CubeLoad: A Parametric Generator of Realistic OLAP Workloads

Stefano Rizzi and Enrico Gallinucci

DISI – University of Bologna,
V.le Risorgimento 2, 40136 Bologna, Italy
{stefano.rizzi,enrico.gallinucci2}@unibo.it

Abstract. Differently from OLTP workloads, OLAP workloads are hardly predictable due to their inherently extemporary nature. Besides, obtaining real OLAP workloads by monitoring the queries actually issued in companies and organizations is quite hard. On the other hand, hardware and software benchmarking in the industrial world, as well as comparative evaluation of novel approaches in the research community, both need reference databases and workloads. In this paper we present CubeLoad, a parametric generator of workloads in the form of OLAP sessions, based on a realistic profile-based model. After describing the main features of CubeLoad, we discuss the results of some tests that show how workloads with very different features can be generated.

Keywords: OLAP, Data Warehouse, Business intelligence, Benchmarks.

1 Introduction

The term *OLAP* (On-Line Analytical Processing) is now widely used to refer to multidimensional databases and to data warehouse systems. However, originally, it was meant to denote a specific class of queries characterized by high interactivity and flexibility, small formulation effort, read-only access, and data aggregation, run by decision makers to analyze their business trend and effectively explore key figures and indicators. While OLTP (On-Line Transactional Processing) queries are normally grouped into transactions that support the everyday operational processes in a company, OLAP queries are typically sequenced into *sessions*. Users create sessions by applying a sequence of OLAP operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, starting from an initial query that is usually predefined [15]. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of information. The resulting sequences of queries are strongly related to the issuing user, to the analyzed phenomenon, and to the current data.

Differently from OLTP workloads, that are 90% frozen within operational applications, OLAP workloads are hardly predictable due to their inherently extemporary nature. Besides, obtaining real OLAP workloads by monitoring the queries actually issued in companies and organizations is quite hard because (i) OLAP queries are at the core of the decision-making process, hence they are jealously guarded by managers and administrators, and (ii) reconstructing OLAP sessions by interpreting the query log of a multidimensional engine operating in a multi-user context is very complex.

On the other hand, hardware and software benchmarking in the industrial world, as well as comparative evaluation of novel approaches in the research community, both need reference databases and workloads. To this end, some efforts have been done over the years to provide standard benchmarks. Specifically, in the OLAP context, the TPC-DS benchmark [14] has been recently developed; it is based on a fixed set of star schemata including 7 fact tables and 17 dimension tables, and it provides a workload featuring queries that address complex business problems and use a variety of access patterns.

The TPC-DS benchmark is carefully designed and offers a solid reference. However, especially in research papers, there is often a need for using benchmarks based on schemata with varying characteristic and on multiple alternative workloads with different features. For instance, it could be interesting to understand how the performance of a proposed approach varies with the number of dimensions in a cube, with the average branching factor of hierarchies, with the maximum length of sessions, or with the average selectivity of queries. In particular, generating parametric OLAP workloads is crucial to the experiments made in the context of OLAP prediction and recommendation, where the features of sessions and queries may have a strong impact on the approach effectiveness and efficiency. So, the papers in this context often rely on synthetically generated OLAP workloads, where queries and session are built in a completely random way based on a set of structural and statistical parameters [1–4]. Unfortunately, while these synthetic workloads serve well for efficiency tests, they cannot provide significant results for effectiveness tests because they do not lean on a realistic user model.

To fill this gap, in this paper we present *CubeLoad*, a parametric generator of OLAP workloads. The main features of CubeLoad are:

– No predefined multidimensional schema is used. The benchmarker[1] can create a workload for any multidimensional schema provided it has been exported in XML compliant with the Mondrian format.
– The workload is generated in the form of sessions, each including a variable number of aggregate queries. The main parameters used are related to a realistic profile-based workload model.
– Sessions are generated according to a set of four templates, that model recurrent types of user analyses.
– If an instance of the multidimensional schema is available (in particular, in the form of a set of dimension tables), its data are used for generating instance-dependent (hence, more realistic) workloads.
– The generated workload is exported in XML to ensure maximum usability.

[1] To distinguish users of OLAP front-ends from the users of CubeLoad, we will call *benchmarkers* the latter.

CubeLoad is written in Java and can be downloaded at http://big.csr.unibo.it/downloads/CubeLoad.zip. It can be freely used by researchers, practitioners, and vendors whenever they need to create parametric bulk OLAP workloads for benchmarking and testing.

The paper outline is as follows. After discussing some related literature in Section 2, in Section 3 we describe the overall functional architecture of CubeLoad. Then we present our workload model and the session templates we defined so far in Sections 4 and 5, respectively. Finally, in Section 6 we discuss the results of some tests we made to profile the generated workloads and in Section 7 we draw the conclusions.

2 Related Works

A milestone in OLAP benchmarking is the TPC-DS [14], that models the decision support functions of a retail product supplier relying on multiple snowflake schemata with shared dimensions. The TPC-DS provides four classes of queries; in particular, the class of *iterative OLAP queries* is distinguished by the tendency of one query to be related to the previous query so as to create sequence of queries —essentially, OLAP sessions. Queries are randomly generated starting from four templates; however, there is no way of parameterizing the generation of sessions.

In [7] the authors introduce the concept of *workload profile* as a way for summarizing the features of an OLAP workload to support designers during logical and physical design. However, the profile used there has a merely statistical nature, and has no relationship with classes of users. Besides, only stand-alone queries are generated.

A workload for evolutionary analytics is proposed in [10] together with several test metrics and with a methodology for running the workload. The emphasis there is not on standard OLAP sessions but rather on queries that evolve over time (which may imply much more drastic changes than those obtained through OLAP operations) and are formulated over changing data and schemata.

A *Data Warehouse Engineering Benchmark* (DWEB) that allows to generate various ad-hoc synthetic data warehouses and workloads is presented in [6]. Though the DWEB workload is parameterized to fulfill data warehouse design needs, it does not create queries in sessions and is ruled by statistical parameters rather than by realistic assumptions.

The author of [13] starts from the query generator of the TCP-DS to define a set of rules that transform a SQL query into another SQL query similar to the original. However, this transformation works at a merely syntactical level (e.g., a new query can be created by changing the comparison operator in the selection predicate) and does not consider OLAP operations such as slicing and drilling.

In [18] the authors introduce a query generator to evaluate the quality of a query optimizer. Similarly to ours, the generator presented is schema-independent and is able to produce valid queries on any database. However, only OLTP queries are generated and, therefore, there is no mention of query sessions.

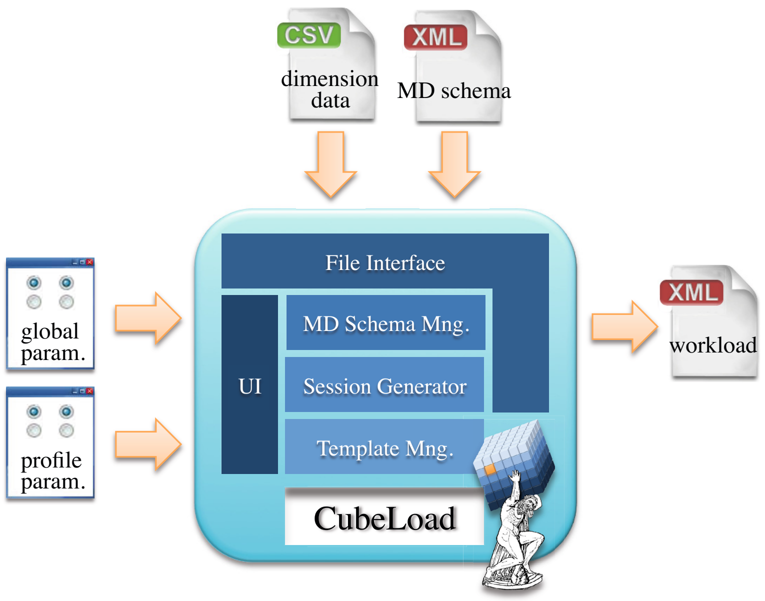

Fig. 1. Functional overview of CubeLoad

Finally, a benchmark on star schemata that extends the TPC-H is presented in [12]; the emphasis here is more on data schemata than on queries, so only 4 non-parameterized OLAP sessions (called query flights here) are provided.

3 Overview

A functional overview of the CubeLoad architecture is sketched in Figure 1. The main input is the multidimensional schema on which the workload is to be generated. To provide this input we adopt the XML specification used by Mondrian for its metadata [9].

Example 1. IPUMS is a public database storing census microdata for social and economic research [11]. An excerpt of the XML specification for its CENSUS multidimensional schema is given below.

```
<?xml version="1.0"?>
<Schema name="Ipums">
  <Cube name="CENSUS">
    <Table name="FACT500K"/>
    <Dimension name="CITY" foreignKey="CITY">
      <Hierarchy hasAll="true" primaryKey="IDCITY" allLevelName="AllCity" allMemberName="All">
        <Table name="CITY"/>
        <Level name="Region" column="REGION" type="String" uniqueMembers="true"/>
        <Level name="State" column="STATE" type="String" uniqueMembers="true"/>
        <Level name="City" column="CITY" type="String" uniqueMembers="false"/>
      </Hierarchy>
    </Dimension>
    <!– other dimensions –>
    <Measure name="SumCostGas" column="COSTGAS" aggregator="sum"/>
    <Measure name="SumIncTot" column="INCTOT" aggregator="sum"/>
    <!– other measures –>
  </Cube>
</Schema>
```

Here, a CITY hierarchy is declared that features three aggregation levels, Region, State, and City besides the AllCity level. Besides, two measures SumCostGas and SumIncTot are declared. □

To maximize interoperability, the workloads generated by CubeLoad are coded using XML; an example is shown below:

```
<Benchmark>
  <!– parameters –>
  <Session profile="Manager" progressive="1" template="Goal Oriented">
    <Query progressive="1">
      <GroupBy>
        <Element> <Hierarchy Value="CITY"/> <Level value="State"/> </Element>
        <!– other group-by elements –>
      </GroupBy>
      <Measures>
        <Element value="MaxCostGas"/> <Element value="SumCostGas"/>
      </Measures>
      <SelectionPredicates>
        <Element>
          <Hierarchy value="OCCUPATION"/> <Level value="Category"/>
          <Predicate value="Dentists"/>
          <YearPrompt value="false"/> <SegregationPredicate value="false"/>
        </Element>
      </SelectionPredicates>
    </Query>
    <!– other queries –>
  </Session>
  <!– other sessions –>
</Benchmark>
```

(an explanation of the parameters and of the other workload elements mentioned in this XML will be given in Section 4).

To generate realistic selection predicates and enable report sizes to be estimated, dimension data are needed. These data can be fed into CubeLoad using the CSV (comma-separated values) format, which can be easily obtained by benchmarkers by exporting dimension tables.

Internally, CubeLoad includes five components:

1. The **user interface**, that allows benchmarkers to select the XML multidimensional schema to be used and choose values for global and profile parameters.
2. The **file interface**, in charge of reading and parsing XML and CSV input files and of writing XML output files.
3. The **multidimensional schema manager**, that builds an internal representation of cubes and dimension data.
4. The **session generator**, that runs the basic procedures for creating sessions respecting the constraints posed by global and profile parameters.
5. The **template manager**, that gives the session generator additional rules for creating sessions based on each template.

4 The Workload Model

The output of CubeLoad is an OLAP workload, defined as a set of *sessions.* A session is a sequence of queries. In the current implementation, we support a basic form of multidimensional query consisting of (i) a *group-by* (i.e., a set of hierarchy levels on which measure values are grouped); (ii) one or more measures whose values are returned (the aggregation operator used for each measure is defined by the multidimensional schema); and (iii) zero or more *selection predicates*, each operating on a hierarchy level. We call *report* the result of a query; its size is the number of facts returned. Roughly, the size of a report can be estimated as the product of the domain cardinalities for all levels in the query group-by, reduced by considering the selectivity factors of the selection predicates; more accurate estimates can be computed if the sparsity of the cube is known [5]. Two consecutive queries within a session are normally separated by the application of one OLAP operation, that changes either the group-by, or the selection predicate, or the set of measures returned, as shown in the following example.

Example 2. An example of a session starting from seed query q_1 is $s = \langle q_1, q_2, q_3, q_4 \rangle$; the group-by's, selection predicates, and returned measures for the queries involved are shown in Table 1. Query q_2 is obtained from q_1 by drilling-down the cube along the CITY dimension; q_3 is obtained from q_2 by slicing-and-dicing the cube; q_4 is obtained from q_3 by changing the measure returned. □

Table 1. Queries for the sample session in Example 2

Query	Group-by	Selection predicate	Measures
q_1	State, Year	Region='South Atlantic'	SumCostGas
q_2	City, Year	Region='South Atlantic'	SumCostGas
q_3	City, Year	Occupation='Dentists'	SumCostGas
q_1	City, Year	Occupation='Dentists'	SumIncTot

In company settings, users of OLAP front-ends are normally grouped into profiles with different skills (e.g., CEO, marketing analyst, department manager) and involved in business analyses with different features (e.g., more or less repetitive, more or less complex). Importantly, different profiles generally have quite different permissions for accessing data; often, a profile has one or more *segregation predicates*, i.e., it can only view a specific slice of the cube data (e.g., a department manager can only access the sales for her department).

When a user logs to the OLAP front-end, she is typically shown a page where some predefined queries (which we call *seed queries*) are linked. Sometimes seed queries include a *prompt*, meaning that the front-end asks the user to select one value out of the domain of a level (often, the year). After choosing and executing one of these queries, the user starts applying a sequence of OLAP operations that progressively transform a query into another so as to build an analysis session.

Features such as the number of seed queries available, the maximum size and complexity of reports returned by seed queries, and the average length of sessions may significantly depend on the typical ICT skills and business understanding for the users of each profile —besides on the quality of the OLAP fron-end.

To simulate the above setting, CubeLoad uses a set of parameters that rule workload generation and are distinguished into *global parameters* and *profile parameters*. The global parameters rule:

- the **number of distinct user profiles** to be simulated. Each profile simulates a specific class of OLAP users and is characterized by different values of the profile parameters. Each session is generated for one profile, so the sessions in the resulting workload can be naturally grouped into clusters; the more different the parameters for the profiles, the sharper the clusters.
- the **maximum number of measures** that can be returned by a single query. A report including several measures is hardly readable by anyone, so the value for this parameter mainly depends on how sophisticated the visualization modes supported by the OLAP front-end are.
- the **minimum and maximum size of seed query reports**. The size (i.e., number of cells) of a query result depends on the query group-by and on the presence of selection predicates. While during an unconstrained OLAP sessions users can (either consciously or unconsciously) formulate a query that returns a report with either negligible or huge size, seed queries are typically created by front-end programmers in such a way that their report size is reasonable. This is reason the reason why in our model the size of seed query reports ranges within a parametric interval.
- the **number of surprising queries**, whose meaning will be explained in Section 5 in relationship to the explorative template.

Each profile is then associated to a further set of parameters, that rule:

- the **number of seed queries**. Specialists' profiles have a large number of seed queries; managers' profiles may have a low number of seed queries.
- the **minimum and maximum length of sessions**. The values for these parameters depend on the ICT skills of the users of each profile and on the complexity of the analyses they usually carry out.
- the **number of sessions** to be created. The more intensive the use of the OLAP front-end for the users of a profile, the higher the value of this parameter.
- the **fraction of seed queries that include a year prompt**. This fraction depends on the time scope of decision-making tasks for each profile (operative profiles typically analyze daily to monthly trends, while managerial profiles are often interested in yearly trends).
- the **presence of a segregation predicate**. A segregation predicate is typically present in departmental or geographically-distributed profiles (e.g., production manager and sales manager for Italy).

The workload model is summarized in Figure 2 in the form of a UML class diagram.

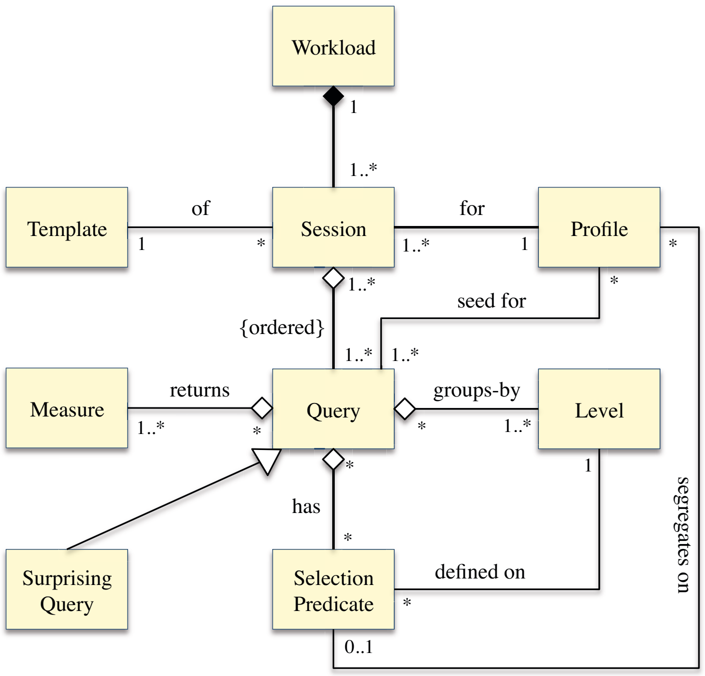

Fig. 2. UML workload model

5 Session Templates

Each session generated by CubeLoad for a given profile starts from one of the seed queries for that profile and evolves, consistently with global and profile parameters, according to a *template*. In its current implementation, CubeLoad uses four different templates for generating sessions:

1. **Slice-and-Drill.** In several OLAP front-ends, the default behavior when a user clicks on a row/column of a pivot table is to disaggregate the values for that row/column into its components, which in OLAP terms means slicing and drilling down. For instance, starting from a report showing sales per state and year, clicking on 2013 would trigger a query showing sales per state and month of 2013, while clicking on Florida would trigger a query showing sales per Florida cities and year. In sessions based on this template, (non-segregated) hierarchies are progressively navigated by choosing a hierarchy h, a member v of the current group-by level $l \in h$ and creating a new query with selection predicate $l = v$ and group-by on the level l that precedes l within h.
2. **Slice-All.** Users are sometimes interested in navigating a cube by slices, i.e., in repeatedly running the same query but with different selection predicates. In sessions based on this template, a level l of the group-by of the seed query

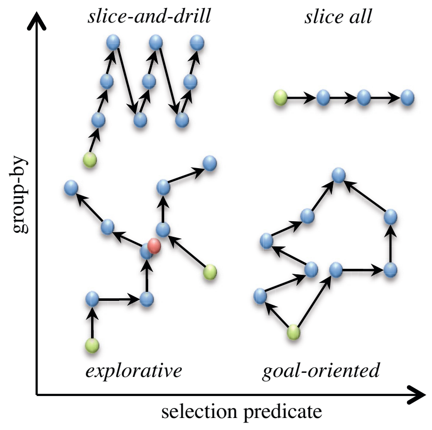

Fig. 3. Session templates (seed queries in green, surprising queries in red)

is chosen, and new queries are generated by keeping the same group-by and adding selection predicates on the different members of l. For instance, starting from a query asking for the monthly sales by state for the video department, the subsequent queries could ask for the same report for the audio, the photo, and the PC departments.

3. **Explorative.** Some queries may return reports that are particularly interesting for most users, for instance because they show unexpected results (e.g., they show that the impact of a social policy is not the one that had been predicted) or have a strong impact on business (e.g., they show that the level of qualified employment in a given area is extremely low, which requires a corrective action to be taken). Following [16], we call them *surprising queries*. The motivation for this template is the assumption that several users, while exploring the cube in search of significant correlations, will be "attracted" by one surprising query. So, sessions based on this template tend to converge "near" to one of the surprising queries, then they evolve casually. Note that the overall number of surprising queries is fixed by a global parameter, while each surprising query is randomly generated.
4. **Goal-Oriented.** Sessions of this type are run by users who have a specific analysis goal, but whose OLAP skills are limited so they may follow a complex path to reach their destination. All the goal-oriented sessions starting from the same seed query q end in the same (randomly-generated) query p, but the sequence of OLAP operations to be applied to reach p from q is generated randomly.

Figure 3 shows an intuition of sessions based on the four templates in a qualitative group-by/selection predicate space.

6 Experiments

To verify that the CubeLoad parameters and templates actually allow a wide spectrum of workloads to be generated, and to help benchmarkers better understand the relationships between those parameters/templates and the workload features, we use a similarity function that was specifically proposed in [2] for comparing OLAP queries and sessions. The query similarity function, σ_{que}, is a combination of three components: one related to group-by's, one to selection predicates, and one to measure sets.

Definition 6.1 (Similarity of OLAP queries). *Let q and q' be two queries on the same n-dimensional schema. The* similarity *between q and q' is*

$$\sigma_{que}(q,q') = 0.35\sigma_{gbs}(q,q') + 0.50 \cdot \sigma_{sel}(q,q') + 0.15 \cdot \sigma_{meas}(q,q') \in [0..1]$$

where:

- *The similarity between the group-by's of q and q', $\{l_1,\dots,l_n\}$ and $\{l'_1,\dots,l'_n\}$ respectively, is*

$$\sigma_{gbs}(q,q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{lev}(l_i,l'_i)}{L_i-1}}{n}$$

  where L_i is the total number of levels in the i-th hierarchy, h_i, and $Dist_{lev}(l_i,l'_i) \in [0..L_i-1]$ is the distance between its two levels l_i and l'_i.
- *The similarity between the selection predicates of q and q', $\{p_1,\dots,p_n\}$ and $\{p'_1,\dots,p'_n\}$ respectively, is*

$$\sigma_{sel}(q,q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{pred}(p_i,p'_i)}{L_i}}{n}$$

  where the distance $Dist_{pred}(p_i,p'_i)$ between predicates p_i and p'_i, both formulated on levels of hierarchy h_i, is 0 if they are expressed on the same level and using the same constant, 1 if they are defined on the same level but not on the same constant, greater than 1 if they are defined on different levels.
- *The similarity between the measure sets returned by q and q', M and M' respectively, is*

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

The session similarity function, $\sigma_{ali}(s,s') \in [0..1]$, is based on the best alignment between the queries belonging to sessions s and s'. The best alignment is computed by means of the Smith-Waterman algorithm, which efficiently matches subsequences of two given sequences by ignoring the non-matching parts [17]. It is a dynamic programming algorithm based on a matrix whose value in position (i,j) expresses the score for aligning subsequences of s and s' that end in queries s_i and s'_j, respectively. This score is computed starting from the similarity between the queries included in the aligned subsequences [2].

Table 2. CubeLoad parameters used for generating the three sample workloads

Sample workload	$W1$	$W2$	$W3$
Number of profiles	1	1	1
Max number of measures	2	2	2
Size of seed query reports	$10 \div 100$	$10 \div 100$	$10 \div 100$
Number of surprising queries	5	2	1
Number of seed queries	50	5	1
Length of sessions	$7 \div 12$	$7 \div 12$	$7 \div 12$
Number of sessions per seed query	4	40	200
Year prompt fraction	0.25	0.50	1.00
Segregation predicate	No	Yes	Yes

To explore the range of possibilities of CubeLoad we generated three sample workloads with the following "extreme" features:

1. Workload $W1$ is a sparse one, i.e., the sessions generated are quite different one from another. This result is mainly obtained by using a high number of seed queries and generating a few sessions per seed.
2. Workload $W2$ is a clustered one, i.e., the sessions generated are similar to each other in five groups. This is mainly obtained by defining five seed queries.
3. Workload $W3$ is a dense one, i.e., the sessions generated are all quite similar to each other. This is mainly obtained by defining a single surprising query and by generating all sessions starting from the same seed query.

For a fair comparison, all three workloads include the same numbers of sessions (200); the values for the other parameters are summarized in Table 2.

A qualitative analysis of these three workloads can be made by observing Figure 4, that shows for each of them the session-to-session similarity. Each row and column corresponds to one of the 200 sessions of the workload, so each cell shows the similarity between two different sessions of the same workload: white means $\sigma_{ali} = 0$, black $\sigma_{ali} = 1$, gray shades mean $0 < \sigma_{ali} < 1$. As expected, in Figure 4.a we find a very low average similarity between sessions, while in Figure 4.c the average similarity is much higher. In Figure 4.b we can easily find the five cluster as areas with higher-than-average similarity. A quantitative confirmation of this fact can be found in Figure 5, that shows for each workload the average session-to-session similarity and its standard deviation: they are both lower for the sparse workload $W1$ (where all sessions are different), while they increasingly grow higher for the clustered workload $W2$ (where sessions in the same cluster are very similar to each other and very different from those in the other clusters) and the dense workload $W3$ (in the latter case, the standard deviation is high because the four templates adopted inevitably introduce a scattering in the sessions generated).

Figure 5 also shows the propensity of each workload to being clustered. The indicator we adopted to this end is the *Hopkins statistics* [8]. Given a workload W, i.e., a set of N sessions, we first generate a set S of m fake sessions ($m \ll N$)

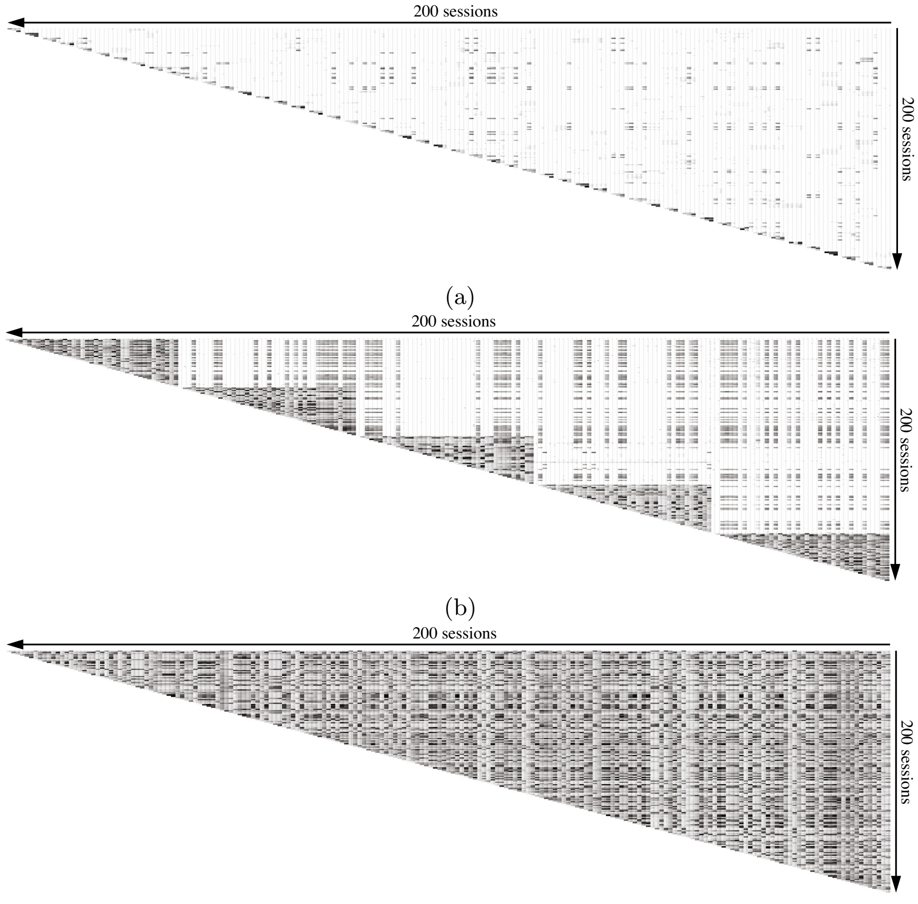

Fig. 4. Session-to-session similarities for the three sample workloads

that are randomly and uniformly distributed in the space of possible sessions. For each fake session $s_i \in S$, let u_i be its distance from the nearest-neighbor session in W (where $Distance(s, s') = 1 - \sigma_{ali}(s, s')$). Then, m sessions are randomly chosen from W; let w_i be the distance of the i-th of these sessions from its nearest-neighbor in W. The Hopkins statistics is then defined as

$$H = \frac{\sum_{i=1}^{m} w_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

For workload $W1$, H is near to 0.5; this means that the distance of each session in $W1$ from its nearest-neighbor is very similar to the distance of each fake session, i.e., that $W1$ has a random distribution. For $W2$ is quite small; this is because the w_i's are small, which means that sessions are well clustered. For $W3$ H is even smaller, because all sessions are part of a single, dense cluster.

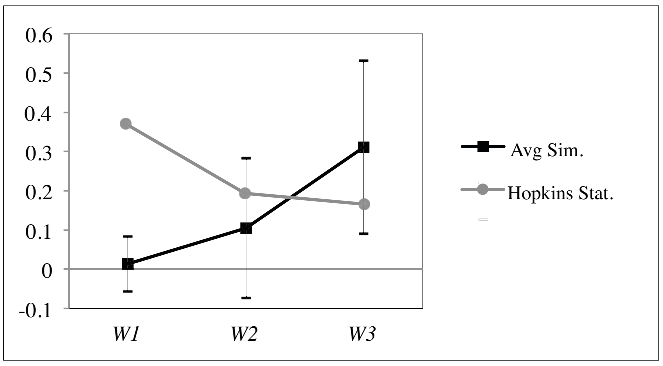

Fig. 5. Average session-to-session similarity and Hopkins statistics for the three sample workloads

Finally, Figure 6 gives a quantitative explanation of the differences between our four templates by showing the similarity σ_{que} between the first query and the subsequent queries for sessions based on each template. In the slice-and-drill template, the saw-tooth trend arises because when a sequence of slice-and-drill clicks along hierarchy h leads to a query grouped by the finest level of h, the simulated user behavior is to go back to the seed query and start a new slice-and-drill sequence along a different hierarchy (three such sequences are clearly visible in the figure). In the slice-all template, only the specific member appearing in the query selection predicate is changed during the session, so the query similarity is mostly constant and quite high. In the explorative template, the session rapidly converges towards the surprising query (the sixth query in the session in this case), then it moves randomly in the query space (in this case, it tends to reapproach the seed query). Finally, in the goal-oriented template the session randomly moves towards its goal query.

7 Final Remarks

In this paper we have described the features of CubeLoad, a generator of OLAP sessions aimed at simulating realistic workloads. The sessions generated are currently based on four templates and ruled by a set of parameters. The template features and the impact of parameters on the resulting workload have been discussed with the support of some tests using a similarity function specifically devised for OLAP sessions.

Some comparison between CubeLoad and TPC-DS is useful at this point. Overall, the focus in the TPC-DS is more on the complexity of single queries rather than on query sessions. Indeed, while the query model is more expressive than in CubeLoad because nesting is supported, three of the four classes of queries provided in the TPC-DS (namely, *ad hoc queries*, *reporting queries*, and *data mining queries*) only include stand-alone queries; as such, they could be

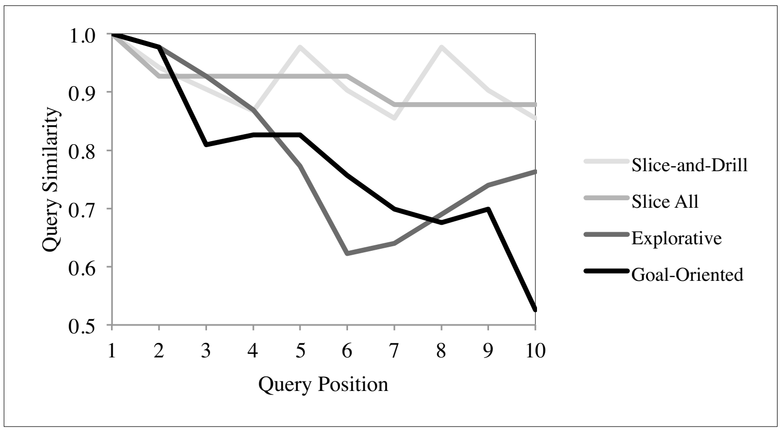

Fig. 6. Intra-session query similarity for the four templates

generated with CubeLoad by setting the maximum length of sessions to 1 and properly tuning the maximum size of seed query reports (differently from the first two classes, data mining queries are characterized by high cardinality of the results). Conversely, the class of *iterative OLAP queries* comprises four base sessions each including exactly 2 queries; more sessions can be generated from each base session by randomly changing a selection predicate. In two of the base sessions, the subsequent queries are not related by the application of a single OLAP operator like in CubeLoad, so they can be quite “distant” from each other, but still they are finalized to the same analysis goal. In the other two base sessions, the two subsequent queries differ from their selection predicate. Thus, an effective way to generate sessions like these ones with CubeLoad is to use the goal-oriented and the slice-all templates and fix the number of seed queries to 4, with a session length equal to 2.

Our future work on this topic will be mainly aimed at enhancing the capabilities of CubeLoad in three directions: (i) by allowing benchmarkers to distinguish *skilled* and *non-skilled* profiles, so as to enable a finer tuning of the workload features; (ii) by defining other templates, so as to make CubeLoad more flexible and usable for a wider array of benchmarks; and (iii) by adopting a more complex query model, so as to make the generated workloads more realistic still. From the engineering point of view, we plan to refactor the CubeLoad code according to an open architecture where each benchmarker can write her own templates in the form of a plugin.

Acknowledgements. We would like to thank Luca Spadazzi for his support in implementing and testing CubeLoad, and Patrick Marcel (Université François Rabelais, Tours, France) for the fruitful discussions about the features of a realistic OLAP workload.

References

1. Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., Turricchia, E.: Mining preferences from OLAP query logs for proactive personalization. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 84–97. Springer, Heidelberg (2011)
2. Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., Turricchia, E.: Similarity measures for OLAP sessions. In: KAIS (to appear, 2014)
3. Aligon, J., Marcel, P.: A framework for user-centric summaries of OLAP sessions. In: Proceedings EDA, Bordeaux, France, pp. 103–117 (2012)
4. Aufaure, M.-A., Kuchmann-Beauger, N., Marcel, P., Rizzi, S., Vanrompay, Y.: Predicting your next OLAP query based on recent analytical sessions. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 134–145. Springer, Heidelberg (2013)
5. Ciaccia, P., Golfarelli, M., Rizzi, S.: Efficient derivation of numerical dependencies. Inf. Syst. 38(3), 410–429 (2013)
6. Darmont, J., Bentayeb, F., Boussaid, O.: DWEB: A data warehouse engineering benchmark. CoRR abs/0705.1453 (2007)
7. Golfarelli, M., Saltarelli, E.: The workload you have, the workload you would like. In: Proceedings DOLAP, New Orleans, Louisiana, pp. 79–85 (2003)
8. Hopkins, B., Skellam, J.G.: A new method for determining the type of distribution of plant individuals. Annals of Botany 18, 213–227 (1954)
9. Hyde, J.: Mondrian documentation (2011), `http://mondrian.pentaho.com/documentation/schema.php`
10. LeFevre, J., Sankaranarayanan, J., Hacigümüs, H., Tatemura, J., Polyzotis, N.: Towards a workload for evolutionary analytics. CoRR abs/1304.1838 (2013)
11. Minnesota Population Center: Integrated public use microdata series (2008), `http://www.ipums.org`
12. O'Neil, P., O'Neil, E., Chen, X., Revilak, S.: The star schema benchmark and augmented fact table indexing. In: Nambiar, R., Poess, M. (eds.) TPCTC 2009. LNCS, vol. 5895, pp. 237–252. Springer, Heidelberg (2009)
13. Poess, M.: Controlled SQL query evolution for decision support benchmarks. In: Proceedings WOSP, Buenes Aires, Argentina, pp. 38–41 (2007)
14. Pöss, M., Smith, B., Kollár, L., Larson, P.Å.: TPC-DS, taking decision support benchmarking to the next level. In: Proceedings SIGMOD Conference, Madison, Wisconsin, pp. 582–587 (2002)
15. Sapia, C.: PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 224–233. Springer, Heidelberg (2000)
16. Sarawagi, S.: User-adaptive exploration of multidimensional data. In: Proceedings VLDB, Cairo, Egypt, pp. 307–316 (2000)
17. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology 147, 195–197 (1981)
18. Stillger, M., Freytag, J.C.: Testing the quality of a query optimizer. IEEE Data Eng. Bull. 18(3), 41–48 (1995)