A Holistic Approach to OLAP Sessions Composition: The Falseto Experience

Julien Aligon, Kamal Boulil, Patrick Marcel, Verónika Peralta
University of Tours
Blois, France
firstname.lastname@univ-tours.fr

ABSTRACT

OLAP is the main paradigm for flexible and effective exploration of multidimensional cubes in data warehouses. During an OLAP session the user analyzes the results of a query and determines a new query that will give her a better understanding of information. Given the huge size of the data space, this exploration process is often tedious and may leave the user disoriented and frustrated.

This paper presents an OLAP tool named Falseto (Former AnalyticaL Sessions for IEss Tedious Olap), that is meant to assist query and session composition, by letting the user summarize, browse, query, and reuse former analytical sessions. Falseto’s implementation on top of a formal framework is detailed. We also report the experiments we run to obtain and analyze real OLAP sessions and assess Falseto with them. Finally, we discuss how Falseto can be seen as a starting point for bridging OLAP with exploratory search, a search paradigm centered on the user and the evolution of her knowledge.

1. INTRODUCTION

While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating. OLAP queries are usually formulated in the form of sequences called OLAP sessions. During an OLAP session the user analyzes the results of a current query and, depending on the specific data she sees, determines a new query that will give her a better understanding of information. This new query is often constructed by modifying the previous query, applying simple primitives like roll-up, drill-down or slice/dice. Noticeably, given the huge number of such modification possibilities, this exploration process is often seen as tedious and may leave the user disoriented and frustrated.

To make the user experience less disorientating, it has recently been suggested that database systems could take advantage of query logs [15]. Logs can indeed be used for learning user preferences [13], for query recommendation [7, 22], for query auto-completion [16], or for query composition [17]. Similar efforts were also made in the context of OLAP systems [4, 10, 2], but so far no usable OLAP tool implements them, and this leads us to the development of Falseto.

Falseto (Former Analytical Sessions for IEss Tedious Olap) aims at assisting the user analyzing data cubes, by letting her summarize, browse, query, and reuse former analytical sessions. In particular, Falseto’s recommender system automatically suggests an analytical session considered as relevant regarding her current session. Falseto also lets the user query and browse former sessions, which is particularly useful when a user initiates her session and the recommender system cannot suggest queries as no current session exists. Since the set of former sessions can be very large, Falseto presents first summarized versions of the sessions in a way that makes it easy to navigate and query them.

The paper is structured as follows. Section 2 introduces Falseto’s functionalities by means of examples. Section 3 describes the logical framework underlying Falseto and Section 4 presents its implementation. Section 5 reports the lessons learned through various tests aiming at obtaining and analyzing real OLAP sessions and assessing Falseto with them. After briefly reviewing related work, we discuss how Falseto can be a starting point of future researches to bridge the gap between OLAP and exploratory search in Section 6. Section 7 concludes the paper.

2. GETTING FAMILIAR WITH FALSETO

Assume you are an OLAP user who has to analyze census data obtained from the IPUMS (Integrated Public Use Microdata Series) website [18], and gathered under the form of a multidimensional cube whose schema is depicted Figure 1. It has 5 hierarchies describing people sex, race, occupation and residence place, as well as census year. They are organized as follows:

- Sex is a one-level hierarchy, containing two values: Male and Female.
- Race is a three-level hierarchy. MRN (stands for Major Races Number) indicates the number of major races mix of a person (e.g., 2). Racegroup groups races in...
major categories (e.g. Black and Asian) and Race lists detailed races (e.g. Black and Korean).

- Occupation is a four-level hierarchy. Category indicates 9 major occupation categories, that are specialized in Sub-categories and Branches. Finally, Occupation lists detailed occupations.

- City is a three-level hierarchy describing people residence place. Region indicates 10 United States regions. State indicates state codes (e.g., CA for California) and City lists United States cities.

- Year is a one-level hierarchy, containing six values, from 2000 to 2005.

- The schema describes six census indicators: total personal income (INCTOT), annual property insurance cost (PROPINSR), annual gas cost (COSTGAS), annual water cost (COSTWATR), annual electricity cost (COSTELEC) and person weight (PERWT). Four measures are defined for each indicator according to 4 aggregation functions (sum, max, min and average), for example SUMINCTOT, MAXINCTOT, MININCTOT and AVGINCTOT. A 25th measure (EVENTCOUNT) counts the number of facts.

More specifically, assume that you have to answer the following, rather vague, analytical question: What is the profile of female workers that better resists to a drop of average income? You would probably try to navigate the IPUMS cube, playing with aggregation levels and filters, to discover this profile.

Falseto, whose main user interface is depicted in Figure 2, can be used to graphically create the queries of your session, execute them, view the results and decide on what the next query should be. Designing queries is done in the query design tab (bottom left corner of Figure 2). The interface refers to the Dimension Fact Model [11] and allows to edit each component of the query, i.e., the measure set, the group-by set and the selection set. The reason for choosing this way of representing OLAP queries is, aside the popularity of the DFM, to have a simple, uniform and non-textual model to depict the queries using the schema of the cube, which allows to focus on the three components of the query. Falseto also enables to devise a sequence of OLAP queries, your OLAP session, in order to answer your analytical question.

Each time a query is created and evaluated, it is recorded and placed in the Session Viewer panel (top left corner of Figure 2). This is the viewer for the past queries of your session. Each query can be subsequently retrieved, zoomed, reevaluated, edited, and used as a parameter for the holistic functionalities of Falseto.

The advanced features of Falseto are called holistic functionalities since they consider sessions in their entirety and do not treat queries independently. Of course, all queries retrieved by means of these functionalities can be reused and edited.

The first holistic functionality allows to browse the past sessions recorded in the query log. The log browsing tab of Falseto provides an overview of these past sessions (see Figure 3). Each of the different panels represents a cluster (i.e., a subset) of past sessions that are similar. The text in each panel indicates the fragments that appear in all the sessions of this subset (group-by levels are in yellow, selections in green and measures in black). A left-click on a cluster displays a summary of the cluster under the form of a sequence of queries (bottom of Figure 3). Each query of this session summarizes a portion of all the sessions of the cluster by displaying the common measures and selections, and the coarsest group-by set of the portion. It is also possible to scroll down the detailed view in order to display each past session that composes the cluster. Another possibility is to split a
cluster into several clusters of shorter sizes, or to merge clusters to have a coarser view of the log. Because the browsing of each cluster or individual session can be tedious, Falseto also offers filtering operators to reduce the number of clusters and sessions to browse. The specialization-based search retrieves from the set of past sessions only those that include at least one query containing the selections, measures and finer (or identical) group-by sets of the query given as the input of the operator. The similarity-based search retrieves from the set of past sessions only those that include at least one query that is similar to the query given as input of the operator. These two operations can also be used by giving not a query but a session as input.

Finally, Falseto implements a recommender system that suggests a sequence of queries based on what you have done in your current session. This suggestion comes from a past session (recorded in the log) whose beginning is found similar to your session, and can be seen as a potential and customized future for your current session. This recommender system can be seen as an instantiation of the generic framework proposed in [9, 10] with some of the similarity measures described in [5]. More details are given in Section 4 and the full description can be found in [3, 2].

Note that the examples and figures used in this section come from the Falseto tutorial, available online on the Falseto web site [8], where Falseto can also be downloaded.

3. THE LOGICAL FRAMEWORK

The logical framework underlying Falseto relies on the separation between queries and logs. The query design functionalities of Falseto rely on a simple query model and language described in Section 3.1, while holistic functionalities rely on a log manipulation language described in Section 3.2.

3.1 The query edition language

Query edition is done with a simple language for manipulating dimensional queries [11]. We now introduce the data model, the query model and finally the operators of this language. To keep the formalism simple, we consider cubes un- computed, the query model and finally the operators of this lan- guage. To keep the formalism simple, we consider cubes un-

Let \( \mathbb{M} \) be a finite set of hierarchies, each defined on \((\mathbb{H}, \mathbb{L}),\mathbb{D})\) for \(i \in \{1, \ldots, n\}\) defines a partition of \(A\);

- \(M\) is a finite set of measure attributes, each defined on a numerical domain \(\text{Dom}(m)\).

Given a schema \(\mathbb{M} = (A, H, M)\), let \(\text{Dom}(H) = \text{Lev}(h_1) \times \ldots \times \text{Lev}(h_n)\); each \(G \in \text{Dom}(H)\) is called a group-by set of \(\mathbb{M}\). In what follows we will use the classical partial order on group-by sets, defined as follows. A group-by set \(G\) is more general than a group-by set \(G'\), noted \(G \succeq G'\), if \(G = (a_1, \ldots, a_n), G' = (a'_1, \ldots, a'_n)\) and \(\forall i \in [1, n]\) it is \(a_i \succeq a'_i\).

**Definition 3.1 (Fragment-based OLAP query).** A query \(q^2\) over schema \(\mathbb{M} = (A, H, M)\) is a triple \(q = (G, P, \text{Meas})\) where:

1. \(G \in \text{Dom}(H)\) is the query group-by set;
2. \(P = \{p_1, \ldots, p_n\}\) is a set of Boolean predicates, one for each hierarchy, whose conjunction defines the selection predicate for \(q\); they are of the form \(l = v\), or \(l \in V\), with \(l\) a level, \(v\) a value, \(V\) a set of values. Conventionally, we note \(p_i = \text{TRUE}\) if no selection on \(h_i\) is made in \(q\) (all values being selected);
3. \(\text{Meas} \subseteq M\) is the measure set whose values are returned by \(q\).

**Definition 3.2 (OLAP Session and log).** Let \(\mathbb{M} = (A, H, M)\) be a multidimensional schema and \(\mathbb{S}\) be a set of queries over \(\mathbb{M}\). A session \(s\) of \(k\) queries over \(\mathbb{M}\), noted \(s = \langle q_1, q_k \rangle\), is a function from an ordered set \(\text{pos}(s)\) of integers (called positions) of size \(k\) to \(\mathbb{S}\). A log \(L\) is a finite set of sessions, noted \(L = \{s_1, \ldots, s_p\}\).

The operators of the query edition language allow to derive a query expression from another one by affecting a particular component of the query (group-by set, selection set or measure set). The operators are similar to those traditionally used in OLAP (such as roll-up, drill-down, or slice-and-dice) but instead of being applied on cubes (i.e. the instances), they are applied on queries (i.e. the expressions).

The first two operators allow to modify the group-by set \(G\), by changing a level, for a given hierarchy \(h_i\), to a coarser or finer granularity level. Two operators modify the selection set \(P\). The first operator adds a member \(c_i'\) of a level \(l_i'\) to a set of members \(X_i\) of the same level as \(l_i\) of a predicate \(p_i \in P\), for a given hierarchy \(h_i\). The second operator removes a member \(c_i'\) of a level \(l_i'\) from a set of members \(X_i\) of the same level as \(l_i\) of a predicate \(p_i \in P\), for a given hierarchy \(h_i\). Finally, two operators allow modifying the measure set \(\text{Meas}\), by either adding or removing a measure fragment.

**Definition 3.3 (Query edition operators).** Let \(q = (G, P, \text{Meas})\) be a query, \(h_i\) be a hierarchy, \(i \in \{c_i'\}\) be a selection predicate and \(m\) a measure.

- **Rollup**act \(q; h_i\) = \((G', P, \text{Meas})\) where \(l_i \in \text{Lev}(h_i)\) and \(G' = \langle l_1, \ldots, \text{Rollup}(l_i), \ldots, l_n \rangle\), if \(l_i \neq \text{ALL}\), undefined otherwise.
- **Drilldown**act \(q; h_i\) = \((G', P, \text{Meas})\) where \(l_i \in \text{Lev}(h_i)\) and \(G' = \langle l_1, \ldots, \text{Drilldown}(l_i), \ldots, l_n \rangle\), if \(l_i \neq \text{DIM}\), undefined otherwise.
- **AddSelection** \(q; p_i \in \{c_i'\}\) = \((G', P, \text{Meas})\) where \(p_i \in P'\) with \(p_i = l_i \in X_i'\) such as \(X_i' = X_i \cup \{c_i'\}\)

\(\text{This definition can be extended to add more query operators (e.g. selections on measure values)\)
The holistic functionalities of Falseto can be easily described by expressions in a language that manipulates logs, i.e., sets of sessions [6]. This language features five operations and is inspired by the relational algebra. Two of them are unary operations and is inspired by the relational algebra. For the sake of simplicity, the names and symbols of the operations are those used in the relational algebra. Two of them are binary: selection \( \sigma \) and grouping/aggregation \( \pi \). Three of them are binary: join \( \bowtie \), union \( \cup \), and difference \( \setminus \).

The selection operation enables to select from a log \( L \) those sessions satisfying a given condition. This condition is given over sessions.

\[ \sigma_{\theta,s}(L) = \{ s' \in L \mid \theta(s,s') \} \]

As logs are defined as sets of sessions, the set union and set difference operations can be used to manipulate logs. The definitions are straightforward:

\[ L \cup L' = \{ s \in L \cup L' \mid s \in L \text{ or } s \in L' \} \]

The join operation enables to combine the sessions in two logs, that are related through a relation \( \theta \).

\[ L \bowtie_{\theta,s} L' = \{ f(s,s') \mid s \in L, s' \in L', \theta(s,s') \} \]

Note that intersection can be simulated with the join operation.

The grouping/aggregation operator allows to group sessions and is inspired by the join operation defined in [1] for joining distant two queries, or two sessions, are. We start by introducing relations between queries and sessions at various levels of details. The second type are specialization relations, allowing the description of how close or distant two queries, or two sessions, are. We start by introducing relations between queries and sessions.

\[ \pi_{\theta,\text{agg}}(L) = \{ \text{agg}(\{ s' \in L \mid \theta(s,s') \}) \mid s \in L \} \]

We briefly review the properties of the language in terms of closeness, completeness and minimality. It can easily be seen that the language is closed under composition, the result of any operation being a set of sessions. The language is complete. Indeed, for any pair of logs \( L \) and \( L' \), there is an expression that enables the transformation \( L \) into \( L' \). The transformation would proceed as follows:

1. Pick a session \( s \) in \( L \) with \( \sigma \).
2. Transform it into a session \( s' \) of \( L' \) using \( \pi \).
3. Repeat the previous steps until all sessions of \( L' \) are formed.
4. Union all transformed sessions to form \( L' \).

The language is not minimal since \( \sigma \) can be simulated with \( \bowtie \). Indeed, \( \sigma_{\theta,s}(L) = L \bowtie_{\theta,f,s} L \) where \( L' = \{ s \} \) and \( f(s,s') = s' \). It can easily be seen that removing \( \sigma \) makes the language minimal.

4. UNDER THE HOOD OF FALSETO

Falseto is implemented as a standalone Java application, on top of the Mondrian OLAP engine, and uses the Weka machine learning library. Falseto can be used with any database system compatible with Mondrian and can be used to query any cube with a star schema.

In terms of query edition, Falseto can be used to construct a query on a given schema from scratch, edit the query using the query edition operators, and evaluate the query over an instance of the schema. All evaluated queries are kept in an in-memory structure representing the current session of the user, who can go back to them for edition or reevaluation purpose. The sessions of the log, that can be retrieved using the holistic functionalities, are loaded in memory when Falseto is started, and their queries can also be edited and evaluated.

The holistic functionalities of Falseto are described by expressions in the log manipulation language introduced above. Noticeably, important inputs of these expressions are the relations over sessions that are used to parametrize the operators (the \( \theta \) in the definitions above). This is why, before giving the expressions corresponding to the holistic functionalities, we start by introducing relations between queries and sessions which will be used as parameters of the operators.

4.1 Relations over queries and sessions

To describe the holistic functionalities of Falseto, we describe two types of relations. The first type are specialization relations, enabling the descriptions of groups of queries or sessions at various levels of details. The second type are similarity relations, allowing the description of how close or distant two sessions, or two queries, are. We start by the specialization relations.

Considering two group-by sets \( G \) and \( G' \), recall that we note \( G \succeq_H G' \) if \( G \) is more general than \( G' \) in the sense of the group-by lattice, whose top element \( G^{+} \) is the coarsest group-by set. For two sets of selection predicates \( P, P' \), we note \( P \preceq_P P' \) if \( P \) selects less values than \( P' \), i.e., \( \text{val}(P) \subseteq \text{val}(P') \), where \( \text{val}(X) \) is the set of values used in selection predicate \( X \). Finally, for two measure sets \( M, M' \), we note \( M \succeq_M M' \) if \( M \subseteq M' \).

\[ \sigma_{\theta,\text{agg}}(L) = \{ \text{agg}(\{ s' \in L \mid \theta(s,s') \}) \mid s \in L \} \]

We briefly review the properties of the language in terms of closeness, completeness and minimality. It can easily be seen that the language is closed under composition, the result of any operation being a set of sessions. The language is complete. Indeed, for any pair of logs \( L \) and \( L' \), there is an expression that enables the transformation \( L \) into \( L' \). The transformation would proceed as follows:

1. Pick a session \( s \) in \( L \) with \( \sigma \).
2. Transform it into a session \( s' \) of \( L' \) using \( \pi \).
3. Repeat the previous steps until all sessions of \( L' \) are formed.
4. Union all transformed sessions to form \( L' \).

The language is not minimal since \( \sigma \) can be simulated with \( \bowtie \). Indeed, \( \sigma_{\theta,s}(L) = L \bowtie_{\theta,f,s} L \) where \( L' = \{ s \} \) and \( f(s,s') = s' \). It can easily be seen that removing \( \sigma \) makes the language minimal.

4. UNDER THE HOOD OF FALSETO

Falseto is implemented as a standalone Java application, on top of the Mondrian OLAP engine, and uses the Weka machine learning library. Falseto can be used with any database system compatible with Mondrian and can be used to query any cube with a star schema.

In terms of query edition, Falseto can be used to construct a query on a given schema from scratch, edit the query using the query edition operators, and evaluate the query over an instance of the schema. All evaluated queries are kept in an in-memory structure representing the current session of the user, who can go back to them for edition or reevaluation purpose. The sessions of the log, that can be retrieved using the holistic functionalities, are loaded in memory when Falseto is started, and their queries can also be edited and evaluated.

The holistic functionalities of Falseto are described by expressions in the log manipulation language introduced above. Noticeably, important inputs of these expressions are the relations over sessions that are used to parametrize the operators (the \( \theta \) in the definitions above). This is why, before giving the expressions corresponding to the holistic functionalities, we start by introducing relations between queries and sessions which will be used as parameters of the operators.

4.1 Relations over queries and sessions

To describe the holistic functionalities of Falseto, we describe two types of relations. The first type are specialization relations, enabling the descriptions of groups of queries or sessions at various levels of details. The second type are similarity relations, allowing the description of how close or distant two sessions, or two queries, are. We start by the specialization relations.

Considering two group-by sets \( G \) and \( G' \), recall that we note \( G \succeq_H G' \) if \( G \) is more general than \( G' \) in the sense of the group-by lattice, whose top element \( G^{+} \) is the coarsest group-by set. For two sets of selection predicates \( P, P' \), we note \( P \preceq_P P' \) if \( P \) selects less values than \( P' \), i.e., \( \text{val}(P) \subseteq \text{val}(P') \), where \( \text{val}(X) \) is the set of values used in selection predicate \( X \). Finally, for two measure sets \( M, M' \), we note \( M \succeq_M M' \) if \( M \subseteq M' \).

\[ \sigma_{\theta,\text{agg}}(L) = \{ \text{agg}(\{ s' \in L \mid \theta(s,s') \}) \mid s \in L \} \]
Figure 4: Specialization relation over sessions

DEFINITION 4.2 (SESSION SPECIALIZATION). A session \( s = (q_1, \ldots, q_n) \) is more general than another session \( s' = (q'_1, \ldots, q'_{n'}) \), written \( s \succeq s' \), if there exists a sequence of \( n \) integers \( \{i_1, \ldots, i_{n'}\} \) with \( i_1 = 1, i_{n'} = n \) and, for \( k \in \{1, n_{i_{k-1}}\} \), \( i_k \leq i_{k+1} \), such that, for all \( j \in \{1, \ldots, n\} \), it is \( q_i \succeq q'_i \).

The intuition of this specialization relation is given in Figure 4. Note that if \( s \succeq s' \) then the sequence of integers \( \{i_1, \ldots, i_{n'}\} \) defines a partition \( P = \{p_1, \ldots, p_{n'}\} \) of queries \( \{s'\} \) where the \( p_i \)'s can be ordered according to the queries of \( s \), and the queries in each \( p_i \), each less general than \( q_i \), constitute a sub-session of \( s' \). More details on this specialization relation can be found in [6].

Finally, we suppose a function \( \text{dist} \) applied to pairs of sessions, giving the distance between two sessions. Similarity relations over sessions can be defined from such a function. For instance, \( \text{sim}(s, s') \equiv \text{dist}(s, s') \leq \alpha \) for some real \( \alpha \) indicates that two sessions \( s \) and \( s' \) are similar if their distance is below a threshold. Falseto implements such a relation, that is based on an extension of the Smith-Waterman alignment algorithm [21], whose goal is to efficiently find the best alignment between subsequences of two given sequences by ignoring their non-matching parts. This extension was proven a good measurement of OLAP session similarity in [5].

4.2 Holistic functionalities

Recall from Section 2 that Falseto includes three holistic functionalities: log searching, log browsing and session recommendation.

Log search

To search the log for useful sessions, Falseto implements different versions of the log selection \( \sigma_{\text{cur}} \). A first version enables the filtering of a log using for \( \theta \) the \( \text{sim} \) relation based on the extension of the Smith-Waterman algorithm. This operation returns those sessions of the log that are close to the session \( s \) given as a second parameter. Likewise, a log can be filtered using for \( \theta \) the specialization relation over sessions defined above. This version of the selection is very useful as it allows to search a log for queries by giving only a partial description of these queries. For instance, to search the log \( L \) for sessions containing queries with selection \( \text{year} = 2013 \), it is sufficient to use \( \sigma_{\text{cur}} (G_{\text{agg}} : L \triangleright (\text{year} = \text{2013})) \), where \( G_{\text{agg}} \) is the coarsest group-by set.

Log summarization and browsing

To allow for a user-friendly browsing of the log, Falseto implements operations for summarizing it, using the \( \pi \) grouping/aggregation operator. The basic summarization operation allows to form pairs of most similar sessions, that are subsequently aggregated using the specialization relation. The \( \theta \) relation used for grouping corresponds to \( \theta_{\text{sim}} (s, s') \equiv \exists s'' \in L \) such that \( \text{sim}(s, s') > \text{sim}(s, s'') \) and \( \text{sim}(s'', s'') > \text{sim}(s, s') \). The aggregation function merges pairs of sessions into a session more general than the two elements of the pair. It uses a tilted-time window [12] to favor a precise summary in proportion as queries are recent between the sessions. The interest being to provide the least amount of information on old queries. It can be declaratively written as follows: \( \text{agg}_{\text{merge}} (s, s') = \{s''\} \) such that both \( s'' \succeq s \) and \( s'' \succeq s' \) and for all \( i \in [1, |s''|] \), \( s'' \) is the least session w.r.t. \( \geq \) such that both \( (q''_i) \succeq s_i \) and \( (q''_i) \succeq s'_i \) where the \( q''_i \)'s are the queries of \( s'' \) and \( s_i \) is the portion of a session \( s \) that corresponds to \( q''_i \) in the sense of the tilted-time window.

The \( \pi \) operation is a generalization of the basic summarization operation, and summarizes a log so as to obtain only one session generalizing all the sessions of the log. It is implemented as an ascendant hierarchical clustering that at each steps uses \( \theta_{\text{sim}, \text{agg}_{\text{merge}}} \) to group pairs of sessions. When Falseto is started, it computes the complete clustering of the log, stores the resulting dendogram and displays a summary of the log in its log viewer. This summary is not the session that summarizes the complete log since such a session would most likely be too much general to be useful. Instead, Falseto proposes nodes of the dendogram that correspond to summarized sessions that contains at least one query that is not the most general one. To save space on the viewer, these nodes are represented as tag clouds, where the tags are the query fragments. On demand, the user can see the details of the corresponding cluster. The dendogram can also be browsed: each cluster displayed on the viewer can be split to display its two direct descendants. This “drill-down” browsing operation exploits the results of a \( \pi \) operation, in the sense that, from any session \( s \) of the dendogram resulting of \( \pi \), the pair of sessions summarized with \( \text{agg}_{\text{merge}} \) to obtain \( s \) is given.

Recommender system

Finally, the Falseto recommender system [3, 2] can be described by the three following phases:

1. The \( \text{Alignment} \) phase selects from the log \( L \) a set of sessions that are similar to the current session \( s_{\text{cur}} \) and determines from them a set \( F \) of potential futures for \( s_{\text{cur}} \). This is implemented using a join operation \( F = \{s_{\text{cur}}\} \bowtie_{\text{sim}, \text{future}} L \) where \( \text{sim} \) is the similarity relation and \( \text{future} \) is the function that determines if a session in \( L \) is a potential future for \( s_{\text{cur}} \).

2. The \( \text{Ranking} \) phase chooses a base recommendation \( r \) as a subsession of a candidate recommendation in \( F \) by considering both its similarity with \( s_{\text{cur}} \) and

\[ \]
its frequency in \( L \). It is implemented by \( \{ r \} = \pi_{\text{all}, \text{agg}, \text{rank}}(F) \) where all \((s, s')\) holds for any \( s, s' \) of \( F \) and agg,rank is the function that compares pairwise the subsessions of \( F \) and returns the most relevant one.

3. The Fitting phase adapts \( r \) to \( s_{\text{cur}} \) by looking for relevant patterns in the query fragments of \( s_{\text{cur}} \) and \( r \) and using them to make changes to the queries in \( r \), so as to deliver a recommendation \( r' \). It is implemented by a join \( \{ r' \} = \{ s_{\text{cur}} \} \bowtie_{\text{all}, \text{fit}} \{ l \} \) where \( l \) is the log session from which \( r \) is extracted and \( \text{fit} \) is the function used for adapting \( r \) to \( s_{\text{cur}} \).

We close this section by noting that the holistic functionalities of Falseto, and therefore the specialization relations and similarity relations so far used in Falseto, are exclusively based on the query expressions. This is done for the sake of achieving an interactive response time. Indeed, if query answer is used instead of query syntax, this would have a substantial impact on the time taken to compute a recommendation or summary of the log. Studying performant ways of incorporating query answers in the holistic functionalities of Falseto is part of our future work.

5. EXPERIMENTS

We tested our approach by running experiments with OLAP users analyzing the cube whose schema is described in Section 2. We first describe the test protocol and then report the lessons learned.

Our objectives for running these tests were to obtain real OLAP sessions and automatically assess Falseto’s holistic functionalities using these sessions. Due to the limited time we had with the users, the goal was not to ask the users to directly assess Falseto’s holistic functionalities. Such assessment is part of our future work as explained in the following section.

5.1 Test protocol

The test consisted of developing sessions for four analytical questions on the IPUMS cube:

1. (Q1) Is there a trend in the evolution of the average cost of gas for some profiles?
2. (Q2) Are energy costs following the evolution of the average income for some profiles?
3. (Q3) What are the individual profiles to target for a campaign of energy cost reduction (i.e., those who pay too much)?
4. (Q4) Where is it better to live in terms of energy costs, for an individual profile?

Note that these questionnaires can be found online via the Falseto web site [8].

The 17 OLAP users engaged in the test were students of Master’s programs specialized in BI: 7 of them were students of the European Erasmus Mundus IT4BI program (http://it4ibi.univ-tours.fr/) and 10 were students of the French program in Information systems and BI of the University of Tours. The test was not part of the program, was not graded and all the participants were volunteers.

Students were first shown the Falseto tool (mostly the query design functionalities) and then they had 2 hours to develop sessions. Then they were asked to identify in the sessions relevant queries, in the sense that the results of these relevant queries should be included in a report summarizing the session. They were also asked to justify the relevance of these queries.

With this corpus of annotated sessions, we tested Falseto’s holistic functionalities as follows. We developed different naive algorithms using Falseto’s holistic functionalities to automatically generate sessions, and we compared these synthetic sessions with the real ones. More precisely:

1. To understand Falseto’s log search capacities, let \( A \) be an analytical question and \( A_i \) be the subset of log queries developed for \( A \). We first randomly choose a query among the initial queries of \( A_i \). Then, with an increasing probability for ending the session, at each step the current query is used to filter the log and retrieve among the resulting session the query that is the closest to the current one. This query becomes the new current query (and is removed from the log). For these generated sessions, we report the recall and precision compared to the log queries, computed as follows. Recall \((R)\) is the number of queries of the generated sessions that also belong to \( A_i \) (noted \( A_j \)) over the total number of queries in \( A_i \), \( R = \frac{|A_j|}{|A_i|} \). Precision \((P)\) is the number of queries of the generated sessions that also belong to \( A_i \) \((A_j)\) over the total number of queries in the generated session \((S_j)\), \( P = \frac{|A_j|}{|S_j|} \). We also measure recall and precision w.r.t. relevant queries. In this case, Recall is the number of relevant queries in the generated session \((R_j)\) over the number of relevant queries in \( A_i \) \((R)\), \( R' = \frac{|R_j|}{|R|} \), while precision is the number of relevant queries in the generated session \((S_j)\) over the number of queries in the generated session \((S)\), \( P' = \frac{|S_j|}{|S|} \).

2. To understand Falseto’s capacities of suggesting new information (i.e., not found in the log) when answering an analytical need, we generated sessions by picking a session in the log, removing the final part of the session and replacing it with the session suggested by Falseto’s recommender system. For this we report the session novelty (the fraction of the generated session that corresponds to queries that cannot be found in the log), the coverage (fraction of times the recommender has something to suggest) and the average similarity of the generated session to the sessions developed for answering the analytical question (to make sure that the generated session does not diverge from its goal).

3. Finally, to understand the overall benefit of Falseto assuming that the recommender system is used in combination with the log search facility, we use Falseto’s recommender system on the sessions generated with the first test, and we report the metrics listed above.

5.2 Lessons learned

We begin by describing the log obtained. 27 sessions (308 queries, 108 of them are relevant) were collected. A global analysis of the overall log is reported in Table 1 (effort is the inverse ratio of relevant queries per session, i.e., the number of queries it takes to obtain one relevant query).
Analysis of the log min max avg std dev.

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>max</th>
<th>avg</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of session</td>
<td>4</td>
<td>31</td>
<td>11.4</td>
<td>6.35</td>
</tr>
<tr>
<td>Relevant queries / session</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>2.61</td>
</tr>
<tr>
<td>Ratio of relevant queries</td>
<td>0.12</td>
<td>0.71</td>
<td>0.37</td>
<td>0.19</td>
</tr>
<tr>
<td>Effort</td>
<td>1.4</td>
<td>8</td>
<td>3.08</td>
<td>1.76</td>
</tr>
</tbody>
</table>

Table 1: Analysis of the log

Figure 5: OLAP operations for a typical session

Figure 5 shows the use of OLAP operations along an average session. It plots the average number of OLAP operations used between two consecutive queries, the average aggregation level and the average number of filters of queries, and the average number of relevant queries for the sessions of the log (all sizes normalized). The aggregation level indicates the position in the group-by set lattice and varies between 0 (coarsest level) and 12 (finest level). The number of filters indicates the number of filters per hierarchy and varies between 0 (no selection) and 5 (selections on all dimensions). The overall pattern is that the number of filters and level of detail increase along the session, while the number of OLAP operations between consecutive queries varies before a final drop at the end of the session. This indicates that users favor mainly a top-down exploration whose aim is to collect filters. Interestingly, the curve for the average number of OLAP operations and that of average number of relevant queries are mostly inversely correlated. When the average number of operations increase, the average number of relevant queries decrease, and vice versa. This may indicate that relevant queries are found when search is more focused. In particular, the final drop of the number of operations seems to indicate that the sessions become more focused in the end.

Figure 6 reports the distribution of the positions of relevant queries (in red) in the sessions. Note that given the relative small size of the log, manual inspection was possible and done, and the queries in yellow color (around 9%) indicate the presence of a query that does not seem to contribute to the session (being a repetition of another query of the session, or comporting edition mistakes like forgetting to incorporate the good measure attribute to analyze).

An analysis by question is reported in Table 2. We have characterized the average similarity of the sessions in the log, using the similarity defined in [5], which is quite low (0.14) as expected, as well as the intra-question similarity and the inter-question similarity. It can be noted that the intra-question similarity shows important variations, which indicates that sessions developed for answering a given analytical question can be quite different. However, this intra-question similarity is almost always greater than the average session similarity of the complete log. A notable exception is Q3, where the intra-question similarity (0.13) is lower than the average similarity to the sessions of both questions 2 (0.15) and 4 (0.16). Interestingly, this is the question that generated the highest number of queries per sessions, which can be understood as the difficulty of the question.

We then report the results of the 3 naive automatic session generation algorithms described above. For each of the questions and each of the tests, 100 sessions were generated. The result of the first test, aiming at understanding Falseto’s log search capacity, is reported in table 3, that shows recall

<table>
<thead>
<tr>
<th>Question analysis</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sessions</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Number of queries</td>
<td>82</td>
<td>57</td>
<td>86</td>
<td>56</td>
</tr>
<tr>
<td>Number of relevant queries</td>
<td>26</td>
<td>21</td>
<td>34</td>
<td>27</td>
</tr>
<tr>
<td>Queries / session</td>
<td>10.25</td>
<td>8.14</td>
<td>14.33</td>
<td>11.2</td>
</tr>
<tr>
<td>Relevant queries / session</td>
<td>3.25</td>
<td>3</td>
<td>5.67</td>
<td>5.4</td>
</tr>
<tr>
<td>Intra-question similarity</td>
<td>0.55</td>
<td>0.16</td>
<td>0.13</td>
<td>0.21</td>
</tr>
<tr>
<td>Inter-question similarity</td>
<td>Q1 0.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q2 0.02</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q3 0.11</td>
<td>0.15</td>
<td>0.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q4 0.14</td>
<td>0.08</td>
<td>0.16</td>
<td>0.21</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Question analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>0.16</td>
<td>0.91</td>
<td>0.29</td>
<td>0.93</td>
<td>0.39</td>
<td>0.95</td>
</tr>
<tr>
<td>Q2</td>
<td>0.11</td>
<td>0.80</td>
<td>0.14</td>
<td>0.67</td>
<td>0.14</td>
<td>0.57</td>
</tr>
<tr>
<td>Q3</td>
<td>0.05</td>
<td>0.79</td>
<td>0.08</td>
<td>0.69</td>
<td>0.11</td>
<td>0.64</td>
</tr>
<tr>
<td>Q4</td>
<td>0.04</td>
<td>0.38</td>
<td>0.08</td>
<td>0.35</td>
<td>0.09</td>
<td>0.31</td>
</tr>
</tbody>
</table>

Table 3: Log search capacity
Gb of RAM and a 2.5 GHz Intel Core i5.

The tests were run on MAC-OSX Mavericks with 16

Table 4: Recommender system capacity

<table>
<thead>
<tr>
<th>Recommender system</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average novelty</td>
<td>0.25</td>
<td>1</td>
<td>0.42</td>
<td>0.39</td>
</tr>
<tr>
<td>Average coverage</td>
<td>0.88</td>
<td>1</td>
<td>0.68</td>
<td>1</td>
</tr>
<tr>
<td>Average recommendation similarity to question</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommendations for Q1</td>
<td><strong>0.65</strong></td>
<td>0.03</td>
<td>0.14</td>
<td>0.27</td>
</tr>
<tr>
<td>Recommendations for Q2</td>
<td>0.03</td>
<td><strong>0.33</strong></td>
<td>0.20</td>
<td>0.15</td>
</tr>
<tr>
<td>Recommendations for Q3</td>
<td>0.12</td>
<td>0.15</td>
<td><strong>0.28</strong></td>
<td>0.21</td>
</tr>
<tr>
<td>Recommendations for Q4</td>
<td>0.19</td>
<td>0.12</td>
<td>0.23</td>
<td><strong>0.39</strong></td>
</tr>
</tbody>
</table>

Table 5: Combining search facilities with recommendations

<table>
<thead>
<tr>
<th>Search and recommendations</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average recall</td>
<td>0.32</td>
<td>0.16</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>Average precision</td>
<td>0.92</td>
<td>0.61</td>
<td>0.61</td>
<td>0.34</td>
</tr>
<tr>
<td>Average novelty</td>
<td>0.08</td>
<td>0.54</td>
<td>0.33</td>
<td>0.5</td>
</tr>
<tr>
<td>Average coverage</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Average recommendation similarity to question</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommendations for Q1</td>
<td><strong>0.66</strong></td>
<td>0.03</td>
<td>0.14</td>
<td>0.15</td>
</tr>
<tr>
<td>Recommendations for Q2</td>
<td>0.02</td>
<td><strong>0.33</strong></td>
<td>0.17</td>
<td>0.19</td>
</tr>
<tr>
<td>Recommendations for Q3</td>
<td>0.15</td>
<td>0.17</td>
<td><strong>0.23</strong></td>
<td>0.24</td>
</tr>
<tr>
<td>Recommendations for Q4</td>
<td>0.05</td>
<td>0.25</td>
<td>0.26</td>
<td><strong>0.29</strong></td>
</tr>
</tbody>
</table>

Finally, we tested the responsiveness of Falseto, to check if the holistic functionalities are compatible with an interactive use. The tests were run on MAC-OSX Mavericks with 16 Gb of RAM and a 2.5 GHz Intel Core i5.

(21)

Figure 7: Falseto’s responsiveness

Figure 7 shows the average time it takes to compute i) the initial summary using clustering (in blue), ii) a search in the log (in green) and iii) a recommendation (in yellow), for synthetic logs of increasing sizes (a maximum of 200 sessions corresponding to 1888 queries). It can be seen that log search and recommendation times are perfectly compatible with an interactive use (the maximum average was 152 ms in our tests, for computing recommendations with a log of 200 sessions). The time to compute the initial clustering is much more substantial and increases with the size of the log, but it remains acceptable since it is executed only once when Falseto is started. This test confirms Falseto’s responsiveness, and, importantly, justifies the use of only the query syntax (and not the query answer, as explained in Section 4) in the holistic functionalities of Falseto.

6. BRIDGING THE GAP BETWEEN OLAP AND EXPLORATORY SEARCH

6.1 Related work

It is now admitted in the database community that query logs can be leveraged not only for physical tuning, but also for user empowerment [15]. Logs can indeed be used for learning user preferences [13], for query recommendation [7, 22], for query auto-completion [16], or for query composition [17]. In particular, the study of [17] showed that SQL query composition time can be heavily reduced when users are provided with a tool to browse, search and reuse former SQL queries organized in sessions.

In the OLAP context, the navigational nature of analytical sessions over data cubes is well known [19], and approaches have been developed for improving query performance holistically at the session level [19, 14], or suggesting queries or facts to assist the user in her navigation [20, 10]. Noticeably, while many efforts have been devoted to speedup OLAP query processing, the question of how effective an OLAP session has attracted little attention, while it is predominant in other fields like information retrieval or even more specifically, exploratory search.

Exploratory search [23] can be defined as a search paradigm centered on the user and the evolution of her knowledge, aiming at a better support for information understanding by moving beyond the traditional query-browse-refine model. As illustrated by Figure 8, exploratory search consists of two main activities: exploratory browsing and focused searching. Exploratory browsing essentially helps relate the problem context to similar documented experiences, while focused searching is generally intended to help the
user follow a known or expected trail. Exploratory search is particularly driven by the quality of the user’s experience, and suitable metrics for measuring it have been proposed \cite{23}: engagement and enjoyment (the number of actionable events), information novelty (the rate at which new information is encountered), task success (do users reach a particular target and get enough detail en route to their goal), task time (the time spent to complete the task), and learning and cognition (the evolution of individual knowledge).

Exploratory search attracted the interest of the database community, and was even the topic of a recent post on the SIGMOD blog (http://wp.sigmod.org/) with a specific section that compares it to OLAP. The conclusion of this section is that OLAP has serious limitations in supporting exploratory search, especially in the domain of discovery, adaptability to new information needs, and the support of all types of relevant data and users. We next discuss how Falseto could help to bridge the gap between OLAP and exploratory search, and overcome these limitations.

### 6.2 Bridging the gap

Figure 8 can be used as a first driver for bridging the gap between OLAP and exploratory search. Interestingly, the test we did with users showed that OLAP sessions could follow this model (see Figure 5). The two languages implemented by Falseto and presented in Section 3 can be seen as targeting the two activities of exploratory search. On the one hand, the query edition language can be used for focused searching, and on the other hand, the log manipulation language can be used to describe exploratory browsing tasks that take advantage of former sessions. With this idea in mind, we designed a simple protocol to understand, via user tests, how Falseto can be used to better support exploratory search in the context of OLAP. This protocol relies on production of sessions in response to a given analytical question, and production of reports by the inclusion and comment, in a document, of the relevant queries from the session.

We first start by adapting to the OLAP context the metrics introduced above for assessing exploratory search applications. These metrics could be computed as follows:

- User engagement is measured as the number of queries found relevant and included in the report created while answering a given analytical question. It is also measured as the number of clicks on the holistic functionalities.
- Information novelty is measured as the number of queries in a report that are relevant and that appear in no other reports of the same analytical need.
- Task time is measured as the time taken to complete a session (to answer the analytical question). The time spent on holistic functionalities (exploratory browsing) is also reported. We also measure the user effort as the ratio between the number of queries in the session and the number of queries in the report.
- Task success is measured as the precision and recall of the report that answers to a given analytical question. Precision is the number of relevant queries in a report over the overall number of queries in the report. Recall is the number of queries of a report that appear in (or that are included in a query of) another report.
- Finally, learning and cognition can be measured by recording a user profile and assess i) how this profile evolves through consecutive uses of Falseto and ii) if this user profile leads to improvements of the other metrics through consecutive uses of Falseto.

We now sketch the protocol more precisely. The complete protocol can be downloaded from the Falseto website \cite{8}. It corresponds to a test that is organized in three phases. The first phase aims at giving the analysts the sufficient knowledge to use both the basic and holistic functionalities of Falseto. The analysts are asked to complete the Falseto tutorial. After this phase, the test supervisors should constitute three groups of analysts (named Basic, Advanced and Complete from now on), ensuring in each group an even distribution of skills with Falseto.

In the second phase, the analysts in the Basic group, will use a basic version of Falseto that does not include the holistic functionalities. In the Advanced group, the analysts will use a version of Falseto that only includes the holistic functionalities, but does not allow the analyst to edit queries. In the Complete group, the analysts will use the complete version of Falseto that includes all the functionalities. All analysts are asked to solve an exercise consisting of two analytical questions that can be solved by navigating the data cube. The second need is deliberately more fuzzy than the first one, and probably requires a longer navigation to get answered. The analysts should answer one requirement after the other. The form of the answer to the question is twofold: the log that records the session (automatically generated by Falseto) and a report. This report is a document that contains a list of annotated query results, taken from the session.

During the last phase, each analyst should evaluate all the reports that were produced to answer a particular analytical question. They should indicate, for each query, if this query is relevant (does it contribute to answer the need, according to the analyst) and mark all the reports that include this query. If a report does not include the query strico sensu but a superset of it (for instance the query filters out one year but the report includes a query that includes all the years but which apart from that is exactly the same) this report should also be marked as including the query.
Implementing this protocol is part of our future work, and will allow to compute the metrics introduced above. As mentioned already, OLAP is limited in its exploratory search capabilities in terms of discovery, adaptability to new information needs, and the support of all types of relevant data and users. Implementing this protocol will at least enable to answer to the limitations in terms of discovery (can Falseto lead the user towards interesting findings they would have missed without Falseto) and in term of supporting different types of users (naive and expert).

7. CONCLUSION

This paper introduced Falseto, an OLAP tool that is meant to facilitate OLAP query and session composition. During a session with Falseto, the user can summarize, browse, query and reuse former sessions recorded in a query log, and can take advantage of sessions suggested by Falseto's recommender system. This tool relies on a logical framework that includes a language for editing queries and a language for manipulating logs. We report the lessons learned from analyzing real OLAP sessions and using them to assess Falseto's functionalities. Finally, we discussed how Falseto can link OLAP to the field of exploratory search, a search paradigm centered on the evolution of the user's understanding.

Falseto can be seen as the starting point of a platform for developing user-centric OLAP facilities. The first short-term perspective of this work is to better assess Falseto's holistic functionalities with user tests, so as to improve the existing functionalities and devise new ones. A second short-term perspective is to make Falseto compatible with logs provided by standard OLAP tools. The challenge being to translate the queries of these logs according to the format supported by Falseto, by limiting information loss. Another short-term perspective is to find better ways of graphically displaying sessions and summaries. On the long run, our aim is to be able to benchmark OLAP activities in terms of effectiveness, which would be possible by bridging the gap between OLAP and exploratory search.

8. REFERENCES


