Coupling Multi-Criteria Decision Making and Ontologies for Recommending DBMS

Lahcène Brahimi
LIAS/ISAE-ENSMA,
Futuroscope, France
lahcene.brahimi@ensma.fr

Ladjel Bellatreche
LIAS/ISAE-ENSMA,
Futuroscope, France
bellatreche@ensma.fr

Yassine Ouhammou
LIAS/ISAE-ENSMA,
Futuroscope, France
yassine.ouhammou@ensma.fr

ABSTRACT

In the last decade, DBMSs have gained a great importance in research and industry. Nowadays, we are getting more than 318 DBMSs in the market, where each one has its own features and fits some specific requirements in terms of storage, management and query processing. This situation contributes in complicating the choice of a company to deploy a new database applications and/or migrating it from one DBMS to another. Certainly, database experts have all competencies to choose their appropriate DBMS, but their diversity may complicate this choice. The main goal is to provide a recommender system assisting experts and companies to perform the selection of their DBMS based on the features of DBMSs. To the best of our knowledge, there is no tool that assists companies to make a good choice when selecting a DBMS that fitful their requirements and constraints. To satisfy this objective, understanding the whole architecture of any DBMS (its parameters) is necessary. In this paper, we propose firstly, a domain ontology describing common and uncommon features of DBMSs that help us in understanding the sense of each parameter of any DBMS. Secondly, we present an adaptation of the multi-criteria decision making technique exploiting this ontology to assist and advise companies to choose their appropriate DBMS. Finally, we give a case study to illustrate our proposal and the utility of our ontology.

1. INTRODUCTION

Certainly, relational database technology has its success story for several decades, where all companies used relational DBMSs to store and manage their data. But, with the explosion of the NoSQL technology, this technology got its reign threatened. As a consequence, the major DBMS editors and start up companies have made great efforts to improve their DBMSs, by integrating new features, or launching new DBMSs to satisfy the new needs of companies (such as data analytics). DB-Engine site[1] has identified more than 318 DBMSs, where each one has its own specificities and utility. Furthermore, various DBMSs categories have been proposed to meet many specific needs.

[1]http://db-engines.com/en/ranking

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice, and the full citation on the first page. To copy otherwise, or to republish the work, requires prior permission from CSI SIGDATA or from the authors.
International Conference on Management of Data (COMAD), 2017
Copyright 2017 Computer Society of India (CSI) SIGDATA.

For example, the event stores DBMSs category considers data as series of immutable events and they keep all state changing events for an object together with a timestamp. Also, time series DBMS category has been constructed to deal with time series data. Moreover, search engines DBMSs category has designed for data content search (Figure 1).

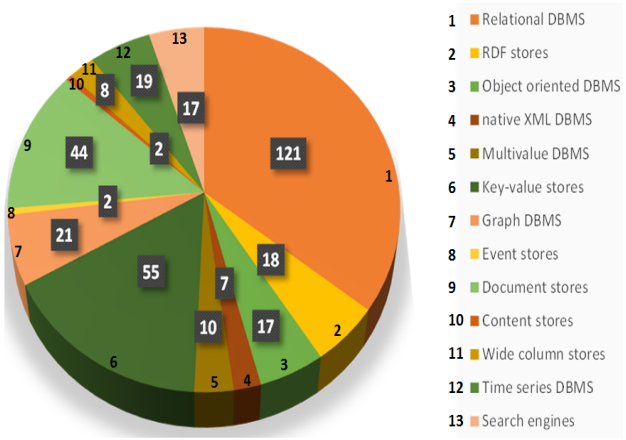

Figure 1: Number of systems per category

Figure 1 and 2 give an overview of the number of systems per category in addition to the DBMSs ranking over the last three months according to DB-Engine.
Given this wealth, it is often difficult for a company to choose the appropriate DBMS that fits its requirements. These requirements are mostly related to the type and size of data owned by the company and also related to the kind of queries. It should be noticed that data are becoming more and more complex. They include multimedia data, graphics, and photographs [11]. Moreover, the number of users who can simultaneously access to the data, upload and share files, photos and videos is exponentially growing. The expected performance is about the response time, energy, resource consumption and availability of data processing functions. The scalability in addition to the cost of acquisition, training and maintenance are also to be taken into account. All the above aspects represent constraints that impact the DBMS selection process. Thus, they should be addressed when making a decision about a DBMS selection. Indeed, in [1], the authors pointed out that the decision of choosing a relevant DBMS is hard because it requires several DBMS parameters and constraints to be satisfied. Due to these constraints and DBMSs diversity, the process of selecting DBMSs that meet specific requirements is a challenging issue. Note that each constraint has an impact on the overall choice.

Rank			DBMS	Database Model	Score		
Nov 2016	Oct 2016	Nov 2015			Nov 2016	Oct 2016	Nov 2015
1.	1.	1.	Oracle	Relational DBMS	1413.01	-4.09	-67.94
2.	2.	2.	MySQL	Relational DBMS	1373.56	+10.91	+86.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	-0.38	+91.48
4.	↑ 5.	↑ 5.	PostgreSQL	Relational DBMS	325.82	+7.12	+40.13
5.	↓ 4.	↓ 4.	MongoDB	Document store	325.48	+6.67	+20.87
6.	6.	6.	DB2	Relational DBMS	181.46	+0.90	-21.07
7.	7.	↑ 8.	Cassandra	Wide column store	133.97	-1.09	+1.05
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	125.97	+1.30	-14.99
9.	9.	↑ 10.	Redis	Key-value store	115.54	+6.00	+13.13
10.	10.	↓ 9.	SQLite	Relational DBMS	112.00	+3.43	+8.55

Figure 2: Ranking of database management systems: 10 most popular DBMSs

By examining the features of DBMSs, we can understand the difficulty of finding a system that simultaneously satisfies all constraints expressed by the user. This could be achieved by designing a common environment that represents a pivotal concept of the DBMSs. To do so, the best way is to develop a domain ontology describing all the features of DBMSs.

Note that ontologies have been widely developed in engineering domain [2]. Contrary to conceptual models, an ontology aims at describing in a consensual way the whole knowledge of a domain. This description is agreed and shared by domain experts allowing them to understand each other [2]. The presence of a such ontology allows us instantiating any DBMS, by considering only its *applicable (rigid) properties* [2]. Thus, ontologies have largely contributed in defining recommender systems [16].

So, to remedy these lacks and facilitate the selection task, it is essential to find a methodology that helps the user to choose the most appropriate system(s). More precisely, the goal consists to provide for companies recommendations ensuring the best compromise by respecting the majority of constraints.

Our main contributions are summarized in the following points:

1. We design domain ontology to describe in details and in explicit manner any DBMS system. This is possible thanks to the large usage experience and maturity of ontology.

2. We propose an adaptation of the multi-Criteria decision making (MCDM) techniques through the weighted sum method on DBMS selection.

The layout of this paper is structured as follows: we present in Section 2 a detailed related work. Basic definitions and concepts related to our studied problem are provided in Section 3. We detail in Section 4 a motivating example to understand the problem posed. A definition of a new model inspired by previous research and based on domain ontology are explained in Section 5. Section 6 proposes our adaptation of the weighted sum method on DBMS selection. Section 7 discusses in a case study example the first benefits of our proposal. Finally, Section 8 concludes the paper with some directions to future work.

2. RELATED WORK

The problem of DBMS selection got less attention compared to other traditional problems in the database field. This problem has been implicitly treated, because, previously, companies are referring to the experts opinion of the domain of databases. In the literature, four existing closer approaches may deal indirectly with our problem: (i) DBMS classification, (ii) DBMS benchmarking, (iii) DBMS conceptual modeling, and (iv) DBMS recommending based on conceptual modeling. DBMSs classification usually follows the evolution of DBMSs technology (Relational, Object Oriented, XML DBMS, Multidimensional, Graph Databases, NoSQL). Each database generation has its own DBMS [9]. A fine grained classification may exist based on some features, which can be supported or not by DBMSs such as ACID properties (Atomicity, Consistency, Isolation, Durability), which are supported by relational DBMSs [13]. The BASE properties (Basically Available, Soft state, Eventual consistency), which are supported by NoSQL systems such CouchDB and Cassandra [7]. Unlike to the relational, NoSQL supports horizontal scaling while providing flexibility in the data model. From then on, NoSQL appears as a solution to the company wishing to manage high loads and volumes. The difficulty in managing the low data consistency for developers has led the great tenors of the web to develop NewSQL. This new RDBMS allows horizontal scalability, schema flexibility and strong data consistency through ACID transactions (Figure 3).

The 451 research group[2] developed a data platforms landscape map with a very large volume of features. This work allows us to extract several DBMSs classification. We can cite the classification by families, relational DBMSs, NoSQL systems and its four classes (Key-value oriented, document oriented, column oriented, graph oriented), and NewSQL systems and its three classes (MySQL ecosystem, advanced clustering/sharding, NewSQL databases). Thus, we can deduce a classification based on the type of use of the DBMSs, the DBMSs intended for: general purpose, analytic, as-a-service (e.g. cloud) and big data.

[2]https://451research.com/

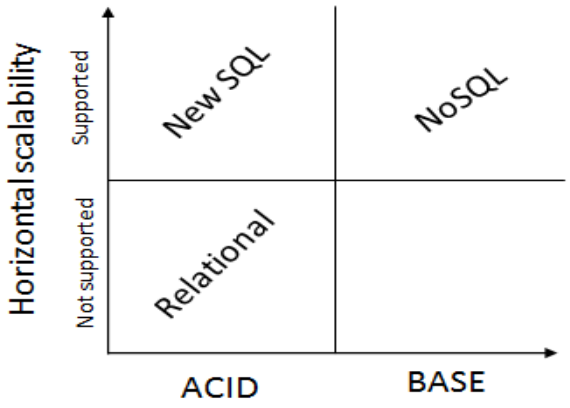

Figure 3: Relational, NoSQL and NewSQL DBMSs

Benchmarking can be another alternative to recommend a DBMS. This can be done by empirical testing of the most popular DBMS based on the company's requirements. The TPC(Transaction Processing Performance Council)[3] spends a lot of efforts in testing the most important DBMS belonging to various generations of databases (TPC-C for Online Transaction Processing, TPC-H for Decision Support applications, Transaction Processing - OLTP, TPC-VMS for Virtualization and TPCx-HS for Big Data). However, benchmarking costs in terms of time and budget and requires a preliminary stage of functional choice. Another limitation of this alternative is that it is based on *popularity of DBMS* instead of considering the relevance and the completeness dimensions. Unfortunately, several DBMSs are rich in terms of supported features but are ignored because they are not well known.

DBMS conceptual modeling is another opportunity to deal with our problem. This issue has been motivated by the slogan *"one size fits all"* [20]. Several studies incorporate the variability in DBMS design. They borrowed expertise from Software Product Line (SPL) community. This allows to design specialized systems by varying some parameters of the original DBMSs to respect new requirements such as energy. To do so, efforts have been elaborated to conceptual model DBMS features. In [20], the authors presented a work on *the FAME-DBMS research project* [4]. The objective is to develop, extend and evaluate techniques and tools to implement and customize DBMSs. These techniques must take into account the particular requirements of embedded systems. For this, they used the software product line (SPL) approach based on the static composition of features.

The complexity and the less predictable of the existing database management systems (ie, performance consistency with increased functionality and data growth is not certain), leading researchers and engineers of databases to review DBMS architectures to meet the needs of new hardware and application trends.

To this end, the authors of [22] proposed a biological-based DBMS called *cellular DBMS*. This architecture was inspired by the FAME-DBMS project allowing the development of highly customizable and autonomous DBMS. The main limitation of these efforts is that they consider only some concepts and properties (parameters) from the *most popular DBMS* and ignore other relevant DBMS [12].

[3]www.tpc.org

[4]https://www4.cs.fau.de/Research/FAME-DBMS/

Recently, some research efforts have been conducted to propose *recommender systems* based on DBMS conceptual model. In [5], the authors have proposed a recommender DBMS system to satisfy non-functional requirements such as performance expressed by a manifest using machine learning techniques. This can be done by the construction of a repository schema storing all experiment tests that are available in Web sites such as TPC. This approach shares the limitation of modeling approaches.

3. DECISION MAKING PROCESS

In this section first, we present multi-criteria decision making (MCDM) techniques. Then we will focus on the best MCDM methods: *the weighted sum model* that will be used in our proposal.

3.1 Multi-criteria decision making (MCDM)

Multi-criteria decision making (MCDM) has grown as a part of operations research, concerned with designing computational and mathematical tools for supporting the subjective evaluation of a set of alternatives in terms of a set of multiple, usually conflicting, decision criteria. MCDM problems are common in everyday life. In personal context, a house or a car one buys may be characterized in terms of price, size, style, safety, comfort, etc. In business context, MCDM problems are more complicated and usually of large scale. For example, many companies in Europe are conducting organizational self-assessment using hundreds of criteria and sub-criteria set in the EFQM (European Foundation for Quality Management) business excellence model. Purchasing departments of large companies often need to evaluate their suppliers using a range of criteria in different area, such as after-sale service, quality management, financial stability, etc. In recent years several previous studies have employed MCDM tools and applications to solve area problems such as energy, environment, quality management, safety and risk management, manufacturing systems, technology and information management, operation research and soft computing, etc. MCDM methods have been designed to designate a preferred alternative and classify alternatives according to a subjective preference order. MCDM is a generic term for all methods that exist for helping people make decisions according to their preferences [15].

3.2 Weighted sum model

The weighted sum model (WSM) is the earliest and probably the most used [21]. It is known as the best and the simplest multi-criteria decision making/analysis (MCDM/MCDA) [3]. It combines the different objectives and weights corresponding to those objectives to create a single score for each alternative to make them comparable. So, the best alternative is chosen as the one which has the maximum WSM score. The different objectives are assumed to be positive [14]. Given a set of m alternatives denoted $A_1, A_2, .., A_m$ and a set of n decision criteria denoted $C_1, C_2, .., C_n$. Furthermore, let assume that all the criteria are benefit criteria, and w_j denotes the relative weight of importance of the criterion C_j and a_{ij} is the performance value of the $i-th$ alternative when it is evaluated in terms of the $j-th$ criterion. Then, the total importance of the $i-th$ alternative, denoted as $A_i^{WSM-score}$, is defined as follows [21]:

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}, \quad for \quad i = 1,2,3,...m. \tag{1}$$

Consequently, the best alternative is the one that satisfies (in the maximization case) the following expression:

$$A_*^{WSM-score} = \max_i \sum_{j=1}^{n} w_j a_{ij}, \quad for \quad i = 1,2,3,...m. \tag{2}$$

4. MOTIVATING EXAMPLE

In this section, we illustrate some examples for the sake of understanding the difficulty of the discussed choice and for explaining the interest in a first phase to construct a DBMS model and its instances.
Let consider table 1 containing some examples of widely used DBMSs. Here, three are relational DBMSs, namely Oracle[5], PostgreSQL [6] and MySQL[7]. Two are NoSQL DBMSs, namely, MongoDB [8] a document oriented system and Cassandra[9] a column oriented system. In our example, six features are presented: storage model, availability of operating systems, programming language supported, query language, structuring of data and supported access methods.

Now, let us assume that a company/user is looking for a DBMS that meets the following requirements:
(i) the stored data are not structured and
(ii) SQL is the query language mastered by users.
By examining the entries of Table 1, we can figure out that no system satisfies simultaneously these two constraints. Since the requirements are contradictory referring to the available six features (parameters). Indeed, the choice can be made based on the query language and we choose from: Oracle, MySQL or PostgreSQL, either based on the nature of data and we select from: MongoDB and Cassandra. Therefore, with the proposed systems, it was impossible to arrive at a solution satisfying both constraints simultaneously, because several aspects are missing, such as:

- A comprehensive model initially representing a functional environment;
- All instances of DBMSs;
- The relationships between features (equivalence, autonomy,...).

5. NEW DATABASE SYSTEMS MODEL

Generally, computer products have multiple functions, particularly in storage, management and processing solutions. The first scan of systems functionalities shows that we can categorize them into three groups:

- Common to all products: price, popularity, feedback, documentation, website.
- Common to all software products: License (proprietary or open source), server operating systems (Windows, Linux, Unix), etc.
- Common and specific to DBMSs: database model (relational, object oriented, key-value oriented, document oriented, column oriented, graph oriented, etc.), architect (centralized, cloud, etc.), supported programming languages (Java, C, C++, etc.), query language (SQL, XQuery, SPARQL, etc.) [17], data manipulation techniques and access methods.

Despite the new technologies of databases, practically each system consists of two main parts: the first part is linked to the system environment and the second part is linked to the database.

[5]http://docs.oracle.com/
[6]http://www.postgresql.org/
[7]https://www.mysql.fr/
[8]http://www.mongodb.org/
[9]http://cassandra.apache.org/

Usually the first part consists of four components: the machine, the operating system, the DBMS and the performance provided by these three elements taking into account the characteristics of the database. According to this principle, our proposal consists to develop a common model for database systems (DBS), which is composed of four concepts (Figure 4): DBMS, Operating System, Hardware and Performance and each being composed of other elements.

- The availability of operating systems with the inherent characteristics is a very important factor for users to ensure an easy integration of new DBMS with the overall corporate information system. It has a strong interaction with the hardware;
- The hardware represents the device characteristics, the frequency and the RAM size. The clock speed and the number CPU cores, in addition to the capacity and speed of the hard disk, are important characteristics;
- DBMS represents the core of the DBS and ensures interaction between the user and the database to capture and analyze the data. It is the main actor in performance database systems;
- The last component is the performance expressed as the usability, the quality, the security, the performance, the normalization, the integrity, etc. The percentage of resource consumption such as RAM, CPU and hard disk and the evaluation results obtained computing response time, energy consumption, etc. using multiple benchmarks as YCSB, TPC and others [9] [10] are very important constraints that play a pivotal role in the evaluation and validation of the database systems.

The goal of modeling database systems is to understand its structure, functioning and presenting in a form of concepts, attributes and instances related to the domain. First, it ensures the interaction between components. Then, it can provide a well detailed functional hierarchy.

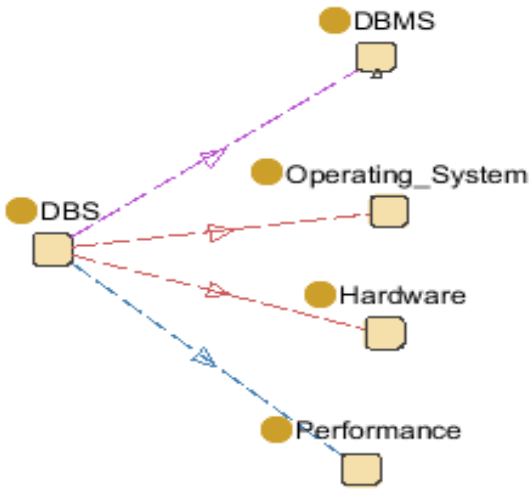

Figure 4: Database systems ontology

Table 1: Example of DBMSs with six different features: Oracle, PostgreSQL, MySQL, MongoDB and Cassandra

DBMS	Oracle	MySQL	PostgreSQL	MongoDB	Cassandra
Model	Relational	Relational	Relational	document Oriented	column Oriented
Operating system	Windows, Unix, Linux	Windows Unix, Linux	Windows Unix, Linux	Windows Unix, Linux	Windows Unix, Linux
API	Java, c ...	Java, c ...	Java, c ...	Java, c ...	Java, c ...
Query language	SQL	SQL	SQL	Command line	CQL
Data structured	Structured	Structured	Structured, Semi structured	Unstructured	Unstructured
Access methods	Index join, B-tree	B-tree, Hash	B-tree, Hash, GIST, GIT	Single and compound field index	Single index

5.1 Database management system class

A few years ago, there were a dozen Database Management Systems (DBMSs) that companies and educational organizations shared to develop database applications. Today we are witnessing a proliferation of new DBMSs developed to meet the new needs motivated by the era of web and Big Data (Figure 5).

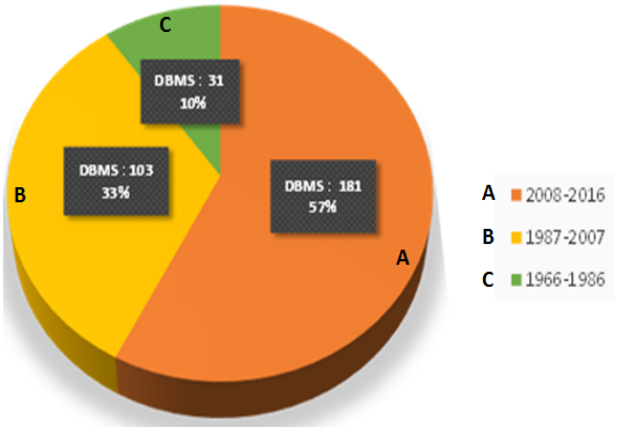

Figure 5: DBMSs development over the years

Many DBMSs are available on the market, from open source DBMSs to commercial DBMSs, from DBMSs specially aimed at professionals to DBMS for beginners and from specific DBMSs such as search engines and data indexing (e.g., Elasticsearch and others) to DBMSs for general use (e.g. Oracle and others). The common point is that all systems have more functionality but more expensive. This situation sometimes returns at cost price of commercial systems and sometimes returns at the cost of processing, storage, maintenance, training , Etc., particularly for open source systems. For that, we focus on the DBMS component and its features. As said previously, unavailability of a classification for DBMS models and the associated features was the motivation for taking a step towards the development of a DBMS model by studying and analyzing their features and functioning, in particular concerning data access, manipulation and processing, and data distribution and storage model (Figure 6). The integration of system features in the same model facilitates the search of a targeted system. Our DBMS model, mainly based on previous research on DBMSs (FAME-DBMS) in [20] and Cellular DBMS in [22], especially in the decomposition and modeling features. We also benefit from the ontology domain through its ease of representation and reasoning.

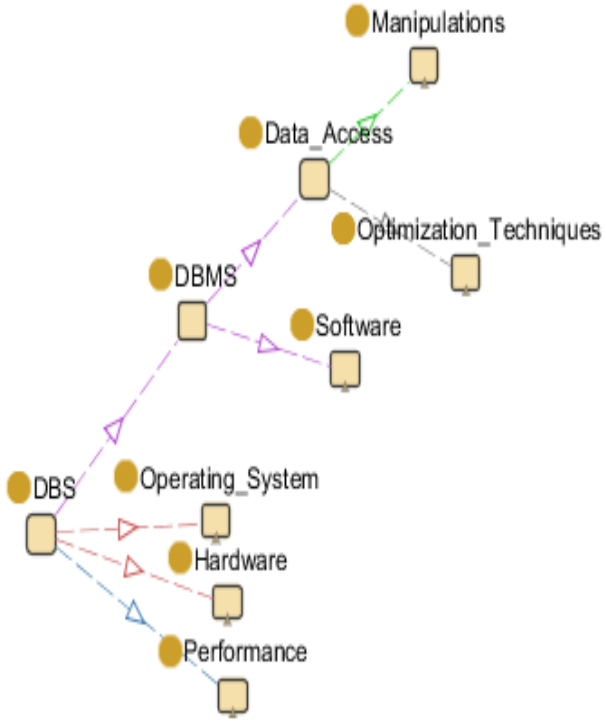

Figure 6: Database Systems ontology: DBMS class

The purpose of making a special model for all DBMSs is to try to develop a common model containing all the features (common or uncommon) distributed in categories according to their role in data management.

The DBMS component is considered as the core of a database system (DBS). We split it in two parts (Figure 6):

- The first is related to the data access techniques describing the various optimization structures, indexes, materialized views, vertical/horizontal fragmentation, etc. and the data manipulation methods, query language, aggregation, etc.
- The second part is linked to the software technology that describes data sharing, cloud, replication and other techniques.

5.2 Data Access class

Given the multiplicity and the difficulty to detail all the features, the data access part (Figure 6) is highlighted for both data manipulation and Data Access Methods. Data manipulation (Figure 7) represents the functionality of query languages supported by the DBMSs for creating and manipulating databases. *Select-From-Where* (or equivalent in other languages) grouping, transaction and aggregation (sum, count, avg, min, max) represent ordinary needs of queries. Stored procedures, triggers, spatial, temporal and stream processing represent specific needs of queries [19].

In the other side, there are wide optimization structures supported by DBMSs that can enhance the query performance. Selecting the appropriate optimization structures for a given workload is NP-hard problem [4]. Several algorithms and approaches have been proposed to deal with this problem. They may be divided into two classes [4] based on their used resources:

- The first class concerns the redundant optimization structures that include vertical fragmentation, materialized views, indexes and management of buffer. These structures require additional storage cost to implement the structures on existing data. In addition, updating the data (insertion, deletion or modification) affects these structures and requires an additional maintenance cost.
- The second class concerns the non-redundant structures (e.g. horizontal fragmentation, parallel processing and scheduling queries, etc.). Unlike redundant structures, they do not require any additional storage costs because they relate to the distribution of data or queries.

5.3 Software class

The software component (Figure 8) represents the storage model (relational, Key-value store, document oriented, column oriented, graph oriented, etc.), the data storage (file system, in memory, etc.), the supported operating systems (windows, Unix, Linux), the architecture (Centralized, distributed, cloud, etc.).

6. A RECOMMENDER SYSTEM FOR SELECTING DBMS

To respond to the question asked previously in the introduction, we believe that recommender systems may assist companies in selecting their favorite DBMS. Recommender systems are software tools and techniques providing suggestions for items to be of use to a user. The suggestions relate to various decision-making processes, such as what DBMS to choose [18]. They have become fundamental applications and have been largely used in several domains, in electronic commerce, information access, media and entertainment industry providing suggestions that effectively prune large information spaces so that users are directed toward those items that best meet their requirements and preferences. A variety of techniques have been proposed for performing recommendation, including collaborative recommender system, content-based recommender system, demographic based recommender system, utility based recommender system, knowledge based recommender system These methods have sometimes been combined in hybrid recommender systems [8]. 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 information that are obtained directly from users.

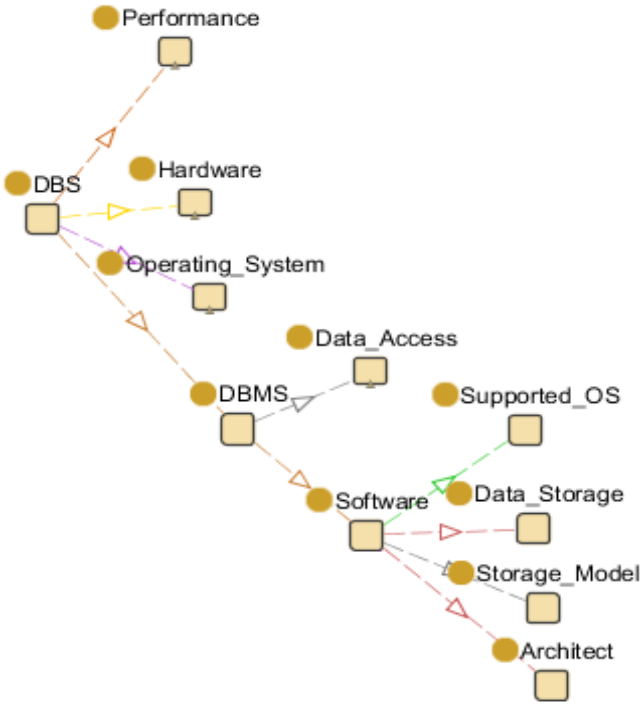

Figure 8: DBMS class: software classes

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 targeted features of DBMSs and the company components, the database schema, the workload, the platform, etc., and wants getting an advise to choose a relevant DBMS that fulfills its requirements. Our recommender system is founded on two levels (see Figure 9):

1. *The first level* represents the search of DBMS features using our adaptive of the weighted sum method on the DBMS selection based on the multi-criteria decision making techniques
2. *The second level* that we presented in [5] and [6], consists to design a repository containing data result tests. Using machine learning techniques, we can predict other test results without doing so. This prediction is based on the functional and non-functional requirements (e.g. query response time) expressed by the users.

This recommender system consists of three parts as follows:

- Users requirements called *Manifest* [6];
- Our adaptation of the weighted sum method on the DBMS selection using our ontology;
- Our approach based on the machine learning techniques using the tests repository [5].

We consider that in *level 1*, all the DBMSs instantiated in our ontology are under the test of our recommendation system. In Level 2, the DBMS under test are the deliverable of *level 1*. We can also and depending on the importance of the user wishing to seek for a DBMS taking into account that performance to set the *level 2* as optional level.

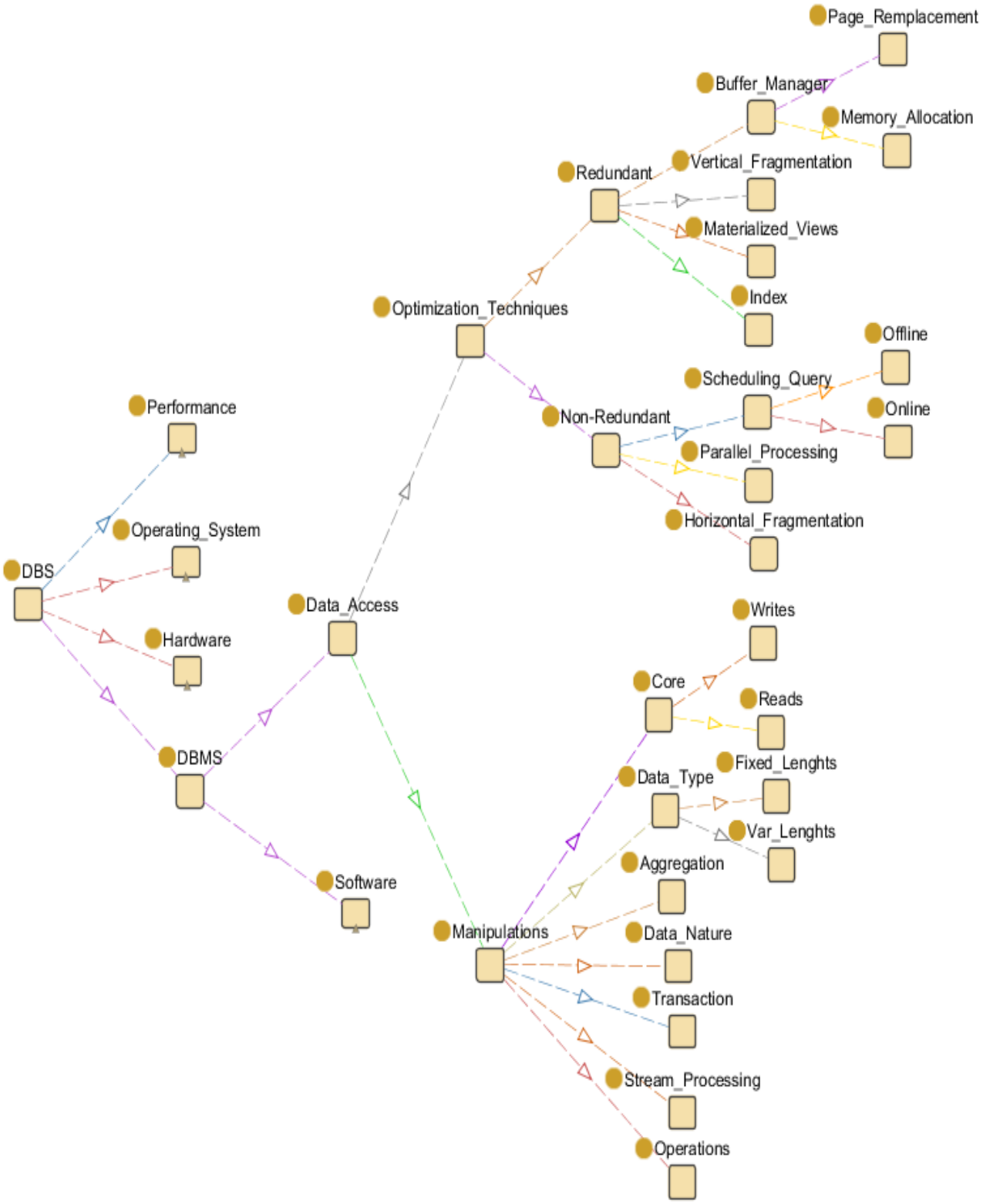

Figure 7: DBMS class: data access and optimization techniques

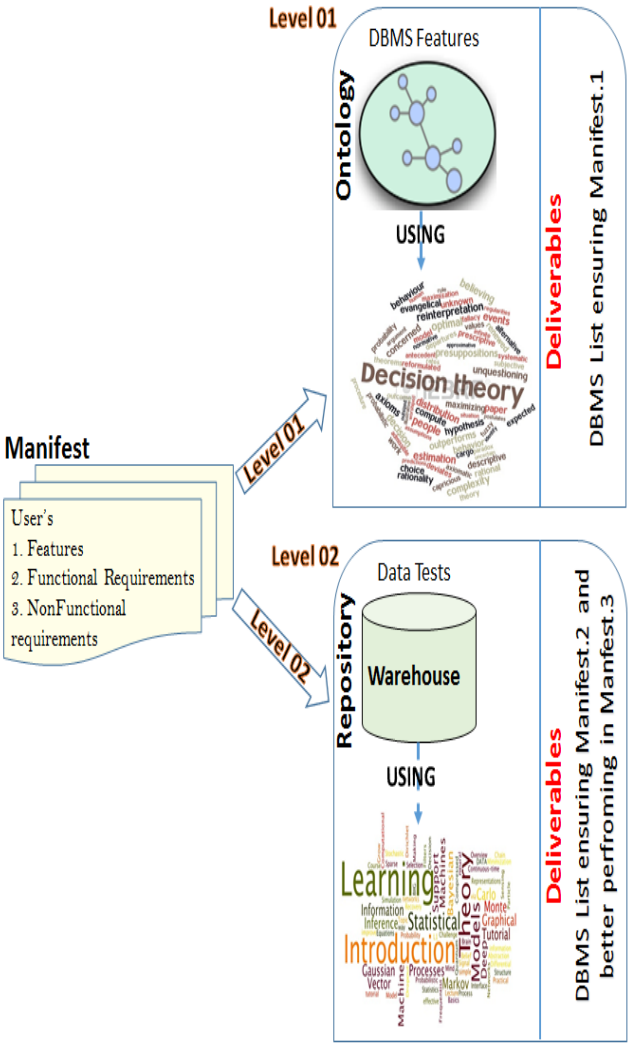

Figure 9: Recommender system levels

6.1 Expressing a Manifest

We assume that a design office of a company comes up with constraints, and wants to get a recommendation to choose the best DBMS that meets its requirements. These constraints are divided into three categories:

- Desired features supported by the DBMSs;
- Functional requirements related to the database schema, data, workload, platform, operating system, deployment architecture, etc.,
- Non-functional requirements representing the performance to be achieved.

These informations components 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.

To clarify the utility of the *Manifest* concept, Figure 10 represents of a *Manifest*, where DBMS and performance metric, that estimate the queries response time, are missing. However, in this paper, we focus to satisfy the first part of the manifest.

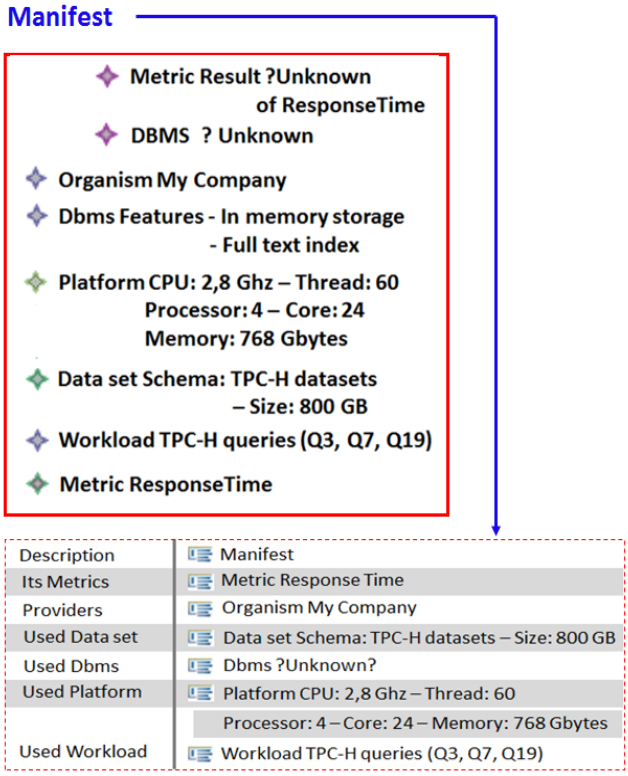

Figure 10: Example of a Manifest

6.2 Adaptation of the weighted sum method on the DBMS selection

The main goal is to find systems that guarantee the company/user requirements. These systems should satisfy the maximum requirements with respect to the importance defined by the companies/users.

Note that our problem is quite similar to the problem of multi-criteria decision making (MCDM). It consists in evaluating a number of alternatives in terms of a number of decision criteria.

Due that we are in the context of adapting multi-criteria decision making techniques on our problem mainly focused on a best DBMS selection method, we decided not to make a comparison between the MCDM techniques because our goal is to focus on DBMS and not on techniques.

To meet this goal, our proposal focuses on the adaptation of MCDM techniques on the DBMS selection using our ontology presented in Section 5. Our ontology represents the knowledge base of DBMS features and the weighted sum method represents the technique for evaluating a number of DBMSs basing on a number of decision criteria.

The idea is to build a decision matrix A from the instance of our DBMS model, while the alternatives A_i represent DBMS, the criteria C_j (the vector criteria C) represent the features and the performance value a_{ij} receive 0 in case of the absence of this feature for the given DBMS and 1 in the contrary case, and w_j denotes the relative weight of importance of the criterion C_j.

For example, if we assume this scenario: company seeks a system ensuring the following criteria: management of geospatial data; storage (in memory); using stored procedures; several text search queries that require optimization by using the full text index and pipelines aggregation. These criteria are classified by importance from the highest to the lowest.

- First, to measure the importance of one criterion relative to the others from the point of view of the company, we attribute to each importance a weight (value), a max weight for the highest and min for the lowest importance, in our case (Table 3) we have assigned the weight 5 to the third criterion (Pipeline aggregation), the weight 4 to the fourth criterion (Stored procedure), the weight 3 to the fifth criterion (Full text index), the weight 2 to the second criterion (Geospatial data) and the weight 1 to the first criterion (In memory storage).

- Second, we apply the formula 1, where A is our matrix of DBMS and features, and C is our criteria vector to evaluate the performance of each alternative (DBMS).

- Then, we sort the alternatives according to the maximum WSM score and we choose the best one which has the maximum score (the maximum number of cases).

7. CASE STUDY

First to build our ontology, we used Protégé-OWL to design our model. The latter is a widely used and well-known open-source ontology edition. In addition, SPARQL query language has been used.
Second, we instantiated 11 DBMSs in our ontology[10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] of different classes (Relational and NoSQL) with all common and uncommon features. It is noted that the instantiation of DBMSs is not an easy task because of the difficulty of finding data sources detailing each DBMS, thus the complexity of functioning the most popular DBMSs such as Oracle, MySQL, etc. and the lack of detail for others.

Table 3: company/user criteria and their importance

Criterion C_j	Importance	Weight w_j
In memory storage	Low	1
Geospatial data	Medium	2
Pipeline aggregation	+High	5
Stored procedure	High	4
Full text index	+Medium	3

[10] http://docs.oracle.com/
[11] https://www.mysql.fr/
[12] http://www.postgresql.org/
[13] http://www.mongodb.org/
[14] http://cassandra.apache.org/
[15] https://voltdb.com/
[16] http://redis.io/
[17] https://www.microsoft.com/fr-fr/server-cloud/products/sql-server/
[18] http://neo4j.com/
[19] http://aws.amazon.com/fr/dynamodb/
[20] http://memcachedb.org/

Table 2 shows an extract of the instances of our ontology including the following thirteens (13) features: INM (In memory), SCX (Secondary index), CPX (Composite index), FTX (Full text index), GSD (Geospatial Data), PTX (Partial index), MVW (Materialized view), CMP (Compression), PIP (Pipelines), SPR (Stored procedure), TRG (Trigger), MPR (Map Reduce) and REP (Replication).
We consider the same previous scenario with five criteria and their weight (Table 3).
To complete the decision matrix (Table 2), we interrogate our DBMS ontology using the SPARQL query language for each criterion. Figures 11 and 12 show, respectively, the SPARQL queries of *In memory* and *Geospatial* criteria. Then, we apply our proposal in section 6.2 to this matrix as follows:

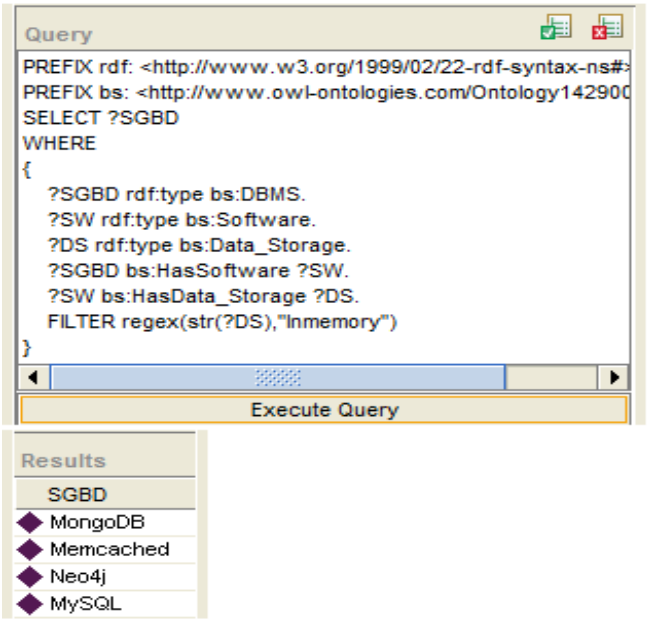

Figure 11: SPARQL Query for In memory criteria

First, we multiply each value of each column in Table 2 by its weight (Multiplying a vector bu a scalar). For example, we multiply the column PIP values by 5 (Pipelines weight), and we put the results in a matrix A^*. Table 4 shows the matrix A^* whose columns have at least a non-zero value.
Second, we compute the values of each row (alternative or DBMS). Finally, we rank the performance results (Table 5).

The result shows that the best alternatives are MySQL and MongoDB (they have the maximum WSM score which is equal to 13). Furthermore, the alternatives PostgreSQL and Oracle possesses the same WSM score which is equal to 12 (Table 5).
We have presented our proposal to introduce mathematical tools in DBMS selection process. Then, we showed a selection scenario of a system according the expressed constraints. However, this methodology depends on the knowledge base instances. The lack of informations and details for DBMSs makes this process incomplete. But we can conclude that our model responds perfectly taking into account the instantiated systems. Finally, we note that it is possible to choose a naive solution using SPARQL query. That consists in relaxing the query of every empty result and then to eliminate the lowest importance criteria. However, if we consider a very high number (n) of criteria, the naive solution may lead to a very large number of executions ($\leq 2^n$).

A=

A \ C	INM	SCX	CPX	FTX	GSD	PTX	MVW	CMP	PIP	SPR	TRG	MPR	REP
MySQL	1	1	1	1	1	0	0	1	1	1	1	0	1
PostgreSQL	0	1	1	1	1	1	1	1	1	1	1	0	1
Oracle	0	1	1	1	1	1	1	1	1	1	1	1	1
MongoDB	1	1	1	1	1	0	1	1	1	1	0	1	1
Cassandra	0	1	1	0	0	0	1	1	0	1	1	1	1
VoltDB	0	1	1	0	0	1	0	0	0	1	0	1	1
Redis	0	1	0	0	1	0	0	1	0	0	1	0	1
MSQL server	0	1	1	1	1	1	1	1	0	1	1	0	1
Neo4j	1	1	0	1	1	0	0	1	0	0	1	0	1
DynamoDB	0	1	1	0	0	0	0	1	0	1	1	1	1
Memcached	1	0	0	0	0	0	0	0	0	0	0	1	0

Table 2: DBMS matrix and their features

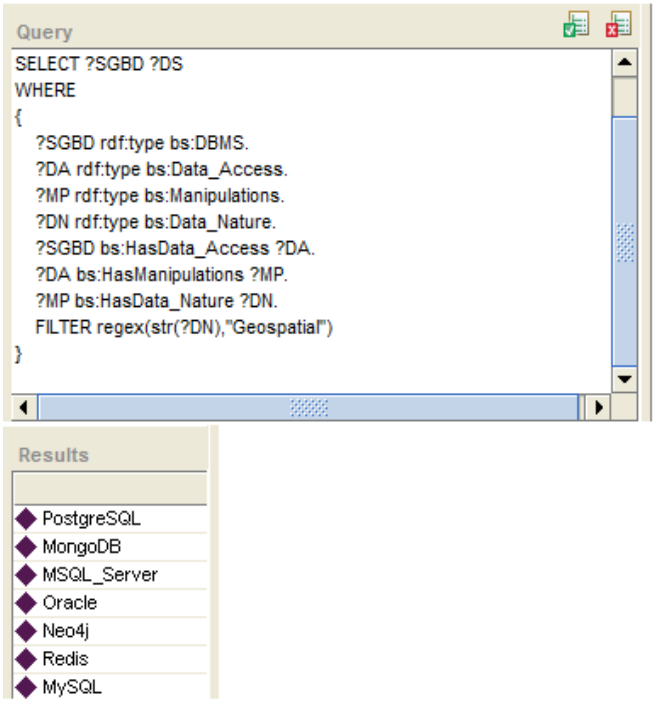

Figure 12: SPARQL Query for Geospatial criterion

A^{*} =

A \ C	INM	FTX	GSD	PIP	SPR
MySQL	1	1	2	5	4
PostgreSQL	0	1	2	5	4
Oracle	0	1	2	5	4
MongoDB	1	1	2	5	4
Cassandra	0	0	0	0	4
VoltDB	0	0	0	0	4
Redis	0	0	2	0	0
MSQL server	0	1	2	0	4
Neo4j	1	1	2	0	0
DynamoDB	0	0	0	0	4
Memcached	1	0	0	0	0

Table 4: Application of our proposal based on WSM

8. CONCLUSION

In this paper, we made a think tank about the problem of recommending DBMS for companies to satisfy its requirements in terms of technical features offered by DBMSs.
First of all, we have identified the limitations of existing ranking DBMS systems. As a consequence, we proposed the usage of a domain ontology to describe in consensual way all concepts and properties of a given DBMS. This ontology represents a conceptual pivot of DBMSs features. It has been validated by the members of our Laboratory. We identified a similarity between our problem and the one of multi-criteria decision making.

Therefore, our ontology has been exploited by the weight sum model to offer companies their relevant DBMSs based on their requirements expressed by a manifest [5].
A case study has thus been elaborated to show the effectiveness of our finding, using the coupling between our ontology and the multi-criteria decision making techniques. At the time, the problem of selecting DBMS is solved by the experts in the field of databases, because they know what different DBMSs can do and what features are more important than others, but now their task is not easy because of the DBMS multiplicity.
So our proposal can not completely replace the experts' work. Moreover, companies do not rely on the final decision of the preferred DBMS to use only on the recommendation system and, therefore, other criteria that must be taken into account such as the price and the training proposed by the founder, etc. Also the opinion of the experts is necessary at the last stage of this selection process.
Currently, we are exploiting the reasoning capabilities of ontologies to substitute a DBMS by other ones offering equivalent features.

	$A^{WCM\text{-}score}$	Rank
MySQL	13	1
PostgreSQL	12	3
Oracle	12	3
MongoDB	13	1
Cassandra	4	6
VoltDB	4	6
Redis	2	10
MSQL server	7	5
Neo4j	4	6
DynamoDB	4	6
Memcached	1	11

	W
INM	1
GSD	2
PTX	0
MVW	0
CMP	0
PIP	5
SPR	4
TRG	0
MPR	0
REP	0
SCX	0
CPX	0
FTX	3

Table 5: Result of our proposal based on WSM

9. REFERENCES

[1] V. Abramova and J. Bernardino. Nosql databases: Mongodb vs cassandra. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pages 14–22. ACM, 2013.

[2] Y. Ait-Ameur, M. Baron, L. Bellatreche, S. Jean, and E. Sardet. Ontologies in engineering: the ontodb/ontoql platform. *Soft Computing*, pages 1–21, 2015.

[3] H. O. Alanazi, A. H. Abdullah, and M. Larbani. Dynamic weighted sum multi-criteria decision making: Mathematical model. *International Journal of Mathematics and Statistics Invention*, 1:16–18, 2013.

[4] L. Bellatreche and A. Kerkad. Query interaction based approach for horizontal data partitioning. *IJDWM*, 11(2):44–61, 2015.

[5] L. Brahimi, L. Bellatreche, and Y. Ouhammou. A recommender system for DBMS selection based on a test data repository. In *ADBIS*, pages 166–180, 2016.

[6] L. Brahimi, Y. Ouhammou, L. Bellatreche, and A. Ouared. More transparency in testing results: Towards an open collective knowledge base. In *Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016, Grenoble, France, June 1-3, 2016*, pages 1–6, 2016.

[7] E. A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, July 16-19, 2000, Portland, Oregon, USA.*, page 7, 2000.

[8] R. D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, 12(4):331–370, 2002.

[9] R. Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2010.

[10] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *ACM symposium on Cloud computing*, pages 143–154, 2010.

[11] S. P. Coy. Security implications of the choice of distributed database management system model: Relational vs. object-oriented. In *NISSC*, volume 96, pages 428–437, 2008.

[12] I. Giurgiu, M. Botezatu, and D. Wiesmann. Comprehensible models for reconfiguring enterprise relational databases to avoid incidents. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1371–1380. ACM, 2015.

[13] V. N. Gudivada, D. Rao, and V. V. Raghavan. Nosql systems for big data management. In *IEEE World Congress on Services*, pages 190–197, 2014.

[14] F. Helff, L. Gruenwald, and L. d'Orazio. Weighted sum model for multi-objective query optimization for mobile-cloud database environments. In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016.*, 2016.

[15] A. Mardani, A. Jusoh, K. MD Nor, Z. Khalifah, N. Zakwan, and A. Valipour. Multiple criteria decision-making techniques and their applications–a review of the literature from 2000 to 2014. *Economic Research-Ekonomska Istraživanja*, 28(1):516–571, 2015.

[16] S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[17] E. Redmond and J. R. Wilson. Seven databases in seven weeks. *The Pragmatic Bookshelf*, 2012.

[18] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. 2011.

[19] M. Rosenmüller, C. Kästner, N. Siegmund, S. Sunkle, S. Apel, T. Leich, and G. Saake. Sql á la carte-toward tailor-made data management. In *BTW*, pages 117–136. Citeseer, 2009.

[20] M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich, O. Spinczyk, and G. Saake. Fame-dbms: tailor-made data management solutions for embedded systems. In *EDBT workshop on Software engineering for tailor-made data management*, pages 1–6. ACM, 2008.

[21] E. Triantaphyllou. Multi-criteria decision making methods. In *Multi-criteria Decision Making Methods: A Comparative Study*, pages 5–21. Springer, 2000.

[22] S. S. ur Rahman, V. Köppen, and G. Saake. Cellular dbms: An attempt towards biologically-inspired data management. *JDIM*, 8(2):117–124, 2010.