Query recommendations for OLAP discovery driven analysis

Arnaud Giacometti, Patrick Marcel, Elsa Negre, Arnaud Soulet
Université François Rabelais Tours - France
Laboratoire d'Informatique
firstname.lastname@univ-tours.fr

ABSTRACT

Recommending database queries is an emerging and promising field of investigation. This is of particular interest in the domain of OLAP systems where the user is left with the tedious process of navigating large datacubes. In this paper we present a framework for a recommender system for OLAP users, that leverages former users' investigations to enhance discovery driven analysis. The main idea is to recommend to the user the discoveries detected in those former sessions that investigated the same unexpected data as the current session.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Data warehouse and repository; H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Algorithms, Design

Keywords

OLAP analysis, MDX queries, recommendation

1. INTRODUCTION

One of the goal of recommender systems is to help users navigating large amounts of data. Existing recommender systems are usually categorized into content-based methods and collaborative filtering methods [1]. Content-based methods recommend to the user items similar to the ones that interested him in the past, whereas collaborative filtering methods recommend to the user items that interested similar users.

Applying recommendation technology to database, especially for recommending queries, is an emerging and promising topic [12, 5]. It is of particular relevance to the domain of multidimensional databases, where OLAP analysis is inherently tedious since the user has to navigate large datacubes to find valuable information, often having no idea on what his/her forthcoming queries should be. This is often the case in discovery driven analysis [20] where the user investigates a particular surprising drop or increase in the data.

In our earlier works [7, 8] we proposed to adapt techniques stemming from collaborative filtering to recommend OLAP queries to the user. The basic idea is to compute a similarity between the current user's sequence of queries and the former sequences of queries logged by the server. In this work, we extend this principle to better take into consideration what the users were looking for. Our work is inspired by what is done in web search and e-commerce applications (e.g., [14]) where inferred properties of former sessions are used to support the current session. In our case, the idea is to infer, for every former sessions on the OLAP system, what the user was investigating. This has the form of a pair of cells showing a significant unexpected difference in the data. If it is found that indeed this difference was investigated during the session, then the discoveries of this former session can be suggested to the current user.

This paper is organized as follows. Next section motivates our approach with a simple yet realistic example. Section 3 reviews related work. Our formal framework is presented in Section 4, and the algorithms are presented in Section 5. Section 6 illustrates the use of the framework on the example given in Section 2. Section 7 briefly introduces our ongoing implementation of the framework. We conclude and draw perspectives in Section 8.

2. MOTIVATING EXAMPLE

In this section we illustrate our approach with an artificial yet realistic motivating example. This example uses typical discovery driven analysis sessions of a simple datacube containing sale results of various products in various locations at different times. These sessions are sequences of queries, the result of which are depicted in Figure 1 (the meaning of the arrows is explained at the end of this present section). On the left are three sessions from the log of the OLAP server and on the right is a current session. This current session consists of only one query (q) that asks for the aggregated sales of cheese in 2007 and 2008 in Europe and USA. The current user may wonder where to navigate the cube further. We will now show how the information in the log can be exploited to provide her/him with some suggestions.

Let us first describe the sessions contained in the log of the OLAP server. In what follows, query q_i^j denotes the i^{th} query of the j^{th} session. The first session first asked for the sales for various dairy products (cheese, milk, and butter)

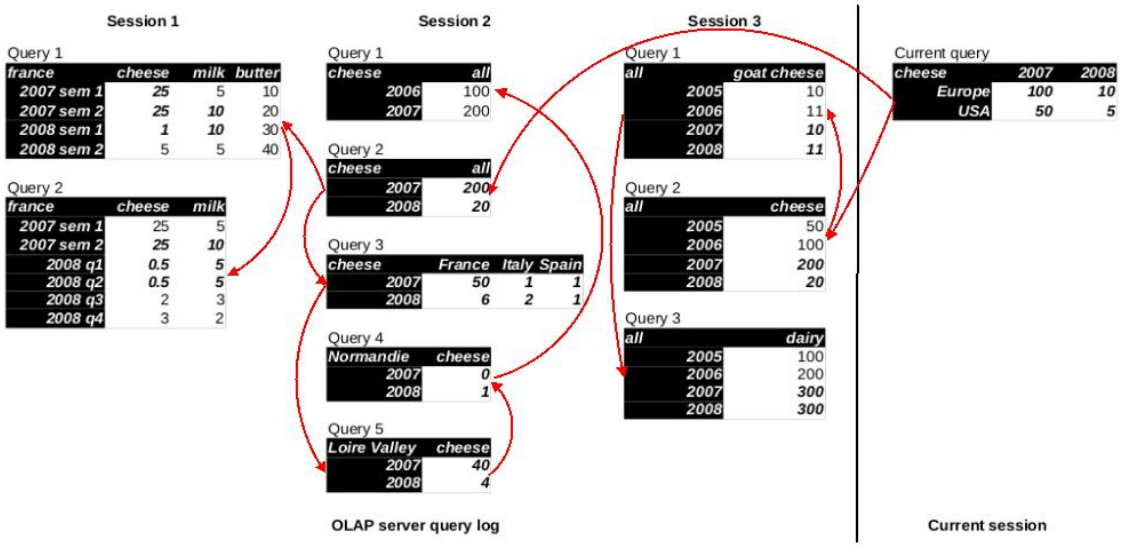

Figure 1: Example of a log and a current session, with computed recommendations

in France for each semester of 2007 and 2008 (query q_1^1), and then for the sales of cheese and milk in France for each quarter of 2008 and each semester of 2007 (q_2^1).

The second session first asked for the sales of cheese in 2006-2007 (q_1^2) and then the sales of cheese in 2007-2008 (q_2^2). Next it asked for the sales of cheese for 2007-2008 in France, Italy and Spain (q_3^2). Then for the sales of cheese for 2007-2008 in two french regions, namely Normandie (q_4^2), and finally the Loire Valley (q_5^2).

The third session asked for the sales of goat cheese for 2005-2008 (q_1^3). Then the user rolled up and asked for the sales of cheese for 2005-2008 (q_2^3). Finally, he/she rolled up again to obtain the sales of all dairy products for 2005-2008 (q_3^3).

By observing these sessions one can notice that each of them is concerned with a general difference that is a drop of the sales of cheese from 2007 to 2008. It appears for instance for query q_2^3 of session 3 and for query q_2^2 of session 2. In the log, there is no difference that can be said to be more general than this one (note for instance that the sales of dairy products is stable from 2007 compared to 2008). Hence this particular difference, the drop of sales for cheese from 2007 to 2008, is said to be the *most general difference* (see formal definitions in Section 4). The queries whose result displays this difference are called *most general difference queries* (*mgdq* for short). This difference also appears for queries q of the current session, q_1^1, q_3^2, q_5^2 and q_1^3, at lower levels of detail. These queries can be said to confirm this difference and are called *drilldown differences* in what follows. On the other hand, query q_4^2 shows that the sales of cheese in a particular region increased from 2007 to 2008. This query is said to be an *exception* to the general difference in what follows.

Suppose now that the log is processed to find the more general differences it contains, as well as their drilldown differences and exceptions. A recommender system for OLAP analysts should recognize that the current user's query is a drilldown difference of one of the mgdq of the log. It would then suggest to the user to navigate the cube to see the mgdq, its drilldown differences and exceptions. On Figure 1, this suggestion consists of a graph of queries that allows to navigate the three sessions of the log, starting from the current query q. Each arrow can be interpreted as "if you have evaluated this query then you might be interested by that next query". For instance, it is recommended to the user to evaluate query q_2^3, whose result displays the mgdq, then q_1^3 whose result displays an exception to this general difference, and finally q_3^3 which is the only remaining query of session 3.

Our framework can be used as a basis for such a recommender system. It is composed of two parts, the processing of the log, and the computation of recommendations, that are detailed in Section 5.

3. RELATED WORK

3.1 Anticipating database queries

In a recent paper, [12] point out the need for systems leveraging former sessions to support database users analyzing large amount of data. The only work we know that proposes to recommend SQL queries for supporting database exploration is that of [5]. Although this work shares some common features with ours, it differs on two important aspects: First it deals only with SQL Select-Project-Join queries and second, the fact that a session is a sequence of queries is not taken into account.

In the context of multidimensional databases, our previous works [7, 8] propose a framework and a system for recommending OLAP queries to a current user by taking advantage of former analytical sessions. This framework is based on the proposal of methods for evaluating the distance between multidimensional queries on the one hand, and the distance between sessions on the other hand. Following a classical collaborative filtering approach, the current session is compared to the sessions found in the log and the sessions close to the current session are used for computing recommendations.

In the present paper we are enriching this approach by taking users' discoveries into consideration. The idea is no

more to recommend queries of sessions that are close to the current session. Instead, our framework recommends queries based on sessions that investigated the same general difference as the one investigated by the current user.

In the same context, the recent work of [11] recommend OLAP queries to the current user by transforming the current query with user preferences in the spirit of what is done in [2]. This work is more of a content-based method as it does not take former sessions into account.

Finally, note that the work of Sapia [16, 17] shares with our work the goal of predicting the forthcoming OLAP query. However the main concern of this work is to prefetch data, not to guide the user towards interesting parts of the cube.

3.2 Discovery driven analysis of datacubes

To support interactive analysis of multidimensional data, Sarawagi et al. introduced discovery driven analysis of OLAP cubes in [20]. This and subsequent work resulted in the definition of advanced OLAP operators to guide the user towards unexpected data in the cube or to propose to explain an unexpected result. We now present two of these operators that can be thought of as implementations of some of the operators of our framework, and that are indeed used for implementing it (see Section 7).

The DIFF operator proposed in [18] explores the reasons why an aggregate is lower (or higher) in one cell compared to another. It takes as parameter two cells c and c', and looks into the two isomorphic subcubes C and C' that detail the two cells (i.e., that are aggregated to form the observed c and c'). As a result, it summarizes the differences in these two subcubes by providing the top-N informative cells from the unvisited part of the cube.

For instance, on the example given in Section 2, a DIFF computed on the two cells of the result of query q_2^2 would propose the two cells of the result of query q_5^2 as one of its answer.

In [21] a RELAX operator has been proposed that can be thought of as the opposite to the DIFF operator. Indeed RELAX tries to confirm at a lesser level of detail a particular significant difference, and summarizes the exceptions to this difference. In its basic form, the RELAX operator takes as parameter two cells c and c', and rolls up to less detailed levels to check if the difference between c and c' also occurs at these levels. For each of these rollups, the most relevant exceptions to the difference are computed.

For instance, on the example given in Section 2, a RELAX computed on the two cells of the result of query q_5^2 would propose as part of its answer both the two cells of the result of query q_2^2 (as a general difference) and the two cells of the result of query q_4^2 (as an exception).

Discussion.

First, note that both DIFF and RELAX are slightly different from the other classical OLAP operators (rollup, slice, etc.), in the sense that they do not produce a cube nor a cross-tab as a result, but a list of cells of the navigated cube. This list can be large. Now the main difference with our present work is that these operators are applied only on query results and they do not take into consideration what other users have discovered. Taking the queries of the log into account can be viewed as a way of filtering the result of these operators, and to propose to the current users only those query results that the former users did find relevant. Indeed, consider the example given Section 2. Suppose the current user applies the RELAX operator on the result of her/his current query to search for differences that generalize the difference of sales of cheese in USA for 2007-2008. The answer can contain general differences for the sales result at higher levels of dimension products (dairy, food, consumable, etc.), combined with higher levels of dimension location (North-America, America, Outside Europe, etc.), combined with higher levels of dimension time (21^{st} century, etc.). However, in the log there are three sessions that focused on the drop of the sales of cheese, and thus our framework will propose to the current user to search in this direction first.

Finally, note that discovery driven analysis is still attracting attention. Indeed, two recent works use a data mining approach to inform the user of potentially interesting regions of a cube by either automatically detecting interesting cells [3] or proposing interesting drill paths [4]. In the former case, the goal is simply to highlight in a given query result the cells whose measure deviates the most from a theoretical value computed under independence model hypothesis. In the latter case however, the goal can be seen as recommending drilldown queries to the user. This approach does not take into account former explorations and thus it is very close to the DIFF operator described above.

3.3 Session properties used in Information Retrieval

The idea of using former sessions to improve current search is very popular in Information Retrieval ([1]) and Web Usage Mining ([22]).

In recent works, properties of the session are inferred to support subsequent searches. For instance, in [6], the *information goal* of a session is defined as the last URL visited during the session or alternatively the last click on a search engine result page.

In [14] in the domain of e-commerce, the session goal is a particular event occurring in the session. In this case of the Ebay site, the goal of a session is a buy event. This allows to enrich all the sessions (and especially the queries of the sessions) with the description of the item bought, which is called the *context* of the session. The authors show how defining the context of a session helps recovering from null result in subsequent searches, provides a better understanding of the queries in the session, or helps generating recommendations.

These works have influenced our approach. Indeed, the mgdq detected in each OLAP session can be viewed as the session context and the drilldown differences and exceptions can be viewed as the session goals.

4. THE FORMAL FRAMEWORK

4.1 Overview of the approach

Recommendations are computed on the basis of the differences discovered in the log. The idea is to detect which difference the current session is investigating and to recommend the sessions in the log that investigated the same difference.

More precisely, the log are preprocessed offline in the following way: 1/ Each session is examined to infer what is the query whose result displays the most general difference (mgdq) that was observed. As there can be more than one mgdq per session, for each such mgdq, one *investigation* is

created that records the mgdq, its drilldown differences, and exceptions. 2/ Once all sessions are examined, the mgdq of the mgdq detected are computed and the investigations are grouped by common mgdq.

Recommendations are computed online each time a current query is added to the current session by the current user. The current query is analyzed to detect to which mgdq of the log it corresponds (this query may be itself a mgdq, a drilldown difference, or an exception of what is detected in the log). Then a navigation plan (a set of sessions arranged in a graph of queries) is proposed for the current user to see drilldown differences or exceptions to the mgdq, by using the queries of the preprocessed sessions.

In what follows, we detail the framework, starting with basic definitions, then explaining how the log is processed and finally how recommendations are computed.

4.2 Preliminaries

An n-dimensional *cube* $C = \langle D_1, \ldots, D_n, F\rangle$ is defined as the classical $n+1$ relation instances of a star schema, with one relation instance for each of the n dimensions and one relation instance for the fact table. Given a particular dimension table D_i, the members of the dimension are the values in this table and they are arranged in a hierarchy denoted $<_i$.

Given an n-dimensional cube $C = \langle D_1, \ldots, D_n, F\rangle$, a cell *reference* (or reference for short) is an n-tuple $\langle m_1, \ldots, m_n\rangle$ where m_i is a member of dimension D_i for all $i \in [1, n]$. A cell is a tuple of $cube(F)$ where $cube(F)$ denotes the datacube [10] of the fact table F. In what follows we will use $measure(c)$ to denote the measure of the cell c.

As in our previous work [7, 8], we define multidimensional queries as sets of references in the following way: Given an n-dimensional cube $C = \langle D_1, \ldots, D_n, F\rangle$, let R_i be a set of members of dimension D_i for all $i \in [1, n]$, a *query* over C is the set of references $R_1 \times \ldots \times R_n$. In this paper we restrict to queries q whose result is the set of cells defined by $cube(F) \bowtie q$. A session is a sequence of queries, and a log is a set of sessions. In what follows, we note $r \in q$ to denote that r is a reference of a query q and $c \in q$ to denote that c is a cell of the result of a query q. $r(i)$ denotes the i^{th} member of a reference r. When the context is clear, a query q will be confounded with its result, and we note $cells(q)$ the set of cells of a query q. We denote the set of queries of a session s by $queries(s)$.

Example 1. Consider the example given in Section 2 that will be used as a running example throughout this section. The current query q is the set of references $\{cheese\} \times \{2007, 2008\} \times \{Europe, USA\}$. Its result is the set of cells $\{\langle cheese, 2007, Europe, 100\rangle, \langle cheese, 2008, Europe, 10\rangle, \langle cheese, 2007, USA, 50\rangle, \langle cheese, 2008, USA, 5\rangle\}$.

The classical specialization relation over cell references is defined: $r <_{cells} r'$ if for all dimensions i, either $r(i) = r'(i)$ or $r(i) <_i r'(i)$. This relation is extended to cells as follows: For two cells c, c' of an n-dimensional cube C, $c <_{cells} c'$ if $r <_{cells} r'$ where r is the reference of c and r' is the reference of c'.

4.3 Difference pairs

We now define the pairs of cells that will be considered during the processing of the log. First note that the specialization relation over cells can be extended to pairs of cells in the following way.

Definition 1. (specialization over pairs) Let C be a cube and c, c', c'', c''' be four cells of C. The pair $\langle c, c'\rangle$ is a generalization of $\langle c'', c'''\rangle$, noted $\langle c, c'\rangle <_{cells} \langle c'', c'''\rangle$ if both $c <_{cells} c''$ and $c' <_{cells} c'''$.

Example 2. Let $c = \langle cheese, 2007, all, 200\rangle$, $c' = \langle cheese, 2008, all, 20\rangle$, $c'' = \langle cheese, 2007, France, 50\rangle$, and $c''' = \langle cheese, 2008, France, 6\rangle$ be four cells of the cube analyzed, that appear in the results of query q_2^2 and q_3^2. We have $\langle c, c'\rangle <_{cells} \langle c'', c'''\rangle$.

If we have $\langle c, c'\rangle <_{cells} \langle c'', c'''\rangle$, we will say that $\langle c, c'\rangle$ is a *rollup pair* of $\langle c'', c'''\rangle$ and $\langle c'', c'''\rangle$ is a *drilldown pair* of $\langle c, c'\rangle$. Moreover, if $\langle c'', c'''\rangle$ is a drilldown pair of $\langle c, c'\rangle$ and $sign(measure(c'') - measure(c''')) \neq sign(measure(c) - measure(c'))$ we will say that $\langle c'', c'''\rangle$ is an *exception pair* of $\langle c, c'\rangle$. Given a set S of pairs of cells, the most general pairs are the pairs of S that have no rollup pairs in S.

Definition 2. (most general pairs) Let S be a set of pairs of cells. The most general pairs of S are the set $max_{<_{cells}}(S)$. For a given pair of cells $\langle c, c'\rangle$ of S, the most general pairs for $\langle c, c'\rangle$ in S is the set $max_{<_{cells}}(\{\langle c'', c'''\rangle \in S | \langle c'', c'''\rangle$ is a rollup pair of $\langle c, c'\rangle\})$.

In what follows we will call a significant difference (or difference for short) a pair of cells such that their measure differ significantly. This significance is computed by a user-defined function d on which we do not impose particular requirements. Examples of such functions are given in Section 7.

Definition 3. (difference pair) Let C be a cube, d be a boolean function over the pairs of cells of C and c', c be two cells of C. The pair $\langle c, c'\rangle$ is a difference pair for C and d if $d(c, c') = true$.

Example 3. If the function d outputs true if the measures differ by a factor of 10, then the pair of cells $\langle c, c'\rangle$ of Example 2 is a difference pair. In Figure 1 every cell that is part of a difference pair has its measure in bold face. $\langle c, c'\rangle$ is a rollup pair of $\langle c'', c'''\rangle$. If S is the set of all pairs of cells of the queries in session 2, then $\langle c, c'\rangle$ is the most general pair of $\langle c'', c'''\rangle$ in S.

4.4 Difference queries

We define a difference query to be a query whose result displays one or more difference pairs. A query is a rollup (resp. drilldown) difference query of a difference query if its result confirms the difference at a higher (resp. lower) level of detail. An exception is a query which result contradicts a difference at a lower level of detail. The following definitions formalize these notions.

Definition 4. (difference query) Let C be a cube, d be a boolean function over the pairs of cells of C. A query q over C is a difference query if there exists two cells $c, c' \in q$ such that the pair $\langle c, c'\rangle$ is a difference pair for C and d.

Definition 5. (rollup/drilldown difference query) Let q and q' be two queries. We say that q is a rollup difference query for q' (resp. q' is a drilldown difference query for q) w.r.t. the pairs $\langle c, c'\rangle$ and $\langle c'', c'''\rangle$, if $c, c' \in q$ and $c'', c''' \in q'$, one of $\langle c, c'\rangle$, $\langle c'', c'''\rangle$ is a difference pair and $\frac{(measure(c'') - measure(c'''))}{(measure(c) - measure(c'))} > \beta$ for some threshold $\beta > 0$.

If q is a rollup (resp. drilldown) difference query for q' w.r.t. the pairs $\langle c, c' \rangle$ and $\langle c'', c''' \rangle$, we say that q is a rollup (resp. drilldown) difference query for $\langle c'', c''' \rangle$ (resp. $\langle c, c' \rangle$).

Example 4. Continuing Example 3, q_2^2 is a difference query, as is the current query q. Thus q_2^2 is a rollup difference query for q, and a rollup difference query for the difference pair $\langle \langle cheese, 2007, Europe, 100 \rangle, \langle cheese, 2008, Europe, 10 \rangle \rangle$.

For a difference pair $\langle c, c' \rangle$ and a set Q of queries, we define a most general difference query for $\langle c, c' \rangle$ (or mgdq for short) to be a query of Q at the highest level of detail that contains a rollup pair of $\langle c, c' \rangle$.

Definition 6. (mgdq) Let Q be a set of queries and $\langle c, c' \rangle$ be a difference pair. An mgdq of $\langle c, c' \rangle$ w.r.t. Q is a query $q \in Q$ such that q contains a most general pair of $cells(Q)$, where $cells(Q) = \cup_Q cells(q)$.

Definition 7. (exception difference query) Given two queries q and q', q' is an exception difference query of q if there exists four cells $c, c' \in q$ and $c'', c''' \in q'$ such that $\langle c'', c''' \rangle$ is an exception pair of $\langle c, c' \rangle$. It is said that q' is an exception difference query for $\langle c, c' \rangle$.

Example 5. Consider the set of queries of session 2 $Q = \{q_1^2, \ldots, q_5^2\}$, and the difference pair $\langle \langle cheese,$ 2007, $Europe$, 100$\rangle$,$\langle cheese,$ 2008, $Europe$, 10$\rangle \rangle$. q_2^2 is the mgdq of Q. Query q_4^2 is an exception difference query of q_2^2.

4.5 Operators for detecting differences and exceptions

Obviously the idea is not to detect all the differences that appear in a cube. We focus on the differences that are detected by the users in the result of their queries. These differences are extracted from the query log with the following operators.

The first operator, *difference* outputs the pairs of cells of a query q that are difference pairs, i.e., $difference(d, q) = \{\langle c, c' \rangle \in q | d(c, c') \; is \; true\}$ for some boolean function d over pairs of cells.

The next two operators detect, for a pair $\langle c, c' \rangle$ and a set of queries Q, which are the queries of Q that are rollup (drilldown) difference queries for $\langle c, c' \rangle$. Formally, $rollupDifference(c, c', Q) = \{q \in Q | q$ is a rollup difference query for $\langle c, c' \rangle\}$, and $drilldownDifference(c, c', Q) = \{q \in Q | q$ is a drilldown difference query for $\langle c, c' \rangle\}$.

The operator $mgdq$ computes the mgdq of $\langle c, c' \rangle$ that occur in a set of queries Q, i.e., $mgdq(c, c', Q) = \{q \in Q | q$ is a mgdq for $\langle c, c' \rangle\}$.

Finally, the last operator detects for a pair of cells $\langle c, c' \rangle$ and a set of queries Q, the exceptions to $\langle c, c' \rangle$ in Q: $exceptions(c, c', Q) = \{q \in Q | q$ is an exception difference query for $\langle c, c' \rangle\}$.

5. THE ALGORITHMS

This section introduces the algorithms underlying our approach.

5.1 Processing the log

We begin with the algorithm used to discover differences and exceptions from a log file.

First, Algorithm 1 processes each session to discover the mgdq, their drilldown differences and exceptions. This algorithm outputs a set of what we call *investigations*, i.e., sessions with added information about the investigated differences. Note that for each session of the log, one investigation is created by mgdq discovered in the session, provided the mgdq comes with some drilldown differences or exceptions.

Definition 8. (investigation) An investigation i for a session s is a tuple $\langle c, c', m, D, E, O \rangle$ where $\langle c, c' \rangle$ is a most general pair that is investigated in s, m is a query that contains $\langle c, c' \rangle$, D is the set of difference drilldown queries of m, E is the set of difference exception queries of m and O is the set of the remaining queries of s.

Algorithm 1: session processing

```
Input: A log L, a boolean function d
Output: A set I of investigations
I ← ∅
foreach session s of L do
    foreach query q of s do
        foreach difference pair (c, c') of difference(d, q) do
            // detect the rollup diff. of (c, c')
            R ← rollupDifference(c, c', queries(s))
            // find the mgdq of R
            M ← mgdq(c, c', R)
            foreach mgdq m ∈ M do
                foreach (t, t') ∈ m such that t <_cells c and t' <_cells c' do
                    // detect the drilldown diff. of m
                    D ← drilldownDifference(t, t', queries(s))
                    // detect the exceptions of m
                    E ← exceptions(t, t', queries(s))
                    // create investigation
                    if D ≠ ∅ or E ≠ ∅ then
                        I ← I ∪ createInvestigation(m, t, t', D, E, s)
```

In Algorithm 1, $createInvestigation$ creates an investigation i using the queries of session s in the following way. The queries of i are that of s and they are ordered as in s. The mgdq and the queries of D and E are labeled with their type (mgdq, drilldown difference or exception) and are associated with their pair of cells that is the difference pair w.r.t. the mgdq. Note that a query can be at the same time mgdq or drilldown difference or exception. Each investigation i is labeled with its mgdq denoted $mgdq(i)$ and this mgdq is associated with its difference pair denoted $differencePair(i)$ and called the difference pair of the investigation.

Example 6. Consider session 1 of Figure 1. Suppose the pair of cells $t_1 = \langle cheese, 2007sem1, France, 25 \rangle$, $\langle cheese, 2008sem1, France, 1 \rangle$ is detected as a difference pair for query q_1^1. There is no rollup differences detected in session 1 for this pair, and thus it is the mgdq of this session. Then query q_2^1 is detected as a drilldown difference query of q_1^1. Therefore an investigation i is created, with t_1 as difference pair, q_1^1 as mgdq and q_2^1 as drilldown difference query. Similarly, note that q_2^2 will be the mgdq of an investigation i' constructed from session 2, and q_2^3 will be

the mgdq of an investigation i'' constructed from session 3 (see Figure 2).

Note that the investigations created by Algorithm 1 can investigate the same most general pair of the cube. More precisely, the difference pair of an investigation i can be a rollup pair of the difference pair of another investigation i'. Thus to complete the processing of the log, it remains to invoke, for each investigation the operator $mgdq(c, c', Q)$ where $\langle c, c' \rangle$ is the difference pair of the investigation and Q is the set of the mgdq of all the investigations. Then, each investigation is updated with its new mgdq and new difference pair if any. If there is more than one mgdq, the one with the least number of cells is used. Finally, the investigations are grouped by their difference pairs.

Example 7. Consider the investigations i, i' and i'' of Example 6. They all investigate the same most general pair of the cube, namely $\langle cheese, 2007, all, 200 \rangle$, $\langle cheese, 2008, all, 20 \rangle$. Consider investigation i. Query q_2^2 and query q_2^3 are mgdq of the mgdq of i. Thus i is modified in the following way: Query q_2^2 becomes its new mgdq and query q_1^1 becomes a drilldown difference query.

5.2 Computing recommendations

Given a current session sc, a current query q and a set of investigations I, the recommender system first identifies in I the difference pairs m to which q can be related. q can be either a drilldown difference of m, a rollup difference of m, or an exception of m. In each case a specific recommendation is issued using the investigations associated with m.

Algorithm 2: Recommending sessions

Input: A current session sc, a current query q, a set I of investigations, a boolean function d
Output: A graph G of recommended queries
$G \leftarrow \langle \emptyset, \emptyset \rangle$
$M \leftarrow$ the mgdq of I
foreach *difference pair* $\langle c, c' \rangle$ *of* $difference(q, d)$ **do**
  `// first check if q is a drilldown diff.`
  $C \leftarrow rollupDifferences(c, c', M)$
  if $C \neq \emptyset$ **then**
    $G \leftarrow G \cup recommendDrilldown(sc, q, C)$
  `// then check if q is a rollup diff.`
  $C \leftarrow drilldownDifference(c, c', M)$
  if $C \neq \emptyset$ **then**
    $G \leftarrow G \cup recommendRollup(sc, q, C)$
  `// finally check if q is an exception to a difference`
  foreach *difference pair* $\langle x, x' \rangle$ *of* $m \in M$ **do**
    $C \leftarrow exceptions(x, x', \{q\})$
    if $C \neq \emptyset$ **then**
      $G \leftarrow G \cup recommendException(sc, q, C)$

Function *recommendDrilldown* is given below. The idea is to recommend each session which mgdq is a rollup difference query of the current query, with the queries of the session arranged in a given order (first the mgdq, then the drilldown differences of the current query, etc.). The recommendation is a navigation plan, i.e., a graph of queries rooted in the current query q. The other functions used in Algorithm 2 for recommending sessions follow the same general principle. For instance, if the current query q is detected as an exception of the mgdq of an investigation, then it makes sense to present first the exceptions of the mgdq that are the rollup difference queries of q, and then the exceptions of the mgdq that are drilldown difference queries of q.

Function `recommendDrilldown(`sc, q, C`)`

Input: A current session sc, a current query q, a set C of mgdq
Output: A graph $G = \langle E, V \rangle$ of queries
$E \leftarrow \emptyset$
$V \leftarrow \emptyset$
foreach *mgdq* $m \in C$ **do**
  foreach *investigation* i *having mgdq* m **do**
    $E \leftarrow E \cup queries(s)$
    `// arrange the queries of` i
    let Q be the queries of i in the following order:
      1. the mgdq
      2. the drilldown differences of q
      3. the exceptions to q
      4. the drilldown differences of m
      5. the exceptions to m
      6. the remaining queries of i
    `// remove the queries with pairs appearing in` sc
    $Q \leftarrow Q \setminus \{q \in queries(sc) | \exists c, c' \in q, \exists q' \in Q$ with $c, c' \in q'\}$
    `// update the vertices`
    foreach $j \in [1, |Q|[$ **do**
      $V \leftarrow V \cup \langle q_j, q_{j+1} \rangle$

Note that Algorithm 2 outputs what can be seen as a set of sessions. These sessions can be ordered for presentation to the user. The ordering can be computed based on how close the recommended sessions are from the current session. Algorithms for computing similarity between sessions are given in our earlier work [7, 8].

6. A COMPLETE EXAMPLE

In this section, we describe the use of the algorithms given in the previous section on the sessions described in Section 2. We begin by describing how Algorithm 1 processes each session of the log. The result of this step is synthesized in Figure 2.

In what follows, a pair of cells is detected as a difference pair if the measure of one is around ten times lower or higher than the measure of the other.

Session 1 is processed in the following way. First, four relevant difference pairs are detected. They concern the differences of sales of cheese for 2007 semester 1 with 2008 semester 1 (t_1), sales of cheese for 2007 semester 2 with 2008 semester 1 (t_2), and the differences of sales between cheese and milk (for milk 2007 semester 2 to cheese 2008 semester 1 (t_3) and for milk 2008 semester 1 to cheese 2008 semester 1 (t_4)). Note that there are other difference pairs (like e.g., the difference of butter and cheese for 2008 semester 1) but they do not give rise to drilldown differences, nor do they give rise to exceptions and thus they are not taken into con-

sideration (see last condition of Algorithm 1). Note also that q_2^1 is a drilldown of q_1^1 for 2008 semester 1 and 2. This will give rise to 4 investigations i_1, i_2, i_3, i_4 where q_1^1 is detected as the mgdq of each investigation, and q_2^1 is a drilldown difference of q_1^1. For i_1 the difference pair of the mgdq is t_1, for i_2 the difference pair of the mgdq is t_2, for i_3 the difference pair of the mgdq is t_3, and for i_4 the difference pair of the mgdq is t_4.

Session 2 is processed in the following way. The relevant difference pairs are the two cells of query q_2^2 (t_5), the cells $\langle cheese, France, 2007, 50\rangle, \langle cheese, France, 2008, 6\rangle$ of query q_3^2 (t_6) and the two cells of query q_5^2 (t_7). Note that there are other difference pairs (e.g., cells $\langle cheese, France, 2007, 50\rangle$ and $\langle cheese, Italy, 2007, 1\rangle$) but they are not taken into consideration due to the last condition of Algorithm 1. Starting with difference t_5, q_2^2 is detected as the mgdq of session 2, with q_3^2 and q_5^2 as drilldown differences and q_3^2 and q_4^2 as exceptions. A first investigation i_5 is created for session 2. Processing difference pairs t_6 and t_7 results in the same investigation i_5 since q_2^2 is detected as a rollup difference of both q_3^2 and q_5^2.

Finally session 3 is processed in the following way. No difference pair is detected in query q_1^3. One difference pair t_8 is detected in query q_2^3, namely the sales of cheese in 2008 is ten times less than the sales of cheese in 2007. No difference pair is detected in query q_3^3. Query q_2^3 is the mgdq of session 1, and query q_1^3 is detected as an exception to q_2^3 since the sales of a particular type of cheese (goat cheese) is greater in 2008 compared to 2007 (difference pair t_9). No drill down difference of q_2^3 is detected. Finally, session 3 gives rise to one investigation i_6, that is composed of q_1^3 labeled *exception* with the difference pair t_9 and of q_2^3 labeled *mgdq* with the difference pair t_8.

Once the sessions are processed with Algorithm 1, the mgdq of the mgdq of each investigation is computed. In our example, the set of the mgdq of the investigations contains 6 elements: q_2^3 for i_6, q_2^2 for i_5 and q_1^1 for each of the 4 investigations computed from session 1. The mgdq q_2^3 and q_2^2 are considered identical since their difference pair is the same. The investigations i_5 and i_6 are left unchanged since there is no mgdq in M that is an mgdq of their mgdq. The investigations i_1 and i_2 are modified: Their new mgdq (q_2^2) is added to the session. The investigations i_3 and i_4 are left unchanged.

Finally, the recommender system leverages these investigations to issue recommendations to the current user. First, Algorithm 2 detects 3 difference pairs in q. Each of this difference is a drilldown difference of q_2^3 (or q_2^2). Thus investigations i_1, i_2, i_5 and i_6 are considered for recommendation. Function *recommendDrilldown* arranges each of these investigations for presentation to the user (see Figure 1). For instance, i_6 is presented in the following order: First q_2^3, then q_1^3 then q_3^3. The recommended investigation i_5 presents first q_2^2, then q_3^2, q_5^2, q_4^2 and finally q_1^2. The recommended investigation i_1 or i_2 presents first q_2^2, then q_1^1 and finally q_1^2.

7. IMPLEMENTATION

In this section we briefly describe the implementation of the framework that we are currently doing. We are using Java and the Mondrian OLAP engine [15] to process and recommend sessions of MDX [13] queries.

In the framework a function d is used as a parameter of Algorithm 1 for detecting difference pairs. This can be done by testing, for a given query result, if the measures of every pair of cells differ significantly. Our implementation proposes two very basic functions for this test: The first one detects the pair $\langle c, c'\rangle$ as a difference pair if $measure(c)/measure(c') > \alpha$ for some threshold α. The second one detects the pair $\langle c, c'\rangle$ as a difference pair if $measure(c) - measure(c') > \alpha$ for some threshold α.

We are also considering more elaborated functions. For instance, it can be searched for difference pair along a particular dimension (commonly, the Time dimension) present in the query result. Also, the work of [3] can be adapted to detect the difference pairs.

The operators described in Section 4.5 for detecting the various type of difference queries are implemented with the RELAX and DIFF operators proposed by Sarawagi [18, 21]. We are using the Java implementation named iCube that is freely available for download [19]. It is to be noticed that, due to the lack of a standard Java API for OLAP, part of the implementation effort has been spent on the interoperability between Mondrian and iCube.

Recall from Section 3 that the RELAX and DIFF operators both output sets of cells. Thus they cannot implement directly the operators defined in Section 4.5. In our implementation, we use the function *detect* that, given a difference pair for a query q detects in a session s if there are rollup differences, drilldown differences or exceptions to the pair.

Function `detect`(q, c, c', op, Q)

Input: A query q, a pair of cells $\langle c, c'\rangle$, an operator *op*, a set of queries Q
Output: A set S of queries
foreach $(x, x') \in op(q, c, c')$ **do**
  foreach *query* $q' \in Q$ **do**
    if $(x, x') \in q'$ **then**
      $S \leftarrow S \cup \{q'\}$

In function *detect*, the *op* operator can be either DIFF or RELAX. For instance, if DIFF is used and Q is the set of queries of a session s, the call $detect(q, c, c', Q)$ detects the queries of $queries(s)$ that are drilldown difference queries of q w.r.t. $\langle c, c'\rangle$. In addition, the RELAX operator is used to detect both rollup differences and exceptions. Indeed, each time RELAX is invoked for collecting rollup differences, the returned exceptions are cumulated. In the future, we will check whether the work of [4] is an interesting alternative to the DIFF operator.

Finally note that processing the log requires the results of the queries. So far in our prototypical implementation, the queries of the log are resubmitted. A more sophisticated implementation should treat each query on the fly.

8. CONCLUSION

In this paper we propose a framework for recommending queries to support OLAP discovery driven analysis. The key idea is to infer from the log of the OLAP server what former users were investigating, and to use this information as a basis for helping the current user to navigate the cube. This framework is under implementation with Java and the Mondrian OLAP engine to recommend MDX queries. Completing an efficient implementation is the first of our future work.

Our long-term goal is to provide OLAP users and ad-

investigation	original session	difference pair	mgdq	drilldown queries	exceptions	other
i_1	session 1	$\langle cheese, 2007, all, 200\rangle$ $\langle cheese, 2008, all, 20\rangle$	$\{q_2^2\}$	$\{q_1^1, q_2^1\}$	$\emptyset$	$\emptyset$
i_2	session 1	$\langle cheese, 2007, all, 200\rangle$ $\langle cheese, 2008, all, 20\rangle$	$\{q_2^2\}$	$\{q_1^1, q_2^1\}$	$\emptyset$	$\emptyset$
i_3	session 1	$\langle milk, 2007sem1, France, 25\rangle$ $\langle cheese, 2008sem1, France, 1\rangle$	$\{q_1^1\}$	$\{q_2^1\}$	$\emptyset$	$\emptyset$
i_4	session 1	$\langle milk, 2007sem2, France, 25\rangle$ $\langle cheese, 2008sem1, France, 1\rangle$	$\{q_1^1\}$	$\{q_2^1\}$	$\emptyset$	$\emptyset$
i_5	session 2	$\langle cheese, 2007, all, 200\rangle$ $\langle cheese, 2008, all, 20\rangle$	$\{q_2^2\}$	$\{q_3^2, q_5^2\}$	$\{q_3^2, q_4^2\}$	$\{q_1^2\}$
i_6	session 3	$\langle cheese, 2007, all, 200\rangle$ $\langle cheese, 2008, all, 20\rangle$	$\{q_2^3\}$	$\emptyset$	$\{q_1^3\}$	$\{q_3^3\}$

Figure 2: Result of the processing of the log

ministrators with a platform for computing various types of recommendations. This platform will integrate the present framework with our earlier work [7, 8]. This platform should also include content-based techniques [5] as well as context-aware methods combined with user profiles [11, 2, 9]. We are working in that direction.

In addition to this, we will conduct experimentations on real data sets with feedback from users. This will allow not only to improve the overall quality of the recommended queries but also to determine to which context a particular approach for computing recommendations is adapted. To this end we are currently working with IRSA (a French social security health examination center) to analyze over 500.000 health care examination questionnaires.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[2] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for olap queries. In *DOLAP*, pages 9–18, 2005.

[3] V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F. Guisnel, and H. Klajnmic. Built-in indicators to automatically detect interesting cells in a cube. In *DaWaK*, pages 123–134, 2007.

[4] V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F. Guisnel, and H. Klajnmic. Built-in indicators to discover interesting drill paths in a cube. In *DaWaK*, pages 33–44, 2008.

[5] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, pages 3–18, 2009.

[6] D. Downey, S. T. Dumais, D. J. Liebling, and E. Horvitz. Understanding the relationship between searchers' queries and information goals. In *CIKM*, pages 449–458, 2008.

[7] A. Giacometti, P. Marcel, and E. Negre. A framework for recommending OLAP queries. In *DOLAP*, pages 73–80, 2008.

[8] A. Giacometti, P. Marcel, and E. Negre. Recommending multidimensional queries. In *DaWaK*, 2009.

[9] M. Golfarelli and S. Rizzi. Expressing olap preferences. In *SSDBM*, pages 83–91, 2009.

[10] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

[11] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Preference-based recommendations for olap analysis. In *DaWaK*, 2009.

[12] N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu. A case for a collaborative query management system. In *CIDR*, 2009.

[13] Microsoft Corporation. Multidimensional expressions (MDX) reference. Available at `http://msdn.microsoft.com/en-us/library/ms145506.aspx`, 2008.

[14] N. Parikh and N. Sundaresan. Inferring semantic query relations from collective user behavior. In *CIKM*, pages 349–358, 2008.

[15] Pentaho Corporation. Mondrian open source OLAP engine. Available at `http://mondrian.pentaho.org/`, 2009.

[16] C. Sapia. On modeling and predicting query behavior in OLAP systems. In *DMDW*, pages 2.1–2.10, 1999.

[17] C. Sapia. Promise: Predicting query behavior to enable predictive caching strategies for OLAP systems. In *DaWaK*, pages 224–233, 2000.

[18] S. Sarawagi. Explaining differences in multidimensional aggregates. In *VLDB*, pages 42–53, 1999.

[19] S. Sarawagi. I3: Intelligent, interactive inspection of cubes. Available at `http://www.cse.iitb.ac.in/~sunita/icube/`, 2009.

[20] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *EDBT*, pages 168–182, 1998.

[21] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *VLDB*, pages 531–540, 2001.

[22] M. Spiliopoulou, J. Srivastava, R. Kohavi, and B. M. Masand. Webkdd 2000 - web mining for e-commerce. *SIGKDD Explorations*, 2(2):106–107, 2000.