A Recommender System for DBMS Selection Based on a Test Data Repository

Lahcène Brahimi(✉), Ladjel Bellatreche, and Yassine Ouhammou

LIAS/ISAE-ENSMA, Poitiers University, Futuroscope, Poitiers, France
{lahcene.brahimi,ladjel.bellatreche,yassine.ouhammou}@ensma.fr

Abstract. Nowadays, we see an explosion in the number of Database Management Systems (DBMS) in the market. Each one has its own characteristics. This spectacular development of DBMS is mainly motivated by the need for storing and exploiting the deluge of heterogeneous data for analytical purposes. As a consequence, companies and users are faced with huge range of choices and sometimes it is hard for them to find the relevant DBMS. Some Web sites such as DB-Engines (http://db-engines.com/en/) provide monthly a classification of hundreds of DBMS (303 in April 2016) using metrics related to usage and user feedbacks. These criteria are not always sufficient to help companies and users to make a good choice. Therefore, they have to be enhanced by qualitative measurements obtained by *testing the activities of DBMS* for a set of non-functional requirements. In this perspective, some council such as Transaction Processing Council publish non-functional requirement results of DBMS using their own benchmarks. Another serious producer of test data is the researchers via their scientific papers. Each year they publish a large amount of results of new solutions. To facilitate the exploitation of these test results by small companies and researchers from developing countries, the construction of a test data repository connected to recommender system is an asset for companies/users. In this paper, we first propose a repository for structuring and storing test data. Secondly, a recommender system is built on the top of this repository to advise companies to choose appropriate DBMS based on their requirements. Finally, a proof of concept of our recommender system is given to illustrate our proposal.

1 Introduction

Nowadays, every science discipline (e.g. smart Grids [22], health-care [18], and telecommunication [5]) needs the services offered by the DBMS. The development of efficient database applications represents a crucial issue for companies. This issue has to deal with the diversity, the deluge of data, the emerging technologies, the continuously need for satisfying several non-functional requirements (e.g., the usability, the quality, the security, the response time, the energy consumption, etc.), etc. The diversity covers several aspects: (a) the manipulated data, (b) the database models (relational, XML, Semantic, Graphs, etc.), (c) the DBMS, (d) the deployment platforms (centralized, distributed/parallel, cloud,

© Springer International Publishing Switzerland 2016
J. Pokorný et al. (Eds.): ADBIS 2016, LNCS 9809, pp. 166–180, 2016.
DOI: 10.1007/978-3-319-44039-2_12

data clusters, etc.), (e) the type of workload (Online Transaction Processing (OLTP), Online Analytical Processing (OLAP) or OLTP/OLAP) [1], etc. The V of Big Data defining the volume makes the satisfaction of certain non-functional requirements, such as performance, more difficult. This situation encourages data management editors to propose solutions and new DBMS in order to fitful these requirements. As a consequence, companies are faced to a problem of choosing their DBMS. Recent initiatives have been launched for this purpose. For instance, the objective of DB-Engines[1] is to collect and present information on DBMS and provides monthly a classification of hundreds of DBMS (303 in April 2016) using metrics related to usage and user feedbacks.

Note that the satisfaction of non-functional requirements strongly depends on the used DBMS and the platform. Faced to the diversity of DBMS, a legitimate question that companies have to ask when they develop new database projects is: *what is the favorite DBMS for my application?*

An equivalent question has already been asked in 80s, when companies and organizations start dealing with projects for new types of data and applications. For instance, in [7], the authors attempt to select a DBMS for agricultural record keeping for United States Department of Agriculture (USDA). Recently, with the explosion of advanced platforms, several studies endeavor to evaluate a set of non-functional requirements of a priori known DBMS deployed in a given platform for a specific activity. In [10], the authors evaluate the performance of the *MongoDB* deployed on a *Hadoop* platform for scientific data analysis. This situation is easier for companies, since it supposes the knowledge of the DBMS and the platform.

The response to the above question can be done thanks to the subjective evaluation of the used non-functional requirements by performing intensive testing activities. Note that a testing activity consists in *stimulating* a system in order to observe its response [19]. A stimulus and a response both have values, which may coincide, as when the stimulus value and the response are both real. In the context of the problem of choosing a DBMS, the stimulus includes the values of parameters, e.g. the deployment platform setting, the database schema/instances, the constraints, the access methods, and so on. Observations include values of the metrics describing the used non-functional requirements.

Notice that the testing in the database covers all phases of the life cycle: user requirement collection, conceptual [23], ETL (Extract, Transform, Load), logical, deployment, physical [3] and analysis [12]. In this paper, we concentrate only on the *deployment phase* in which the DBMS hosting the database application and the platform are chosen.

Test activities are time and money consuming. As quoted in [15], *Microsoft spends 50 per cent of its development costs on testing*. Big companies can spend money to test their database solutions deployed in a DBMS. As a consequence, they can tune their solutions to satisfy their requirements. Other organisms and council such as Transaction Processing Council published regularly the performance of *well known* DBMS and platforms based on their benchmark data[2].

[1] http://db-engines.com/en/.
[2] www.tpc.org.

For other companies with a large expertise in simulation, they can simulate the behavior of a set of DBMS and develop mathematical cost models to evaluate the different metrics measuring the asked non-functional requirements. To be more accurate, these metrics have to consider relevant parameters of the database environment such as the schema, the population, the workload, the deployment platform, the DBMS, the used algorithms, the used optimization structures (e.g. indexes, materialized views). Based on the results of the simulation, they can choose the best DBMS that satisfies their requirements. Usually, companies consuming the database technology, especially those belonging to developing countries cannot afford the luxury of Big companies and they do not have enough expertise to develop their own simulations. Thus, another alternative has to be found.

On the other hand, the database community spent a great effort in testing their findings. If we consider only database and information systems conferences and journals, each year, more than 80 % of scientific papers provide intensive experiments to evaluate and compare their proposals. This situation contributes in generating a mass of test data that have to be analyzed. Through this paper, we would like to *think-tank* about the following topic: *are the available test data well structured, presented and stored (in a transparency manner) to be publicly exploited?*

In this study, we propose a "DBLP-like"[3] repository persisting test data to offer researchers and companies the possibility to exploit it. Then, researchers can make a good decision to choose their DBMS, platforms, etc. The repository exploitation can be ensured by recommender systems and machine learning techniques.

In this paper, we present in Sect. 2 basic definitions and concepts related to our studied problem. Section 3 proposes our recommender system and its different components. Section 4 reports a proof of concept for our proposal. Finally, Sect. 5 concludes the paper and highlights some open issues.

2 Background

In this section, we first present the metrics measuring non-functional requirements that a DBMS has to satisfy, then the schema of our repository.

2.1 Database Benchmark Metrics

In the database field, the functional requirements describe the functionalities, the functioning, and the usage of the DBMS and its components. They are specifying a behavioral input/output system such as the calculation, data manipulation and processing, identification, creation, insert, delete, update and others. In general, they are detailed in the system design [16].

[3] http://dblp.uni-trier.de/.

Non-functional requirements [20], also called quality attributes are either optional requirements or needs/constraints, they are detailed in system architecture. Non-functional requirements describe how the system will do. In the context of the advanced databases, the non-functional requirements are usually difficult to test. As a consequence, they are evaluated subjectively [6,14].

To evaluate a non-functional requirement corresponding to the deployment phase, several metrics are used which have to be either maximized or minimized. We can cite some traditional metrics:

- *Query-per-Hour Performance* (QphH@size): it is a measure used to determine the performance of a database system. This metric represents the number of queries executed for one hour relative to the size of the database. The TPC-H[4] which is one of the most popular benchmarks uses this metric.
- *Execution-time*: it represents the time needed for execution resources of the system to process a query.
- *Latency or response time*: it represents the time between the launch of a query and the arrival-time of the first answer. The best response time value of a query corresponds to its run-time.
- *Throughput*: it gives the number of queries performed per time.
- *Utilization rate of a resource*: it is the proportion of the time that the resource is used in a given time.
- *Transmission rate:* it gives the number of tuples produced per time.

2.2 Test Data Repository

The basic idea behind our test data repository was inspired from the presence, in numerous scientific papers of a section describing *Experimental Study*. The analysis of this section allows us to identify repetitive informations that describe the experimental environment and the obtained test results.

This environment contains: the *used platforms*, *the DBMS*, the *operating systems*, the *database* (schema and instances), the *workload*, the *used algorithms*, the *mathematical cost models*, the *hypothesis*, the *metrics* (with their units), the *type of experiments* (simulation, real), the used *external material* to compute the cost of consumed resources such as the energy, etc. From a scientific paper, we can deduce other information such that the *affiliation of the authors*, the *period of the test*, etc. The test data represents the obtained measures of metrics of non-functional requirements. Table 1 gives an example of the experimental environment of [21] that deals with the problem of designing of an energy-aware DBMS. The used metrics represent the consumed energy consumption, the Inputs-Outputs (*IO*) and the CPU cost when executing a workload.

From these informations, we embodied a data warehouse schema ()as a star schema) (Fig. 1) [4]. It is composed of the following dimensions:
Dim_Platform, *Dim_Deployment*, *Dim_DBMS*, *Dim_OS*, *Dim_Dataset*, *Dim_Query*, *Dim_Algorithms*, *Dim_AccessMethods*, *Dim_Hypothesis*, *Dim_Metrics*, *Dim_Laboratory* and *Dim_Time*.
The fact table contains the mathematical and real measures related to metrics (CPU, IO, Network, Energy, etc.).

[4] http://www.tpc.org/tpch/.

Table 1. Testing environment

Laboratory	LIAS/ENSMA
Time	14/05/2015
Platform	Marque: Dell precision T1500 CPU: Intel Core i5 2.27 GHz, Memory: 4 GB of DDR3
Dataset	Star Schema Benchmark (SSB), Size: 100 GB
Operating System	Ubuntu 14.04 LTS kernel 3.13
Workload	Star schema Benchmark (SSB) queries
Deployment	Centralized
Optimization Structures	Materialized views
DBMS	Oracle 11gR2
Algorithm	Nondominated Sorting Genetic Algorithm NSGA II
Hypothesis	Without cache
Metrics	Response time CPU_Cost IO_Cost Energy
External material	Watts UP? Pro ES[a]
Type of experiments material	simulation and real

[a] https://www.wattsupmeters.com/

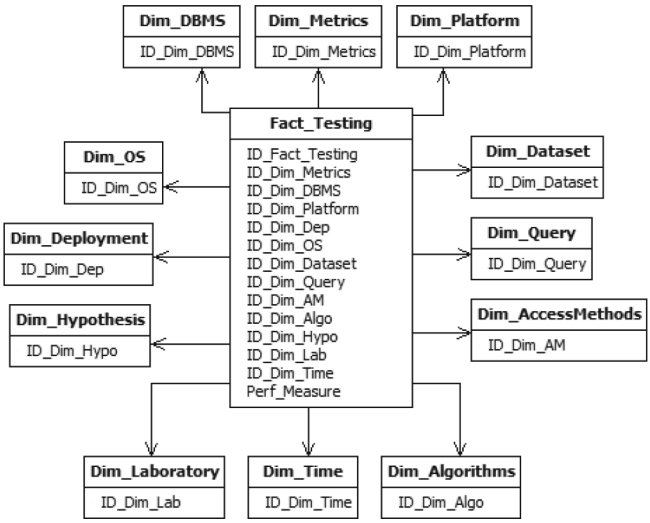


Fig. 1. Our test data repository

Our data warehouse can be exploited by traditional reporting tools (For example, the OLAP Slice and Dice operations shown in Fig. 2), exploration [11, 17], data mining algorithms [2], etc.

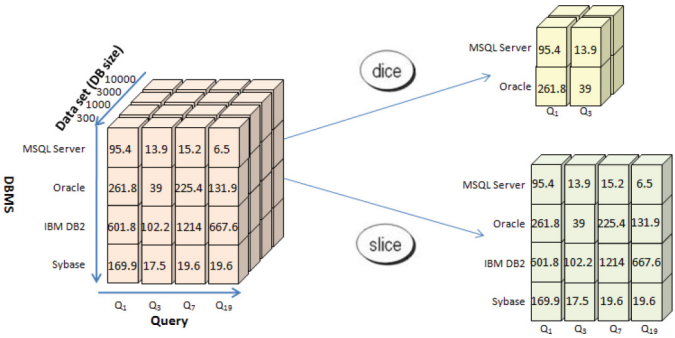


Fig. 2. An example of OLAP slice and dice

3 A Recommender System for Choosing DBMS

To respond to the question that we asked in Introduction, we believe that recommender systems may assist companies in selecting their favorite DBMS. Recommender systems have been largely used in several domains. Three main types of recommender systems exist: collaborative filtering, content-based and knowledge-based. They differ from the information that they use to propose recommendations. The collaborative filtering uses similarities between users and items. Content-based uses static information about users or items. However, knowledge-based depends on informations that are obtained directly from users [13].

3.1 Components of Our Recommender System

The recommendation scenario in our context is the following: We assume that a company/user comes up with a database application with its characteristics related to the database schema, the workload, the platform, etc., and wants getting an advise to choose a relevant DBMS that fulfills its requirements. These informations are described through a document called the *manifest*. Two categories of information are available: (i) *given information* and (ii) *missing information*. The first category defines the valued attributes that a company has, whereas the second one represents the attributes with missing values that the company is looking for.

Note that all attributes used in the manifest belong to the schema of our warehouse. Figure 3 represents an example of a manifest, in which the DBMS and performance metric (estimating QphH) are missing. This means that the company is looking for a DBMS and its performance for its application. Our recommender system has to consider the manifest explores the warehouse to find fragment of test data corresponding to the manifest, and then propose the company a DBMS. To highlight the work-flow related to the test seeking, we describe the steps shown in Fig. 4.

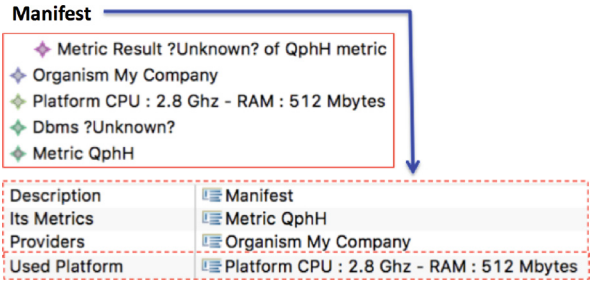


Fig. 3. Example of a manifest

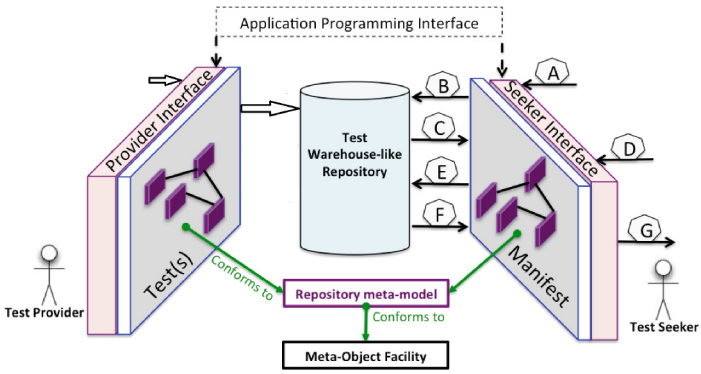


Fig. 4. Overview of the test warehouse-like repository usage

(A) The company chooses to play the role of a test seeker.
(B) The seeker interface transforms the request to a set of queries to select all the dimensions with their values and the metrics (without values) which exist in the test repository.
(C) The seeker interface loads the result of B and presents it to the company (e.g. seeker). This instance corresponds to an empty *Manifest*.
(D) The company enriches the manifest by expressing it needs based on the existing content. Of course, users can add new values related to the dimensions when it is necessary. However, adding new metrics is not possible, because the objective is to orient designers to choose a test configuration depending on the metrics that exist in the repository.
(E) The seeker interface generates from the manifest a set of appropriate SQL queries to explore the test repository.
(F) Based on the *manifest* queries and the repository content, a set of possible tests and their specific configurations, in which missing informations are replaced by the recommended values, are proposed to the seeker via the interface. Note that this problem is quite similar to the problem of clustering with missing data [24]. Several research efforts have been done to solve the above-mentioned problem. Usually, they propose algorithms and

methods to predict the missing values [24]. These algorithms are defined at the attribute level and not at the dimension level. This motivates us to develop our own algorithm. The basic idea is to discard the dimensions which are not expressed in the *manifest*. Based on the obtained results, we estimate the attributes values of unknown dimension(s). This can be done by using machine learning techniques (Fig. 5). The details of this algorithm is presented in Sect. 3.2.

(G) Finally, the user can download information related to the proposed solution. Note that the searching results shall correspond to a repository containing one or several tests depending on the seeker requests. The aim is to allow seekers to download customized repositories referring to their needs.

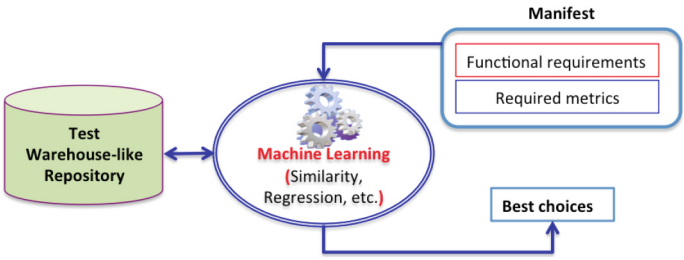


Fig. 5. The structure of recommender system

3.2 Machine Learning Algorithms

At the beginning, we used a linear regression technique to deal with our problem. However, the obtained results were poor in terms of prediction. This is due to occurrences of DBMS in the repository which are not enough for prediction. For instance, in our repository, there is 40 tests involving MS SQL Server, but only 10 for Oracle DBMS.

To avoid the problem of occurrences of tests, we use another algorithm based on similarity between *Manifest* and tests. Before detailing this algorithm, some definitions are given.

Definition 1. *The **similarity** is a comparison between two objects to determine the most important and useful relations between them* [8].

Definition 2. *The **distance** is the inverse measure of the similarity. Several distance functions exists such as Euclidean distance defined as follows:*
Let $P_1(x_1, x_2, ..., x_k)$ and $P_2(y_1, y_2, ..., y_k)$ be two points of a vector space. The distance between P_1 and P_2 is given by the following equation:

$$Distance = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \quad (1)$$

Let x and y be two scalable values. x and y are similar if they verify the following relations [9]:

$$\begin{array}{l} \textit{Relative relation} : \dfrac{x}{y} \approx 1 \quad \text{if } \dfrac{x}{y} \in [1-\epsilon, \dfrac{1}{1-\epsilon}] \\ \textit{Absolute relation} : x - y \approx 0 \ \text{ if } |x-y| \in [0, \epsilon] \end{array} \tag{2}$$

where ϵ is the smallest value in the scale of x or y. Among the two above relations, the relative relation fits better our problem. Therefore, the similarity can be assimilated to the ratio between the estimated and the real measures.

Definition 3. ***Normalization.*** *It is a property of the similarity and requires that all values belonging to the interval [0, 1]. There are various normalizations in statistics. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a sample of n valued items. The normalized value of x_i may be given by:*

$$N = \frac{x_i - Min(x_i)}{Max(x_i) - Min(x_i)} \tag{3}$$

If the distance (D) is normalized, the similarity S is can be given by: $S = 1 - D$.

Now, we have all ingredients to describe and illustrate our algorithm. Let us consider an office design company comes with a *Manifest*, where DBMS and performance metric that estimate QphH are missing. Since the following lines describe the algorithm, Table 2 shows the whole process and its results step by step.

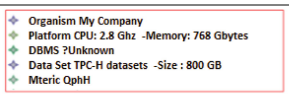
- **step 1:** analyzing of the company *Manifest* to identify the presence of dimensions;
- **step 2:** getting a fragment of the data cube satisfying these dimensions (using Slice and Dice);
- **step 3:** normalizing all the dimension's values using formula 3;
- **step 4:** computing the similarity between the company *Manifest* and each instance of the data cube fragment. Note that an instance represents a test;
- **step 5:** selecting the best propositions based on the result of sorting. Indeed, tests are sorted in relation to similarity results for each DBMS.
- **step 6:** the company can choose its favorite DBMS based on its requirements such as price.

Our algorithm can be extended by considering missing measures, by extracting the fragment of the data cube corresponding on the given dimensions.

4 Proof of Concept

To stress our proposal, we consider real test data available at the TPC website. They correspond to the execution cost (in a single stream) of queries running on four well-known DBMS: Oracle, MS SQL Server, DB2 and Sybase. These data are manually inserted into our repository (about ten tests of each DBMS). Two cases of manifest are considered (Table 3).

Table 2. Example process of our recommender system

<table>
<tr><td>Algorithm's steps</td><td>Example</td></tr>
<tr><td>Step 1
Input: Manifest</td><td>


</td></tr>
<tr><td>Step 2
Input: DW_TEST</td><td>Output:
<table>
<tr><th>DBMS</th><th>Test</th><th>Size</th><th>CPU</th><th>Memory</th><th>QphH</th></tr>
<tr><td rowspan="5">MS SQL Server</td><td>Test1</td><td>1000</td><td>2,8</td><td>1536</td><td>588831</td></tr>
<tr><td>Test2</td><td>3000</td><td>2,5</td><td>3072</td><td>725686</td></tr>
<tr><td>Test3</td><td>3000</td><td>2,5</td><td>3072</td><td>700392</td></tr>
<tr><td>Test4</td><td>3000</td><td>2,8</td><td>3072</td><td>461837</td></tr>
<tr><td>Test5</td><td>10000</td><td>2,8</td><td>4096</td><td>652239</td></tr>
<tr><td rowspan="5">Oracle</td><td>Test6</td><td>1000</td><td>1,5</td><td>64</td><td>9853</td></tr>
<tr><td>Test7</td><td>3000</td><td>2,88</td><td>512</td><td>198907</td></tr>
<tr><td>Test8</td><td>3000</td><td>3</td><td>1024</td><td>205792</td></tr>
<tr><td>Test9</td><td>10000</td><td>1,5</td><td>288</td><td>108099</td></tr>
<tr><td>Test10</td><td>30000</td><td>1,6</td><td>1024</td><td>156960</td></tr>
<tr><td rowspan="5">DB2</td><td>Test11</td><td>100</td><td>3,6</td><td>4</td><td>1894</td></tr>
<tr><td>Test12</td><td>300</td><td>3</td><td>32</td><td>10165</td></tr>
<tr><td>Test13</td><td>1000</td><td>1,7</td><td>32</td><td>20221</td></tr>
<tr><td>Test14</td><td>1000</td><td>1,9</td><td>32</td><td>26156</td></tr>
<tr><td>Test15</td><td>3000</td><td>2,6</td><td>16</td><td>38672</td></tr>
</table>
</td></tr>
<tr><td>Step 3 and 4
Input: Table in above with the following formulas: 1, 3 and S</td><td>
<table>
<tr><th>DBMS</th><th>Test</th><th>Size</th><th>N1</th><th>CPU</th><th>N2</th><th>Memory</th><th>N3</th><th>QphH</th><th>Distance</th><th>N</th><th>S</th></tr>
<tr><td rowspan="5">MS SQL Server</td><td>Test1</td><td>1000</td><td>0,03</td><td>2,8</td><td>0,62</td><td>1536</td><td>0,37</td><td>588831</td><td>0,19</td><td>0,17</td><td>0,83</td></tr>
<tr><td>Test2</td><td>3000</td><td>0,10</td><td>2,5</td><td>0,48</td><td>3072</td><td>0,75</td><td>725686</td><td>0,59</td><td>0,52</td><td>0,48</td></tr>
<tr><td>Test3</td><td>3000</td><td>0,10</td><td>2,5</td><td>0,48</td><td>3072</td><td>0,75</td><td>700392</td><td>0,59</td><td>0,52</td><td>0,48</td></tr>
<tr><td>Test4</td><td>3000</td><td>0,10</td><td>2,8</td><td>0,62</td><td>3072</td><td>0,75</td><td>461837</td><td>0,57</td><td>0,50</td><td>0,50</td></tr>
<tr><td>Test5</td><td>10000</td><td>0,33</td><td>2,8</td><td>0,62</td><td>4096</td><td>1,00</td><td>652239</td><td>0,87</td><td>0,77</td><td>0,23</td></tr>
<tr><td rowspan="5">Oracle</td><td>Test6</td><td>1000</td><td>0,03</td><td>1,5</td><td>0,00</td><td>64</td><td>0,01</td><td>9853</td><td>0,64</td><td>0,57</td><td>0,43</td></tr>
<tr><td>Test7</td><td>3000</td><td>0,10</td><td>2,88</td><td>0,66</td><td>512</td><td>0,12</td><td>198907</td><td>0,10</td><td>0,09</td><td>0,91</td></tr>
<tr><td>Test8</td><td>3000</td><td>0,10</td><td>3</td><td>0,71</td><td>1024</td><td>0,25</td><td>205792</td><td>0,14</td><td>0,12</td><td>0,88</td></tr>
<tr><td>Test9</td><td>10000</td><td>0,33</td><td>1,5</td><td>0,00</td><td>288</td><td>0,07</td><td>108099</td><td>0,70</td><td>0,62</td><td>0,38</td></tr>
<tr><td>Test10</td><td>30000</td><td>1,00</td><td>1,6</td><td>0,05</td><td>1024</td><td>0,25</td><td>156960</td><td>1,13</td><td>1,00</td><td>0,00</td></tr>
<tr><td rowspan="5">DB2</td><td>Test11</td><td>100</td><td>0,00</td><td>3,6</td><td>1,00</td><td>4</td><td>0,00</td><td>1894</td><td>0,42</td><td>0,37</td><td>0,63</td></tr>
<tr><td>Test12</td><td>300</td><td>0,01</td><td>3</td><td>0,71</td><td>32</td><td>0,01</td><td>10165</td><td>0,20</td><td>0,18</td><td>0,82</td></tr>
<tr><td>Test13</td><td>1000</td><td>0,03</td><td>1,7</td><td>0,10</td><td>32</td><td>0,01</td><td>20221</td><td>0,55</td><td>0,49</td><td>0,51</td></tr>
<tr><td>Test14</td><td>1000</td><td>0,03</td><td>1,9</td><td>0,19</td><td>32</td><td>0,01</td><td>26156</td><td>0,46</td><td>0,41</td><td>0,59</td></tr>
<tr><td>Test15</td><td>3000</td><td>0,10</td><td>2,6</td><td>0,52</td><td>16</td><td>0,00</td><td>38672</td><td>0,22</td><td>0,19</td><td>0,81</td></tr>
<tr><td></td><td>Manifest</td><td>800</td><td>0,02</td><td>2,8</td><td>0,62</td><td>768</td><td>0,19</td><td></td><td>0,00</td><td>0,00</td><td>1,00</td></tr>
</table>
</td></tr>
<tr><td>Step 5 and 6
Result:</td><td>
<table>
<tr><th>DBMS</th><th>QphH</th><th>S</th></tr>
<tr><td>MS SQL Server</td><td>588831</td><td>0.81</td></tr>
</table>
</td></tr>
</table>

Table 3. The cases of the experimental study

	Dataset	Workload	Platform	DBMS
Case 1	✓	✓	✓	?
Case 2	✓	✓	?	?

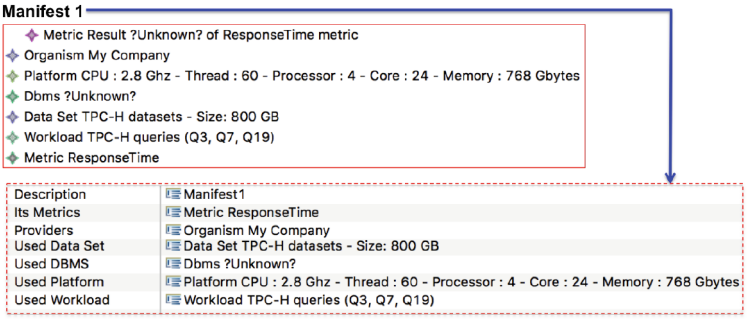


Fig. 6. Excerpt of the *manifest* corresponding to case 1

Case 1. It corresponds to the scenario where a company looks for a DBMS. It can expresses its requirement through a *manifest* as it is shown in Fig. 6.

This means that the user would like to know the response-time of the TPC-H queries (i.e. Q_3, Q_7, Q_{19}) depending on specific platform and dataset. Moreover, referring to the result related the response-time metric; we can recommend a list of suitable DBMS that matches its requirements.

Table 4. Q_3, Q_7, Q_{19} response time(s) with four DBMS

	Oracle	MS SQL Server	DB2	Sybase
Q3	6.80	5.40	102.50	35.50
Q7	34.30	2.80	677.80	37.50
Q19	50.30	2.50	262.20	19.30
Similarity	0.74	0.81	0.48	0.49

Table 4 represents the results obtained that shows the response time of Q_3, Q_7, Q_{19} with MS SQL Server, Oracle, DB2 and Sybase. So, according to the obtained results, we can sort the found DBMSs. In first position, we find MS SQL Server which shows performances of speed (Response time) $Q_3 = 4.37\,\text{s}$, $Q_7 = 2.26 * 0.99\,\text{s}$ and $Q_{19} = 2.02\,\text{s}$ (Response time * Similarity). The overall performance of that Sybase and DB2 DBMS is high. Notice that Sybase outperforms Oracle for the query Q_{19}. Therefore, we can recommend MS SQL Server to satisfy this manifest.

Case 2. It corresponds to the scenario in which a company looks for both a DBMS and a platform. Its *manifest* is shown in Fig. 7. Let us assume that this company uses the same configuration used in the case 1, except the platform is missing. We would like to precise that in Case 2, the company does not ignore the platform dimension, but it looks for a relevant platform and a DBMS.

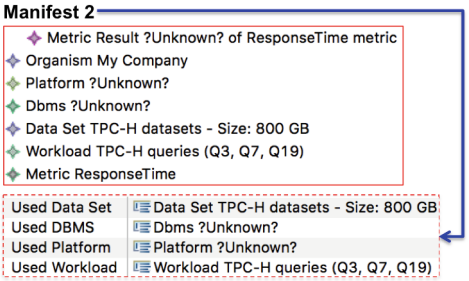


Fig. 7. Excerpt of the *manifest* that corresponds to the case 2

Table 5. Selected DBMS and platforms based on the response-times of Q_3, Q_7, Q_{19}

	Oracle	MS SQL Server	DB2	Sybase
CPU	1.3	2.8	1.9	2.8
Proc	64	4	8	2
Threads	64	120	32	4
Cores	64	60	16	4
Memory	256	1536	32	16
Q3	6.8	4.7	27.3	1429.4
Q7	34.3	2.8	150.4	573.8
Q19	50.3	2.3	163.6	469.2
Similarity	0.99	0.97	0.98	0.93

Table 5 represents the results obtained that shows the response-times of queries Q_3, Q_7, Q_{19}. These response-times are categorized based on DBMS and the platform configurations. We can see that MS SQL Server is the best DBMS according to the computed response-times. Moreover, it is related to the following platform configuration (i.e. CPU: 2.8 GHz - Proc: 4 - Threads: 120 - Cores: 60 - Memory: 1536 GB).

5 Related Work

Before reviewing the important organisms and councils whose the main activity is publishing test data, let us notice that in our recent work [4], concerns the *static part of warehouse*. We only concentrated on proposing a test data repository and we show the interest of using model-driven engineering techniques to perform this design and describe the manifest.

The transaction processing council offers a large panoply of benchmarks covering: transaction processing - OLTP (TPC-C TPC-E), Decision Support (TPC-H, TPC-DS, TPC-DI), virtualization (TPC-VMS, TPCx-V), Big Data (TPCx-HS, TPCx-BB) and common specifications (TPC-Energy, TPC-Pricing).

This council works in close collaboration with industrial partners by delivering them trusted results.

In computational science such as physics and automatics, we recently assist in the development of repository persisting the results of experiments and simulations. The *cTuning* repository[5] is open-source, customizable Collective Knowledge Repository for physics domain. It aggregates developments, ideas and techniques, and allows users to share, cross-link and reference any object and knowledge (workloads, data sets, tools, optimization results, predictive models, etc.) as a reusable component with a unified JSON API via GitHub. *AiiDA*[6] is a flexible and scalable informatics' infrastructure to manage, preserve, and disseminate the simulations, data, and work-flows of modern-day computational science to ensure reproducibility.

6 Conclusion

The data warehousing and recommender systems have been applied in numerous domains manipulating huge amount of historical data. Scientific papers, councils and research foundations represent rich test data sources that have to be exploited by researchers and companies for developing countries. In this paper, we attempt to federate the database community around the importance of the available test data and to motivate them to build "DBLP-like" repository that can play the role of a test data warehouse. Its dimensions represent several aspects of a test environment: database, dataset, workload, platform, DBMS, algorithms, hypothesis, non-functional requirements, unit of measure, etc. The fact table of our warehouse contains all measures corresponding to metrics describing non-functional requirements. This warehouse can be used either by traditional OLAP tools for exploration and reposting activities or by systems recommending companies the relevant DBMS based on their *manifest*. Two case studies are given and showed the utility of our approach.

Our paper opens several issues: (i) the development of comprehensive forms allowing researchers putting their test results in the repository, (ii) providing a mechanism making our system *trustworthy* and (iii) generalization of our repository to consider other phases of the life cycle of database design such as conceptual, logical and ETL.

References

1. Ahmad, M., Aboulnaga, A., Babu, S., Munagala, K.: Interaction-aware scheduling of report-generation workloads. VLDB J. **20**(4), 589–615 (2011)
2. Baralis, E., Meo, R., Psaila, G.: Data mining in data warehouses. In: SEBD, pp. 51–65 (1999)

[5] http://ctuning.org/index.html.
[6] http://www.aiida.net/.

3. Bouchakri, R., Bellatreche, L., Hidouci, K.-W.: Static and incremental selection of multi-table indexes for very large join queries. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS 2012. LNCS, vol. 7503, pp. 43–56. Springer, Heidelberg (2012)
4. Brahimi, L., Ouhammou, Y., Bellatreche, L., Ouared, A.: More transparency in testing results: towards an open collective knowledge base. In: 10th IEEE International Conference on Research Challenges in Information Science, pp. 315–320 (2016)
5. Chen, Q., Hsu, M., Dayal, U.: A data-warehouse/OLAP framework for scalable telecommunication tandem traffic analysis. In: ICDE, pp. 201–210 (2000)
6. Chung, L., do Prado Leite, J.C.S.: On non-functional requirements in software engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 363–379. Springer, Heidelberg (2009)
7. Cross, T.L., Lane, R.J., et al.: Selecting a database management system for agricultural record keeping. Technical report (1988)
8. Cross, V.V., Sudkamp, T.A.:Similarity and compatibility in fuzzy set theory: assessment and applications, vol. 93 (2002)
9. Dague, P., Travé-Massuyès, L.: Raisonnement causal en physique qualitative. Intellectica **38**, 247–290 (2004)
10. Dede, E., Govindaraju, M., Gunter, D., Canon, R.S., Ramakrishnan, L.: Performance evaluation of a mongodb and hadoop platform for scientific data analysis. In: Proceedings of the 4th ACM Workshop on Scientific Cloud Computing, pp. 13–20 (2013)
11. Furtado, P., Nadal, S., Peralta, V., Djedaini, M., Labroche, N., Marcel, P.: Materializing baseline views for deviation detection exploratory OLAP. In: DAWAK, pp. 243–254 (2015)
12. Golfarelli, M., Rizzi, S.: Data warehouse testing: a prototype-based methodology. Inf. Softw. Technol. **53**(11), 1183–1198 (2011)
13. Gomez-Uribe, C.A., Hunt, N.: The netflix recommender system: algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. **6**(4), 13 (2016)
14. Gross, D., Yu, E.: From non-functional requirements to design through patterns. Require. Eng. **6**(1), 18–36 (2001)
15. Haftmann, F., Kossmann, D., Lo, E.: A framework for efficient regression tests on database applications. VLDB J. **16**(1), 145–164 (2007)
16. Lauesen, S.: Task descriptions as functional requirements. Softw. IEEE **20**(2), 58–65 (2003)
17. Ordonez, C., Chen, Z., García-García, J.: Interactive exploration and visualization of OLAP cubes. In: ACM DOLAP, pp. 83–88 (2011)
18. Park, Y., Shankar, M., Park, B., Ghosh, J.: Graph databases for large-scale healthcare systems: a framework for efficient data management and data services. In: Workshops Proceedings of the ICDE, pp. 12–19 (2014)
19. Pezzè, M., Zhang, C.: Automated test oracles: a survey. Adv. Comput. **95**, 1–48 (2015)
20. Rosenmüller, M., Siegmund, N., Schirmeier, H., Sincero, J., Apel, S., Leich, T., Spinczyk, O., Saake, G.: Fame-DBMS: tailor-made data management solutions for embedded systems. In: Proceedings of the 2008 EDBT Workshop on Software Engineering for Tailor-Made Data Management, pp. 1–6 (2008)
21. Roukh, A., Bellatreche, L., Boukorca, A., Bouarar, S.: Eco-DMW: eco-design methodology for data warehouses. In: ACM DOLAP, pp. 1–10 (2015)

22. Siksnys, L., Thomsen, C., Pedersen, T.B.: MIRABEL DW: managing complex energy data in a smart grid. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 443–457. Springer, Heidelberg (2012)
23. Tort, A., Olivé, A., Sancho, M.-R.: An approach to test-driven development of conceptual schemas. Data Knowl. Eng. **70**(12), 1088–1111 (2011)
24. Wagstaff, K.: Clustering with missing values: no imputation required. In: Banks, D., McMorris, F.R., Arabie, P., Gaul, W. (eds.) Classification, Clustering, and Data Mining Applications. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 649–658. Springer, Heidelberg (2004)