A Model&DBMS Independent Benchmark for Data Warehouses

Ibtisam Ferrahi*, Sandro Bimonte**, Kamel Boukhalfa***

* University Mhamed Bougara, Boumerdes, Algeria
ferrahi.ibtisam@umbb.dz,
** TSCF, Irstea. 9 Av. Blaise Pascal, Aubiere, France
Sandro.bimonte@irstea.f
***LSI Laboratory-USTHB, BP 32 ElAlia, Bab Ezzouare, Algiers
kboukhalfa@usthb.dz

Abstract. NoSQL systems support new data models, which propose alternative models to the well-known relational models, and query languages. Due to the lack of a well-accepted logical model for Data Warehouses (DWs), some preliminaries works proposes extensions/redefinition of relational star and snow-flake schemata for NoSQL families. However, many other modeling possibilities remain unexplored, and it is difficult to compare these proposals because of the lack of a well-recognized benchmarking framework for DWs. In this paper, we propose a generic extension of the relational Star Schema Benchmark, called GenSSB, to handle any kind of DBMS in terms of logical models. We validate our framework by instantiating GenSSB for some different logical models and DBMSs.

1 Introduction and motivation

Data Warehouses (DWs) and OLAP systems with their implementation in relational and multidimensional architectures have been widely studied in the last 30 years (Kimball and Ross, 2002). Nowadays, DWs and OLAP systems have reached a great maturity having different kind of applications in several domains such as marketing, health, agriculture, etc. Conceptual, logical and physical issues have been extensively investigated by academic and industrial communities. Several conceptual models based on ER, UML and other formalisms have been proposed, but no standard has been defined yet. Some logical models have been also proposed. Star and snowflake schemas are actually recognized as the de-facto standard logical models for DWs. The star schema denormalizes dimensional attributes to avoid expensive join operators. The snowflake schema is similar to the star schema except that dimensions are normalized into multiple related tables.

Based on these logical models, some optimization techniques (such as index, materialized views, fragmentation), and administration/tuning tools have been proposed. Therefore, specific benchmarks for relational DW (such as SSB, TPC-DS, etc.) (O'Neil et al., 2009) have been designed over these logical models to evaluate and compare performance of these optimization techniques.

However, some new NoSQL Database Management Systems (DBMSs) have been recently proved to be effective Business Intelligence solutions. Different families of NoSQL DBMSs exist: Key-value, Extensible record, Document and Graph. A Key-value database is a collection of data without a schema and organized as a collection of key-value pairs. Data is accessed using the key and its value represents data. An Extensible record database represents data with tables where each row can present different attributes (different columns). A Document database stores information as documents having a complex structure. A Graph database is suited for applications in which there are more interconnections between the data like social networks.

Due to the lack of a well-accepted NoSQL logical model for DWs, some preliminaries works propose extensions/redefinition of relational star and snowflake schemata. However, other modeling possibilities remain unexplored, and it is difficult to compare these proposals because of the lack of a standard logical model for NoSQL DWs, and its associated benchmark. This shows the need for a framework to compare different logical multidimensional models with their possible NoSQL implementations.

Relatively little attention has been paid in the literature to benchmarks for NoSQL DWs (Qin and Zhou, 2013; Shah et al., 2014). The CNSSB benchmark (Dehdouh et al., 2014) is a benchmark proposed to explicitly support two column-oriented logical models. SSB+ (Chevalier et al., 2015) considers both the NoSQL column-oriented and the document-oriented models. CNSSB and SSB+ are based on some new logical models for column and document-oriented DWs.

To best of our knowledge, all existing works for benchmarking DWs are composed of: (i) a data and query generator that are based on a particular logical model, and (ii) a set of methods for writing generated data into particular DBMSs. To conclude, the main limitation of the above described benchmarks is that they are strongly coupled with a particular logical design of the DW, for example the star-schema for the relational DBMSs in SSB, the MLD0 for document DBMSs in SSB+ (Chevalier et al., 2015), etc. Therefore, it is not possible use them to evaluate new logical multidimensional models over NoSQL DBMSs.

To deal with this problem, we propose a generic extension of the relational Star Schema Benchmark, called GenSSB, to handle any relational and NoSQL logical multidimensional model. GenSSB is defined as a simple C library. Finally, we present a comparative study between different models with respect to the loading time of data generated by GenSSB. The paper is organized in the following way: Section 2 presents GenSSB; experiments of GenSSB are described in Section 3, which followed by conclusion and future work.

2 GenSSB Benchmark

In this section, we describe our framework for benchmarking model and DBMS independent DWs, called GenSSB. Our proposal extends the Star Schema Benchmark (SSB) (O'Neil et al., 2009). SSB is based on a relational star schema extending TPC-H for supporting OLAP queries. SSB is defined to analyze sales per PART, SUPPLIER, CUSTOMER and DATE. It defines a set of representative OLAP queries. For example the "Q2" queries of SSB find for the revenue for some product classes, for suppliers in a certain region, grouped by product classes. The SSB data generator uses a parameter called Scale Factor (SF) to generate data at different size. Data is generated proportionally to scale factor. For example, SF=1 corresponds

to $6*10^6$ tuples of the fact table, and SF=50 corresponds to $3*10^8$ tuples. GenSSB generalizes SSB by moving it from the logical level to the conceptual level. GenSSB is composed of: (i) a conceptual multidimensional model and (ii) a data generator that takes as in inputs the same Selectivity Factor parameter of SSB. Contrary to SSB, GenSSB generated data is not associated to any logical model, but to a conceptual multidimensional model. This allows GenSSB to be independent from the used logical model and the DBMS. Indeed, the implementation of a particular logical model in a specific DBMS is in charge of the user of the benchmark. For example, once data have been generated, a user can provide writing functions for generated data in: the star schema with PostgreSQL, and/or the MLD0 and MLD1 with MongoDB. MLD0 and MLD1 are two logical models for document DBMS proposed by (Chevalier et al., 2015).

Using GenSSB, three criteria can be used to evaluate the logical models and their implementation in different DBMSs: (i) Size: the size of data generated, (ii) Load: the time needed to load it into the DBMS, and (iii) Query: the query processing time. Moreover, GenSBB allows a comparative study among logical models belonging to the same family or not.
Therefore, we define two classes of evaluation: (i) **Intra-Family** and (ii) **Inter-Family**.
Intra-Family class contains all models that belong to the same DBMS family, for example for the relational DBMS family it allows compare the star schema over Postgres and Oracle, or for the document DBMS family a study can concern the MLD0 for MongoDB and CouchBase. In details, Intra-Family class contains two groups: *Intra-DBMS* and *Inter-DBMS*, which allow for comparing the same DBMS and different DBMSs respectively. An example of Intra-family evaluation with Intra-DBMS is: perform loading time of data generated into MongoDB using Json file with and without an index. An example of Intra-family evaluation with Inter-DBMS is to perform loading time of data generated (Json file) into MongoDB and CoucheBase.
Inter-Family contains logical models that belong to different families. For the Inter-Family class, only the *Inter-DBMS* group can be defined. For example, it is possible to compare the star schema over Postgres and the MLD0 over MongoDB.

As previously described GenSSB covers an important set of possibilities. Nowadays, with the advent of NoSQL solutions, GenSSB appears mandatory for the choice of the right solution for the right application for the current skills of the enterprise. Indeed, some years ago, relational DBMSs where the **only** possibility offered to BI enterprises that decided to use a particular DBMS solution (for example Postgres) according to the skills of its employees (for example Postgres and MySQL) and the results of the DW benchmarks (e.g. SSB). Nowadays with the proliferation of NoSQL DBMSs (HBase, MongoDB, Neo4j, etc.), BI developers must take into account the skills' employees, but they cannot use existing DW benchmarks to choice the right DBMS for their application. Therefore, GenSSB can be used for easily and fast comparing all possible solutions before to start the BI project development. The overall process for DW implementation using GenSSB is shown in Figure 1.

Therefore, an easy and fast usage of the benchmark is one new important mandatory requirement that we have taken into account for the definition of GenSSB, as shown in Sec 3.3.

Conceptual model: As previously described, GenSSB is not based on a particular logical model, but it generates data based on the conceptual abstraction of the SSB model. The conceptual multidimensional model is presented using the ICSOLAP UML profile (Boulil et al., 2015). We use ICSOLAP since it is based on UML, which is a standard and it can be easily understood.

The Figure 2 presents the conceptual model of our case study based on the SSB bench-

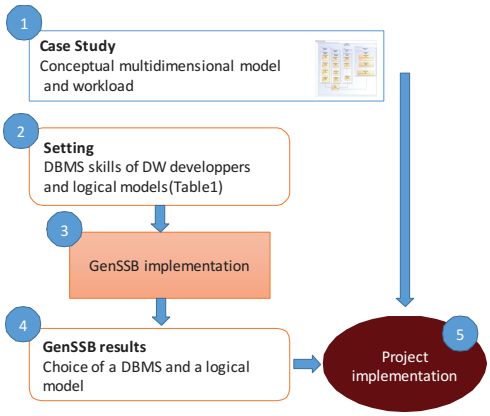

FIG. 1 – *Project development methodology including GenSSB*

mark (O'Neil et al., 2009) using ICSOLAP. It consists of a fact LINEORDER with many measures: QUANTITY, REVENUE, TAX...etc. For reasons of readability we present only quantity. The dimensions are: PART, DATE, CUSTOMER and two spatial dimensions with the spatial levels: (SUPPLIER, CITY, NATION and REGION) and (CUSTOMER, CITY, NATION and REGION). Using the above DW, it is possible to answer OLAP queries that provide the total revenue of each supplier per year, the total revenue of each of supplier per nation and year.

Data generator: The data generator has been implemented in C language extending the SSB implementation. GenSSB's data generator adapts and adds data structures that are not associated to the relational star schema, but it implements the conceptual model previously described (Figure 2). We have added the structure "$nlineorder\_t$" representing facts and modified the structure of the dimensions replacing the keys (primary keys) by some optional identifiers (XXX_ID), in order to improve performance. These data structures are fulfilled by the MK_XXX functions of SSB.

Once data is generated it must be inserted in a logical schema over a particular DBMS. In order to grant generality, we define this function as a C prototype function: PrintDW ($nlineorder\_t$ *t, File *F). A function prototype is a declaration of a function that specifies the function's name and type signature (cardinality, data types of parameters, and return type), but omits the function body. Inputs of PrintDW are: (i) $nlineorder\_t * t$: fact and dimensions data previously generated using the MK_XXX functions, and (ii) $File * F$: file where data is written (such as .TBL, .JSON, .CSV etc.). Then, the PrintDW function must be implemented for each logical model and DBMS chosen for the experiments.

3 Experiments and validation

In this section, we present the implementation of GenSSB using a real case study (Sec 3.1). In particular, we detail the different implementations of the PrintDW function of GenSSB for

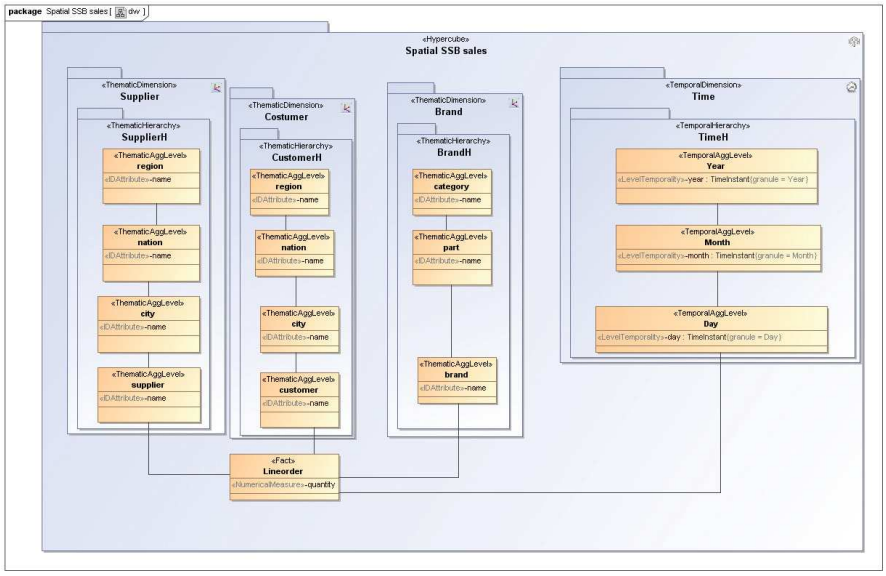

FIG. 2 – *GenSSB conceptual model*

various logical models and DBMSs (Sec 3.3). We also present some results applied to our case study (Sec 3.4).

3.1 Case study overview

Our case study concerns the monitoring of electric use in dairy production (Bimonte et al., 2013). In particular, sensors data coming from dairy equipment is warehoused to be analyzed in a quasi-real-time approach. Data is collected each 10 seconds. A data warehouse representing electric use has been designed in (Bimonte et al., 2013). This DW should allow the analysis of weekly data. Therefore, as defined in (Bimonte et al., 2016) warehoused data is loaded and dropped at the beginning and the end of the week, respectively. According to this approach, data loading must be accomplished in a short time (3-4 hours) in order to allow the good working of the overall system. In (Bimonte et al., 2016) we study a relational DBMS (i.e. Postgres) implementation of the DW. Therefore, in this work we investigate other DW solutions trying to improve loading time, and consequently allow the usage more sensors data. In particular, according to the skills of our engineer team, we are interested to implement GenSSB for supporting the following DBMSs: Postgres, CouchBase, C assandra and MongoDB, and some existing multidimensional logical models. Thus, GenSSB allows us to fast obtain preliminaries results comparing these solutions, without implement data generators from scratch, as described in Figure 1.

3.2 Case study experimental setup

In the context of our case study, the tests were performed using the following configuration setup: Experiments were conducted on a virtual machine with a 6 VCPU, 32 GB of main memory, a 9 TB hard disk under Windows Server 2012. Storage processing were performed using CouchBase (version 4.6.0), MongoDB (version 2.4.1), Postgres (version 9.2), Cassandra (version 3.9).

The logical models used for our case study are MLD0, MLD1, MLD2 for document DBMS (MongoDB and CouchBase), star schema for relational DBMS (Postgres) and CNSSB model pour column DBMS (Cassandra).

Using this configuration, we will evaluate the following experiments scenario : For first we compare the three families (Experiment 1) DBMSs. Then, we compare the different logical models for one family (Experiment 2). Finally, once we have chosen the family and the logical model, we will compare the different DBMSs (Experiment 3).
Experiment 1: *Inter-family*; *Inter-DBMS*; MLD2-MongoDB Vs Star schema-Postgres Vs CNSSB model-Cassandra;
Experiment 2: *Intra-family*; *Intra-DBMS*; MLD2 Vs MLD1 Vs MLD0 MongoDB;
Experiment 3: *Intra-family*;*Inter-DBMS*; MLD2 Vs MongoDB Vs CouchBase.

Relational family: Regarding to star schema, it presents a fact table (LINEORDER) and 4 shared dimensions tables (CUSTOMER, SUPPLIER, PART, DATE). More details can be found in (O'Neil et al., 2009). Postgres is a relational DBMS allowing for transactional storage of data [1].

Document family: For document family logical models, we use ones proposed by (Chevalier et al., 2015): MLD0, MLD1 and MLD2. MLD0 represents facts measures and dimensions with one collection, and one document per fact. An example of one document is shown in Figure 3a. MLD1 is similar to the MLD0, but it presents a collection for facts, with a subdocument per dimension (see Figure 3b for an example). MLD2 is similar to the star schema. It presents a collection for the facts, and a collection per dimension. An example is shown in Figure 3c. MongoDB is a free and open-source cross-platform document-oriented database program [2]. MongoDB uses JSON-like documents with schemas. CouchBase Server, originally known as Membase, is an open-source, distributed (shared-nothing architecture) multi-model NoSQL document-oriented database software package that is optimized for interactive applications [3].

Column family: For Column family logical model, we use renormalized star schema model proposed by (Dehdouh et al., 2014) named CNSSB. This model denormalizes SSB tables into a single table. Consequently, fact table LINEORDER is denormalized, and dimensions tables PART, SUPPLIER, CUSTOMER and DATE are combined into a single LINEORDER table. Using this model, attributes of dimensions are regrouped into four column families: $CF\_CUSTOMER$, $CF\_SUPPLIER$, CF_PART and CF_DATE.
For instance, the column family $CF\_CUSTOMER$ allows to group all attributes of CUSTOMER dimension and so on. We have tested CNSSB under Cassandra, which is column-

1. https://www.postgresql.org/, visited on 29/1/2017
2. https://www.mongodb.com/, visited on 29/1/2017
3. https://www.couchbase.com, visited on 29/1/2017

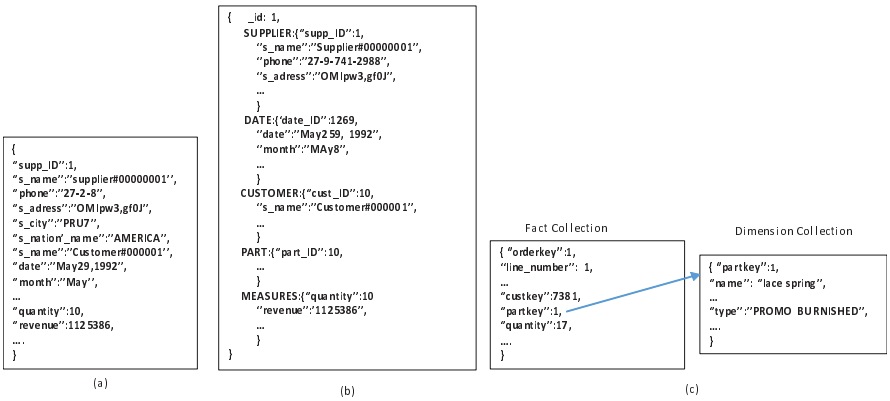

FIG. 3 – *Document family: a) MLD0, b) MLD1, c) MLD2*

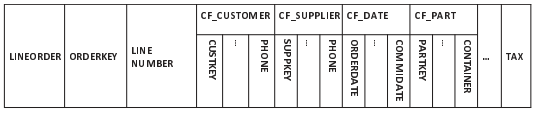

FIG. 4 – *CNSSB data model*

oriented database management system. Cassandra [4] is an apache project originally developed by Facebook, and BigTable by Google. The CNSSB model is presented in figure 4.

3.3 GenSSB use: PrintDW implementations

In this section, we describe the implementations of the PrintDW for some of the previously described logical models. These functions create three files (CSV file, TBL file and JSON file) for the different chosen DBMSs as shown in Figure 5.

The PrintDW function of relational star schema simply modifies the original one provided by SSB.

The PrintDW function for the MLD1 model (Figure 6) generates a set of MongoDB documents in the JSON format (see figure 3a).

3.4 GenSSB use: Evaluation

In this section, we present the results of the above described experiments. We use different scale factors (sf) namely sf=1, sf=10, sf=20 and sf=30. The scale factor sf=1 generates approximately 10^7 lines for the LINEORDER fact. Table 1 shows the sizes of the generated files regarding to scale factor for star schema model (Postgres), MLD0, MLD1, MLD2 for (Couch-

4. http://cassandra.apache.org, visited 29/01/2017

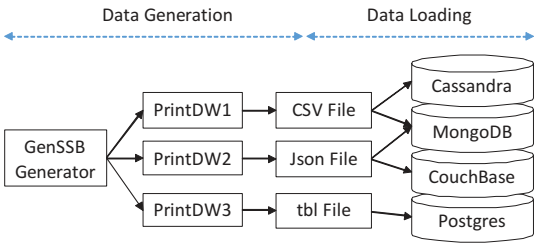

FIG. 5 – *PrintDW implementation*

```
PrintDW(int format, FILE *target, void *data, int len, int sep)
{
...

Pr_nline(nlineorder_t *n, int mode);
...
}

pr_nline(nlineorder_t *n, int mode)
{

insertCustFields(fp_l,&n->cust);
insertPartFields(fp_l,&n->part);
insertSuppFields(fp_l,&n->sup);
insertDateFields(fp_l,&n->date);

}
```

```
int insertSuppFiels(FILE *p_fp, supplier_t *sup)
{
PR_STR(fp, "\"SUPPLIER:{\" : ",12);
PR_STR(fp, "\"SUPPKEY:{\" : ",3);
PR_INT(fp,supp->suppkey);
PR_STR(fp, "\", \n\n"name\" : \" ' ",3);
PR_VSTR(fp,supp->name,S_NAME_LEN);
PR_STR(fp, "\", \n\"adress\" : \" ",3);
PR_VSTR(fp, supp->, (columar)?(long)(ceil(S_ADDR_LEN * V_STR_HGH)) : supp->alen);
PR_STR(fp,"\",\n\"city\" : \" ",3);
PR_STR(fp,supp->city, CITY_FIX);
PR_STR(fp,"\",\n\"nation_name\" : \" ",3);
PR_STR(fp,supp->nation_name, S_NATION_NAME_LEN);
PR_STR(fp,"\",\n\"region_name\" : \" ",3);
....
PR_STR(fp, "}",1); return(0);
}
```

FIG. 6 – *Instance of PrintDW function implementation for MLD1 (JSON file)*

Base and MongoDB), and CNSSB (Cassandra). Data is loaded into MongoDB, CouchBase, Postgres and Cassandra using native instructions.

TAB. 1 – *Data size by model and by scale factor*

SF	SF=1	SF=10	SF=20	SF=50
Star schema	573 Mo	5.68 Go	11 Go	28 Go
MLD0	5.86 Go	58.7 Go	117 Go	294 Go
MLD1	5.96 Go	59,8 Go	118.1 Go	296 Go
MLD2	2,11 Go	21 Go	42 Go	102 Go
CNSSB	2.36 Go	23,8 Go	47,4 Go	120,3 Go

Experiment 1: This experiment aims to compare loading time of (JSON, TBL AND CSV) files generated by GenSSB for three DBMS systems (MongoDB, Postgres and Cassandra). We consider the JSON files generated for MLD2, TBL files generated for star schema model, and CSV file generated for CNSSB. Our results show that MongoDB is faster when it comes to loading compared to Cassandra and Postgres. In the rest of the paper, we present experiments regarding to document-oriented model using MongoDB and CouchBase DBMSs.

Experiment 2: This experiment aims to compare loading time of Json file generated by GenSSB for MLD0, MLD1 and MLD2 into MongoDB. Since, we are mostly interested to load performance in our case study, MLD2 is the best choice. Indeed our results show that

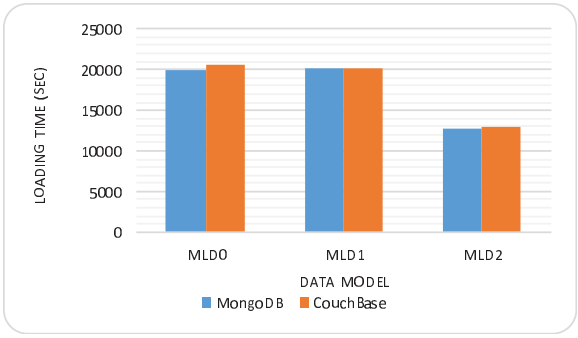

FIG. 7 – *Data loading time of Json file into MongoDB and CouchBase (sf=20)*

its loading time is lower than the other models because of its non-redundant approach (MLD2 needs less data to load as shown in Table 1).

Experiment 3: In this experiment, we compare loading time of Json files generated for MLD0, MLD1 and MLD2 in MongoDB and CouchBase. As previously shown, the MLD2 model is better than the other models, but no significant differences can be noted about is implementation in MongoDB and CouchBase (see Figure 7). Therefore, the choice between the two document DBMSs needs to compare query performance.

3.5 Case study discussion

From the experiments previously described, it appears evident that for our case study the MLD2 model with MongoDB DBMS is a feasible solution. Indeed, the size of sensors data that will be warehoused corresponds more or less to the fact table of GenSSB with SF=20, and the loading time remains acceptable (few minutes). Moreover, as stated in Section 3.1, the implementation of the printDW functions takes only one day, for a 2 days trained engineer. Finally, all these experiments were conducted in one day. To conclude, with few implementation efforts we were able to choice a feasible solution in terms of logical model and DBMS for the future implantation of our energy use DW.

4 Conclusion

In the context of NoSQL DWs, the lack of a widely recognized logical model for NoSQL DWs implies the need for new benchmarks for these new DBMSs. Therefore, in this paper we present GenSSB, a proposal for generic data benchmark for data warehouses. Our proposal is based on conceptual model to generate data that can be supported by many DBMSs with any logical multidimensional model. As future work, taking into account complex multidimensional structures such as complex hierarchies (non-strict, non-onto and non-covering), and complex facts (multi-granular and many to many facts-dimensions is also necessary since these structures usually characterize real DW applications.

References

Bimonte, S., M. Pradel, D. Boffety, A. Tailleur, G. André, R. Bzikha, and J.-P. Chanet (2013). A New Sensor-Based Spatial OLAP Architecture Centered on an Agricultural Farm Energy-Use Diagnosis Tool:. *Int. Jour. of Decision Support System Technology 5*(4), 1–20.

Bimonte, S., M. Schneider, and O. Boussaid (2016). Business Intelligence Indicators:: Types, Models and Implementation. *Int. Jour. of Data Warehousing and Mining 12*(4), 75–98.

Boulil, K., S. Bimonte, and F. Pinet (2015). Conceptual model for spatial data cubes: A UML profile and its automatic implementation. *Computer Standards & Interfaces 38*, 113–132.

Chevalier, M., M. El Malki, A. Kopliku, O. Teste, and R. Tournier (2015). Benchmark for OLAP on NoSQL technologies comparing NoSQL multidimensional data warehousing solutions. pp. 480–485. IEEE.

Dehdouh, K., O. Boussaid, and F. Bentayeb (2014). Columnar NoSQL Star Schema Benchmark. In Y. Ait Ameur, L. Bellatreche, and G. A. Papadopoulos (Eds.), *Model and Data Engineering*, Volume 8748, pp. 281–288. Cham: Springer International Publishing. DOI: 10.1007/978-3-319-11587-0_26.

Kimball, R. and M. Ross (2002). *The data warehouse toolkit: the complete guide to dimensional modeling* (2nd ed ed.). New York: Wiley.

O'Neil, P., E. O'Neil, X. Chen, and S. Revilak (2009). The Star Schema Benchmark and Augmented Fact Table Indexing. In *Performance Evaluation and Benchmarking*, Volume 5895, pp. 237–252. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-10424-4_17.

Qin, X. and X. Zhou (2013). A Survey on Benchmarks for Big Data and Some More Considerations. In *Intelligent Data Engineering and Automated Learning - IDEAL 2013*, Volume 8206, pp. 619–627. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-41278-3_75.

Shah, S. M., R. Wei, D. S. Kolovos, L. M. Rose, R. F. Paige, and K. Barmpis (2014). A Framework to Benchmark NoSQL Data Stores for Large-Scale Model Persistence. In *Model-Driven Engineering Languages and Systems*, Volume 8767, pp. 586–601. Cham: Springer International Publishing. DOI: 10.1007/978-3-319-11653-2_36.

Résumé

Les systèmes NoSQL se basent sur de nouveaux modèles de données différents du relationnel. En raison de l'absence d'un modèle logique pour les entrepôts de données (EDs) accepté par les communautés académique et industrielle, quelques travaux préliminaires proposent des extensions / redéfinition des modèles relationnels en étoile et de flocons de neige pour les SGBDs NoSQL. Cependant, de nombreuses autres possibilités de modélisation restent inexplorées, et il est difficile de comparer ces propositions en raison de l'absence d'un modèle de référence reconnu pour les EDs. Dans cet article, nous proposons une extension générique du Star Schema Benchmark, appelée GenSSB, pour gérer tout type de SGBD en termes de modèles logiques. Nous validons notre proposition en utilisant différents modèles logiques et SGBSs NoSQL.