Similarity Measures for OLAP Sessions

Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, Elisa Turricchia

Under consideration for publication in Knowledge and Information Systems

Similarity Measures for OLAP Sessions

Julien Aligon[1], Matteo Golfarelli[2], Patrick Marcel[1], Stefano Rizzi[2] and Elisa Turricchia[2]

[1]Laboratoire d'Informatique – Université François Rabelais Tours, France;
[2]DISI – University of Bologna, Italy

Abstract. OLAP queries are not normally formulated in isolation, but in the form of sequences called OLAP sessions. Recognizing that two OLAP sessions are similar would be useful for different applications, such as query recommendation and personalization; however, the problem of measuring OLAP session similarity has not been studied so far. In this paper we aim at filling this gap. First, we propose a set of similarity criteria derived from a user study conducted with a set of OLAP practitioners and researchers. Then we propose a function for estimating the similarity between OLAP queries based on three components: the query group-by set, its selection predicate, and the measures required in output. To assess the similarity of OLAP sessions we investigate the feasibility of extending four popular methods for measuring similarity, namely the Levenshtein distance, the Dice coefficient, the tf-idf weight, and the Smith-Waterman algorithm. Finally, we experimentally compare these four extensions to show that the Smith-Waterman extension is the one that best captures the users' criteria for session similarity.

Keywords: OLAP; Similarity measures; Query comparison; Sequence comparison

1. Introduction and Motivation

The OLAP paradigm has revolutionized the way users access information in multidimensional databases. This paradigm achieves the ambitious goal of coupling a large querying expressiveness with a small query formulation effort, by providing a set of operators (such as drill-down and slice-and-dice) to transform one multidimensional query into another. As a consequence, OLAP queries are not normally formulated in isolation, but in the form of sequences (*OLAP sessions*). During an OLAP session focused on a phenomenon –such as sales– the user analyzes the results of a query and, depending on the specific data she

Received xxx
Revised xxx
Accepted xxx

sees, interactively chooses to apply one operator to determine a new query that will give her a better view of that phenomenon. The extemporary sequences of queries that are created this way are strongly related to the issuing user, to the analyzed phenomenon, and to the current data.

The capability of recognizing that two OLAP sessions are similar would be quite beneficial for different classes of applications, in particular:

- *Query recommendation.* Based on the current session and on the similar sessions that were issued in the past, the system suggests further queries to help users navigating the cube (Giacometti, Marcel and Negre, 2009).
- *Query personalization.* By comparing the current session with the similar ones, the system extracts a set of preferences to better focus the analysis process (Golfarelli, Rizzi and Biondi, 2011).
- *Query formulation support.* By comparing the current session with similar past sessions, the system either auto-completes the query that a non-expert user is currently writing (Khoussainova, Kwon, Balazinska and Suciu, 2010), or lets her browse similar past sessions for query reuse (Khoussainova, Kwon, Liao, Balazinska, Gatterbauer and Suciu, 2011).

The problem of measuring query similarity has been largely investigated in the literature, mostly in the contexts of information retrieval and collaborative filtering (see for instance Ögüdücü, 2010). Though some works are focused on assessing the similarity between OLAP queries (Aouiche, Jouve and Darmont, 2006; Golfarelli, 2003; Sapia, 2000), similarity of OLAP sessions has been only marginally taken into account. The similarity of sessions of SQL queries, disregarding order, is assessed by Chatzopoulou, Eirinaki, Koshy, Mittal, Polyzotis and Varman (2011). Aouiche et al. (2006) propose a basic measure for similarity between sets of OLAP queries (again disregarding query order) aimed at clustering a workload. Giacometti et al. (2009) compare OLAP sessions based on the order of queries, using edit distance, but at the extensional level —which may create efficiency problems. However, no systematic study exist to compare different similarity measures for OLAP sessions; in particular, though both Giacometti et al. (2009) and Chatzopoulou et al. (2011) aim at assisting the user, no users were apparently involved in the design of the similarity measures proposed.

The contributions we give to fill this gap can be summarized as follows:

1. We propose a set of criteria for OLAP sessions similarity derived from the results of a user study conducted with a set of practitioners and researchers in the OLAP field.
2. We propose a function for estimating the similarity between OLAP queries based on three components: the query group-by set, its selection predicate, and the measures required in output.
3. To assess the similarity of OLAP sessions we investigate the feasibility of two-level extensions (i.e., that compare query sequences based on the similarity between their elements) of four popular methods for measuring similarity, namely the Levenshtein distance, the Dice coefficient, the tf-idf weight, and the Smith-Waterman algorithm.
4. We experimentally compare these four extensions from both points of view of efficiency and effectiveness. The results clearly show that the Smith-Waterman extension is the one that best captures the users' criteria for session similarity.

Noticeably, our focus is on an *application-independent evaluation* of the exten-

sions proposed. For this reason, effectiveness tests will be based on the results of the user study and on a set of templates that model intuitive notions of what similar OLAP sessions might be. Evaluating our extensions with specific reference to the different applications outlined above is outside the scope of this paper.

The paper is structured as follows. Section 2 lists a number of requirements an approach for computing OLAP session similarity should have, and Section 3 uses these requirements to critically review the literature for candidate approaches. Section 4 introduces the formalization we adopt for the multidimensional model. Section 5 defines our query and session model and describes in detail the extensions we propose, while Section 6 compares them through a set of experimental tests. Finally, Section 7 draws the conclusions.

2. Requirements for OLAP Session Similarity

The goal of this section is to list a number of requirements to be used for (i) understanding which approaches, among all those proposed in the literature for query and sequence comparison, are eligible for the OLAP context; and (ii) driving the adaptation and extension of the eligible approaches towards the development of an original approach to OLAP session comparison.

We start by proposing a first set of requirements, suggested by the specific features of the OLAP context and by our experience in the field:

♯1 Multidimensional databases store huge amounts of data, and OLAP queries may easily return large volumes of results. Computing similarity at the extensional level, i.e., by comparing the data resulting from queries, would pose serious efficiency problems in this context, and would discourage the use of the approach for recommendation and personalization —that require a fast interaction with users. Indeed, as noted by Chatzopoulou et al. (2011) in the case of recommendation of SQL queries, there is a clear trade-off between efficiency and quality, when a fragment based model or a tuple based model is used. For this reason we compute similarity at the intensional level, i.e., considering only query expressions.

♯2 It is unlikely that two OLAP sessions share identical queries; this feature is better managed by having comparisons of single queries result in a score rather than in a Boolean.

♯3 A typical OLAP query is defined by the fact to be analyzed, one or more measures to be computed, a set of hierarchy levels for aggregating measure values, a predicate for filtering a subset of events, and a presentation. Though the presentation chosen for displaying the results of an OLAP query (e.g., a cross-tab or a pie-chart) certainly has an influence on how easily users can interpret these results, it does not affect the actual informative content, so it should not be considered when comparing queries.

To discover additional requirements for OLAP sessions similarity, we conducted a user study. We prepared a questionnaire asking to give a qualitative evaluation of the similarity between couples of OLAP queries and couples of OLAP sessions over a simple multidimensional schema (more details will be

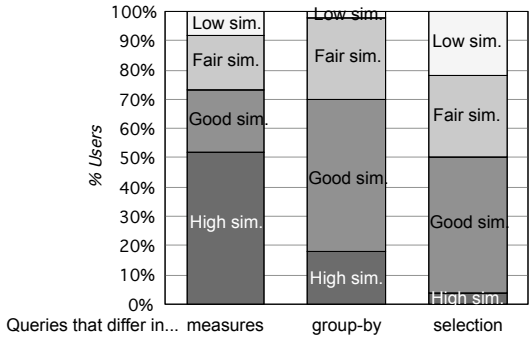

Fig. 1. Perceived similarities for OLAP queries only differing in one of their three main components

given in Section 6.1). The questionnaire[1] was submitted to all the teachers and PhD students of the First European Business Intelligence Summer School (eBISS 2011)[2], as well as to the master students of two specialistic courses on data warehouse design at the Universities of Bologna (Italy) and Tours (France). All people involved had some experience as OLAP users, most of them had some practice of multidimensional design too. Overall, 41 answers were collected. The additional requirements emerging from an analysis of the questionnaire results can be summarized as follows:

♯4 The selection predicate is the most relevant component in determining the similarity between two OLAP queries, followed by the group-by set. The less important component is the set of measures to be returned.

♯5 The order of queries is relevant in determining the similarity between two sessions, i.e., two sessions sharing the same queries but in different orders have low similarity.

♯6 Recent queries are more relevant than old queries in determining the similarity between two OLAP sessions. Since the time actually elapsed between two consequent queries in a session depends on several unpredictable factors (e.g., the query execution time, the size and complexity of the data returned, the user's query formulation skills), only the *order* of queries will be considered.

♯7 The longest the matching fraction of two sessions, the highest their similarity.

♯8 Two sessions that match with one or more gaps (i.e., one or more non-matching queries are present) are similar, but their similarity is lower than the one of two sessions that match with no gaps.

In particular, as to point ♯4, in Figure 1 we show the percentages of users that perceive a given level of similarity for couples of queries that only differ in either their measure sets, or their selection predicates, or their group-bys. Apparently, measures are the less important component in determining similarity since most users perceive as highly similar two queries that only differ in their measures. The opposite holds for the selection predicate component.

[1] Available at `http://www.julien.aligon.fr/recherche/similarityform.aspx`.
[2] `http://cs.ulb.ac.be/conferences/ebiss2011/`

3. A Critical Review of Similarity Approaches

This section reviews the literature for similarity functions that could possibly be used to compare OLAP sessions. Since OLAP sessions are sequences of queries, we first review the approaches for comparing sequences (Subsection 3.1) and then those for comparing database queries (Subsection 3.2). At the end of each subsection, the requirements expressed in Section 2 are used to restrict the set of approaches that are candidate to be adopted in the OLAP context.

3.1. Sequence Comparison Approaches

Comparing sequences has attracted a lot of attention especially in the context of string processing, with applications like information retrieval, spell-checkers, bioinformatics, and record linkage (Cohen, Ravikumar and Fienberg, 2003; Moreau, Yvon and Cappé, 2008). The existing approaches are inspired by different principles.

In *token-based approaches* sequences are treated as bags of elements, and classical set similarity functions like Jaccard and Hausdorff, and all their variants, can be used or adapted. Of course, these approaches are not sensible to the order of sequence elements. When the sequences to be compared are taken from a *corpus*, the popular *term frequency-inverse document frequency* (tf-idf) weight can be adopted, which weighs each element of a sequence using (positively) their frequency in the sequence and (negatively) their frequency in the corpus. A cosine is then used to measure the similarity between two vectors of weights.

Some approaches compare two sequences by comparing their subsequences. A basic approach here is to use the size of the longest common subsequence (LCS).[3] An approach often used in statistical natural language processing relies on n-grams, i.e., substrings of size n of a given sequence (Brown, Pietra, de Souza, Lai and Mercer, 1992). A popular similarity function using n-grams is the *Dice coefficient*, an extension of the Jaccard index defined as twice the number of shared n-grams over the total number of n-grams:

$$Sim_{Dice}(s, s') = \frac{2|ngrams(s) \cap ngrams(s')|}{|ngrams(s)| + |ngrams(s')|}$$

Other approaches compare sequences based on their *edit distance*, i.e., in terms of the cost of the atomic operations necessary to transform one sequence into another. Many edit distances have been proposed that differ on the number, type, and cost of the edit operations. The most popular are the *Levenshtein distance*, that allows insert, delete, and substitute, and the *sequence alignment distance*, that allows match, replace, delete, and insert (Cohen et al., 2003; Navarro, 2001).

Finally, in *two-level* approaches sequences are compared based on the similarity between their elements. A simple example is the Hausdorff distance between sets, that relies on the distance between elements of the set. In (Monge and Elkan, 1997) the similarity between sequences s and s' is the average of the

[3] Note that, while substrings are consecutive parts of a string, subsequences need not be.

highest similarities between pairs of elements of s and s':

$$Sim_{M\&E}(s, s') = \frac{1}{|s|} \sum_{s_i \in s} max_{s'_j \in s'} \{Sim_{elem}(s_i, s'_j)\}$$

where Sim_{elem} measures the similarity between single elements. In *soft tf-idf* (Cohen et al., 2003), the tf-idf weight is extended using the similarity of sequence elements; more precisely,

$$Sim_{soft}(s, s') = \sum_{s_i \in Close_\theta(s,s')} T(s_i, s) \cdot T(s_i, s') \cdot max_{s'_j \in s'} \{Sim_{elem}(s_i, s'_j)\}$$

where $T(s_i, s)$ is a normalized form of the tf-idf of element s_i within sequence s, θ is a threshold, and $Close_\theta(s, s')$ is the set of elements $s_i \in s$ such that there is at least an element $s'_j \in s'$ with $Sim_{elem}(s_i, s'_j) > \theta$. While the two previous two-level approaches do not consider the ordering of elements within sequences, the *Smith-Waterman algorithm* relies on element ordering; it can be used to efficiently find the best alignment between subsequences of two given sequences by ignoring the non-matching parts of the sequences (Smith and Waterman, 1981). It is a dynamic programming algorithm based on a matrix H whose value in position (i, j) expresses the score for aligning subsequences of s and s' that end in elements s_i and s'_j, respectively. This matrix is recursively defined based on the following formula:

$$H(i,j) = max \left\{ \begin{array}{c} 0; \\ H(i-1, j-1) + Sim_{elem}(s_i, s'_j); \\ max_{k \geq 1}\{H(i-k, j) - cost_k\}; \\ max_{k \geq 1}\{H(i, j-k) - cost_k\} \end{array} \right\}$$

where $cost_k$ is the cost of introducing a gap of length k in the matching between s and s'. Note that, here, the similarity between two elements can be negative, to express that there is a mismatch between them; intuitively, the algorithm seeks an optimal trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a poorly matching pair of elements.

We conclude this overview with a couple of brief observations about the features a sequence comparison approach should have to be used for OLAP sessions:

- In OLAP sessions, the order of queries is relevant (requirement ♯5), which discourages from taking token-based approaches.
- Mostly, OLAP sessions do not share the very same queries (requirement ♯2). This makes two-level approaches, that take advantage of a similarity function for OLAP queries, more suitable for our purposes.
- Following requirement ♯8, it is important to be able to determine similar regions in two globally different sessions, which favors a sequence alignment approach.

3.2. Query Comparison Approaches

We can distinguish two main motivations for comparing database queries. The first one is query optimization, where a query q to be evaluated is compared to another query q', with the goal of finding a better way of evaluating q. This

motivation attracted a lot of attention, and covers classical problems like view usability (Garcia-Molina, Ullman and Widom, 2008; Gupta and Mumick, 1999), query containment (Abiteboul, Hull and Vianu, 1995), plan selection (Ghosh, Parikh, Sengar and Haritsa, 2002), view selection (Aouiche et al., 2006; Golfarelli, 2003), and data prefetching (Sapia, 2000). The second, more recent, motivation is to suggest a query to the user without focusing on its evaluation. In this context, a query is compared to another one with the goal of helping the user exploring or analyzing a database. This includes query completion (Yang, Procopiuc and Srivastava, 2009) and query recommendation (Stefanidis, Drosou and Pitoura, 2009; Drosou and Pitoura, 2011; Chatzopoulou, Eirinaki and Polyzotis, 2009; Chatzopoulou et al., 2011; Akbarnejad, Chatzopoulou, Eirinaki, Koshy, Mittal, On, Polyzotis and Varman, 2010; Giacometti et al., 2009).

From a technical point of view, the approaches found in the literature can be classified according to (i) the *query model* they adopt, i.e., the structure used to compactly represent queries; (ii) the *information source* from which the representation of each query is derived; and (iii) the *function* used to compute similarity.

Query models range from a string corresponding to the uninterpreted SQL sentence (Yao, An and Huang, 2005) to the set of tuples resulting from the query evaluation (Stefanidis et al., 2009; Drosou and Pitoura, 2011). Queries can also be modeled as vectors of features with either a score or a Boolean for each feature (Akbarnejad et al., 2010; Agrawal, Rantzau and Terzi, 2006; Aouiche et al., 2006; Ghosh et al., 2002), or as sets of *fragments*, each representing a particular part of the query, such as the attributes required in output (SELECT clause) or the table names in the cross product (FROM clause) (Sapia, 2000; Aligon, Golfarelli, Marcel, Rizzi and Turricchia, 2011). Finally, queries are sometimes modeled as graphs, following the database schema like in (Yang et al., 2009).

As to the information source, it can be the query expression, e.g., the uninterpreted query text (Yao et al., 2005) or the list of query fragments (selection predicates, projection, etc.) (Garcia-Molina et al., 2008; Yang et al., 2009). When fragments are used, only some of them may be taken into account; for instance, only the selection attributes are used by Agrawal et al. (2006) and Yang et al. (2009) whereas all fragments are used by Garcia-Molina et al. (2008) and Gupta and Mumick (1999). The information source can also be related to the database queried; more precisely, it can be:

- The database instance, e.g., the query result or the active domain of the database attributes (Agrawal et al., 2006; Chatzopoulou et al., 2009; Chatzopoulou et al., 2011; Giacometti et al., 2009; Stefanidis et al., 2009; Drosou and Pitoura, 2011). In the former case, the query can be evaluated either fully (Stefanidis et al., 2009; Drosou and Pitoura, 2011) or partially (Giacometti et al., 2009). In this category we also include an approach for measuring similarity between multidimensional cubes (Baikousi, Rogkakos and Vassiliadis, 2011), because obviously an OLAP query returns a multidimensional cube.
- The statistics used by the query optimizer, like table sizes and attribute cardinalities (Ghosh et al., 2002).
- The database schema, e.g., the keys defined or the index used to process a selection (Ghosh et al., 2002; Golfarelli, 2003).
- The query log, if the query model relies on other queries that have previously been launched on the same database. For instance, Chatzopoulou et al. (2009),

Table 1. Query comparison approaches at a glance

Ref.	*Motivation*	*Model*	*Source*	*Similarity Function*
Gupta and Mumick (1999)	optimization	sets	S, P, C	fragment tests
Chatzopoulou et al. (2011)	recommend.	vector	db instance, log	cosine
Akbarnejad et al. (2010)	recommend.	vector	S, P, log	cosine
Agrawal et al. (2006)	optimization	vector	S, db instance	cosine
Aouiche et al. (2006)	optimization	vector	S, P, log	Hamming distance
Ghosh et al. (2002)	optimization	vector	S, C, db statistics	Hamming distance
Stefanidis et al. (2009) (1)	recommend.	vector	log	inner product
Stefanidis et al. (2009) (2)	recommend.	set	db instance	Jaccard index
Giacometti et al. (2009)	recommend.	set	db instance	Hausdorff distance
Sapia (2000)	optimization	sets	S, P	query repres. equality
Golfarelli (2003)	optimization	set	P, db schema & statistics	group-by lattice
Yao et al. (2005)	recommend.	string	SQL sentence	entropy
Yang et al. (2009)	recommend.	graph	S, P, C	query repres. equality

Chatzopoulou et al. (2011), Akbarnejad et al. (2010), Aouiche et al. (2006), and Stefanidis et al. (2009) model a query in terms of its links with other queries or how many times it appears in the log.

Finally, the result of query comparison can be a Boolean or a score, usually normalized in the [0..1] interval. The first case applies when queries are tested for equivalence (Abiteboul et al., 1995) or view adaptation (Gupta and Mumick, 1999), or when the goal is to group queries based on some criteria (Sapia, 2000; Yang et al., 2009). In this case, the comparison can be a simple equality test of the query representations (Sapia, 2000; Yang et al., 2009) or it can be based on separate tests of query fragments (Gupta and Mumick, 1999). In the second case, the comparison is normally based on classical functions applied to the query representations. For instance, if the query is modeled as a vector, cosine (Agrawal et al., 2006; Akbarnejad et al., 2010; Chatzopoulou et al., 2009; Chatzopoulou et al., 2011), inner product (Stefanidis et al., 2009), or Hamming distance (Aouiche et al., 2006) can be used; if the query is modeled as a set, the Jaccard index (Stefanidis et al., 2009) or the Hausdorff distance (Giacometti et al., 2009) can be used. Sometimes, more sophisticated similarity functions are used. For instance, Yao et al. (2005) use a measure based on entropy to cluster queries modelled as strings. In (Golfarelli, 2003), similarity between OLAP queries is computed based on the relative position of the query group-by sets within the group-by lattice.

Table 1 summarizes the approaches reviewed in this section. Note that Stefanidis et al. (2009) propose two ways of comparing queries: (1) based on the frequency of the query in the log, and (2) based on the query result. Letters S, P, and C indicate the fragments used by the approach (S for selection, P for generalized projection —including the group-by set and the aggregation operator—, and C for cross-product).

We conclude this overview with some brief observations about the features a query comparison approach should have to be used for OLAP queries:

– Following requirement ♯1, we solely rely on query expressions to derive query representations. Then we exclude the approaches based on query evaluation (Giacometti et al., 2009; Stefanidis et al., 2009; Drosou and Pitoura, 2011), those depending on database instances (Chatzopoulou et al., 2009; Chatzopoulou et al., 2011; Agrawal et al., 2006; Baikousi et al., 2011), and those

using query logs (Aouiche et al., 2006; Akbarnejad et al., 2010; Stefanidis et al., 2009).
- Our goal is not query optimization, so we drop the approaches aimed at optimization like Ghosh et al., 2002. In that particular work, the idea is to reuse execution plans, that heavily rely on "physical" properties (like statistics and presence of indexes); thus, query similarity is more related to how queries are evaluated than to what they mean to users. This means that two queries that should be very similar for our purposes could be found to be very dissimilar using that approach if their execution plans are different (for instance, if one has a WHERE clause and the other does not).
- According to requirement ♯2, query comparison should result in a score. So, Boolean approaches like Gupta and Mumick, 1999 and Yang et al., 2009 are less relevant in our context.
- OLAP queries are expressed using a friendly visual interface, and the syntax of the underlying query language (e.g., MDX) is typically transparent to users. This discourages the adoption of uninterpreted approaches like Yao et al., 2005.
- According to requirement ♯3, the OLAP semantics is carried by a number of different components (e.g., the aggregation level), which encourages the adoption of a fragment-based query model like in Sapia, 2000, also taking into account the peculiarities of the multidimensional model like in Golfarelli, 2003.

Among the query similarity functions proposed in the OLAP area, the one that captures the above requirements at best is Aouiche et al., 2006. In that approach, similarity between queries q and q' is based on the number of attributes they share within their SELECT, WHERE, and GROUP-BY clauses; the normalized form we adopt here for comparison purposes (Section 6.1) is

$$\sigma_{AJD}(q, q') = \frac{|L \cap L'|}{|L \cup L'|}$$

where L and L' are the attributes appearing in q and q', respectively.

4. Formal Background

In this section we define the multidimensional model we will use to formalize our approach and introduce a working example. For simplicity, we will define hierarchies as total orders instead of partial orders, i.e., we will assume hierarchies have no branches.[4]

Definition 4.1 (Multidimensional Schema). A *multidimensional schema* (or, briefly, a *schema*) is a triple $\mathcal{M} = \langle L, H, M \rangle$ where:

- $L = \{l_1, \ldots l_p\}$ is a finite set of *levels*, i.e., categorical attributes;
- $H = \{h_1, \ldots, h_n\}$ is a finite set of *hierarchies*, each characterized by (1) a subset $Lev(h_i) \subseteq L$ of levels and (2) a *roll-up* total order $\succeq_{h_i}$ of $Lev(h_i)$;

[4] While this enables a simpler formalization for group-by sets (see Definition 4.2), it does not significantly impact on the overall approach. Indeed, partially ordered hierarchies could be easily dealt with by extending Definition 5.4 to measure the distance between two group-by sets on the multidimensional lattice as suggested by Golfarelli (2003).

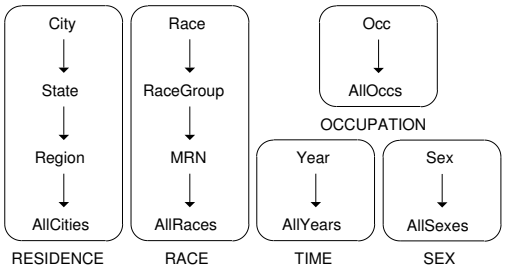

Fig. 2. Roll-up orders for the five hierarchies in the CENSUS schema (MRN stands for MajorRacesNumber)

– $M = \{m_1, \ldots, m_l\}$ is a finite set of *measures*, i.e., numerical attributes.

For each hierarchy h_i, the bottom level is denoted by ALL_i, has a single possible value, and determines the coarsest aggregation.

A group-by set includes one level for each hierarchy, and defines a possible way to aggregate data.

Definition 4.2 (Group-by Set). Given schema $\mathcal{M} = \langle L, H, M \rangle$, let $Dom(H) = Lev(h_1) \times \ldots \times Lev(h_n)$; each $g \in Dom(H)$ is called a *group-by set* of $\mathcal{M}$.

Example 4.1. IPUMS is a public database storing census microdata for social and economic research (Minnesota Population Center, 2008). Its CENSUS multidimensional schema has five hierarchies, namely RACE, TIME, SEX, OCCUPATION, and RESIDENCE, and measures AvgIncome, AvgCostGas, AvgCostWtr, and AvgCostElect. It is City $\succeq_{\text{RESIDENCE}}$ State (the complete roll-up orders are shown in Figure 2); examples of group-by sets are:

$g_1 = \langle$State, Race, Year, AllSexes, Occ$\rangle$
$g_2 = \langle$State, RaceGroup, Year, AllSexes, Occ$\rangle$
$g_3 = \langle$Region, AllRaces, Year, Sex, Occ$\rangle$
$g_4 = \langle$AllCities, AllRaces, AllYears, AllSexes, AllOccs$\rangle$

The last group-by set specifies total aggregation. □

5. Measuring OLAP Session Similarity

As observed in Section 3, OLAP session comparison should rely on a two-level approach that compares query sequences based on the similarity between their elements. Consistently with this, after proposing in Subsection 5.1 our query and session model, in Subsection 5.2 we separately discuss the function we adopt for computing query similarity. Then we propose four techniques for computing session similarity. The first one (Subsection 5.3) is an extension of the Levenshtein distance, and considers the atomic operations necessary to transform one session into another. The second one (Subsection 5.4) is an extension of the Dice coefficient, and is based on the number of common subsequences shared by two sessions. The third one (Subsection 5.5) is an extension of the soft tf-idf method and is based on the relative importance of queries within a corpus —in

our case, the log of OLAP sessions. The last one (Subsection 5.6) is an extension of the Smith-Waterman algorithm and considers the cost for aligning common subsequences in two sessions.

5.1. Model

We consider a basic form of OLAP query centered on a single schema and characterized by an aggregation and a selection expressed through a conjunctive predicate. To be independent of the details related to logical design of multidimensional schemata and to specific query plans, we express queries using an abstract syntax.[5] Following the observations made in Section 3, we opt for a fragment-based query model with three components:

Definition 5.1 (OLAP Query). A *query* on schema $\mathcal{M} = \langle L, H, M \rangle$ is a triple $q = \langle g, P, Meas \rangle$ where:

1. $g \in Dom(H)$ is the query group-by set;
2. $P = \{c_1, \ldots, c_n\}$ is a set of Boolean clauses, one for each hierarchy, whose conjunction defines the *selection predicate* for q; conventionally, $c_i = TRUE_i$ if no selection on h_i is made in q;
3. $Meas \subseteq M$ is the measure set whose values are returned by q.

An OLAP session is an ordered sequence of correlated queries formulated by a user on a schema; typically (but not necessarily), each query in a session is derived from the previous one by applying an OLAP operator (such as roll-up, drill-down, and slice-and-dice).

Definition 5.2 (OLAP Session). An *OLAP session* of length v is a sequence $s = \langle q_1, \ldots, q_v \rangle$ of v queries on schema $\mathcal{M}$.

Example 5.1. All the examples in this section will be based on a simple log that consists of three sessions:

$$s = \langle q_1, q_2, q_3 \rangle$$
$$s' = \langle q_4, q_5, q_6, q_7, q_8 \rangle$$
$$s'' = \langle q_9, q_{10} \rangle$$

Table 2 represents each query in terms of our query model; the involved group-by sets are those used in Example 4.1, while the selection predicates are:

$$P_1 = \{TRUE_{\mathsf{RESIDENCE}}, TRUE_{\mathsf{RACE}}, (\mathsf{Year} = 2005), TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$
$$P_2 = \{TRUE_{\mathsf{RESIDENCE}}, (\mathsf{RaceGroup} = Chinese), TRUE_{\mathsf{TIME}}, TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$
$$P_3 = \{TRUE_{\mathsf{RESIDENCE}}, (\mathsf{RaceGroup} = Chinese), (\mathsf{Year} = 2005), TRUE_{\mathsf{OCCUPATION}}, TRUE_{\mathsf{SEX}}\}$$

[5] In a relational implementation, a multidimensional schema is translated into a star schema; in this case, the queries we consider can be classified as *GPSJ - Generalized Projection / Selection / Join* queries (Gupta, Harinarayan and Quass, 1995), based on a star join between the fact table and the dimension tables.

Table 2. Queries for Example 5.1

		Queries									
		q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
Group-by set		g_1	g_2	g_2	g_2	g_2	g_3	g_3	g_2	g_1	g_1
Measures	AvgCostWtr	✓	✓	✓	✓	✓	✓	✓	✓		
	AvgCostElect	✓	✓	✓		✓	✓		✓		
	AvgCostGas			✓						✓	✓
	AvgIncome							✓	✓		✓
Selection predicates		P_1	P_1	P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

The MDX formulation of query q_4 is:

```
SELECT AvgCostWtr ON COLUMNS,
      Crossjoin(OCCUPATION.Occ.members,
            Crossjoin(TIME.Year.members,RESIDENCE.State.members)) ON ROWS
FROM CENSUS WHERE RACE.RaceGroup.[Chinese]
```

□

5.2. Query Similarity

In this section we define the similarity function used in our two-level approach to compare OLAP queries. As remarked in the Section 3.2, this function must consider the peculiarities of the multidimensional model, be computable based on query expressions only, and result in a score. Consistently with Definition 5.1, the function we propose is a combination of three components: one related to group-by sets, one to selection predicates, and one to measure sets.

To define group-by set similarity, we first introduce the notion of distance between levels in a hierarchy.

Definition 5.3 (Distance between hierarchy levels). Let $\mathcal{M} = \langle L, H, M \rangle$ be a schema, $h_i \in H$ be a hierarchy, and $l, l' \in Lev(h_i)$ be two levels. The *distance* between l and l', $Dist_{lev}(l, l')$, is the difference between the positions of l and l' within the roll-up order $\succeq_{h_i}$.

Definition 5.4 (Group-by set similarity). Let q and q' be two queries, both on schema $\mathcal{M}$, with group-by sets g and g', respectively, and let $g.h_i$ ($g'.h_i$) denote the level of h_i included in g (g'). The *group-by set similarity* between q and q' is

$$\sigma_{gbs}(q, q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{lev}(g.h_i, g'.h_i)}{|Lev(h_i)|-1}}{n}$$

where n is the number of hierarchies in $\mathcal{M}$.

Our definition of selection similarity takes into account both the levels and the constants that form the selection predicates. In particular, for each hierarchy, two identical clauses are given maximum similarity, and non-identical clauses are given decreasing similarities according to the distance between the hierarchy levels they are expressed on.

Definition 5.5 (Distance between selection clauses). Let $\mathcal{M} = \langle L, H, M \rangle$

be a schema, and c_i and c'_i be two selection clauses over hierarchy $h_i \in H$. Let $c_i.h_i \in Lev(h_i)$ denote the level of h_i involved in c_i (conventionally, $TRUE_i.h_i = ALL_i$). The *distance* between c_i and c'_i is

$$Dist_{clau}(c_i, c'_i) = \begin{cases} 0, \text{ if } c_i = c'_i; \\ Dist_{lev}(c_i.h_i, c'_i.h_i) + 1, \text{ otherwise} \end{cases}$$

According to this definition, the distance between two selection clauses on h_i is 0 if they are expressed on the same level and the same constant, 1 if they are defined on the same level but not on the same constant, greater than 1 if they are defined on different levels.

Definition 5.6 (Selection similarity). Let q and q' be two queries, both on schema $\mathcal{M}$, with selection predicates P and P', respectively, with $P = \{c_1, \ldots, c_n\}$ and $P' = \{c'_1, \ldots, c'_n\}$. The *selection similarity* between q and q' is

$$\sigma_{sel}(q, q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{clau}(c_i, c'_i)}{|Lev(h_i)|}}{n}$$

Finally, to define the measure similarity, we use the Jaccard index.

Definition 5.7 (Measure similarity). Let q and q' be two queries, both on schema $\mathcal{M}$, with measure sets $Meas$ and $Meas'$, respectively. The *measure similarity* between q and q' is

$$\sigma_{meas}(q, q') = \frac{|Meas \cap Meas'|}{|Meas \cup Meas'|}$$

We can now define the similarity between two OLAP queries as the weighted average of the three similarity components defined above.

Definition 5.8 (Similarity of OLAP queries). Let q and q' be two queries, both on schema $\mathcal{M}$. The *similarity* between q and q' is

$$\sigma_{que}(q, q') = \alpha \cdot \sigma_{gbs}(q, q') + \beta \cdot \sigma_{sel}(q, q') + \gamma \cdot \sigma_{meas}(q, q')$$

where α, β, and γ are normalized to 1.

Example 5.2. The similarity between queries q_1 and q_4 of Example 5.1 is computed as follows:

$$\begin{aligned} \sigma_{gbs}(q_1, q_4) &= 1 - \frac{(0/3 + 1/3 + 0/1 + 0/1 + 0/1)}{5} = 0.933 \\ \sigma_{sel}(q_1, q_4) &= 1 - \frac{(0/4 + 3/4 + 2/2 + 0/2 + 0/2)}{5} = 0.650 \\ \sigma_{meas}(q_1, q_4) &= \frac{1}{2} = 0.500 \\ \sigma_{que}(q_1, q_4) &= 0.694 \end{aligned}$$

(assuming for simplicity $\alpha = \beta = \gamma = 0.333$). The overall query similarities for sessions s and s' are summarized in Table 3. □

5.3. Edit-Based Session Similarity

The Levenshtein distance compares two strings in terms of the cost of the atomic operations (typically insertion, deletion, and substitution of a character) neces-

Table 3. Query similarities for Example 5.2

	q_4	q_5	q_6	q_7	q_8
q_1	0.694	0.927	0.844	0.622	0.866
q_2	0.716	0.950	0.866	0.644	0.888
q_3	0.661	0.838	0.755	0.616	0.833

sary to transform one string into another (Ristad and Yianilos, 1998). Given two strings s and s' of v and v' characters, respectively, a $(v+1) \times (v'+1)$ distance matrix D of reals is recursively defined in terms of the deletion, insertion, and substitution costs; the Levenshtein distance between s and s' is found in the bottom-right cell of D, that represents the minimum sum of the operation costs to transform s in s'.

In the traditional formulation, an operation is applied in absence of a perfect match (i.e., of an identity) between the compared characters. In our case this is too restrictive, because OLAP queries are complex objects whose match is not effectively captured by identity (see requirement ♯2). So we consider two queries as matching when their similarity is above a given threshold θ, and we apply a transformation operation when the similarity is under θ. Besides, we normalize distances using the length of the longest of the two sessions involved, so that the cost of a single mismatch is lower for longer sessions.

Definition 5.9 (Edit-Based Similarity of OLAP Sessions). Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively. Given a matching threshold θ, the distance matrix for s and s' is a $(v+1) \times (v'+1)$ matrix D_θ of reals recursively defined as follows:

$$D_\theta(i,j) = \begin{cases} 0, \textit{ when } i = 0 \textit{ or } j = 0 \\ D_\theta(i-1, j-1), \textit{ when } i,j > 0 \textit{ and } \sigma_{que}(s_i, s'_j) \geq \theta \\ min \left\{ \begin{array}{c} D_\theta(i-1,j)+1; \\ D_\theta(i,j-1)+1; \\ D_\theta(i-1,j-1)+1 \end{array} \right\}, \textit{ when } i,j > 0 \textit{ and } \sigma_{que}(s_i, s'_j) < \theta \end{cases}$$

where s_i is the *i-th* query of session s. The *edit-based similarity* between s and s' is:

$$\sigma_{edit}(s, s') = 1 - \frac{D_\theta(v, v')}{max\{v, v'\}}$$

Note that, like in most applications of the Levenshtein distance, all transformation costs are set to 1.[6] As to complexity of this function, in the general case it is $O(v \cdot v')$ where v and v' are the lengths of the two sessions (Wagner and Fischer, 1974).

Example 5.3. With reference to Example 5.1 and using $\theta = 0.7$, the minimum cost to transform s' to s is obtained by matching queries as follows: $\langle q_1, q_5 \rangle, \langle q_2, q_6 \rangle, \langle q_3, q_8 \rangle$ and deleting q_4 and q_7. Thus, it is $\sigma_{edit}(s, s') = 1 - \frac{2}{5} = 0.60$. □

[6] In the formula, the three rows of the *min* argument deal with deletions, insertions, and substitutions, respectively.

5.4. Subsequence-Based Session Similarity

An n-gram is a substring of size n of a given string (Brown et al., 1992). A popular string similarity function based on n-grams is the *Dice coefficient*, an extension of the Jaccard index defined as twice the number of shared n-grams over the total number of n-grams in the two strings.

In the OLAP context, the concept of "shared" n-grams becomes that of "similar" n-grams. Two n-grams r and r' are similar if their queries are pairwise similar, i.e., if their similarity is above threshold θ. To ensure symmetry while being consistent with the original definition, in our two-level extension similarity is defined as follows.

Definition 5.10 (Subsequence-Based Similarity of OLAP Sessions). Let s and s' be two OLAP sessions on schema $\mathcal{M}$, and $n \geq 1$. Given a matching threshold θ, the *subsequence-based similarity* between s and s' is

$$\sigma_{sub}(s, s') = \frac{2 \times min\{|SNgram_\theta(s, s')|, |SNgram_\theta(s', s)|\}}{|Ngram(s)| + |Ngram(s')|}$$

where $Ngram(s)$ is the set of n-grams of s and $SNgram_\theta(s, s') \subseteq Ngram(s)$ is the set of n-grams of s that have a similar n-gram in s':

$$SNgram_\theta(s, s') = \{r \in Ngram(s) | \exists r' \in Ngram(s'), \sigma_{que}(r_i, r'_i) \geq \theta \; \forall i = 1, \ldots, n\}$$

The complexity of this function is that of finding the n-grams of the two sessions, which is $O(v)$ (where v is the length of the longest one), plus that of computing the sets $SNgram_\theta(s, s')$, which is $O((v - n)^2)$.

Example 5.4. Applying the above definition to Example 5.1, with n=1, we obtain $\sigma_{sub}(s, s') = \frac{2 \times min\{1,2\}}{1+2} = 0.67$. □

5.5. Log-Based Session Similarity

In the tf-idf approach, the similarity between two sets of tokens (in information retrieval applications, tokens are lemmas and sets of tokens are documents) depends on both the frequency of each token in the sets and its frequency in a corpus. In our context, this approach can be adopted if the OLAP sessions to be compared are taken from a log, to penalize the non-distinctive queries (i.e., those that are more frequent in the log) when assessing similarity.

To propose an extension of the tf-idf method we start by applying the definition of *soft tf-idf* given by Moreau et al. (2008):

$$Sim_{soft}(s, s') = \sum_{s_i \in Close_\theta(s,s')} T(s_i, s) \cdot T(s'_{j_i}, s') \cdot \sigma_{que}(s_i, s'_{j_i})$$

where θ is a threshold,

$$Close_\theta(s, s') = \{s_i \in s | \exists s'_j \in s', \sigma_{que}(s_i, s'_j) > \theta\},$$

$$T(s_i, s) = \frac{tfidf(s_i, s)}{\sqrt{\sum_{s_k} tfidf(s_k, s)^2}},$$

$$tfidf(s_i, s) = tf(s_i, s) \cdot idf(s_i, s) = \frac{n_{s_i,s}}{|s|} \cdot log \frac{|L|}{|\{s \in L | s_i \in s\}|},$$

$$s'_{j_i} = argmax_{s'_j \in s'} \{\sigma_{que}(s_i, s'_j)\},$$

$n_{s_i,s}$ is the number of times s_i appears in s, and L is the set of OLAP sessions in the log. Intuitively, $Close_\theta(s, s')$ is the set of queries in sessions s that have some similarity to a query in session s'; $tfidf(s_i, s)$ is directly proportional to the frequency of query s_i in session s and inversely proportional to the frequency of s_i in the log L ($tfidf(s_i, s) = 0$ when all session in L include s_i); $T(s_i, s)$ is a normalized form of $tfidf(s_i, s)$; s'_{j_i} is the query in s' that is most similar to s_i.

This definition cannot be immediately used in our case for the following reasons:

1. It uses the "crisp" definition of tf-idf in the definition of T whereas in our case, given that it is unlikely to find the same query twice in an OLAP log, a "soft" version (i.e., one based on query similarity) should be used instead.
2. The soft tf-idf is not symmetric, which is not desirable for a similarity function.
3. There may be more than one query s'_{j_i} in s' that maximizes σ_{que} with s_i, which may not be relevant in the context of named entity matching (Moreau et al., 2008), but is definitely relevant in the OLAP context.
4. As pointed out by Moreau et al. (2008), there is a problem with counting that makes the similarity not normalized.

To cope with the first issue, we inject the similarity σ_{que} in the definition of tf-idf. By replacing equality with similarity, a two-level tf-idf can be computed as:

$$tfidf_2(s_i, s) = \frac{|Close_\theta(s_i, s)|}{\sum_{s_k \in Q} |Close_\theta(s_k, s)|} \cdot log \frac{|L|}{|\{s \in L | Close_\theta(s_i, s) \neq \emptyset\}|}$$

where Q is the set of all queries in L and $Close_\theta(s_i, s)$ is the set of queries of s that are similar to s_i.

Symmetry can be achieved by modifying the definition of similarity to work on pairs of queries, each relating a query in one session with one of its closest queries in the other session. This set of pairs is defined by:

$$R_\theta(s, s') = \{\langle s_i, s'_k \rangle | s_i \in s, s'_k \in Closest_\theta(s_i, s'))\} \cup \{\langle s_l, s'_j \rangle | s'_j \in s', s_l \in Closest_\theta(s'_j, s)\}$$

where $Closest_\theta(s_i, s)$ is the set of queries of s that have maximum similarity with s_i. Note that a query in a session appears more than once in $R_\theta(s, s')$ if there is more than one query in the other session with maximum similarity. This solves the third issue.

Finally, to cope with the fourth issue, the similarity is computed as the cosine of the two vectors obtained by taking the $tfidf_2$ of all the first (respectively, second) queries of the pairs.

Definition 5.11 (Log-Based Similarity of OLAP Sessions). Let s and s' be two OLAP sessions on schema $\mathcal{M}$. The *log-based similarity* between s and s' is

$$\sigma_{log}(s, s') = \sum_{\langle s_i, s'_j \rangle \in R_\theta(s,s')} T_2(s_i, s, s') \times T_2(s'_j, s', s) \times \sigma_{que}(s_i, s'_j)$$

where

$$T_2(s_i, s, s') = \frac{tfidf_2(s_i, s)}{\sqrt{\sum_{\langle s_i, s'_j \rangle \in R_\theta(s,s')} tfidf_2(s_i, s)^2 + \sum_{Closest_\theta(s_i, s')=\emptyset} tfidf_2(s_i, s)^2}}$$

$$T_2(s'_j, s', s) = \frac{tfidf_2(s'_j, s')}{\sqrt{\sum_{\langle s_i, s'_j \rangle \in R_\theta(s,s')} tfidf_2(s'_j, s')^2 + \sum_{Closest_\theta(s'_j, s)=\emptyset} tfidf_2(s'_j, s')^2}}$$

The complexity of this function should obviously be expressed not only in terms of the sessions to be compared but also in terms of the size of the log; it turns out that the complexity of computing $R_\theta(s, s')$ is $O(v^2)$, while that for computing all the $tfidf_2$ terms it is $O(v \times |Q|)$ where v the length of the longest session in the log.

Note that, as any cosine similarity, σ_{log} can be easily turned into the angle distance $arcos(\sigma_{log})$, which is a metric (Bustos and Skopal, 2011).

Example 5.5. With reference to Example 5.1, we focus on computing the log-based similarity between s and s'. The set of query pairs used in the computation of $\sigma_{log}(s, s')$ is $R_{0.7}(s, s') = \{\langle q_1, q_5 \rangle, \langle q_2, q_5 \rangle, \langle q_3, q_5 \rangle, \langle q_2, q_4 \rangle, \langle q_2, q_6 \rangle, \langle q_2, q_8 \rangle\}$; the two components of the $tfidf_2$ weights for each of these queries are as follows:

$$\begin{aligned}
tf_2(q_1, s) &= 0.333\ , & idf_2(q_1, s) &= 0.176 \\
tf_2(q_2, s) &= 0.333\ , & idf_2(q_2, s) &= 0.176 \\
tf_2(q_3, s) &= 0.333\ , & idf_2(q_3, s) &= 0.000 \\
tf_2(q_4, s') &= 0.117, & idf_2(q_4, s') &= 0.176 \\
tf_2(q_5, s') &= 0.235, & idf_2(q_5, s') &= 0.176 \\
tf_2(q_6, s') &= 0.235, & idf_2(q_6, s') &= 0.176 \\
tf_2(q_8, s') &= 0.235, & idf_2(q_8, s') &= 0.000
\end{aligned}$$

Note that, though q_3 and q_8 are similar (same group-by set, same selection predicate, and nearly the same set of measures) and should positively contribute to the similarity of s and s', they do not actually enter in the computation of $\sigma_{log}(s, s')$. Indeed, queries similar to q_3 and q_8 can be found in each session of the log, making their idf weight 0. By applying Definition 5.11 we get $\sigma_{log}(s, s') = 0.479$, while $\sigma_{log}(s, s'') = \sigma_{log}(s', s'') = 0$. □

5.6. Alignment-Based Session Similarity

As emerged in Section 3.1, a comparison of OLAP sessions should support subsequence alignment, keep query ordering into account, and allow gaps in the matching subsequences. The Smith-Waterman algorithm mentioned in Section

3.1 has all these features. It relies on a distinction between *matching* elements (whose similarity is positive) and *mismatching* elements (whose similarity is negative), and is based on a matrix whose cells show the score for aligning two sequences starting from a specific couple of elements. Each score is the result of a trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a mismatching pair of elements.

Unfortunately, none of the implementations available in the literature can be directly applied here for different reasons:

- The algorithm was originally aimed at molecular comparison, so sequence elements were taken from a set that is known a priori (the set of all amino acids). This allows matching and mismatching pairs to be enumerated and a similarity score to be assigned in advance to each possible couple of elements. In the OLAP context matching elements are queries, and the domain of the possible OLAP queries is huge (requirement ♯2); besides, the similarity between two queries is always positive, so separating matching and mismatching queries requires the adoption of a threshold.
- For the same reason mentioned above, in all previous implementations the cost for introducing a gap could be assigned in advance to each possible couple of elements. Conversely, in our case it must be determined at runtime based on the two specific sessions being compared (requirement ♯8).
- In all previous implementations all matchings were considered to be equally important, while in OLAP sessions a matching between recent queries should be given more relevance (requirement ♯6).

To address all these issues, we propose an extension of the Smith-Waterman algorithm that relies on the matrix defined below. The value in position (i, j) of this matrix is a score that expresses how "well" two sessions s and s' match when they are aligned ending in queries s_i and s'_j. Intuitively, each score is recursively calculated by progressively adding the similarities between all pairs of matching queries in the two sessions. Threshold θ is used to distinguish matches from mismatches; a *time-discounting* function $\rho(i, j)$ is used to promote alignments based on recent queries; finally, a *gap penalty* δ is used to discourage discontinuous alignments.

Definition 5.12 (OLAP Session Alignment Matrix). Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively. Given a matching threshold θ, the *(OLAP session) alignment matrix* for s and s' is a $(v+1)\times(v'+1)$ matrix A of reals recursively defined as follows:

$$A(i,j) = \begin{cases} 0, \text{ when } i = 0 \text{ or } j = 0 \\ max \left\{ \begin{array}{c} 0; \\ A(i-1,j-1) + (\sigma_{que}(s_i, s'_j) - \theta) \cdot \rho(v-i, v'-j); \\ max_{1 \le k < i}\{A(k,j) - \delta \cdot (i-k)\}; \\ max_{1 \le k < j}\{A(i,k) - \delta \cdot (j-k)\} \end{array} \right\}, \text{ else} \end{cases}$$

where δ is the average similarity between all couples of queries in s and s' whose similarity is above θ:

$$\delta = avg_{(i,j):\sigma_{que}(s_i,s'_j) \ge \theta}\{\sigma_{que}(s_i, s'_j)\} \ ,$$

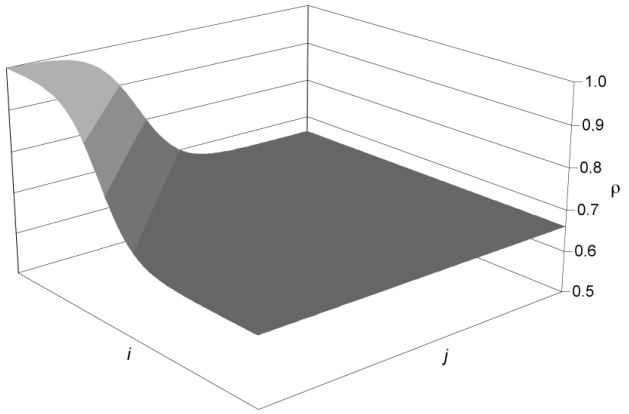

Fig. 3. The time-discounting function $\rho(i, j)$ with $\rho_{min} = 0.66$ and $slope = 4$

ρ is a two-dimensional logistic sigmoid function:

$$\rho(i, j) = 1 - \frac{1 - \rho_{min}}{1 + e^{slope - i - j}} ,$$

ρ_{min} is the minimal value assumed by ρ (i.e., the maximum time discount), and *slope* rules the position where the slope is steepest (Figure 3).

Some observations on the above definition:

- The use of the term $\sigma_{que}(s_i, s'_j) - \theta$ implies that query pairs whose similarity is above (below) θ are considered as matches (mismatches). Although a "sharp" threshold is used, the score of a matching pair and the cost of a mismatching pair turn out to be proportional to the distance of that pair similarity from θ.
- The definition given of the gap penalty δ is such that it guarantees a gap penalty to be payed if it enables a good match (i.e. a match higher than the average). Note that a penalty only related to the threshold could lead to underestimating or overestimating the impact of a gap on the overall similarity.
- The time-discounting function ρ leads match and mismatch scores to decay when moving backwards along the two sessions; it is maximum and equal to 1 for the ending queries of the two sessions.

The optimal alignment between s and s' is determined by the highest value in A, $\overline{A}$, that we call *alignment score*. The positions $\overline{i}$ and $\overline{j}$ such that $A(\overline{i}, \overline{j}) = \overline{A}$ mark the end of the matching subsequences of s and s'.

The alignment score is not really a similarity value, since it is not limited in the interval [0..1]. This creates problems when comparing sessions with difference length. Then we define OLAP session similarity by normalizing the alignment score:

Definition 5.13 (Alignment-Based Similarity of OLAP Sessions). Let s and s' be two OLAP sessions on schema $\mathcal{M}$, of lengths v and v' respectively (with $v \leq v'$), and let $\overline{A}$ be the alignment score for s and s'. The *alignment-based sim-*

Table 4. Threshold-filtered and discounted query similarities, $(\sigma_{que}(s_i, s'_j) - \theta) \cdot \rho(v-i, v'-j)$, for Example 5.6

	q_4	q_5	q_6	q_7	q_8
q_1	-0.004	0.171	0.120	-0.071	0.160
q_2	0.013	0.208	0.151	-0.053	0.186
q_3	-0.032	0.126	0.053	-0.082	0.132

Table 5. OLAP session alignment matrix for Example 5.6

	q_4	q_5	q_6	q_7	q_8
q_1	0.000	**0.171**	0.120	0.000	0.160
q_2	0.013	0.208	**0.322**	*0.191*	0.186
q_3	0.000	0.139	0.261	0.241	**0.323**

ilarity between s and s' is

$$\sigma_{ali}(s, s') = \frac{\overline{A}}{(1-\theta)\sum_{k=1}^{v} \rho(k,k)}$$

where the normalizing factor is the alignment score for two identical sessions of length v.

Like for edit-based similarity, the complexity of this function is known to be $O(v \cdot v')$ where v and v' are the lengths of the two sessions (Li and Durbin, 2010).

Example 5.6. Again we focus on comparing s and s' of Example 5.1. Table 4 reports the results obtained by filtering query similarities with $\theta = 0.7$ and applying the time-discounting function ρ as shown in Definition 5.12. Note that a negative value represents a mismatch, and a positive one a match. Table 5 shows the OLAP session alignment matrix for s and s'; the cells in bold denote alignments between two queries (e.g., q_1 is aligned with q_5), those in italics refer to gaps. Alignments on recent queries are favored, so q_3 is aligned with q_8. Query q_4 is not involved in the alignment due to the low similarity it has with the other queries in s. In q_7, a gap penalty is paid to gain the good match between q_3 and q_8. The overall similarity between s and s' is 0.323 (the highest value in the matrix). After normalization, we obtain $\sigma_{ali}(s, s')$=0.387. □

The properties of the proposed similarity function can be evaluated in terms of the distance function it induces using the standard transformation $\sigma_{ali} = 1/(1 + Dist_{ali})$. As stated by Bustos and Skopal (2011) for the original Smith-Waterman approach, $Dist_{ali}$ is not a metric because, while it is non-negative and symmetrical, it is not reflexive and it does not satisfy the triangular inequality as shown in Example 5.7. In particular, the triangular inequality cannot be satisfied because this approach is based on a local alignment.

Example 5.7. Let $s = \langle q_1, q_2 \rangle$, $s' = \langle q_1, q_2, q_3, q_4 \rangle$, and $s'' = \langle q_3, q_4 \rangle$ be three sequences, where $\sigma_{que}(q_i, q_j) = 0$ if $i \neq j$. It is

$$Dist_{ali}(s, s'') = \infty\ ,\ Dist_{ali}(s, s') = Dist_{ali}(s', s'') = 0$$

which obviously contradicts the triangle inequality axiom. Besides, s' has zero distance from both s and s'' though $s \neq s' \neq s''$. □

Table 6. Consensus and matching factors for OLAP query comparison user tests

	Consensus		σ_{AJD}		σ_{que}	
	ϕ_{score}	ϕ_{rank}	SM	RM	SM	RM
Test 1	70%	94%	70%	94%	70%	94%
Test 2	56%	70%	56%	56%	56%	70%
Test 3	41%	64%	34%	57%	41%	64%
Test 4	73%	93%	49%	93%	59%	93%

6. Validation and Comparison

This section discusses the outcomes of the tests we run to answer three main questions: *Do the proposed solutions properly capture the idea of similarity as perceived by the users? Do they adequately express the similarity criteria proposed in Section 2? What are their discriminant capabilities?* While the first question will be answered in Subsection 6.1, the remaining two questions will be discussed in Subsection 6.2.

6.1. User Tests

As stated in Section 3.2, we submitted a questionnaire to 41 persons with different OLAP skills. The results have been used in the first stages of this work to understand how OLAP session similarity is perceived by users, and they will be used here to verify if the proposed methods capture the users' perception of similarity. To enable a better interpretation of the results, for each questionnaire test we show the *consensus* ϕ, i.e., the degree of agreement among raters, defined as the percentage of users who gave the majority judgement.

The first four tests of the questionnaire were focused on OLAP query comparison. In each test the users were asked to rate the similarity between a given query q_c and three other queries $\{q_1, q_2, q_3\}$ in both absolute (using four scores: *low*, *fair*, *good*, and *high*) and relative terms (i.e., by ranking queries in order of similarity). All queries were focused on the complete CENSUS schema (including 5 hierarchies and 6 measures); they were basic OLAP queries as of Definition 5.1 and were presented in a graphical way. We used the results obtained in two ways: (i) to compare σ_{que} with function σ_{AJD} mentioned in Section 3.2 in terms of compliance with the users' judgments; and (ii) to set the weights of the three components of our query similarity function σ_{que}.

As to (i), we defined two matching factors as follows:

- The *score matching factor* SM for σ is the percentage of times the score given by a user is the same returned by σ. To compute it, we first discretized the values returned by σ in ranges corresponding to *low*, *fair*, *good*, and *high*.
- The *rank matching factor* RM for σ is the percentage of cases in which the rankings σ provides match with those given by users (e.g., q_c was judged to be more similar to q_i than to q_j, and $\sigma(q_c, q_i) > \sigma(q_c, q_j)$).

As to (ii), we tuned the weights through an optimization process whose goal function was the maximization of the correspondence with the questionnaire results. To avoid overfitting we used a ten folds cross-validation approach. The ranges for the weights were chosen consistently with requirement ♯4 in Section 2: $\alpha \in [0.2, 0.5]$, $\beta \in [0.35, 0.75]$, $\gamma \in [0.05, 0.45]$. The function to be optimized was the average value of RM for σ_{que} in Tests 1 to 4, that measures the percentage

Fig. 4. Questionnaire matching for σ_{que} as a function of weights α and β

of cases in which the rankings provided by σ_{que} match with those given by users. Figure 4 shows the average RM as a function of α and β (γ is set so that they sum up to 1). The optimal weights turned out to be $\alpha = 0.35$, $\beta = 0.5$, and $\gamma = 0.15$ ($\beta > \alpha$, consistently with requirement ♯4); noticeably, RM smoothly decreases for increasing distances from these optimal values, which proves that the setting is robust.

The comparison results are reported in Table 6. For all the tests, σ_{que} matches the users' judgement at least like σ_{AJD} thanks to its fine-grained definition. In particular, σ_{que} returns the same answers given by the majority of the users (i.e. the highest possible values for SM and RM) in Tests 1, 2, and 3, while σ_{AJD} returns the same answers only in Test 1. Note that σ_{AJD} falls short both when there is high user consensus (Test 4) and when user consensus is low because queries are very similar to each other (Tests 2 and 3). Overall, these results confirm a strong correlation between the query similarity computed through σ_{que} and the one perceived by users. Since σ_{que} is more sensitive than σ_{AJD} and it shows better results, in the remaining tests we will focus on the former.

The second part of the questionnaire included five more tests focused on OLAP session comparison. In each test, the users were asked to evaluate the similarity of a given session s_c against three candidate sessions $\{s_1, s_2, s_3\}$ in absolute and relative terms. Sessions were graphically presented to users as sequences of queries, emphasizing the OLAP operator used to move from one query to the next one. The results are summarized in Table 7 for the four functions described in Section 5, by applying SM and RM to sequences rather than to single queries. Note that the edit-based and the subsequence-based approaches, that do not directly incorporate the σ_{que} score in their definitions, are not sensitive enough to rank the sessions proposed in our tests. In fact, they return the same similarity for most sessions involved in each test, so their RM cannot be determined. This also penalizes SM, that is significantly low.

Conversely, both the log-based and the alignment-based approaches perform very well and the scores returned are, in most cases, those of the majority of users

Table 7. Consensus and matching factors for OLAP session comparison user tests

	Consensus		σ_{edit}		σ_{sub}		σ_{log}		σ_{ali}	
	ϕ_{score}	ϕ_{rank}	SM	RM	SM	RM	SM	RM	SM	RM
Test 1	51%	75%	51%	-	29%	-	51%	75%	51%	71%
Test 2	43%	70%	33%	-	9%	-	39%	70%	43%	70%
Test 3	51%	64%	41%	-	4%	-	51%	46%	51%	46%
Test 4	36%	80%	19%	-	26%	-	35%	65%	35%	65%
Test 5	38%	78%	33%	-	13%	-	33%	70%	33%	70%

(i.e., $SM = \phi_{score}$ and/or $RM = \phi_{rank}$, that is the maximum attainable). The errors always involve sequences that are quite similar, making the comparison more subjective. Note that the absolute consensus is always much lower than the relative one; this can be explained considering that scoring entails a 4-valued choice, while ranking only requires choosing between two alternatives (s_c is either more similar to s_i than s_j or not), thus making inter-user agreement more likely. Some more detailed comments for single tests of log-based and alignment-based approaches follow:

- In test 1, candidate sessions differ in the length of the match. s_1 and s_2 are very similar to each other and determine a long match with s_c, while s_3 is quite different from the others. While the log-based approach returns the same results as the majority of users, the alignment-based approach returns an inverted ranking between s_1 and s_2, which is a minor issue due to their strong similarity.
- In test 2, candidate sessions differ in the position of the match. The log-based approach returns a score that is slightly different from the one of the majority group since it does not give different relevance to matches of recent and old queries.
- In test 3, all three candidate sessions are quite similar to each other and to s_c, leading to a difficult ranking operation for both functions.
- In test 4, each candidate session differs from the reference only for one of the components of its queries (group-by set, predicates, and measures). Both approaches agree with the users majority in indicating the session that differs in their selection predicates as the less similar to the reference session. However, both approaches return an inverted ranking between the sessions that differ in their group-by sets and in their predicates, respectively. This is probably due to the weight we use for measure similarity, $\gamma = 0.15$, that in this particular case is not low enough to counterbalance the relevant difference on measure sets.
- In test 5, session s_1 is very similar to s_c; s_2 and s_3 are similar to each other and quite different from s_c. Both approaches agree with the users majority in indicating s_1 as the most similar to s_c, but they disagree in ranking the other two sessions. This is actually not surprising in light of the low relative consensus ($\phi_{rank}(s_2, s_3) = 61\%$).

6.2. Objective Tests

In this section we compare the four functions described in Section 5; for subsequence-based similarity we use 3-grams (empirically tested for best results). All tests were conducted on a 64-bits Intel Xeon quad-core 3GHz, with 8GB RAM, run-

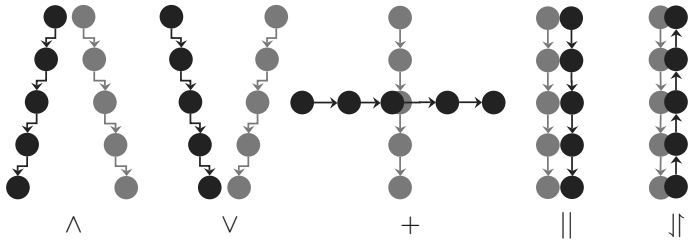

Fig. 5. The templates used to generate sessions. Overlapping circles represent identical queries, near circles represent similar queries. For template ||, the queries are pairwise separated by one atomic OLAP operation

ning Windows 7 pro SP1; the similarity threshold was tuned to $\theta = 0.8$ to achieve the best results.

Our benchmark includes a set of synthetic sessions over the CENSUS schema, generated based on Definition 5.2 with our own log generator developed in Java. A session is generated starting from an initial query and a final query, both obtained by randomly choosing a group-by set, a selection predicate, and a subset of measures. Intermediate queries are then generated by applying, one at a time in a random order, the minimal atomic OLAP operations that transform the initial query into the final one. The atomic OLAP operations considered are: change level along one hierarchy in the group-by set, add or remove a clause from the selection predicate, change the constant appearing in a selection clause, and add or remove a measure.

To generate logs we considered the five templates depicted in Figure 5, that model intuitive notions of what similar sessions might look like:

- In template $\wedge$, the two sessions have similar starting queries then they diverge to radically different queries.
- In template $\vee$, the two sessions have radically different starting queries then they converge to similar ending queries.
- In template $+$, the two sessions converge to the same query then they diverge.
- In template $||$, the second session is constructed by "shifting" all queries in the first session by one OLAP operation.
- In template $\downharpoonleft\upharpoonright$, the two sessions have the same queries in reverse order.

In light of the requirements expressed in Section 2, some of these templates should yield higher similarities. In particular, we want template $\vee$ to yield higher similarities than $\wedge$ due to requirement ♯6. For requirement ♯7, we also expect $||$ to yield higher similarities than $\vee$, $\wedge$, and $+$. As to $\downharpoonleft\upharpoonright$, requirement ♯5 imposes that it yields low similarities.

The first test assesses the capabilities of the similarity functions. In this test, for each template we generated a log as follows (see also Figure 6 for an example):

1. Generate a pair of sessions, s and s', that respect the template.
2. Generate 5 more sessions $s_1, \ldots s_5$ using s as a seed. The first and the last query of s_i are obtained by applying three random atomic OLAP operations to the first and the last query of s, respectively; then, the intermediate queries of s_i are generated as described above.
3. Repeat the two previous steps 5 times.

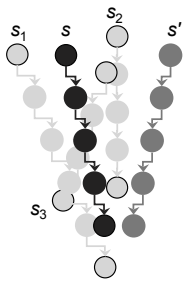

Fig. 6. The seed session s (in black), its mate s' according to template $\wedge$ (in dark gray), and three random sessions (in light gray). The first and last queries of sessions are circled.

Table 8. Ratio τ for template-based OLAP session comparison objective tests

Log	σ_{edit}	σ_{sub}	σ_{log}	σ_{ali}
$\wedge$	1.39	1.16	1.39	2.32
$\vee$	1.46	1.52	1.31	3.21
$+$	1.44	1.23	1.32	2.15
$\|\|$	1.79	1.57	1.51	5.23
$\downarrow\uparrow$	1.08	1.57	1.42	0.78
average	1.40	1.35	1.35	2.51

This means generating overall 5 logs, each including 35 sessions. Then, for each log and each similarity function, we computed the ratio τ of the average similarity σ_t between the two sessions respecting the template and the average similarity σ_r between each seed and the 5 sessions generated from it; the higher τ, the better the function can distinguish a template from the background. Table 8 reports the results. Noticeably, the alignment-based approach largely outperforms the others; besides yielding an average τ that is almost twice that of the other approaches, it meets the expectations as to template similarities. Template $||$ is correctly recognized as the one with highest similarity; $\vee$ clearly yields higher similarities than $\wedge$, while $\downarrow\uparrow$ yields low similarities since it does not fulfill requirement ♯5 about query ordering. The only other function that captures requirement ♯5 is σ_{edit}. Noticeably, though all the other functions return an average ratio τ higher than 1, they are not sensitive enough to distinguish and rank the different templates.

The purpose of the second objective test is to discover how sensitive each function is to the distance between the two sessions that form template $||$; to this end, the number of atomic OLAP operations that separate these two sessions is varied from 1 to 5 (using the same log-generation algorithm explained for the first test). Even in this test σ_{ali} turns out to be more effective than the other functions. Indeed, as shown in Table 9, the ratio τ for σ_{ali} progressively decreases for increasing distances, while for the other functions it is almost constant. This is because σ_{ali} is sensitive to the specific values of similarity between each couple of queries, while for the other functions each couple of queries either match or do not match.

The next test measures the time for computing each similarity function. For this test we generated a log, randomly chose one session s, and compared all prefixes of s with 10 other sessions randomly chosen from the log. Note that, for log-based similarity, we disregard the time for building the frequency matrix used in the computation of all the idf's. We report the results for a minimum prefix of

Table 9. Ratio τ for increasing distances in the || template

|| dist	σ_{edit}	σ_{sub}	σ_{log}	σ_{ali}
1	1.79	1.57	1.51	5.23
2	1.91	1.55	1.51	3.78
3	1.86	1.56	1.45	3.48
4	1.81	1.52	1.42	2.80
5	1.81	1.52	1.55	2.68

1 query and a maximum prefix of 13 queries. As expected, the subsequence-based approach is the most efficient (from 0.4 ms to 3.6 ms for a single comparison), followed by the alignment-based approach (from 1.1 ms to 7.1 ms) and by the edit-based approach (from 1.3 ms to 8.3 ms). Log-based similarity is the less efficient (from 30.4 to 75.1 ms).

We close this section with a final remark related to efficiency. OLAP sessions are inherently interactive; to understand to what extent our approach can realistically be adopted to compare sessions at user-time, we made two tests using the same protocol adopted for the test above:

- We measured how many comparisons can be made for each similarity function during 100 ms, which is usually considered to be the maximum interactive response time (Khoussainova et al., 2010). The number of comparisons ranges from 109 for subsequence-based similarity to 3 for log-based similarity, with alignment-based and edit-based similarity scoring 32 and 31 comparisons, respectively.
- We measured how many comparisons can be made during the average time it takes to evaluate a query. To this end we randomly chose a session in the log and computed the average execution time for its queries, expressed in MDX; we used real data extracted from the IPUMS database (Minnesota Population Center, 2008), corresponding to about 500,000 facts stored on Oracle 11g. The average query execution time turned out to be 553.46 ms, which corresponds to 607 comparisons for subsequence-based similarity, 177 and 175 comparisons for alignment-based and edit-based similarity respectively, and 18 comparisons for log-based similarity.

7. Discussion

In this paper we investigated different approaches for defining a similarity function to compare OLAP sessions, based on the requirements deduced from a user study conducted with practitioners and researchers. We considered and compared two functions for OLAP query similarity and four functions for OLAP session similarity; in particular, the latter were obtained by extending popular approaches for string comparison.

Overall, the experimental results we obtained show that the alignment-based approach (an extension of the Smith-Waterman algorithm, coupled with a three-component query similarity function) is the one that best matches the users' judgements. It is also the one that clearly gives best results on a synthetic benchmark in terms of sensitivity and capability of correctly ranking different templates of session similarity. Finally, from the point of view of efficiency, the time required for comparing two sessions is perfectly compatible with complex applications.

The results presented in this paper are propaedeutic to a broader research in the area of OLAP personalization. In particular, the next step will be to use the alignment-based approach to set up a method for recommending OLAP queries based on the similarity between the session a user is currently involved in, and the sessions that were issued in the past by the same or other users. We will pay a particular attention in mixing intensional and extensional information, as suggested in (Chatzopoulou et al., 2011), in order to support OLAP exploratory analysis. This will have a major impact on improving OLAP-based interactions from both points of view of efficiency (by reducing the query formulation effort) and effectiveness (by suggesting popular/successful trends of analysis).

References

Abiteboul, S., Hull, R. and Vianu, V. (1995), *Foundations of Databases*, Addison-Wesley.

Agrawal, R., Rantzau, R. and Terzi, E. (2006), Context-sensitive ranking, *in* 'Proceedings ACM SIGMOD International Conference on Management of Data', Chicago, IL, pp. 383–394.

Akbarnejad, J., Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., On, D., Polyzotis, N. and Varman, J. S. V. (2010), 'SQL QueRIE recommendations', *PVLDB* **3**(2), 1597–1600.

Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S. and Turricchia, E. (2011), Mining preferences from OLAP query logs for proactive personalization, *in* 'Proceedings ADBIS', Vienna, Austria, pp. 84–97.

Aouiche, K., Jouve, P.-E. and Darmont, J. (2006), Clustering-based materialized view selection in data warehouses, *in* 'Proceedings ADBIS', Thessaloniki, Greece, pp. 81–95.

Baikousi, E., Rogkakos, G. and Vassiliadis, P. (2011), Similarity measures for multidimensional data, *in* 'Proceedings ICDE', Hannover, Germany, pp. 171–182.

Brown, P. F., Pietra, V. J. D., de Souza, P. V., Lai, J. C. and Mercer, R. L. (1992), 'Class-based n-gram models of natural language', *Computational Linguistics* **18**(4), 467–479.

Bustos, B. and Skopal, T. (2011), Non-metric similarity search problems in very large collections, *in* 'Proceedings ICDE', Hannover, Germany, pp. 1362–1365.

Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., Polyzotis, N. and Varman, J. S. V. (2011), 'The QueRIE system for personalized query recommendations', *IEEE Data Eng. Bull.* **34**(2), 55–60.

Chatzopoulou, G., Eirinaki, M. and Polyzotis, N. (2009), Query recommendations for interactive database exploration, *in* 'Proceedings SSDBM', New Orleans, LA, pp. 3–18.

Cohen, W. W., Ravikumar, P. D. and Fienberg, S. E. (2003), A comparison of string distance metrics for name-matching tasks, *in* 'Proceedings IJCAI-03 Workshop on Information Integration on the Web', Acapulco, Mexico, pp. 73–78.

Drosou, M. and Pitoura, E. (2011), ReDRIVE: result-driven database exploration through recommendations, *in* 'Proceedings CIKM', Glasgow, United Kingdom, pp. 1547–1552.

Garcia-Molina, H., Ullman, J. D. and Widom, J. D. (2008), *Database Systems: The Complete Book, Second edition*, Prentice Hall.

Ghosh, A., Parikh, J., Sengar, V. S. and Haritsa, J. R. (2002), Plan selection based on query clustering, *in* 'Proceedings VLDB', Hong Kong, China, pp. 179–190.

Giacometti, A., Marcel, P. and Negre, E. (2009), Recommending multidimensional queries, *in* 'Proceedings DaWaK', Linz, Austria, pp. 453–466.

Golfarelli, M. (2003), Handling large workloads by profiling and clustering, *in* 'Proceedings DaWaK', Prague, Czech Republic, pp. 212–223.

Golfarelli, M., Rizzi, S. and Biondi, P. (2011), 'myOLAP: An approach to express and evaluate OLAP preferences', *IEEE TKDE* **23**(7), 1050–1064.

Gupta, A., Harinarayan, V. and Quass, D. (1995), Aggregate-query processing in data warehousing environments, *in* 'Proceedings VLDB', Zurich, Switzerland, pp. 358–369.

Gupta, A. and Mumick, I. (1999), *Materialized views: techniques, implementations, and applications*, MIT Press.

Khoussainova, N., Kwon, Y., Balazinska, M. and Suciu, D. (2010), 'SnipSuggest: Context-aware autocompletion for SQL', *PVLDB* **4**(1), 22–33.

Khoussainova, N., Kwon, Y., Liao, W.-T., Balazinska, M., Gatterbauer, W. and Suciu, D. (2011), Session-based browsing for more effective query reuse, *in* 'Proceedings SSDBM', Portland, OR, pp. 583–585.

Li, H. and Durbin, R. (2010), 'Fast and accurate long-read alignment with Burrows-Wheeler transform', *Bioinformatics* **26**(5), 589–595.

Minnesota Population Center (2008), 'Integrated public use microdata series', http://www.ipums.org.

Monge, A. E. and Elkan, C. (1997), An efficient domain-independent algorithm for detecting approximately duplicate database records, *in* 'Proceedings Workshop on Research Issues on Data Mining and Knowledge Discovery'.

Moreau, E., Yvon, F. and Cappé, O. (2008), Robust similarity measures for named entities matching, *in* 'Proceedings International Conference on Computational Linguistics', Manchester, UK, pp. 593–600.

Navarro, G. (2001), 'A guided tour to approximate string matching', *ACM Comput. Surveys* **33**(1), 31–88.

Ögüdücü, S. G. (2010), *Web Page Recommendation Models: Theory and Algorithms*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers.

Ristad, E. S. and Yianilos, P. N. (1998), 'Learning string-edit distance', *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(5), 522–532.

Sapia, C. (2000), PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems, *in* 'Proceedings DaWaK', London, UK, pp. 224–233.

Smith, T. and Waterman, M. (1981), 'Identification of common molecular subsequences', *Journal of Molecular Biology* **147**, 195–197.

Stefanidis, K., Drosou, M. and Pitoura, E. (2009), "You May Also Like" results in relational databases, *in* 'Proceedings International Workshop on Personalized Access, Profile Management and Context Awareness: Databases', Lyon, France.

Wagner, R. and Fischer, M. (1974), 'The string-to-string correction problem', *Journal ACM* **21**(1), 168–173.

Yang, X., Procopiuc, C. M. and Srivastava, D. (2009), Recommending join queries via query log analysis, *in* 'Proceedings ICDE', Shanghai, China, pp. 964–975.

Yao, Q., An, A. and Huang, X. (2005), Finding and analyzing database user sessions, *in* 'Proceedings DASFAA', Beijing, China, pp. 851–862.