Benchmarking Exploratory OLAP

Mahfoud Djedaini, Pedro Furtado, Nicolas Labroche, Patrick Marcel, Veronika Peralta

Benchmarking exploratory OLAP

Mahfoud Djedaini[1], Pedro Furtado[2], Nicolas Labroche[1], Patrick Marcel[1], and Verónika Peralta[1]

[1] University of Tours, France
firstname.lastname@univ-tours.fr
[2] University of Coimbra, Portugal
pnf@dei.uc.pt

Abstract. Supporting interactive database exploration (IDE) is a problem that attracts lots of attention these days. Exploratory OLAP (On-Line Analytical Processing) is an important use case where tools support navigation and analysis of the most interesting data, using the best possible perspectives. While many approaches were proposed (like query recommendation, reuse, steering, personalization or unexpected data recommendation), a recurrent problem is how to assess the effectiveness of an exploratory OLAP approach. In this paper we propose a benchmark framework to do so, that relies on an extensible set of user-centric metrics that relate to the main dimensions of exploratory analysis. Namely, we describe how to model and simulate user activity, how to formalize our metrics and how to build exploratory tasks to properly evaluate an IDE system under test (SUT). To the best of our knowledge, this is the first proposal of such a benchmark. Experiments are two-fold: first we evaluate the benchmark protocol and metrics based on synthetic SUTs whose behavior is well known. Second, we concentrate on two different recent SUTs from IDE literature that are evaluated and compared with our benchmark. Finally, potential extensions to produce an industry-strength benchmark are listed in the conclusion.

1 Introduction

Supporting exploration of databases is of prime importance, especially in a context of big, distributed and heterogeneous data, as shown in a recent survey of the topic [17]. Both researchers and companies that supply data analysis tools are increasingly focused on mechanisms for improving user experience, in particular aids for effective data exploration. As researchers and companies implement, test and tune alike their Interactive Data Exploration (IDE) solutions, a major issue they face is how to assess and compare solutions, improvements and alternatives.

While there exist a set of benchmarks recognized by the database community as relevant for evaluation and comparison of performance of database systems, such as the benchmarks from TPC organization, there is yet no commonly agreed upon benchmark for evaluating to what extent database systems help the user during data exploration. Our objective is to propose such a benchmark as,

roughly speaking, actual TPC benchmarks assess data retrieval, and not data exploration.

In this work, we focus on the context of OLAP analysis of data, as an important use case of IDE, first because exploration of data has been deeply studied in this context, and second because to our opinion OLAP is an ideal first step before generalizing to other database systems. OLAP is defined as the process of analyzing multidimensional datasets (cubes), online, interactively, summarizing key performance indicators (called measures) from different perspectives or axes of analysis (called dimensions).

In order to motivate our work, let's now consider the following toy example: a user navigating OLAP sales cube faces an unexpected difference between sales in year 2014 and year 2015 for a product P in France. The user will then explore the surrounding region of the cube by means of OLAP operators such as roll-up (at the Europe level for example), drill-down (at the month level for example) and slices (for other products) to find evidences that may explain and corroborate the first fact. The user might even get some support from a system that automatically proposes next moves in the analysis [1,13]. We consider that the surrounding region of the first interesting fact corresponds to a neighborhood that has to be covered to ensure the exploration task success. If one wants to evaluate this particular data exploration, it is then possible to measure several metrics such as the number of queries that the user needed to cover this neighborhood, the ratio of this area that has been finally discovered, the ratio of the rest of the cube that the user had to visit to reach this result etc. So far, the assessment of data exploration through quality measures has been overlooked by the database community, but we can benefit from experience in the fields of information retrieval and exploratory search [33], which are particularly driven by the quality of the user's experience and metrics for measuring it.

This paper covers all the aspects of the implementation of a data exploration benchmark for OLAP. This benchmark was designed with a set of guiding principles in mind. It has to be easy to use by anyone, considering that a developer or researcher working in an OLAP exploration tool or algorithm, should be able to quickly plug his approach to the benchmark and use it, without requiring complex development or setting up of schema, dataset or OLAP exploration characteristics. The benchmark should also return objective evaluation metric results that are independent of the approach being tested. This means that both the mechanisms of the benchmark and the evaluation metrics must be agnostic of the IDE approach. Therefore, the benchmark includes generating skewed data with interesting facts, generating past query logs on this data and simulating OLAP users to evaluate a System Under Test (SUT) that provides support for next data analysis moves to the user. The benchmark can be used with any SUT, to evaluate any strategy that one may design. It reports objective measures for a set of metrics that characterize the degree to which the SUT fulfills certain objectives. Extensive experiments have been conducted to validate our benchmark proposal. First we evaluate the benchmark protocol and metrics based on synthetic SUTs whose behavior is well known. Second, we show that it is possible

		Input			
	Category	DB instance	Query log	Current query	Output
[5]	Automatic exploration	✓		✓	Tuples
[1]	Automatic exploration		✓	✓	Sequence of queries
[12]	Automatic exploration	✓	✓	✓	Queries
[3]	Automatic exploration		✓	✓	Queries
[13]	Visual optimization Automatic exploration	✓		✓	Queries result highlighting
[14]	Visual optimization			✓	Query
[29]	Data prefetching	✓	✓	✓	Tuples
[28,30]	Data prefetching	✓		✓	Tuples
[18]	Data prefetching	✓		✓	Sequence of queries
[27]	Data prefetching		✓	✓	Queries

Table 1. Interactive cube exploration techniques signatures

and meaningful to compare two state-of-the-art SUT from IDE literature [1,13] with our benchmark.

The paper is organized as follows. Section 2 discusses related work. Section 3 explains how interactive explorations can be scored and defines the benchmark metrics. Section 4 presents the benchmark itself. Experimental results are discussed in section 5. Finally, section 6 concludes the paper with considerations on potential extensions to produce industry-strength benchmark. A long version of this paper, with additional details, examples and tests, is available in [8][3].

2 Related Work

The variety of database exploration approaches Many approaches have recently been developed to support interactive database exploration (IDE), as illustrated by a recent survey of the topic [17]. Techniques range from Visual optimization (like query result reduction [4]), automatic exploration (like query recommendation [9]), assisted query formulation (like data space segmentation [31]), data prefetching (like result diversification [19]) and query approximation [16]. The core of most of these approaches consists of a function that, given the database instance and users' history with the database (i.e., past and current queries), computes new relevant queries, tuples or visualizations that are meant to support user exploration.

Given the exploratory nature of OLAP analysis of multidimensional data (see e.g., [29,18]), many exploration techniques have been specifically developed in the context of interactive OLAP exploration of data cubes. Table 1 lists these exploration approaches, indicating their categories (in terms of those proposed in [17]), and details their inputs and outputs. For instance, the PROMISE prefetching approach [27], that predicts a query based on a Markov Model constructed out of the server's log, corresponds to a function with signature $\langle L, \langle q \rangle \rangle \rightarrow \langle q' \rangle$, where L is the query log, q is the current user query and q' is the predicted query.

[3] More information on the benchmark can be found on its web page: http://www.info.univ-tours.fr/~marcel/benchmark.html

Measuring the quality of an exploration Measuring the quality of exploration has attracted a lot of attention in Information Retrieval, in particular in the field of Exploratory Search[4] [33] that can be defined as a search paradigm centered on the user and the evolution of their knowledge. It is particularly driven by the quality of the user's experience, and metrics for measuring it have been categorized as follows. **Engagement and Enjoyment** measures the "degree to which users are engaged and are experiencing positive emotions". It includes "the amount of interaction required during exploration", the "extent to which the user is focused on the task". **Task Success** assesses "whether the user reaches a particular target" and finds a "sufficient amount of information and details" along the way. **Information Novelty** measures the "amount of new information encountered". **Task Time** measures the "time spent to reach a state of task completeness". **Learning and Cognition** measures the "attainment of learning outcomes", "the amount of the topic space covered" and "the number of insights acquired". While these categories have been proposed in the context of web search, they make perfect sense for interactive database exploration, and we next focus on measures that have been proposed in the literature in these categories.

User engagement measures are popular in web search to measure how a user is engaged in using a website or search engine. Many implicit measures have been proposed [22] to track online behavior. These measures are classically categorized in activity (how a website is used), loyalty (return of users to a website) and popularity (how much a website is used). While loyalty and popularity essentially make sense for relative comparison of websites, activity enables measuring engagement for a particular website independently of other websites. The most commonly used activity metrics include number of queries per session, number of clicks, number of clicks per query, dwell (presence) time (see e.g., [10,32]).

Task success is well studied in information retrieval, with even conferences devoted specifically to this, like the TREC conference[5]. Task success is traditionally measured with precision/recall-based measures, which supposes that the target of the task is known. In this case, roughly speaking, recall measures how complete the answer to a query is, while precision measures how noisy the answer to a query is.

Many works have been interested with measuring information novelty in relational databases. For instance, in [11], the authors propose to describe the data space covered by a session with a vector of the tuples accessed by the queries of the session. In [23], the authors propose the notion of access area to capture the portion of the dataset a user is interested in. In [19], the authors use a similar notion to propose query result diversification. In data cubes exploration, Sarawagi [29] assimilates novelty with the most informative constraints so that the expected distribution of a cube's cell values - based on a maximum entropy principle - is closer to the actual observed values. Here, a constraint is defined at an aggregate level of the observed cells and is expressed as a sum over the values

[4] http://wp.sigmod.org/?p=1183
[5] http://trec.nist.gov/

of a subset of the observed cells. It is then expected that bringing more constraints modifies the expected distribution of values and thus allows to reduce the divergence between the observation and the expectation. The constraints that best reduce this divergence is declared to be the most informative.

Measuring task time may seem straightforward, but one needs to carefully define what is timed and how to report it. Performance related metrics like query per hour can be adapted from TPC benchmarks to this end.

Finally, measuring learning and cognition has attracted lots of attention in learner models [7]. Learner models are central components of intelligent tutoring systems, that infer a student's latent skills and knowledge from observed data. A very influential and widespread accepted model is the Knowledge Tracing model [6]. Knowledge tracing is a Bayesian network allowing to measure the probability that a skill is mastered when resolving a problem (opportunity to use the skill). The model relies on four parameters, usually experimentally tuned: $P(L_0)$: the probability the skill is already mastered before the first problem, $P(T)$: the probability the skill will be learned at each opportunity to use the skill (transition from not mastered to mastered), g: the probability the resolution is correct if the skill is not mastered (guess), s: the probability a mistake is made if the skill is mastered (slip). The probability that the skill L at opportunity n is mastered is the probability the skill is learned at step $n-1$ or not learned at step $n-1$ but learned at this step n. It can be computed as:

$$P(L_n|X_n = x_n) = P(L_{n-1}|X_n = x_n) + (1 - P(L_{n-1}|X_n = x_n)) \times P(T) \quad (1)$$

where:

$P(L_{n-1}|X_n = 1) = \frac{P(L_{n-1})(1-s)}{P(L_{n-1})(1-s)+(1-P(L_{n-1}))g}$

$P(L_{n-1}|X_n = 0) = \frac{P(L_{n-1})s}{P(L_{n-1})s+(1-P(L_{n-1}))(1-g)}$

$X_n = 1$ (resp. 0) means problem n has been solved (resp. not solved).

Current benchmarks for decision support, big data and analytic workloads TPC proposes a number of popular benchmarks and metrics for assessing the performance of database systems, covering time, performance, price, availability or energy consumption (see Table 2). However, while TPC acknowledges the importance of the explorative nature of decision support queries (see e.g., the OLAP interactive queries in the TPC-DS benchmark), none of the existing TPC metrics are appropriate for measuring database exploration support in the sense of the categories proposed in Exploratory Search. A recent benchmark targets analytical workloads [21], but it too overlooks assessing the quality of interactive data exploration by proposing metrics covering only query response time, tuning overhead, data arrival to query time, storage size and monetary cost.

OLAP exploration as a relevant use-case Interestingly, the literature on OLAP already provides the building blocks for benchmarking cube exploration. OLAP has been the subject of specific benchmarks, like the TPC-H-based Star Schema Benchmark (SSB) [24]. SSB models a realistic use case of sales analysis, for which

Metrics	TPC-H	TPC-DS	TPC-DI	TPCx-HS	TPCx-BB
Query per hour/minute	✓	✓		✓	✓
Price/performance	✓	✓	✓	✓	✓
Availability date	✓	✓	✓	✓	✓
Power/performance	✓	✓		✓	✓
Power	✓				
Throughput	✓	✓	✓	✓	✓
Load time		✓			✓
Power test elapsed time		✓			✓

Table 2. Metrics of relevant TPC benchmarks

realistic instances with skewed data can be produced with the PDGF data generator [25]. Realistic OLAP workloads can be generated with the CubeLoad session generator [26]. CubeLoad takes as input a cube schema and creates the desired number of sessions according to templates modeling various user exploration patterns: users with limited OLAP skills pursuing a specific analysis goal, more advanced users navigating with a sequence of slice and drill operations, users tracking unexpected results with explorative sessions. OLAP literature also provides techniques for characterizing analytic behaviors [27,3]. In these works, the user's behavior is defined as a Markov model, whose states are built from the past queries of the user, and the transitions between states are weighted by the probability of observing a query after another in the user's query log. Finally, OLAP literature also provides characterizations of interesting data in the multidimensional space. Discovery-driven analysis of data cube [28,29,30,5] aimed at measuring potentially surprising data, knowing already evaluated queries. These work characterize surprising data as being groups of tuples that are connected (usually one OLAP operation apart), and that, taken altogether, appear to be meaningful (usually unexpected, in the sense of e.g., information theoretic measures).

3 Evaluating an exploration

This section describes how interactive explorations can be scored, by implementing the metrics related to user experience identified in the previous section. We first start with presenting formally the definition of an exploration in an OLAP context. A complete formal framework, with illustrative examples, can be found in the full paper [8].

3.1 Exploration in an OLAP context

Our benchmark incorporates the explorative and interactive nature of IDE by considering user sessions as first class citizens. We define a session $s = \langle q_1, \ldots, q_k \rangle$ of length $|s| = k$ as a sequence of k OLAP queries over a data warehouse and a log as a finite set of sessions. In what follows, a log can be associated to one particular user profile (representing this user's activity) or can represent the overall activity (being the union of all user logs).

Without loss of generality, the OLAP queries we consider are dimensional aggregate queries over a data cube [15]. A query is defined as a group by set (identifying the query granularity) and a set of Boolean predicates, one for each hierarchy. During their sessions, after each query is processed, users inspect the cube cells retrieved by the query. A cell c is an element of a cube that has a particular coordinate and a measure value. The answer to a query q, denoted $answer(q)$, is the set of retrieved cells whose coordinates are defined by the query group by set and selection predicates. Thanks to the the popular OLAP operations (roll-up, drill-down, slice-and-dice), users navigate the cube by exploring cells neighborhood, querying at coarser granularity (roll-up), finer granularity (drill-down) or reaching siblings in a hierarchy. This is formalized using classical relations between cells. For two cells c and c', we note $c \succeq c'$ if c' is a roll-up of c, i.e., a coordinate of c' is an ancestor of that of c in some hierarchy, and we note $c \approx c'$ if the two cells are siblings, i.e., their coordinates differ only in one sibling position in some hierarchy.

Definition 1 (Neighborhood of a cell). *The rollup (resp., drill-down, sibling) neighborhood of a cell c is the set of all cells c' such that $c' \succeq c$ (resp. $c \succeq c'$, $c \approx c'$). The OLAP neighborhood of a cell c is the union of its rollup neighborhood, its drilldown neighborhood and its sibling neighborhood.*

The neighborhood of a group of cells C, noted $neighborhood(C)$ is the union of the neighborhoods of each cell of the group. Intuitively, the neighborhood of a group of cells defines a zone of the cube to be explored to analyze this group of cell.

A user is represented by a log, i.e., the user's past explorations. This allows to characterize a user's behavior by constructing a generative model, in the spirit of what has been successfully applied in OLAP for data prefetching [27].

Definition 2 (Generative model). *Let L be a set of sessions characterizing a user. The generative model to represent this user's behavior is a Markov model of order one, i.e., a graph $\langle S, P \rangle$ where S is the set of queries of L and $P : S \times S \rightarrow [0, 1]$ denotes the probability function for the state transition, computed as $P(q_1, q_2) = \frac{sessions(\langle q_1, q_2 \rangle)}{sessions(\langle q_1 \rangle)}$ where q_1 and q_2 are queries and $sessions(s)$ gives the number of sessions where the sequence s appears.*

Definition 3 (User). *Let S be a set of sessions and x be a percentage. A user u_x is a tuple $u_x = \langle s^{log}, s^{seed}, g \rangle$ where $S = s^{log} \cup s^{seed}$, $|s^{log}| = x \times |S|$ and g is the generative model built from s^{log}.*

Finally, a task $\langle s, u \rangle$ for a session consists of a set of cells to be analyzed by a user u. This set of cells is given under the form of a session s, i.e., consists of the cells retrieved by the queries of this session. This session is based on the user's seed sessions.

3.2 Metrics

As explained in Section 2, the benchmark metrics follow the categorization proposed in the field of Exploratory Search [33]. For each category, we propose a

primary metric and a secondary metric, with the idea that secondary metrics can be used to counterbalance primary ones. Metrics of different categories have been defined so that the overlapping between them is minimal: User engagement relates only to the number of queries, novelty to cells, task success to cell neighborhood and task time only to time. Only Learning and cognition overlaps with novelty and task success since it aims at measuring the skill of finding new and relevant information. In what follows, let $u = \langle s^{log}, s^{seed}, g \rangle$ be a user, let $t = \langle s^0, u \rangle$ be a task for user u and let $s = \langle q_1, \ldots, q_k \rangle$ be a session produced for the resolution of a task t.

User engagement and enjoyment We use two popular and simple activity metrics used in web search: click depth as primary metric, to represent overall activity, and number of clicks per query to represent how focused this activity is. Dwell time, another popular activity metric, better fits in the Task time category. In the web search context, a click correspond to following a hyperlink (i.e., an HTTP query). In the context of the benchmark, a click corresponds to a new query. The metrics are defined as follows:

- **Query depth (QD, primary)** $= k$, i.e., the number of queries in the session, needed for resolving a task.
- **Focus (F, secondary)** $= \frac{max(\{|focus(q)||q \in s\})}{|s|}$, where $focus(q) = \langle q, \ldots, q' \rangle \subseteq S$ such that for all $q_i, q_{i+1} \in focus(q)$ the cells retrieved by q_{i+1} are in one of the neighborhood of the cells retrieved by q_i. Intuitively, this is to measure for a query q, the length of the chain of queries starting from q that are successively distant of only one OLAP operation.

Information Novelty Capturing user interest in the data explored can be done by measuring the access area [23]. In our context, this access area would be the set of tuples (recorded in a fact table) contributing to form the cells of a query result. As this area corresponds to tuples that are not actually presented as answers to queries (since, being an OLAP context, these tuples are aggregated), data of interest is better captured with view area, i.e., the cells presented in the answers. This is defined by: given a set of query $Q = \{q_1, \ldots, q_n\}$, the view area of Q is $va(Q) = \bigcup_{q \in Q} answer(q)$.

In a view area, not all data is interesting in the sense that it brings novel knowledge. We measure interestingness degree as a simple normalized entropy: $interest(C) = \frac{(-\sum_{i=1}^{m} p(i) \log(p(i)))}{\log(m)}$, with $|C| = m$, $C(i)$ is the i^{th} value of the set C and $p(i) = \frac{C(i)}{\sum_{i=1}^{m} C(i)}$ denotes the i^{th} cell probability.

The primary metric then quantifies the amount of interesting data found in the session. The secondary metric measures the increase in view area compared to the user's log view area. They are defined as follows:

- **Relevant new information (RNI, primary)** $= 1 - avg_{q \in s}(interest(va(q)))$
- **Increase in view area (IVA, secondary)** $= \frac{|va(s) \setminus va(s^{log})|}{|va(s) \cup va(s^{log})|}$

Task Success Intuitively, a task consists of investigating what can be said of a group of cells C coming from a task $\langle s, u \rangle$. The extent to which a task is complete consists of assessing how much of the neighborhood of this group of cells has been retrieved during the resolution of the task. A simple way of measuring it is with recall and precision. Recall is the primary metrics since consistently with exploratory search, we consider OLAP navigation as a recall oriented activity (what matters most is to minimize the number of false negative). The metrics are defined as follows, for a group of cells C:

- **Recall (R, primary)** $= \frac{|va(s) \cap neighborhood(C)|}{|neighborhood(c)|}$
- **Precision (P, secondary)** $= \frac{|va(s) \cap neighborhood(C)|}{|va(s)|}$

Task Time Measuring task time is done by adapting metrics of existing TPC benchmarks. We need to measure the time for the SUT to produce its output and to process the queries needed for the resolution of the task. The primary metric comes from the TPC-DS benchmark and measures the number of queries per the time taken to resolve the task. The secondary metric simply measures the task elapsed time. The metrics are defined as follows:

- **Query per seconds (QpS, primary)** $= \frac{k}{\sqrt{T_o \times T_e}}$, where T_o is the overall time for the SUT to produce its outputs and T_e is the overall query execution time.
- **Task elapsed time (TET, secondary)** $= T_o + T_e$, where T_o is the overall time for the SUT to produce its outputs and T_e is the overall query execution time.

Learning and cognition We adapt the Knowledge Tracing (KT) model to our context: we assimilate the skill mastering with the ability of finding interesting and novel information in the neighborhood N_C of a group of cells C. In other words, $X_n = 1$ if the nth query finds at least one more unknown cell of N_C where novelty increases for those cells compared to query $n-1$. In this case, we say that the query is successful (from the learning point of view). It is 0 otherwise. The primary metric **Learning (L, primary)** is defined as in the classical KT model, see Equation 1 in Section 2. The challenge is then to define the four parameters of KT: g, s, $P(L_0)$ and $P(T)$ based on the user generative model (UGM) since it represents the past of the user.

- $P(L_0)$ is is the proportion of successful queries in the UGM.
- g (resp. s) is the probability in the UGM of passing from unsuccessful to successful (resp. from successful to unsuccessful) queries.
- $P(T)$ is defined as the average weighted position of successful queries in the sessions of the UGM, giving more importance to queries that happen earlier in the session.

The secondary metric measures the average progression of the learning curve. It is defined by the arithmetic mean of the proportional growth of the probabilities.

– **Learning growth rate (LGR, secondary)** = $\frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{P(L_i|X_i=x_i)-P(L_{i-1}|X_{i-1}=x_{i-1})}{P(L_{i-1}|X_{i-1}=x_{i-1})}\right)$ where n is the session length.

4 The Benchmark

In this section we define the interface between the SUT and the benchmark and how the benchmark runs an experiment.

4.1 Interfacing with a SUT

In order to assess a SUT, the benchmark, simulates a user and interacts with the SUT. The SUT first builds its inner structures, if any, and obtains input metadata from the benchmark. Conceptually, a SUT requires as input all or part of the following parameters: the database (schema and instance), user traces (i.e., sequences of queries collected into the query log) and the active user's current exploration (a sequence of queries). Let D denote the set of all database instances for a given schema, Q denotes the set of all possible queries over this schema, S denotes the set of all sequences of queries (i.e. $Q \times Q \times \ldots \times Q$), and 2^A denotes the power-set of a set A. The functionality of a SUT can be defined generically as doing the transformation: $\langle D, 2^S, S\rangle \rightarrow S$. Once the SUT is ready, the evaluation protocol starts resolving a task, successively calling the SUT to provide suggestions.

In practice, the benchmark is a Java program where SUTs can be plugged to be evaluated. Its code and javadoc are available for SUT programmers on BitBucket[6]. Basically, a SUT is sought twice, (1) before starting the evaluation so it can initialize, and (2) whenever a next move suggestion is requested. From the benchmark point of view, SUTs are only seen as black boxes that perform what they are asked to perform, through a contract. Practically, a SUT is a class that implements an interface that exposes two functions *readMetadata* and *nextSuggestion*. Function $readMetadata$ is called before starting the actual evaluation process, so the SUT can read and initialize its internal structure. Its parameter is a $Metadata$ object whose getters allow to access the cube, the list of users, past user traces, etc. Function $nextSuggestion$ is called many times during a task resolution. It provides to the SUT a given user and a current exploration (sequence of queries), and asks the SUT to suggest. It is the responsibility of the benchmark to orchestrate the whole process, and to make sure the functions are called with the right arguments.

4.2 How the Benchmark Works

The benchmark process is composed of three components. The first component initializes the benchmark. It generates the context: the database (i.e., the cube),

[6] https://bitbucket.org/mdjedaini/ea-benchmark

some sequences of queries (i.e., the log), data skews to simulate interesting observations, and creates user profiles. You do not need to run this component if you reuse an existing context, but you can also create a new context with different schema or generation parameters.

The second component is responsible for the actual evaluation of a SUT. The evaluation is a simulation of a user's actual navigation, whereby the benchmark suggests some initial sequence of queries, asks the SUT for next move suggestions, then proposes some continuation, switches to ask the SUT again, and so on. This allows the benchmark to ask the SUT for suggestions multiple times, in multiple phases and focusing multiple view areas.

The third component is in charge of computing scores and reporting results. It considers the sessions produced with the SUT, and computes values for the quality metrics presented in Section 3.2.

4.3 Component 1: Benchmark Initialization

Initialization consists of the synthesis of an OLAP user environment. It consists mainly of data generation and user creation.

Data generation An OLAP database (schema and instance) and a set of user sessions over it are firstly generated. The default database schema is the one of SSB benchmark [24], but the benchmark can be initialized with any other OLAP schema. We use CubeLoad [26] for automatically generating user sessions. CubeLoad generates realistic OLAP workloads, taking as input a cube schema and the desired number of sessions. Its templates enable the creation of a large number of sessions representing varied explorations and patterns. Finally, a realistic database instance is generated with PDGF [25]. We use the more frequent selection predicates in the log of sessions to produce data skew in the most queried zones of the cube.

Users creation While CubeLoad enables the generation of a large workload and creates feasible exploration patterns, it does not assign sessions to specific users. We use an off-the-shelf clustering algorithm [20], using a similarity measure tailored for OLAP sessions [2] to generate "users". In this way a user is characterized by a set of sessions focusing on some zones of the cube. Each set of sessions is split in two parts: log and seed sessions. The former constitutes the user log that is exposed to the SUT, so that it can build its own knowledge for suggesting next moves. The latter, not shown to the SUT, is used to seed the benchmark tasks. The size of each user's log is ruled by a parameter. This allows the benchmark to evaluate the SUT when working with novice users versus advanced users, creating tasks with different difficulty levels, in the sense that it is more difficult for a SUT to suggest something interesting to a relatively new user. Finally, a generative model is learned from the log, inspired by techniques of the OLAP literature [27,3]. This generative model is a Markov Model that is used by Component 2, for simulating the interaction with a user.

4.4 Component 2: Evaluation of a SUT

This component is responsible for the simulation of a navigation, together with the SUT, in order to resolve a given task. A task can be seen as an exercise that has to be solved by SUTs. Tasks are created just before starting a SUT evaluation. The evaluation protocol first provides a seed session, which is a set of seed queries representing part of a navigation, as a context for continuation of the navigation. Then it asks the SUT for a first next move suggestion that consists of one or more queries. After the SUT suggestion, the benchmark decides if it accepts or refuses the suggestion (a real user would either follows the suggestion or not). The probability of discarding the suggestion is given as parameter. The following step is for the benchmark to indicate the next query (a real user may evaluate their own queries). This is done by finding the closest query in the user model to the current query, and stochastically determining the next query in the user model. This new query is then presented to the SUT to suggest again, and the process continues as such until a stop condition. The simulation is ran for a set of tasks (the number of tasks to run is a user-given parameter), and the whole process is preceded by the definition of tasks to accomplish.

4.5 Component 3: Scoring

All the queries recorded during task resolution are fed to the scoring component so it can compute a score for the SUT using the metrics defined in Section 3.2. For each metric, the scoring component first scores each task, and then, it aggregates scores for the SUT. In practice, a metric can be seen as a function that takes as input a task resolution (the queries that were played), and provides as output a number that represents the score of the metric for the given task.

5 Experiments

In this section we describe and report results on the experiments designed to validate the proposed benchmark. A first version of the benchmark application was coded in Java, using PDGF [25], CubeLoad [26] and Fuzzy C-medoids [20], as explained in section 4. The tested SUTs were plugged to the benchmark application using the interface class. Experiments use the default schema (SSB) [24] with a scale factor of 1, a small global log of 50 sessions, 375 queries and 9 users with 50% of seed sessions. We generated 100 tasks for each SUTs to resolve. Tests were conducted on a laptop equiped with an i5-3210M CPU @ 2.50GHz and 8GB of RAM.

5.1 Experimental setup

Validation. In order to test benchmark ranking, we compared three synthetic SUTs that have simple behavior, and then expected results. 'Random', the one having the worst strategy, returns purely random next move suggestions. 'Naive'

	Engagement		Success		Time		Novelty		Learning	
	QD	**F**	**R**	**P**	**QpS**	**TET**	**RNI**	**IVA**	**L**	**LGR**
User	102	0.082	0.122	0.032	0.223	20.080	1.24E-004	0.012	0.377	0.554
stdev	0	0.053	0.229	0.082	0.264	19.706	3.38E-004	0.039	0.387	0.501
Random	102	0.030	0.189	0.002	0.0016	2260.400	3.23E-004	0.728	0.384	0.554
stdev	0	0.013	0.263	0.003	0.001	4151.932	2.33E-004	0.307	0.386	0.502
Naive	102	0.069	0.293	0.014	0.004	1464.560	7.72E-005	0.569	0.377	0.554
stdev	0	0.039	0.302	0.031	0.008	1180.270	1.10E-004	0.289	0.387	0.502
Cheater	101.4	0.029	0.538	0.119	0.014	319.600	7.83E-005	0.155	0.513	0.557
stdev	3	0.039	0.318	0.235	0.016	340.167	2.20E-004	0.277	0.489	0.504
Falseto	467	0.024	0.575	0.005	0.0005	2205.080	1.37E-004	0.737	0.376	0.559
stdev	25.855	0.001	0.344	0.003	0.0001	706.705	9.13E-005	0.236	0.386	0.506
Cinecube	184.2	0.018	0.398	0.013	0.006	2891.840	2.20E-004	0.908	0.377	0.556
stdev	51.027	0.006	0.333	0.039	0.010	4210.870	5.38E-004	0.092	0.387	0.503

Table 3. Scores of the SUTs

generates queries that are one OLAP operation away from the previous query. It naively tries to stay close from the current query, but still chooses the next move randomly within that neighborhood. 'Cheater' uses 'insider information' in order to return good suggestions. Concretely, it generates queries containing exclusively one cell from the neighborhood N_C of cells in the seed session, which should fit the user's needs in terms of task success. The goal of this experiment is to confirm that the benchmark ranks these approaches as expected.

Benchmarking existing approaches. We created an experimental setup to compare the following approaches: CineCube [13] and Falseto [1]. CineCube is a multifaceted approach focusing on building a user-friendly sequence of explanations for the analysts. The approach highlights relevant cells in current views and explores automatically expansion into two one-distance children and two one-distance sibling queries, also summarizing the findings. Falseto is an OLAP session composition tool that implements a recommender system based on collaborative filtering. It features three phases: (i) search the log for sessions that bear some similarity with the one currently being issued by the user; (ii) extract the most relevant subsessions; and (iii) adapt the top-ranked subsession to the current user's session. As a baseline we also report the scores without a SUT, i.e. when sessions are created only by playing the user generative model ('User').

5.2 Analysis of Experimental Results

Table 3 shows the benchmark results for the tested SUTs. For each SUT, we report its average score and standard deviation for the 100 tasks, for all the benchmark primary and secondary metrics.

Validation. Regarding the three basic SUTs designed, the results globally allow us to rank 'Cheater' highest, followed by 'Naive' and 'Random' with the

poorest performance, as expected. Having access to detailed insider information, 'Cheater' achieved a higher task success and it provides better learning, with a slightly higher learning curve. Theoretically, cheater should suggest all neighbor cells (recall of 1), but in practice, it is stopped by the protocol (number of chances reached). That explains that its recall is good, but not maximal. However, as it plays only queries containing the coverage of the study, increase in view area is lower. 'Random' proposes completely random jumps in the multidimensional space, which is less effective (lower task success). As it slowly contributes to task resolution, the stop condition (50 chances) stops its execution. That is why it obtains maximum query depth for all tasks (stdev=0). Nevertheless, it randomly explores other cube zones, so consequently increases view area and increases learning at the cost of a poor precision. As expected, 'Naive' stays half-way between 'Cheater' and 'Random'. By moving always close to the current query, it was able to stay within relevant regions (so succeeding quite well). As 'Naive', it executes until the stop condition obtaining maximum query depth.

Benchmarking existing approaches Results in Table 3 highlight the differences between Falseto and Cinecube and helps deciding which is best in which case. By definition, Falseto generates longer sessions than Cinecube as reflected by the Query Depth score. Falseto also generates queries that are not only related to the neighborhood of the last queries as Cinecube but that are based on collaborative filtering with user past sessions to recommend next analysis moves. This leads Falseto to produce more diverse queries than Cinecube. This is an advantage when it comes to explore the data as shown by the Recall of Falseto which is slightly better than that of Cinecube. However this comes at the cost of a lower precision, because it explores parts of the cube outside seed neighborhood.

When it comes to compare existing approaches with basic SUTs, we also retrieve coherent and intuitive results. Indeed, the scores allow to globally rank both Falseto and Cinecube better than Naive and worse than Cheater, while being good in some points. Indeed, contrary to Random and Naive that do not seem to effectively support data exploration, Falseto and Cinecube are clearly of great help for the user. According to User scores (i.e. user playing alone), they lead to a more complete exploration of relevant regions with more engagement and better task success.

6 Conclusion

In this paper we proposed the first benchmark for assessing OLAP exploration approaches. Modern OLAP exploration approaches are expected to suggest next moves to users, but an important question is how to evaluate the quality of such suggestions, and how to compare alternatives. Our benchmark uses state of the art techniques to generate data and user traces, and for its metrics definition. The benchmark is easy to use, requiring the SUT tester to write only a well-defined interface, and classifies the SUT according to a set of user-centric metrics. This is an important advance, since existing benchmarks focus almost exclusively on

performance, cost or energy. To validate the approach, we have proved that the benchmark correctly ranks a set of strategies for which the behavior is known.

We plan to make all the details of the benchmark public for anyone to use and improve, and our long-term goal is that it serves as a building block of a more general benchmark for exploratory search over databases in general. We are currently working on turning our proposal into an industry-strength benchmark: we are detailing rules, procedures, reporting procedures and documentation; we are investigating the benchmark robustness and its sensitivity to the data and traces. We are currently studying how to use KT to aggregate our metrics to easily rank SUTs. We are also applying the benchmark to rank other existing exploratory approaches, as a way to create a regular use base.

References

1. J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending OLAP sessions. *DSS*, 69:20–30, 2015.
2. J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. Similarity measures for olap sessions. *KAIS*, 39(2):463–489, 2014.
3. M.-A. Aufaure, N. Kuchmann-Beauger, P. Marcel, S. Rizzi, and Y. Vanrompay. Predicting your next olap query based on recent analytical sessions. In *DaWaK*, pages 134–145, 2013.
4. L. Battle, M. Stonebraker, and R. Chang. Dynamic reduction of query result sets for interactive visualizaton. In *Int'l Conference on Big Data*, pages 1–8, 2013.
5. V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F. Guisnel, and H. Klajnmic. Embedded indicators to facilitate the exploration of a data cube. *IJBIDM*, 4(3/4):329–349, 2009.
6. A. T. Corbett and J. R. Anderson. Knowledge tracing: Modelling the acquisition of procedural knowledge. *UMUAI*, 4(4):253–278, 1995.
7. M. C. Desmarais and R. S. J. de Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. *UMUAI*, 22(1-2):9–38, 2012.
8. M. Djedaini, P. Furtado, N. Labroche, P. Marcel, and V. Peralta. Assessing the effectiveness of olap exploration approaches. Technical Report 315, June 2016. http://www.info.univ-tours.fr/~marcel/RR-DFLMP-1-062016.pdf.
9. M. Drosou and E. Pitoura. Ymaldb: exploring relational databases via result-driven recommendations. *VLDB J.*, 22(6):849–874, 2013.
10. A. Drutsa, G. Gusev, and P. Serdyukov. Future user engagement prediction and its application to improve the sensitivity of online experiments. In *WWW*, pages 256–266, 2015.
11. M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. Querie: Collaborative database exploration. *TKDE*, 26(7):1778–1790, 2014.
12. A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for olap discovery-driven analysis. *IJDWM*, 7(2):1–25, 2011.
13. D. Gkesoulis, P. Vassiliadis, and P. Manousis. Cinecubes: Aiding data workers gain insights from OLAP queries. *IS*, 53:60–86, 2015.
14. M. Golfarelli, S. Rizzi, and P. Biondi. myolap: An approach to express and evaluate OLAP preferences. *TKDE*, 23(7):1050–1064, 2011.
15. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

16. J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD*, pages 171–182, 1997.
17. S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, pages 277–281, 2015.
18. N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE*, pages 472–483, 2014.
19. H. A. Khan, M. A. Sharaf, and A. Albarrak. Divide: efficient diversification for interactive data exploration. In *SSDBM*, pages 15:1–15:12, 2014.
20. R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE-FS*, 9:595–607, 2001.
21. J. LeFevre, J. Sankaranarayanan, H. Hacigümüş, J. Tatemura, and N. Polyzotis. Towards a workload for evolutionary analytics. In *DanaC '13*, pages 26–30, 2013.
22. J. Lehmann, M. Lalmas, E. Yom-Tov, and G. Dupret. Models of user engagement. In *UMAP*, pages 164–175, 2012.
23. H. V. Nguyen, K. Böhm, F. Becker, B. Goldman, G. Hinkel, and E. Müller. Identifying user interests within the data space - a case study with skyserver. In *EDBT 2015*, pages 641–652, 2015.
24. P. E. O'Neil, E. J. O'Neil, X. Chen, and S. Revilak. The star schema benchmark and augmented fact table indexing. In *TPCTC*, pages 237–252, 2009.
25. T. Rabl, M. Poess, H. Jacobsen, P. E. O'Neil, and E. J. O'Neil. Variations of the star schema benchmark to test the effects of data skew on query performance. In *ICPE'13*, pages 361–372, 2013.
26. S. Rizzi and E. Gallinucci. Cubeload: A parametric generator of realistic OLAP workloads. In *CAiSE 2014*, pages 610–624, 2014.
27. C. Sapia. Promise: Predicting query behavior to enable predictive caching strategies for olap systems. In *DaWaK*, pages 224–233, 2000.
28. S. Sarawagi. Explaining differences in multidimensional aggregates. In *VLDB*, pages 42–53, 1999.
29. S. Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, pages 307–316, 2000.
30. G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional OLAP data. In *VLDB*, pages 531–540, 2001.
31. T. Sellam and M. L. Kersten. Meet charles, big data query advisor. In *CIDR*, 2013.
32. Y. Song, X. Shi, and X. Fu. Evaluating and predicting user engagement change with degraded search relevance. In *WWW*, pages 1213–1224, 2013.
33. R. W. White and R. A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool Publishers, 2009.