A benchmark for assessing OLAP exploration assistants

Mahfoud Djedaini*, Nicolas Labroche*, Patrick Marcel*, Veronika Peralta*

*University of Tours, Tours, France
firstname.lastname@univ-tours.fr,

Abstract. In this demonstration paper, we present *InDExBench*, a benchmark designed and developed for evaluating and comparing Interactive Database Exploration (IDE) assistant systems in the context of OLAP. We briefly recall how *InDExBench* works behind the scenes. Then, we explain how it can be used in practice by considering the case of *Sam*, an OLAP IDE assistant author who wants to evaluate how her system performs, and how it compares to competitors.

1 Introduction

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, a recurrent problem is how to assess the effectiveness of an exploratory OLAP approach. In this paper, we describe *InDExBench*, a benchmark for evaluating IDE approaches, referred to as SUTs (for Systems Under Test), relying on an extensible set of user-centric metrics that relate to the main dimensions of exploratory analysis. Basically, SUTs are evaluated by assessing the quality of explorations they help the user to produce. An OLAP exploration (Aligon et al., 2014) is technically a sequence of OLAP queries over a database instance issued by a given user. *InDExBench* achieves its goal by first simulating a complete OLAP system (DB instance, cube schema, users, ...) and then by giving the SUT the opportunity to play wihtin the system. In this paper, we review *InDExBench* features through a realistic use case. Thorough details can be found in (Djedaini et al.).

2 Benchmark overview

In this section, we describe more precisely the metrics, how *InDExBench* generates the OLAP system, and finally how it simulates and scores explorations.

Metrics *InDExBench* scores explorations using five categories of user-centric metrics borrowed from Exploratory search (White and Roth, 2009). Each category is implemented with a primary metric and a secondary to counterbalance it.

User engagement measures how engaged and invested is a user on a system. For this category, we borrow from web search two popular and intuitive metrics. Query Depth (QD) as primary metric, represents the number of queries. Query Focus (QF) as secondary metric, measures the degree to which an exploration is focused around a zone of the cube. **Information Novelty** measures the quantity of Relevant New Information (RNI). We use a normalized entropy as primary metric to measure the quantity of interesting information contained *in the data* retrieved by each query of the exploration. The secondary metric measures the Increase in View Area (IVA), i.e. the increase in the number of viewed cells. Intuitively, information about a group of cells can be obtained by exploring a cube area around it. **Task completeness** is reached when the whole neighborhood around a cell, in the sense of OLAP operation, has been explored. A simple way of measuring it is with recall and precision. Recall (R) is the primary metric since, consistently with exploratory search, we consider OLAP navigation as a recall oriented activity. Precision (P) is then the secondary metric. Measuring **task time** is done by adapting metrics of existing TPC benchmarks. The primary metric comes from the TPC-DS benchmark ((TPC), 2012) and measures the query frequency, i.e. number of queries per second (QPS). The secondary metric simply measures the elapsed time (TET) between the beginning and the end of an exploration. **Learning and cognition** aims at evaluating the user knowledge. Knowledge Tracing (KT) (Corbett and Anderson, 1995) has been proposed originally in e-learning to evaluate students knowledge, based on a sequence of exercises that they have to solve. We adapt KT by considering as an exercise finding OLAP queries with high Information Novelty. The primary metric Learning (L) is then the knowledge level estimated by KT. The secondary metric measures the Learning Growth Rate (LGR).

OLAP environment generation *InDExBench* is capable of generating a complete OLAP database (schema and instance), with users and user explorations over it. A realistic database instance is generated with PDGF (Rabl et al., 2013), a data generator that supports generation of skewed data. By default, *InDExBench* uses the Star Schema Benchmark (SSB) (O'Neil et al., 2009), but the benchmark can be initialized with any other OLAP schema. CubeLoad (Rizzi and Gallinucci, 2014) is used for automatically generating realistic OLAP workloads, taking as input a cube schema and the desired number of sessions. Sessions are then clustered using a metric tailored for OLAP sessions (Aligon et al., 2014). Finally, a Markov inspired generative model is learned from each cluster to simulate a particular user. Sessions of this cluster are considered as the user's past sessions.

Generating and scoring OLAP explorations The evaluation protocol first provides a seed session, which is a set of seed queries representing part of a navigation of a given user, as a context for continuation of the navigation. Then, the simulated user and the SUT play in turn. The simulated user smartly issues new queries using his/her generative model. The SUT uses its internal intelligence and the context (current query, query logs, . . .) to propose new queries. Like in real cases, SUT propositions may or may not be included in the exploration, depending on the simulated user choice. A SUT is allowed to play a given number of times, after which the process is stopped. The obtained exploration is then scored using the metrics described above. The same process is repeated a large number of times for a given SUT. Finally a SUT obtains a global score for each metric, by averaging the scores for all the explorations for the given metric.

3 Scenario

The demonstration scenario will consider the case of *Sam*, a researcher who is thinking about a very interesting idea for implementing a new IDE assistant dedicated to OLAP. *Sam* wants to quickly have detailed information about how her prototype performs. She also would like to compare with competitor algorithms, as well as with baseline algorithms such as a random algorithm or a naive one. We will then describe how *Sam* will use *InDExBench* and how she can benefit from it.

***InDExBench* installation** *InDExBench* is developed in Java, a portable language, as per benchmarks portability requirement and so is mainly distributed as a java library. So, basically, *Sam* creates a Java project, and imports the *InDExBench* jar file. *Sam* may now have a look at the *InDExBench* API documentation [1]. *Sam*'s next step is to let *InDExBench* know about her algorithm.

Interfacing SUT with *InDExBench* Within *InDExBench*, SUTs are recognized as being classes implementing an interface called I_SUT. To evaluate her SUT, *Sam* writes a class that implements I_SUT. Her class represents her algorithm within *InDExBench*. *Sam* can write her whole code withing her new class. If her algorithm is already developed independently, she can just import her library into the project, and call her library features within her class. In the live demo, we will mostly focus on SUTs comparison by showing how to compare different SUTs provided by *InDExBench*.

Evaluation and feedback At this point, *Sam* has installed *InDExBench*, and she has plugged her SUT to it. However, she does not have an OLAP system within easy reach. Provided a JDBC connection string and an OLAP cube schema, *InDExBench* can generate for *Sam* a simulated OLAP system. *Sam* can configure how the OLAP system will be generated. For instance, she can configure how data are generated to populate the database, how many past sessions should be present in the log, how many users the system must have, etc.

To set a comparison with a random algorithm, *Sam* only has to create a $RandomSUT$ instance. Indeed, *InDExBench* provides by default implementation for different SUTs, among which a random algorithm called $RandomSUT$. For comparing with SUTs from the literature, *Sam* will have to interface the SUTs she wants to challenge with *InDExBench*. So far, *InDExBench* provides interfaces for two SUTs from the literature, namely Falseto (Aligon et al.) and Cinecube (Gkesoulis et al., 2015).

As *Sam* wants to compare with other SUTs, she has to clearly ask this to *InDExBench*. When a comparison is performed, *InDExBench* ensures that each SUT is provided under the *exactly* same circumstances. When the evaluation is completed, *Sam* gets a detailed information of how each SUT performed by means of a score for each metric. She can for instance notice that her algorithm takes more time to execute than $RandomSUT$, but that it performs better in terms of precision, user engagement, etc. *Sam* can also get a detailed score per exploration, for example if she wants to analyze how her SUT's performance evolves with time. *InDExBench* provides a very rich feedback, usable at will by evaluators.

1. `http://www.info.univ-tours.fr/~marcel/benchmark.html`

4 Conclusion

In this paper we described several features of *InDExBench* by using a concrete example. We showed how *Sam* can benefit from *InDExBench* features to quickly set up and compare her algorithm from other algorithms from the literature. Actually, *InDExBench* has a lot more features than exposed in this scenario, that could not be detailed here. We created a specific website for *InDExBench* `http://www.info.univ-tours.fr/~marcel/benchmark.html` where we regularly publish material. In this website, can be found *InDExBench* Java library, API documentation, as well as references to published papers.

References

Aligon, J., K. Boulil, P. Marcel, and V. Peralta. A holistic approach to OLAP sessions composition: The falseto experience. In *DOLAP 2014*, pp. 37–46.

Aligon, J., M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia (2014). Similarity measures for olap sessions. *KAIS 39*(2), 463–489.

Corbett, A. T. and J. R. Anderson (1995). Knowledge tracing: Modelling the acquisition of procedural knowledge. *UMUAI 4*(4), 253–278.

Djedaini, M., P. Furtado, N. Labroche, P. Marcel, and V. Peralta. In *TPCTC (2016)*. LNCS 10080 proceedings.

Gkesoulis, D., P. Vassiliadis, and P. Manousis (2015). Cinecubes: Aiding data workers gain insights from OLAP queries. *IS 53*, 60–86.

O'Neil, P. E., E. J. O'Neil, X. Chen, and S. Revilak (2009). The star schema benchmark and augmented fact table indexing. In *TPCTC*, pp. 237–252.

Rabl, T., M. Poess, H. Jacobsen, P. E. O'Neil, and E. J. O'Neil (2013). Variations of the star schema benchmark to test the effects of data skew on query performance. In *ICPE'13*, pp. 361–372.

Rizzi, S. and E. Gallinucci (2014). Cubeload: A parametric generator of realistic OLAP workloads. In *CAiSE 2014*, pp. 610–624.

(TPC), T. T. P. P. C. (2012). Tpc benchmark ds (tpc-ds): The new decision support benchmark standard. http://www.tpc.org/tpcds/.

White, R. W. and R. A. Roth (2009). *Exploratory Search: Beyond the Query-Response Paradigm*. Morgan & Claypool Publishers.

Résumé

Dans ce papier de démonstration, nous présentons *InDExBench*, un Benchmark conçu et développé pour comparer des assistants à l'Exploration Interactive des Données (EDI) dans un contexte OLAP. Nous rappelons brièvement comment *InDExBench* fonctionne en coulisses. Ensuite, nous expliquons comment il peut être utilisé en pratique en considérant le cas de *Sam*, une chercheuse qui a une nouvelle idée d'algorithme d'EDI pour OLAP dont elle souhaite évaluer les performances et comparer avec des algorithmes conccurrents.