'South' cover 'Paris' and 'Marseille' but also 'Blois', that is not present in the log.

Finally, note that so far, we have illustrated the problem of summarizing queries by another query. But a set of queries could be summarized by another set of queries. Moreover, summaries for a log should respect the fact that logs are sequences of queries. For instance, consider again L, this log could be summarized by the sequence $\langle q'_1, q_3 \rangle$ where q'_1 is a summary of q_1 and q_2 asking for the sales of Soda in France in the second semester of 2008.

III. How to summarize queries ?

In this section, we formally define the query manipulation language.

A. Preliminary definitions

As the query manipulation language is tailored for OLAP queries, we first begin with the definition of an OLAP query. Note that in this paper, we do not consider query result, and thus the definition of a query result is not given.

An n-dimensional cube $C = \langle D_1, ..., D_n, F \rangle$ is defined as the classical $n + 1$ relation instances of a star schema, with one relation instance for each of the n dimensions D_i and one relation instance for the fact table F. For a dimension D_i having schema $S = \{L^i_1, \ldots, L^i_{d_i}\}$, a member m is any constant in $\bigcup_{L^i_j \in S} \pi_{L^i_j}(D_i)$. For a dimension D_i, we consider that members are arranged into a hierarchy $<_i$ and we note $m <_i m'$ the fact that the member m' is the ancestor of m in this hierarchy.

Given such a cube, a cell reference (or reference for short) is an n-tuple $\langle m_1, ..., m_n \rangle$ where m_i is a member of dimension $D_i, \forall i \in [1, n]$. We define multidimensional queries as sets of references that can be expressed as Cartesian products of multisets. The reason for having multisets is to be able to define operators that count members' occurrences.

Definition 3.1: Given an n-dimensional cube $C = \langle D_1, ..., D_n, F \rangle$, let R_i be a multiset of members of dimension $D_i, \forall i \in [1, n]$. A query q over C is the multiset of references $q = R_1 \times ... \times R_n$.

In what follows, we assume an n-dimensional cube $C = \langle D_1, ..., D_n, F \rangle$. In the following definitions, i ranges from 1 to n. For a query q, $m_i(q)$ denotes its multiset of members in dimension D_i. A query q will be noted $\times_i m_i(q)$, and multiset $m_i(q)$ will be noted $\langle S_i, f_i \rangle$, where S_i is a set and f_i is a function giving the occurrences of each element of S_i.

Example 3.1: Consider the three queries q_1, q_2 and q_3 of the toy example described in the previous section. q_1 can be expressed in the MDX query language:

$SELECT$
$\{[Drink].[DrinkAll].[Soda].[Pepsi],$
$[Drink].[DrinkAll].[Soda].[Coke]\}$
$ON\ COLUMNS$
$Crossjoin(\{$
$[Country].[CountryAll].[France].[North].[Paris],$
$[Country].[CountryAll].[France].[South].[Marseille]\},$
$\{[Date].[DateAll].[2008].[S2\text{-}08].[July08]\})$
$ON\ ROWS$
$FROM\ Cube$

We have $m_1(q_1) = \{Pepsi, Coke\}$, $m_2(q_1) = \{July08\}$, $m_3(q_1) = \{Paris, Marseille\}$. This query is the set of references: $q_1 = \{Pepsi, Coke\} \times \{July08\} \times \{Paris, Marseille\}$. The set of references for q_2 and q_3 are:

- $q_2 = \{Coke\} \times \{July08\} \times \{North, South\}$

- q_3 = $\{Orangina\} \times \{S2\text{-}08\} \times \{North, South\}$

The language we propose is composed by unary operators and binary operators that manipulate queries and output a query, that is called a summary query (or simply summary for short). The main idea behind the definition of these operators is that they operate dimension-wise: They construct a new query from the one(s) in parameter by treating each dimension independently. We now present formally these operators, starting with the binary operators.

B. The binary operators of QML

The first operators are the classical bag operators [6] extended to multiple dimensions.

Definition 3.2: (Bag operators) Given two queries q_1 and q_2 and $op \in \{\cup_B, \cap_B, \setminus_B\}$, $q_1\ op\ q_2 = \times_i(m_i(q_1)\ op\ m_i(q_2))$.

Example 3.2: Consider the first two queries of Example 3.1, we have:

- $q_4 = q_1 \cup_B q_2 = \{Pepsi, Coke, Coke\} \times \{July08, July08\} \times \{Paris, Marseille, North, South\}$
- q_5 = $q_1 \cap_B q_2$ = $\{Coke\} \times \{July08\} \times \emptyset = \emptyset$
- q_6 = $q_1 \setminus_B q_2$ = $\{Pepsi\} \times \emptyset \times \{Paris, Marseille\} = \emptyset$

The next operators give priority to one query over the other.

Definition 3.3: (Priority operators) Given two queries q_1 and q_2, $q_1 \lhd q_2 = q_1$.

C. The unary operators of QML

Recall that a query can be seen as a Cartesian products of multisets. Our first operator outputs, for a query q in parameter, a query for which only the most frequent members of q in each dimension is retained.

Definition 3.4: (Most frequent operator) Let q be a query with $m_i(q) = \langle S_i, f_i \rangle$ for all i. $mostfreq(q) = \times_i \langle S'_i = \{m \in S_i | \nexists m' \in S_i, f_i(m') > f_i(m)\}, f_{i_{|S'_i}} \rangle$ ($f_{i_{|X}}$ denotes the restriction of a function f_i to the set X)

Example 3.3: $mostfreq(q_4) = \{Coke, Coke\} \times \{July08, July08\} \times \{Paris, Marseille, North, South\}$

Our second operator outputs, for a query q in parameter, a query for which only the most general members of q in each dimension are retained, w.r.t. the hierarchy of the dimension.

Definition 3.5: (Max operator) Let q be a query. $max(q) = \times_i \langle S'_i = \{m \in m_i(q) | \nexists m' \in m_i(q), m <_i m'\}, f_{i_{|S'_i}} \rangle$

Example 3.4: $max(q_4) = \{Pepsi, Coke, Coke\} \times \{July08, July08\} \times \{North, South\}$

Our next operator outputs, for a query q in parameter, a query for which only the lowest common ancestors of the members of q in each dimension are retained, w.r.t. the hierarchy of the dimension.

Definition 3.6: (lca operator) Let q be a query. Let lca be the function that outputs, for a given set of members M in dimension D_i, their common ancestor w.r.t. $<_i$, i.e., $\{m \in D_i | \forall m' \in M, m' <_i m \wedge \nexists m'', m' <_i m'' m'' <_i m\}$. $lca(q) = \times_i lca(m_i(q))$.

Example 3.5: $lca(q_4) = \{Soda\} \times \{S2\text{-}08\} \times \{France\}$

D. Expression of various summaries

We now briefly illustrate how QML can be used. For instance, consider a log L composed by 3 queries: $L = \langle q_1, q_2, q_3 \rangle$. This log can be summarized by the query q_s^1 that retain only the references that appear in all queries, i.e., $q_s^1 = q_1 \cap_B q_2 \cap_B q_3$. Alternatively, L can be summarized by taking into account the frequency of the members used in the log: $q_s^2 = mostfreq(q_1 \cup_B q_2 \cup_B q_3)$. Finally, L can be summarized by a query roughly indicating the parts of the cube that were explored: $q_s^3 = lca(q_1 \cup_B q_2 \cup_B q_3)$. We illustrate these possibilities on our running example.

Example 3.6: Summarizing by retaining the common references of all queries, we obtain: q_s^1 = $q_1 \cap_B q_2 \cap_B q_3$ = $\emptyset$. Summarizing by frequencies on these queries, we obtain: q_s^2 = $mostfreq(q_1 \cup_B q_2 \cup_B q_3)$ = $\{Coke, Coke\} \times \{July08, July08\} \times \{North, North, South, South\}$. Summarizing by lca, we obtain: q_s^3 = $lca(q_1 \cup_B q_2 \cup_B q_3)$ = $\{Soda\} \times \{2008\} \times \{France\}$.

In the following section, we introduce our measure for assessing the quality of summaries expressed with QML.

IV. Quality measure

In this section, we propose a measure that evaluates to which extent a query is a good summary of some other queries. The measure is based on the classical notion of precision and recall. Note that the operators of QML define summaries by adding or deleting references to their operands. It is the case, for instance, of the lca operator which summarizes by adding ancestors.

For instance, in Example 3.5, the added references in the summary are: A = $\{Soda\} \times \{S2\text{-}08\} \times \{France\}$. And in that case, the deleted references in the summary are: D = $\{Coke, Pepsi\} \times \{July08\} \times \{Paris, Marseille, North, South\}$. The intuition behind this example is that we can consider this summary as a good summary with a good quality. However, if we apply the classical precision and recall measures to evaluate its quality, both are null (since $A \cap D = \emptyset$).

For a given set of queries, our idea of good summary is that the more references of the queries it has, the better quality it achieves. But additions or deletions of references should not decrease the quality if these additions cover, at best, the coverage of deleted references (for the references at the most detailed level w.r.t. the hierarchies of the dimensions). In that sense, it would be preferable that the coverage of additions introduces few references not in the coverage of the deleted references. We now formalize these notions.

Definition 4.1: (Coverage) A reference r covers another r' if r=$\langle m_1, ..., m_n \rangle$, r'=$\langle m'_1, ..., m'_n \rangle$ and $\forall_i \in [1, n]$, $m_i >_i m'_i$ or $m_i = m'_i$.

For a set R of references, cover(R)=$\{f \in \Pi_{L_1^1}(D_1) \times \Pi_{L_1^2}(D_2) \times ... \times \Pi_{L_1^n}(D_n) \mid \exists r \in R, r\ covers\ f\}$

In Figure 2, we note L the set of the references of some queries to be summarized, S the set of references of the summary and K as $L \cap S$. The shaded parts are the coverages of D and A respectively, for the most detailed references.

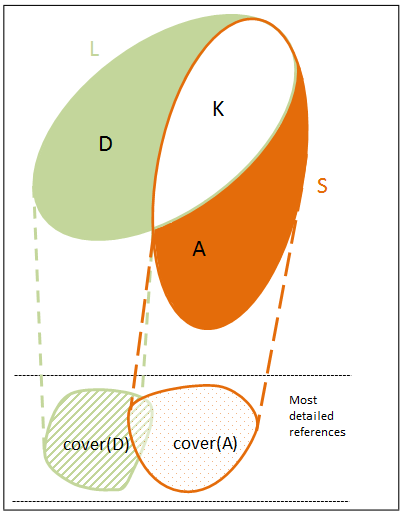

Fig. 2. Principle of the quality measure

For instance, consider Example 3.5. $L = q_1 \cup q_2$, $S = lca(q_4)$ and $L \cap S = \emptyset$ thus both A and $D \neq \emptyset$. So, we compare $cover(A)$ and $cover(D)$. $cover(A) = \{Pepsi, Coke, Orangina\} \times \{July08, August08, September08, October08, November08, December08\} \times \{Paris, Blois, Marseille\}$ and $|cover(A)| = 54$. cover(D) = $\{Pepsi, Coke\} \times \{July08\} \times \{Paris, Marseille\} \cup \{Coke\} \times \{July08\} \times \{Blois\}$ and $|cover(D)| = 5$. We can note that $cover(D) \subset cover(A)$. Intuitively, if a recall measure is used, it would be excellent because all covered references are recalled. On the contrary, a precision measure would be very bad because a lot of references are introduced: The noise is consequent.

To formalize this intuition, our measure of recall is the proportion of covered references existing in $cover(D)$ and found in $cover(A)$ compared with the set of references in $cover(D)$. Moreover, recall favors maximality of K. Recall is defined by: $r = \frac{|K \cup (cover(D) \cap cover(A))|}{|K \cup cover(D)|}$.
The measure of precision is the proportion of covered references existing in $cover(D)$ and found in $cover(A)$ compared with the set of references in $cover(A)$. As for recall, precision encourages maximality of K. Thus we measure the noise in the case when $cover(A)$ would add new covered references in the summary. Precision is formalized as: $p = \frac{|K \cup (cover(D) \cap cover(A))|}{|K \cup cover(A)|}$. If S is the empty set (it is the case of query q_5 in Example 3.2), then we consider that $p = r = 0$.

Note that these recall and precision measures are also relevant in three particular cases:

- If D = $\emptyset$ (there is no deleted references, as for instance for the union operator), recall is perfect and precision is only based on the kept references compared with coverage of A, i.e. $p = \frac{|K|}{|K \cup cover(A)|}$.
- If A = $\emptyset$ (there is no added references, as for instance for the priority operators), precision is perfect and recall is based on the kept references compared to the coverage of D, i.e. $r = \frac{|K|}{|K \cup cover(D)|}$.
- If K = $\emptyset$ (there is no kept references, as for instance for the lca operator), recall and precision are based on the coverage of additions and deletions, i.e. $r = \frac{|cover(A) \cap cover(D)|}{|cover(D)|}$ and $p = \frac{|cover(A) \cap cover(D)|}{|cover(A)|}$.

Finally, our quality measure, simply named $quality$, aggregates our recall and precision measures, with a simple F-measure. Formally, given a set L of references, a query R and a set of dimensions $\{D_1, ..., D_n\}$, $quality(L, R, \{D_1, ..., D_n\}) = 2 \times \frac{p \times r}{p + r}$. It returns a real in [0,1].

So far, we have introduced a way to express summaries of query sets. In the following section, we present our approach for summarizing a query log, i.e., a sequence of queries.

V. How to summarize a log ?

In this section, we present an algorithm for summarizing a log, based on QML and our quality measure. A log is a finite sequence of queries. The main idea is that a summary of a log is also a log, which is computed w.r.t. a quality threshold given by the user.

Note that, if we summarize a set of queries by one query, summary operators can be combined under the form: $u(q_1\ b\ q_2)$ where u is a unary operator, b is a binary operator and q_1, q_2 are two queries of the set to be summarized. The idea is to apply this form in a greedy algorithm (Algorithm Summarize detailed Figures 4 to 6). The algorithm iterates until the quality of the summary falls under a threshold or the summary stops changing. In a first part, the algorithm tests each couple of consecutive queries with a binary operator (line 6 to 17). It takes the best summary w.r.t. the quality measure (line 10 to 15). Then, with this summary query, the algorithm tests each unary operators and the best output is kept (line 19 to 26). Note that the quality is measured w.r.t. the initial query couple and not w.r.t. the output of the binary operator.

Figures 5 and 6 detail the algorithm of the quality measure. Note that the size of coverages are computed without actually computing the coverages. Indeed, we assume that the number of covered members is kept for all members of the dimension tables.

Figure 3 illustrates the algorithm with the following parameters: $L = \langle q_1, q_2, q_3 \rangle$ with q_1, q_2, q_3 the three queries of the toy example, $\mathcal{U}$ and $\mathcal{B}$ are respectively the sets of unary and binary operators of QML, the dimensions of the toy example and the quality threshold of 0.7. All binary operators are tested on each couple of consecutive queries. The couple $\{q_1, q_2\}$ is selected because the quality measure is the highest for the union operator. Then, with the result of binary operator of couple $\{q_1, q_2\}$, all unary operators are tested. max operator is selected. Finally, $\{q_1, q_2\}$ can be summarized by q_{S1}. The algorithm tries to summarize with the new sequence of queries $\langle q_{S1}, q_3 \rangle$ following the same principle. As the computed qualities are all below the threshold, the algorithm stops with result $\langle q_{S1}, q_3 \rangle$.

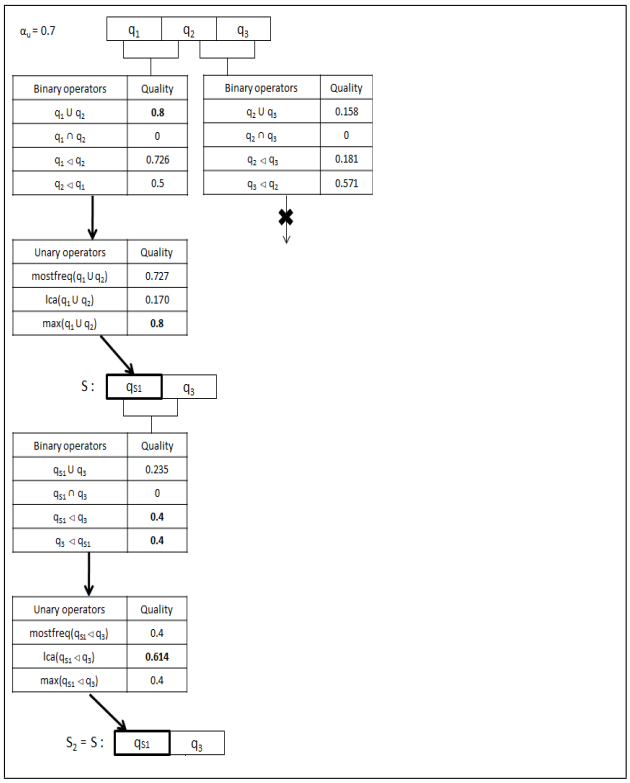

Fig. 3. First step of the algorithm on our toy example

VI. Related Work

Summarization of structured data has attracted a lot of attention in various domain, covering web server log [7] pattern mining (see e.g., [8] that includes a brief survey), sequences of event [9], database [10], multidimensional data stream [11], and datacubes [12].

Many of these works rely on fuzzy set theory ([7], [10]) and/or are compression techniques for which it is important that original data can be regenerated ([12], [8]). Moreover, it can be the case that the summary has not the same type as the data it summarizes. In the domain of database ([10], [11], [12]), summarizing is applied to the database instance where, for OLAP data, measure values are taken into account.

In this paper we address the problem of summarizing an OLAP server query log. Our approach has the following characteristics:

- We do not rely on fuzzy set theory for summarizing. Instead, we leverage the hierarchies described in the dimension tables.
- The type of the summary is the same as the type of the summarized data.
- We do not address the problem of regenerating the summarized data from the summary.
- We do not summarize a database instance, but database queries.

To the best of our knowledge, no work have yet addressed the problem of summarizing a database query log in a suitable and concise representation. As pointed out in [1], many systems provide query logging, primarily for physical tuning, and allow users to view the log. But the way log is displayed, often in flat table or file, is not suitable for browsing. In our earlier work [13], we propose to organize an OLAP query log under the form of a website. But if the log is large, browsing this website may be tedious. An effective log visualization and browsing tool is yet to be designed, and the present work is a step in that direction.

VII. Conclusion and Perspectives

In this paper, we propose a framework for summarizing OLAP query logs. This framework is based on the idea that a query can summarize another query and that a log can summarize another log. Our contributions include a query manipulation language that allows to declaratively specify a summary, a measure to assess the quality of a summary and an algorithm for automatically computing a good quality summary of a query log.

Future work includes the implementation of our approach as well as various tests to assess its efficiency and effectiveness to support on-line analysis. We will also study its extension to a collaborative context where a log, composed by many sequences of queries performed by different users, can be efficiently browsed and searched.

References

[1] N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu, "A case for a collaborative query management system," in *CIDR*. www.crdrdb.org, 2009.

[2] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query recommendations for interactive database exploration," in *SSDBM*, ser. Lecture Notes in Computer Science, M. Winslett, Ed., vol. 5566. Springer, 2009, pp. 3–18.

[3] A. Giacometti, P. Marcel, and E. Negre, "Recommending multidimensional queries," in *DaWaK*, ser. Lecture Notes in Computer Science, T. B. Pedersen, M. K. Mohania, and A. M. Tjoa, Eds., vol. 5691. Springer, 2009.

[4] A. Giacometti, P. Marcel, E. Negre, and A. Soulet, "Query recommendations for olap discovery driven analysis," in *DOLAP*, 2009, pp. 81–88.

[5] K. Stefanidis, M. Drosou, and E. Pitoura, ""You May Also Like" Results in Relational Databases," in *PersDB*, 2009.

[6] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008.

[7] S. Zadrozny and J. Kacprzyk, "Summarizing the contents of web server logs: A fuzzy linguistic approach," in *FUZZ-IEEE*. IEEE, 2007, pp. 1–6.

[8] M. Ndiaye, C. T.Diop, A. Giacometti, P. Marcel, and A. Soulet, "Cube based summaries of large association rule sets," in *sixth International Conference on Advanced Data Mining and Applications (ADMA)*. LNCS, 2010.

[9] W. Peng, C. Perng, T. Li, and H. Wang, "Event summarization for system management," in *KDD*, P. Berkhin, R. Caruana, and X. Wu, Eds. ACM, 2007, pp. 1028–1032.

[10] R. Saint-Paul, G. Raschia, and N. Mouaddib, "General purpose database summarization," in *VLDB*, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, Eds. ACM, 2005, pp. 733–744.

[11] Y. Pitarch, A. Laurent, and P. Poncelet, "Summarizing multidimensional data streams: A hierarchy-graph-based approach," in *PAKDD (2)*, ser. Lecture Notes in Computer Science, M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds., vol. 6119. Springer, 2010, pp. 335–342.

[12] L. V. S. Lakshmanan, J. Pei, and J. Han, "Quotient cube: How to summarize the semantics of a data cube," in *VLDB*. Morgan Kaufmann, 2002, pp. 778–789.

[13] S. Colas, P. Marcel, and E. Negre, “Organisation de log de requêtes OLAP sous forme de site web,” in *6èmes journées francophones sur les Entrepôts de Données et l’Analyse en ligne (EDA 2010), Djerba, Tunisie*, ser. RNTI, vol. B-6. Toulouse: Cépaduès, Juin 2010, pp. 81–95.

INPUT:
L: a log,
$\mathcal{U}$: a set of unary operators,
$\mathcal{B}$: a set of binary operators,
$D_1, \ldots, D_n$: the dimensions,
α: quality threshold.
OUTPUT: A summary of L.
VARIABLES:
SOQ, SOQ': sequences of queries,
$query1_S$, $query2_S$, q_S, $query_{\mathcal{B}}$, $query_{\mathcal{U}}$, $query_1$, $query_2$: queries,
$quality$, $qualityQuery$, max: reals.

$SOQ' \leftarrow \langle\rangle$
$SOQ \leftarrow L$
$qualityQuery \leftarrow \alpha$
while ($qualityQuery >= \alpha$) and ($SOQ' \neq SOQ$) and (sizeOf(SOQ) $\neq$ 1) **do**
  $max \leftarrow 0$
  for $each\ pair\ \langle query_1, query_2\rangle\ of\ consecutive\ queries\ in\ SOQ$ **do**
    for $each\ b\ \in\ \mathcal{B}$ **do**
      $q_S \leftarrow query_1\ b\ query_2$
      $ref \leftarrow query_1 \cup query_2$ {the union of the references of $query_1$ with those of $query_2$}
      $quality \leftarrow$ Quality(ref,q_S,$D_1, \ldots, D_n$) {Function Quality() returning the quality degree of a summarized query}
      if $quality > max$ **then**
        $max \leftarrow quality$
        $query1_S \leftarrow query_1$
        $query2_S \leftarrow query_2$
        $query_{\mathcal{B}} \leftarrow q_S$
      end if
    end for
  end for
  $max \leftarrow 0$
  for $each\ u\ \in\ \mathcal{U}$ **do**
    $q_S \leftarrow u(query_{\mathcal{B}})$
    $ref \leftarrow query1_S \cup query2_S$
    $quality \leftarrow$ Quality(ref,q_S,$D_1, \ldots, D_n$)
    if $quality > max$ **then**
      $max \leftarrow quality$
      $query_{\mathcal{U}} \leftarrow q_S$
    end if
  end for
  $qualityQuery \leftarrow max$
  $SOQ' \leftarrow SOQ$
  $SOQ \leftarrow SOQ$.replace($query1_S$,$query2_S$,$query_{\mathcal{U}}$) {Function replace() replacing the pair $\{query1_S$,$query2_S\}$ by $query_{\mathcal{U}}\}$
end while
return SOQ'

Fig. 4. Algorithm Summarize(L, $\mathcal{U}$, $\mathcal{B}$, $D_1, \ldots, D_n$, α)

INPUT:
ref_i, ref_s: set of references,
$D_1, \ldots, D_n$: the dimensions
OUTPUT: The quality of ref_s w.r.t ref_i.
VARIABLES:
D, A: a set of references, K, C_D, C_A,$C_\cap$: integer, r, p: float

```
1:  K ← |ref_i ∩ ref_s|
2:  D ← ref_i - ref_s
3:  A ← ref_s - ref_i
4:  D ← max(D) {function max() returning a set of reference from D covering every references of D}
5:  A ← max(A)
6:  C_D ← CardinalityOfCoverage(D)
7:  C_A ← CardinalityOfCoverage(A)
8:  Cov_∩ ← 0
9:  M_∩ ← 0
10: C_∩ ← 0
11: for each reference r_D of D do
12:     for each reference r_A of A do
13:         for each dimension d of D_1, ..., D_n do
14:             for each member m_D of r_D do
15:                 for each member m_A of r_A do
16:                     if m_A <_d m_D then
17:                         Cov_∩ = Cov_∩ + |cover(m_A)|
18:                     else {m_D <_d m_A}
19:                         Cov_∩ = Cov_∩ + |cover(m_D)|
20:                     end if
21:                 end for
22:             end for
23:             M_∩ = M_∩ × Cov_∩
24:         end for
25:     end for
26:     C_∩ = C_∩ + M_∩
27: end for
28: r ← (K+C_∩)/(K+C_D)
29: p ← (K+C_∩)/(K+C_A)
30: return 2 × (p × r)/(p + r)
```

Fig. 5. Algorithm Quality(ref_i, ref_s, $D_1, \ldots, D_n$)

INPUT:
S_r: set of references,
$D_1, \ldots, D_n$: the dimensions
OUTPUT: $|cover(S_r)|$.
VARIABLES:
r: a reference, d: a dimension, m: a member,

```
1:  Cov ← 0
2:  T ← 0
3:  C ← 0;
4:  for each reference r of S_r do
5:      for each dimension d of D_1, ..., D_n do
6:          for each member m of r do
7:              Cov = Cov + |cover(m)|
8:          end for
9:          T = T × Cov
10:     end for
11:     C = C + T
12: end for
13: return C
```

Fig. 6. Algorithm CardinalityOfCoverage(S_r, $D_1, \ldots, D_n$)