To Each His Own: Accommodating Data Variety by a Multimodel Star Schema

Sandro Bimonte
University Clermont, TSCF, INRAE
Aubiere, France
sandro.bimonte@inrae.fr

Yassine Hifdi
LIFAT Laboratory, University Tours
Blois, France
hifdiyassine@gmail.com

Mohammed Maliari
ENSA
Tangier, Maroc
mohammedmaliari@gmail.com

Patrick Marcel
LIFAT Laboratory, University Tours
Blois, France
Patrick.Marcel@univ-tours.fr

Stefano Rizzi
DISI, University of Bologna
Bologna, Italy
stefano.rizzi@unibo.it

ABSTRACT

Recent approaches adopt multimodel databases (MMDBs) to natively handle the variety issues arising from the increasing amounts of heterogeneous data (structured, semi-structured, graph-based, etc.) made available. However, when it comes to analyzing these data, traditional data warehouses (DWs) and OLAP systems fall short because they rely on relational Database Management Systems (DBMSs) for storage and querying, thus constraining data variety into the rigidity of a structured schema. This paper provides a preliminary investigation of the performance of an MMDB when used to store multidimensional data for OLAP analysis. A multimodel DW would store each of its elements according to its native model; among the benefits we envision for this solution, that of bridging the architectural gap between data lakes and DWs, that of reducing the cost for ETL data transformations, and that of ensuring better flexibility, extensibility, and evolvability thanks to the use of schemaless models. To support our investigation we present an implementation, based on the UniBench benchmark dataset, that extends a star schema with JSON, XML, spatial, and key-value data; we also define a sample OLAP workload and use it to test the performance of our solution and compare it with that of a classical star schema. As expected, the full-relational implementation performs better, but we believe that this gap could be balanced by the benefits of multimodel in dealing with variety. Finally, we give our perspective view of the research on this topic.

1 INTRODUCTION

Big Data is notoriously characterized by (at least) the 3 V's: volume, velocity, and variety. To handle velocity and volume, some distributed file system-based storage (such as Hadoop) and new Database Management Systems (DBMSs) have been proposed. In particular, four main categories of NoSQL databases have been proposed [2]: key-value, extensible record, graph-based, and document-based.

Although NoSQL DBMSs have successfully proved to support the volume and velocity features, variety is still a challenge [21]. Indeed, several practical applications (e.g. retail, agriculture, etc.) ask for collecting and analyzing data of different types: structured (e.g., relational tables), semi-structured (e.g., XML and JSON), and unstructured (such as text, images, etc.). Using the right DBMS for the right data type is essential to grant good storage and analysis performance. Traditionally, each DBMS has been conceived for handling a specific data type; for example, relational DBMSs for structured data, document-based DBMSs for semi-structured data, etc. Therefore, when an application requires different data types, two solutions are actually possible: (i) integrating all data into a single DBMS, or (ii) using two or more DBMSs together. The former solution presents serious drawbacks: first of all, some types of data cannot be stored and analyzed (e.g., the pure relational model does not support the storage of images, XML, arrays, etc. [29]); besides, even when data can be converted and stored in the target DBMS, querying performances could be unsatisfactory. The latter approach (known as *polyglot persistence* [16]) presents important challenges as well, namely, technically managing more DBMSs, complex query languages, inadequate performance optimization, etc. Therefore, *Multimodel databases* (MMDBs) have recently been proposed to overcome these issues. A MMDB is a DBMS that natively supports different data types under a single query language to grant performance, scalability, and fault tolerance [21]. Remarkably, using a single platform for multimodel data promises to deliver several benefits to users besides that of providing a unified query interface; namely, it will simplify query operations, reduce development and maintenance issues, speed up development, and eliminate migration problems [21]. Examples of MMDBs are PostgreSQL and ArangoDB. PostgreSQL supports the row-oriented, column-oriented, key-value, and document-oriented data models, offering XML, HSTORE, JSON/JSONB data types for storage. ArangoDB supports the graph-based, key-value, and document-oriented data models.

Handling variety while granting at the same time volume and velocity is even more complex in Data Warehouses (DWs) and OLAP systems. Indeed, warehoused data result from the integration of huge volumes of heterogeneous data, and OLAP requires very good performances for data-intensive analytical queries [20]. Traditional DW architectures rely on a single, relational DBMS for storage and querying[1]. To offer better support to volume while maintaining velocity, some recent works propose the usage of NoSQL DBMSs; for example, [8] relies on a document-based DBMS, and [5] on a column-based DBMS. NoSQL proposals for DWs are based on a single data model, and all data are transformed to fit with that model (document, graph, etc.). Overall, although these approaches offer interesting results in terms of volume and velocity, they have been mainly conceived and tested for structured data, without taking into account variety.

[1] More precisely, this is true for so-called *ROLAP* architectures. In *MOLAP* architectures, data are stored in multidimensional arrays. Finally, in *HOLAP* architectures, a MOLAP and a ROLAP systems are coupled.

Furthermore, to facilitate OLAP querying, DWs are normally based on the multidimensional model, which introduces the concepts of facts, dimensions, and measures to analyze data, so source data must be forcibly transformed to fit a multidimensional logical schema following a so-called *schema-on-write* approach. Since this is not always painless because of the schemaless nature of some source data, some recent work (such as [12]) propose to directly rewrite OLAP queries over document stores that are not organized according to the multidimensional model, following a *schema-on-read* approach (i.e., the multidimensional schema is not decided at design time and forced in a DW, but decided by each single user at querying time). However, even this approach relies on a single DBMS.

An interesting direction towards a solution for effectively handling the 3 V's in DW and OLAP systems is represented by MMDBs. A *multimodel data warehouse* (MMDW) can store data according to the multidimensional model and, at the same time, let each of its elements be natively represented through the most appropriate model. Among the benefits we envision for MMDWs, that of bridging the architectural gap between data lakes and DWs, that of reducing the cost for ETL data transformations, and that of ensuring better flexibility, extensibility, and evolvability thanks to the use of schemaless models.

In this paper we conduct a preliminary investigation of the performance of MMDWs to store multidimensional data. To this end we introduce a logical schema for MMDWs and its implementation on PostgreSQL, which gives native multimodel support. Our schema extends the classical star schema introducing semi-structured (JSON, XML, and key-value) data in all the multidimensional elements; thus, it goes in the direction of coupling the pros of schema-on-write approaches (mainly, good performances and simple query formulation with no need for query rewriting) with those of schema-on-read approaches (higher flexibility in ad-hoc querying).

Due to the lack of a benchmark for multimodel data warehouse, in this paper we propose our own OLAP workload to evaluate the performance of our proposal, which we also test against a full-relational implementation on PostgreSQL. To the best of our knowledge, no benchmark dataset for DW (either relational or NoSQL) supports variety; thus, for the experiments we use the schema and data provided by UniBench [30], a benchmark for MMDBs that well represents variety.

The paper outline is as follows. After discussing the related literature in Section 2, in Section 3 we present the UniBench case study. Sections 4 and 5 introduce our logical schema for MMDWs and the related OLAP workload, respectively. Section 6 shows the results of the experiments we made, while Section 7 presents our vision of future MMDW research. Finally, in Section 8 we draw the conclusions.

2 RELATED WORK

Some recent work concerns warehousing and OLAP using NoSQL DBMSs of different kinds. In [11], three different logical models are proposed, using 1 or N document collections to store data in document-based DBMSs and highlighting the utility of nested document and array types [10]. The same authors also investigate how to handle complex hierarchies and summarizability issues with document-based DWs [9]. The introduction of spatial data in document-based DWs has been discussed in [15], which proposes a new spatial multidimensional model to avoid redundancy of spatial data and improve performances. A logical model for column-based DWs has been proposed by [5] and [7] to address volume scalability. In [28], transformation rules for DW implementation in graph-based DBMSs have been proposed for better handling social network data. To the best of our knowledge, only [22] presents a benchmark for comparing NoSQL DW proposals; specifically, this benchmark is applied to MongoDB and Hbase. Some works also study the usage of XML DBMSs for warehousing XML data [24]. Although XML DWs represent a first effort towards native storage of semi-structured data, their querying performances do not scale well with size, and compression techniques must be adopted [4].

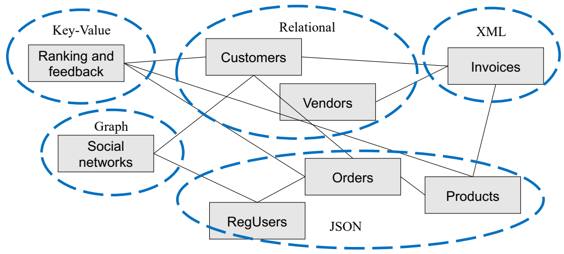

Figure 1: Overview of the UniBench data

Among all these proposals, it is hard to champion one logical and physical implementation for NoSQL and XML DWs, since no approach clearly outperforms the other on the 3 V's. Moreover, these single-model proposals do not address other issues related to warehousing big data, such as reducing the cost of ETL, evolution and improving flexibility.

Recently, some approaches to execute OLAP queries directly against NoSQL data sources were proposed. In [12], a schema-on-read approach to automatically extract facts and hierarchies from document data stores and trigger OLAP queries is proposed. A similar approach is presented in [17]; there, schema variety is explicitly taken into account by choosing not to design a single crisp schema where source fields are either included or absent, but rather to enable an OLAP experience on some sort of "soft" schema where each source field is present to some extent. In the same direction, [13] proposes a MapReduce-based algorithm to compute OLAP cubes on column stores, while [6] aims at delivering the OLAP experience over a graph-based database.

The approaches mentioned above rely on a single-model database. Conversely, [19] proposes a pay-as-you-go approach which enables OLAP queries against a polystore supporting relational, document, and column data models by hiding heterogeneity behind a dataspace layer. Data integration is carried out on-the-fly using a set of mappings. Even this approach can be classified as schema-on-read; the focus is on query rewriting against heterogeneous databases and not on the performances of the approach.

3 CASE STUDY: UNIBENCH

UniBench is a benchmark for multimodel databases proposed in [30]. It includes a retail dataset composed of relational, XML, JSON, key-value, and graph data as shown in Figure 1, which makes it a good representative for variety. However, UniBench was not conceived for OLAP queries. Since our goal is to handle variety with specific reference to DWs, we had to derive a multidimensional schema from UniBench. This adaptation required some modifications, including the addition of descriptive attributes (e.g., LastName), which allows to better test the effectiveness of the proposed approach; as a consequence, some

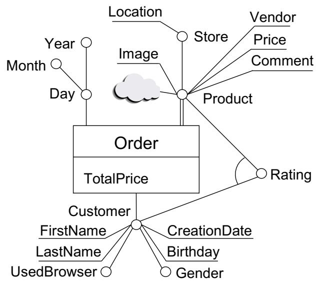

Figure 2: Multidimensional schema for UniBench (the DFM notation [18] is used)

additional data had to be (randomly) generated. The resulting schema represents the Order fact; as shown in Figure 2 it presents three dimensions:

- A Time dimension with levels Day, Month, and Year.
- A Product dimension with one hierarchy including level Store and some descriptive attributes (e.g., Vendor). Interestingly, stores are described by a spatial level, Location. The cloud symbol in the schema denotes that a product can have some additional descriptive attributes not specified at design time.
- A Costumer dimension in which two hierarchies are rooted: one with level Gender, one with UsedBrowser. The customer also has some descriptive attributes, e.g., LastName.

Attribute Rating is cross-dimensional, i.e., its value is jointly determined by Product and Customer (a customer can rate several products). The fact has one measure, TotalPrice.

Finally, since an order is associated to many products, a many-to-many relationship is set between the fact and the product dimension (*non-strict hierarchy*).[2]

4 A MULTIMODEL STAR SCHEMA FOR UNIBENCH

In this section we present a MultiModel, MultiDimensional (in short, M^3D) logical schema for the Order fact introduced above. Essentially, we use a classical star schema with a fact and dimension tables, extended with semi-structured data in JSON and XML form, and with spatial data. Starting from a star schema has several clear advantages: (i) the star schema is supported by all OLAP servers and already in use in a huge number of enterprise DWs; (ii) the best practices for designing a star schema from a conceptual schema are well understood and commonly adopted by practitioners; (iii) fact-dimension relationships are ruled by foreign keys so their consistence is natively checked by the DBMS; (iv) performance optimization of star schema has been long studied and practiced at both the logical (e.g., via view materialization) and the physical (e.g., via indexing) level.

Clearly, several possible alternatives arise for modeling the Order fact with an extended star schema. Defining a set of best practices for designing an M^3D schema that achieves the best trade-off between the five advantages listed in Section 1 is out of the scope of this paper; so, we opted for designing the schema based on a simple guideline: preserve as much as possible the source data variety, i.e., minimizing the transformations to be applied to UniBench source data. Figure 3 shows the M^3D schema that results from applying this guideline to the conceptual schema in Figure 2. It can be described as follows:

- The fact table, Fact_Order, has one tuple for each order and references the order customer and date via foreign keys. Each tuple includes a JSON document that stores the totalPrice measure and an array of orderlines, each specifying a product.
- The customer dimension table, Dim_Customer, specifies each customer's data in the form of XML documents.
- The temporal dimension table, Dim_Date, stores in each tuple a JSON document with the order date; to enable useful aggregations, it also stores the corresponding month and year.
- The product dimension table, Dim_Product, for each product stores its location (as a spatial attribute), vendor, and store, as well as a JSON document with the product name (title), price, and image. Each product also has a Feedback attribute that stores all its ratings in key-value form, with the customer code as a key.
- As shown in Figure 2, each order refers to several products. To model this non-strict hierarchy, rather than opting for the classical relational solution (a many-to-many bridge table [18]), we established a connection between the InfoOrder document stored in the fact table and the Dim_Product dimension table via the asin attribute.

An example of instances of the fact table and of the product dimension table are shown in Figure 4.

The cloud symbol in Figure 2 denotes that the product dimension can include some additional attributes not specified at design time (hence, not included in the JSON schema). For instance, some InfoPrdt documents will have an EU attribute precising the category of product according to the EU classification (see Figure 5), while some InfoOrder documents will have a brand attribute.

5 AN OLAP WORKLOAD FOR UNIBENCH

The workload we introduce to test our M^3D schema is inspired by that of the classical SSB benchmark [23], itself loosely based on the TPC-H benchmark. The SSB workload is meant to functionally cover the different types of star schema queries while varying fact table selectivity. SSB queries are organized in 4 flights, where each flight is a list of 3 to 4 queries. Query flight 1 has restrictions on only 1 dimension, flight 2 has restrictions on 2 dimensions, flight 3 on 3, and flight 4 represents a what-if sequence of the OLAP type. We adopt the same approach, while at the same time classifying queries according to the usage of relational (R)/non relational (NR) measures, relational/non relational group-by levels, and relational/non relational selection levels. We also add a parameter representing the type of join: relational means a join using two relational attributes, while JSON means a join between a JSON attribute and a relational one. Q10 and Q12 use selection attributes that are not part of the JSON schema (brand and EU, respectively). In Table 1, which presents the workload, "by" introduces a group-by and "for" a selection.

6 MULTIMODEL VS. FULL-RELATIONAL

In this section we give a preliminary assessment of the effectiveness and efficiency of MMDWs as compared to those of a classical relational implementation.

[2]We have not considered the graph data of UniBench, since the PostgreSQL DBMS used for implementation does not support them natively.

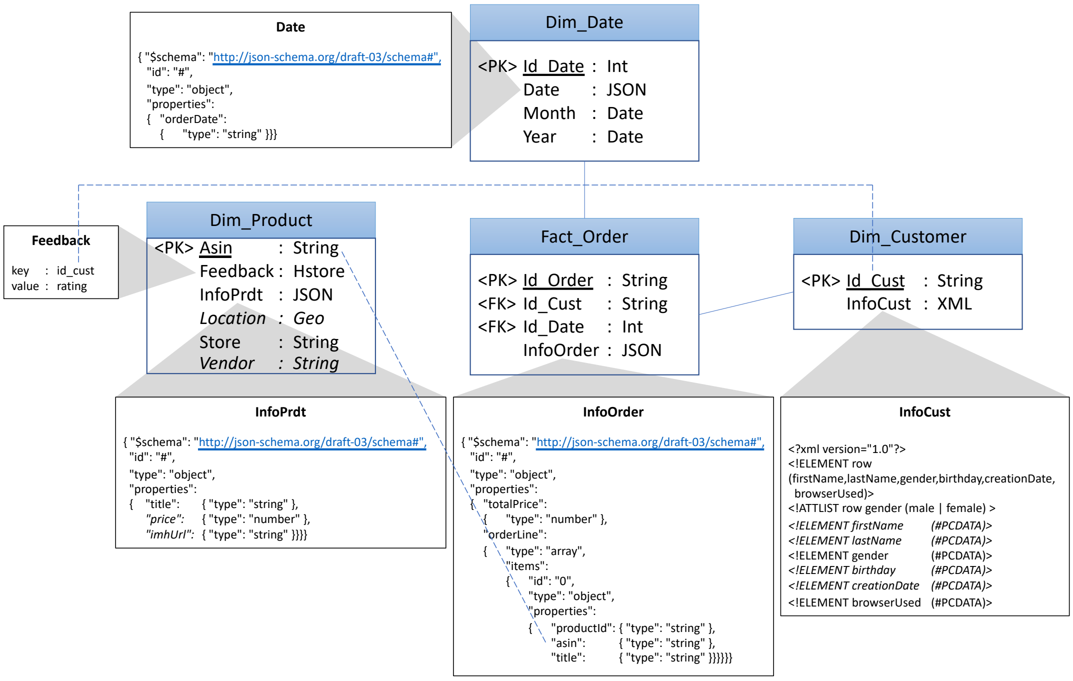

Figure 3: Multimodel star schema in PostgreSQL (solid and dashed lines represent foreign key relationships and implicit relationships between relational and JSON attributes, respectively; in italics, level properties)

orderid [PK] character	id_cust [PK] character varyir	id_date [PK] integer	infoorder jsonb
213507	24189255811864	455	{"Orderline":[{"asin":"B004PJ1J4S","brand":"MYLAPS_Sports_Timing","price":340.65,"title":"Invicta Men s 6566 Subaqua Noma IV Collection Chronograph Black P.
309382	24189255811864	435	{"Orderline":[{"as … R Pathfinder Triple Sensor Tough Solar Digital Multi-Fun.
716103	24189255811864	206	{"Orderline":[{"as … itch Platform 2 Bike Rack","productId":"6515"},{"asin":"B.
807937	24189255812265	211	{"Orderline":[{"as … issTool Spirit Plus Ratchet","productId":"6540"},{"asin":"B
838666	24189255812265	511	{"Orderline":[{"as … at MOLLE Assault Pack Backpack","productId":"6060"},{".
397352	24189255812265	193	{"Orderline":[{"as … rt Kit Deluxe with 1000Xp Scale","productId":"6817"},{"as
97773	24189255812265	382	{"Orderline":[{"as … cal Trainer","productId":"163"},{"asin":"B000M9Q9OK","b
332441	24189255812265	376	{"Orderline":[{"as … escope","productId":"6170"},{"asin":"B000P7ANRI","bran
846313	24189255812265	344	{"Orderline":[{"as … 59P Mid Size Bench","productId":"1991"},{"asin":"B001CJ
873204	24189255812265	514	{"Orderline":[{"asin":"B001CJX50K","brand":"MYLAPS_Sports_Timing","price":389.99,"title":"Kinetic Rock n Roll Trainer w/Road Resistance Unit","productId":"640.
363996	24189255812265	382	{"Orderline":[{"asin":"B004P4HH8U","brand":"TRYMAX","price":509.99,"title":"Barnett Ghost 350 CRT Crossbow Package (Quiver 3 - 20-Inch Arrows and Illuminat

{"Orderline":
[{"asin":"B004PJ1J4S","brand":"MYLAPS_Sports_Timing","price":340.65,"title":"Invicta Men s 6566 Subaqua Noma IV Collection Chronograph Black Polyurethane Watch","productId":"6439"},{"asin":"B0025YCYK8","brand":"Li-Ning","price":207.72,"title":"Spyderco Temperence 2 Canvas Micarta Plain Edge Knife","productId":"2126"},{"asin":"B000OLCP7Q","brand":"Li-Ning","price":179.92,"title":"Profile Design Carbon Stryke Bar (All composite)","productId":"2144"},
{"asin":"B0081MPIBA","brand":"Elan_(company)","price":499.95,"title":"ETEK4 - Planet Eclipse Etek 4 LT / AM Paintball Guns","productId":"6853"},
{"asin":"B004QTWME8","brand":"Peak_Sport_Products","price":546.29,"title":"Trijicon RMR Sight Adjustable (LED 6.5 MOA Red Dot)","productId":"2222"}],"TotalPrice":1774.53}

infoprdt jsonb	store character varying	industry jsonb	country geometry	feedback hstore	asin [PK] character varying
{"price":107.95,"title":"Volcano 20-200 Charcoal Coll...	Schwarzkopf	{"Industry":"Utilities"}	0103000000010...	"2199023262166"=>"5.0", "2199023263018"...	B000FDDM8O
{"price":44.95,"title":"Topeak FlashStand Portable Tu...	Clarins	{"Industry":"Consume...	0103000000010...	"24189255821680"=>"5.0"	B000FIE4VS
{"price":174.72,"title":"Bushnell Banner Mil-Dot Reti...	Shiseido	{"Industry":"Industrial...	0103000000010...	"5"=>"2.0", "236"=>"4.0", "277"=>"5.0", "392...	B000GEWB52
{"price":84.98,"title":"Tasco Target/Varmint 6-24x42...	Clarins	{"Industry":"Telecom"}	0103000000010...	"4398046519433"=>"4.0", "2418925581671...	B000GEY6L4
{"price":150,"title":"Oakley Mens Whisker Sunglasse...	Clearasil	{"Industry":"Industrial...	0103000000010...	"303"=>"5.0", "696"=>"5.0", "892"=>"5.0", "1...	B000GGNR26
{"price":56.58,"title":"Bianchi 111 Cyclone Holster Fit...	LOccitane	{"Industry":"Healthcar...	0103000000010...	"2199023263194"=>"5.0", "4398046522083"...	B000GKMVO2
{"price":109,"title":"Stamina SpaceMate Folding Step...	NARS Cosmetics	{"Industry":"Real Esta...	0103000000010...	"17592186048159"=>"3.0", "219902325627...	B000JC4OXS
{"price":38.61,"title":"Champion Bi-Pod (9 - 13-Inch)"...	Rimmel	{"Industry":"Financial...	0103000000010...	"10995116288536"=>"5.0"	B000JCQBIY
{"price":16.95,"title":"Uncle Mikes Quick Detachable ...	Est?e Lauder	{"Industry":"Healthcar...	0103000000010...	"28587302328959"=>"5.0"	B000JMH5OS
{"price":234.94,"title":"Lyman Reloading Press T-Ma...	NYX Cosmetics	{"Industry":"Financial...	0103000000010...	"178"=>"5.0", "421"=>"5.0", "645"=>"5.0", "8...	B000KKEPJ2

Figure 4: Sample instances of Fact_Order (top) and Dim_Product (bottom)


```
{   "title":    "5 LED Bicyle Rear Tail",
    "price":    8.26,
    "imhUrl":   "http://ecx.images-amazon.com/SY300.jpg",
    "EU":       "Electronics" }}}
```


Figure 5: An InfoPrdt document including an extra-schema attribute, EU

6.1 Efficiency

We have implemented the M^{3}D schema using PostgreSQL with its JSON, XML, key-value, and spatial native storage. Data used to feed dimensions and facts has been extracted from the UniBench benchmark [30]. Specifically, we have 745 dates (|Dim_Date|), 9,949 customers (|Dim_Customer|), 10,116 products (|Dim_Product|), and 640,000 orders (|Fact_Order|).

Table 1: OLAP queries on the Order fact

Query	Measure	Group-by	Selection	Join	Query	Numb. of selections	NR types
Q0	R	R	R	R	Number of orders by months for given months and years	2	–
Q1	R	R	R	JSON	Number of orders by months for given stores and years	2	JSON
Q2	R	R	NR	JSON	Number of orders by months for given years and rating	2	JSON,key-value
Q3	R	NR	R	R	Number of orders by months, gender for given years	1	XML
Q4	R	NR	NR	JSON	Number of orders by months, gender for given products	1	JSON,XML
Q5	NR	R	R	JSON	Total price by year for given stores	1	JSON
Q6	NR	R	NR	R	Total price by year for given genders	1	JSON,XML
Q7	NR	NR	R	JSON	Total price by year, gender for given stores and years	2	JSON
Q8	NR	NR	NR	R	Total price by year of birth for given browsers and genders	2	XML
Q9	NR	NR	NR	JSON	Total price by date, customer for given months, ratings, stores	3	JSON,key-value
Q10	NR	NR	NR opt.	R	Total price by date for given months, genders, brands	3	JSON,XML
Q11	NR	NR	NR	JSON	Total price by date, customer for given months, genders, ratings	3	JSON,XML,key-value
Q12	NR	NR	NR opt.	JSON	Total price by costumer for given EU values	1	JSON,XML

```
Q2:
select count(distinct o.Id_Order) as NumberOfOrders, d.Month
from (  select Id_Order, products->>'asin' as Asin
        from Fact_Order o,
        jsonb_array_elements(o.InfoOrder->'orderLine') as products
     ) as op, Fact_Order o, Dim_Date d,
     (  select skeys(p.Feedback) as Id_Cust, Asin, svals(p.Feedback) as Rating
        from Dim_Product p
     ) as cp
where d.Year='2020' and Rating>4 and o.Id_Date=d.Id_Date
and o.Id_Cust=cp.Id_Cust and o.Id_Order=op.Id_Order and cp.Asin=op.Asin
group by d.Month
```

Figure 6: SQL formulation of query Q2 in PostgreSQL over the M^3D schema

All the OLAP queries proposed in Section 5 have been successfully formulated and executed over the M^3D schema, which confirms the feasibility of using PostgreSQL as a platform for storing and querying MMDWs. Figure 6 shows the SQL formulation of a sample query in PostgreSQL; note that attributes Feedback of type key-value and InfoOrder of type JSON are retrieved as table views to be used for a join or a selection.

In the following, we present some experiments aimed at quantitatively comparing the querying performances of the M^3D schema and those of a full-relational star schema (from now on, FR). For the FR schema we used two bridge tables as shown in Figure 8. The first one, Bridge_Ord_Prod, stores the many-to-many relationship between an order and its products. The second one, Bridge_Cust_Prod, is necessary to store the Rating cross-dimensional attribute. Noticeably, attributes EU and brand are not included here since, as explained in Section 4, they were not known at design time. Clearly, unless some (costly) evolution of the schema is carried out, these attributes cannot be loaded and they cannot be used for querying. The FR schema is also implemented in PostgreSQL; Figure 7 shows the SQL formulation of query Q2 over the FR schema. A comparison between Figures 6 and 7 suggests that the formulation over the M^3D schema is more complex; however, we wish to emphasize that there is no real difficulty in formulating queries on an MMDW in comparison to a traditional star schema, except that some knowledge of the DBMS-specific operators to manipulate key-value, JSON, and XML types is required.

For both implementations, B+trees have been used to index relational attributes. For the M^3D schema, some tests were done in order to find the best optimization plan for the workload queries. The results we report below use the following: (i) a Gist index is used on the Feedback hstore attribute; (ii) B+trees and Gin indexes are used on JSON attributes. All tests have been run

```
Q2:
select count(distinct Id_Order) as NumberOfOrders, d.Month
from Fact_Order o, Dim_Date d, Bridge_Cust_Prod cp, Bridge_Ord_Prod op
where d.Year='2020' and Rating>4 and o.Id_Date=d.Id_Date
and o.Id_Cust=cp.Id_Cust and o.Id_Order=op.Id_Order and cp.Asin=op.Asin
group by d.Month
```

Figure 7: SQL formulation of query Q2 in PostgreSQL over the FR schema

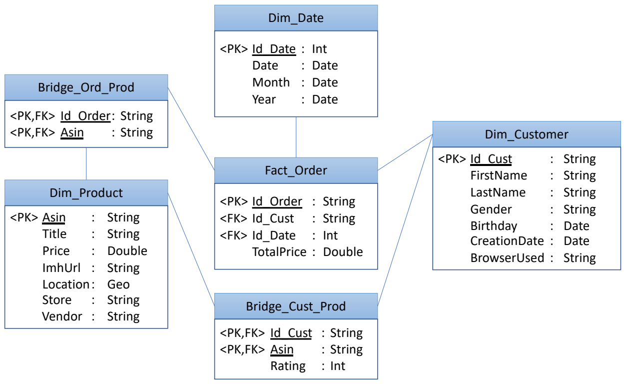

Figure 8: Full-relational star schema in PostgreSQL

Table 2: Performance of benchmark queries (in milliseconds)

Query	M^3D	FR
Q0	253	310
Q1	712	633
Q2	2509	1161
Q3	3996	1023
Q4	1049	175
Q5	1437	714
Q6	1034	197
Q7	1902	711
Q8	817	131
Q9	187	732
Q10	1660	–
Q11	2726	517
Q12	1165	–

on a Core i5 with 4 CPUs @2.3GHz laptop with 16 GB RAM and SSD running MacOS Mojave.

Table 2 shows the query execution in milliseconds against both implementations. Note that Q10 and Q12 cannot be executed on the FR schema because they use attributes (brand and EU, respectively) that were not known at design time so they are not part of that schema. Not surprisingly, the full-relational implementation outperforms the multimodel implementation over most queries. This can partly be explained by recalling that

Table 3: Storage size

Table	M^3D	FR
Fact_Order	603 MB	41 MB
Dim_Product	4160 kB	3896 kB
Dim_Customer	3280 kB	824 kB
Dim_Date	56 kB	40 kB
Bridge_Ord_Prod	–	55 MB
Bridge_Cust_Prod	–	9864 kB

PostgreSQL was originally born as a relational DBMS, so semi-structured and complex data querying is not fully optimized yet. In particular, PostgreSQL lacks specific optimization structures adapted to XML data, thus, the InfoCust attribute cannot be properly indexed; this impacts queries Q3, Q4, Q6, Q8, and Q10. M^3D is also penalized by the necessity to have a JSON attribute in the fact table to be joined with a dimension table (namely, InfoOrder). Additionally, the fact table in M^3D is quite larger than the one in the FR schema, which results in slower star joins (even using the JSONB type instead of JSON, the improvement is very small). The only case where the multimodel implementation significantly outperforms the full-relational one is Q9; this is due to the use of a bridge table in the relational implementation and specifically to the fact that, despite the presence of indexes, the optimizer uses sequential scan to access the bridge table.

Table 3 shows the storage size of both implementations. Unsurprisingly, the relational implementation is more sober than the multimodel one.

Though devising complete guidelines and best practices for multimodel design is out of the scope of this paper, we observe that:

(1) the relational model is still more efficient, so it should be used, during logical design, for the data sources that can be smoothly transformed into relational form (i.e., those whose transformation does not entail loss of information content and can be accommodated within the time frame of ETL);
(2) conversely, the data sources that hardly fit into the fixed structure of a relational schema, e.g., because their schema is not completely known in advance, should be left in their native form.

6.2 Effectiveness

In this section we provide a qualitative comparison of the two solutions in terms of effectiveness from three points of view:

- *Transformation.* The full-relational implementation required all the UniBench data to be translated in relational form according to the star schema in Figure 8. While in the M^3D schema the dimension and fact tables are fed with JSON data with simple INSERT queries, in the FR schema more steps are required. For instance, just to feed the bridge table using the ETL Talend tool we need (i) a job for reading the JSON collection (tFileInputJSOn); (ii) a loop JSON query to read the array of products of each InfoOrder document; (iii) a job for reading the Dim_Product dimension table; and finally (iv) a join operation. This means that transformations may require a significant time and can be error-prone, so they may be unsuitable in specific settings such as those of real-time DWs.
- *Flexibility.* Differently from the FR schema, the M^3D one preserves the data variety existing in the data sources. This is particularly relevant for instance in self-service business intelligence scenarios, where data scientist will write ad-hoc queries to satisfy situational analysis needs [1]. Besides, mixing different models allows, in an MMDW, to achieve higher flexibility in the modeling solutions taken, for instance when dealing with many-to-many relationships.
- *Evolution.* While the multimodel implementation is partially schemaless, so it inherently supports evolution, the situation with the full-relational implementation is quite different. In fact, even adding a couple of simple levels (as EU and brand in our case study) requires, at the very least, changing the relational schema of one or more tables, editing the ETL procedures, and migrating the data from the old schema to the new one. A more complex evolution, e.g., one involving a new many-to-many relationship, would have even more impact because it would require creating new tables. In case users ask for a full versioning of the schemata, the effort would be greater still. An M^3D schema represents a good trade-off here because most evolutions can be handled seamlessly with no impact on tables and ETL; clearly, a more invasive evolution (such as adding a new dimension or measure) would still require a change to the relational part of the schema and to the ETL.

7 A PERSPECTIVE ON MMDW RESEARCH

The experiments we conduct in this work are encouraging enough to set a short- and mid-term perspectives of the research on MMDWs. The advantages we envision for MMDWs can be summarized as follows:

(1) An MMDW will natively and efficiently support OLAP querying over large volumes of multimodel and multidimensional data, thus ensuring support to both volume, velocity, and variety.
(2) Storing data in their native model means reducing the data transformations required; hence, the effort for writing (time-consuming and error-prone) ETL procedures will be reduced in MMDWs, and the freshness of data in the DWs will be increased.
(3) MMDWs will bridge the architectural gap between data lakes and DWs. A data lake ingests heterogeneously-structured raw data from various sources and stores them in their native format, enabling their processing according to changing requirements [25]. Differently from DWs, data lakes support storage of any kind of data with low-cost design, provide increasing analysis capabilities, and offer an improvement in data ingestion; however, analysis tasks are more complex and time-consuming since a schema-on-read approach must be followed. We believe MMDWs will offer an effective architectural trade-off by enabling both OLAP multidimensional analyses and ad-hoc analytics on the same repository.
(4) Schema evolution is a crucial issue in traditional DW architectures, since modifying relational schemata to accommodate new user requirements is a complex and expensive task. MMDWs can store schemaless data, so they will ensure a more effective support to schema evolution [27].
(5) Again thanks to their support of schemaless data, higher flexibility and extensibility will be granted, which will enhance analysis capabilities thus generating added value for users [3].

(6) More specifically, key-value stores on the one hand, and the array constructs supported by document-based databases on the other, provide an alternative solution to model many-to-many relationships appearing in some multidimensional schemata.

In our short-term research agenda on MMDWs we mainly plan to verify and quantify these benefits via an extensive set of experiments based on a more comprehensive case study. This will require, for instance, to measure the effort for writing ETL procedures to transform all data according to a single model; to assess the increase in querying expressiveness achieved by MMDWs in function of the amount of data variety; to simulate dynamic settings so as to evaluate the saving in dealing with schema evolution. In order to overcome performance limitations described in the previous section, we think also that new experiments are mandatory on another multimodel DBMS such as Oracle, which provides other types of implementation for non relational data, and also distributed storage and computation.

In the mid-term, the preliminary work we presented in this paper opens several research issues:

- *Multidimensional design from MMDBs.* The existing data-driven approaches to multidimensional design are based on detecting functional dependencies in single-model data sources, namely, relational, XML, linked-open data, JSON [26]. Using a multimodel data source for design requires integrating different techniques into a synergic methodology.
- *Conceptual models.* Existing conceptual models for DWs are mostly aimed at designing multidimensional schemata with fixed structure. To take full advantage of the flexibility ensured by MMDWs, new models capable of coping with schemaless data (as naively done with the cloud symbol in Figure 2) are needed.
- *Best practices for logical design.* In presence of variety, several alternatives emerge for the logical representation of dimensions and facts [14]. Indeed, some combinations of models may be better than others when coupled with star schemata. A specific set of guidelines for logical design of MMDWs is thus needed to find the best trade-off between performances, fidelity to source schemata, extensibility, and evolvability; this should also include the issues related to view materialization.
- *OLAP benchmark.* Effectively benchmarking MMDBs [21] and non relational DBMSs [22] is still a challenge. Providing a benchmark for MMDWs is a further challenge, since it requires defining a dataset representative of DW volume and multimodel variety, as well as a full range of representative OLAP queries over this dataset.
- *Indexing.* PostgreSQL offers different types of indexes over MMDBs, e.g., B-trees, hash, GiST (for geo data), GIN (for document and hstore data), etc. Ad hoc indexing strategies will have to be devised, in presence of variety, to cope with the specific features of multidimensional data and OLAP queries.
- *OLAP tools.* Last but not least, more sophisticated OLAP tools are required to let users benefit from the additional flexibility introduced by MMDWs while ensuring good performances. Specifically, there is a need for devising techniques to automatically generate efficient SQL queries over MMDWs from the (MDX-like or graphical) language used by the front-end.

8 CONCLUSION

Handling big data variety, volume, and velocity is an important challenge for decision-making information systems. On the one hand, data lakes have been proposed to ensure flexible storage of raw data, but at the price of making analyses more complex. On the other hand, classical DW architectures provide an efficient framework for analyzing transformed and integrated data, but they fall short in natively handling data variety. Motivated by the emerging trend of MMDBs, in this work we have investigated the feasibility of a multimodel approach to DW based on an extension of the well-known star schema with schemaless data as dimensions and facts. Our experiments are encouraging as they show that all queries of our multimodel tailored OLAP workload can run over the proposed multimodel star schema in acceptable time compared to a full-relational implementation. Based on these first results, we have presented many short- and mid-term research perspectives on MMDW.

ACKNOWLEDGEMENT

This work was partially supported by the French National Research Agency as part of the “Investissements d’Avenir” through the IDEX-ISITE initiative CAP 20-25 (ANR-16-IDEX-0001), and the project VGI4bio (ANR-17-CE04-0012).

REFERENCES

[1] Alberto Abelló, Jérôme Darmont, Lorena Etcheverry, Matteo Golfarelli, Jose-Norberto Mazón, Felix Naumann, Torben Bach Pedersen, Stefano Rizzi, Juan Trujillo, Panos Vassiliadis, and Gottfried Vossen. 2013. Fusion Cubes: Towards Self-Service Business Intelligence. *IJDWM* 9, 2 (2013), 66–88.
[2] Paolo Atzeni, Francesca Bugiotti, and Luca Rossi. 2014. Uniform access to NoSQL systems. *Inf. Syst.* 43 (2014), 117–133.
[3] Nabila Berkani, Ladjel Bellatreche, Selma Khouri, and Carlos Ordonez. 2019. Value-driven Approach for Designing Extended Data Warehouses. In *Proc. DOLAP@EDBT/ICDT*. Lisbon, Portugal.
[4] Doulkifli Boukraâ, Mohammed Amin Bouchoukh, and Omar Boussaïd. 2015. Efficient Compression and Storage of XML OLAP Cubes. *IJDWM* 11, 3 (2015), 1–25.
[5] Mohamed Boussahoua, Omar Boussaid, and Fadila Bentayeb. 2017. Logical Schema for Data Warehouse on Column-Oriented NoSQL Databases. In *Proc. DEXA*. Lyon, France, 247–256.
[6] Arnaud Castelltort and Anne Laurent. 2014. NoSQL Graph-based OLAP Analysis. In *Proc. KDIR*. Rome, Italy, 217–224.
[7] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015. Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems. In *Proc. ADBIS*. Poitiers, France, 79–91.
[8] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2015. Implementation of Multidimensional Databases with Document-Oriented NoSQL. In *Proc. DaWaK*. Valencia, Spain, 379–390.
[9] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2016. Document-Oriented Data Warehouses: Complex Hierarchies and Summarizability. In *Proc. UNet*. Casablanca, Morocco, 671–683.
[10] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2016. Document-oriented data warehouses: Models and extended cuboids, extended cuboids in oriented document. In *Proc. RCIS*. Grenoble, France, 1–11.
[11] Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, and Ronan Tournier. 2016. Document-oriented Models for Data Warehouses - NoSQL Document-oriented for Data Warehouses. In *Proc. ICEIS*. Rome, Italy, 142–149.
[12] Mohamed Lamine Chouder, Stefano Rizzi, and Rachid Chalal. 2019. EXODuS: Exploratory OLAP over Document Stores. *Inf. Syst.* 79 (2019), 44–57.
[13] Khaled Dehdouh. 2016. Building OLAP Cubes from Columnar NoSQL Data Warehouses. In *Proc. MEDI*. Almería, Spain.
[14] Ibtisam Ferrahi, Sandro Bimonte, and Kamel Boukhalfa. 2017. A Model & DBMS Independent Benchmark for Data Warehouses. In *Proc. EDA*. Lyon, France, 101–110.
[15] Ibtisam Ferrahi, Sandro Bimonte, Myoung-Ah Kang, and Kamel Boukhalfa. 2017. Design and Implementation of Falling Star - A Non-Redundant Spatio-Multidimensional Logical Model for Document Stores. In *Proc. ICEIS*. Porto, Portugal, 343–350.
[16] Vijay Gadepally, Peinan Chen, Jennie Duggan, Aaron J. Elmore, Brandon Haynes, Jeremy Kepner, Samuel Madden, Tim Mattson, and Michael Stonebraker. 2016. The BigDAWG polystore system and architecture. In *Proc. HPEC*. Waltham, MA, USA, 1–6.
[17] Enrico Gallinucci, Matteo Golfarelli, and Stefano Rizzi. 2019. Approximate OLAP of document-oriented databases: A variety-aware approach. *Inf. Syst.*

85 (2019), 114–130.
[18] Matteo Golfarelli and Stefano Rizzi. 2009. *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, Inc., New York, NY, USA.
[19] Hamdi Ben Hamadou, Enrico Gallinucci, and Matteo Golfarelli. 2019. Answering GPSJ Queries in a Polystore: a Dataspace-Based Approach. In *Proc. ER*. Salvador de Bahia, Brazil.
[20] Ralph Kimball and Margy Ross. 2002. *The data warehouse toolkit: the complete guide to dimensional modeling, 2nd Edition*. Wiley.
[21] Jiaheng Lu and Irena Holubová. 2019. Multi-model Databases: A New Journey to Handle the Variety of Data. *ACM Comput. Surv.* 52, 3 (2019), 55:1–55:38.
[22] Mohammed El Malki, Arlind Kopliku, Essaid Sabir, and Olivier Teste. 2018. Benchmarking Big Data OLAP NoSQL Databases. In *Proc. UNet*. Hammamet, Tunisia, 82–94.
[23] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The Star Schema Benchmark and Augmented Fact Table Indexing. In *Proc. TPCTC*. Lyon, France, 237–252.
[24] Zoubir Ouaret, Rachid Chalal, and Omar Boussaid. 2013. An overview of XML warehouse design approaches and techniques. *IJICoT* 2, 2/3 (2013), 140–170.
[25] Franck Ravat and Yan Zhao. 2019. Data Lakes: Trends and Perspectives. In *Proc. DEXA*. Linz, Austria, 304–313.
[26] Oscar Romero and Alberto Abelló. 2009. A Survey of Multidimensional Modeling Methodologies. *IJDWM* 5, 2 (2009), 1–23.
[27] Stefanie Scherzinger, Meike Klettke, and Uta Störl. 2013. Managing Schema Evolution in NoSQL Data Stores. In *Proc. DBPL*. Riva del Garda, Italy.
[28] Amal Sellami, Ahlem Nabli, and Faïez Gargouri. 2018. Transformation of Data Warehouse Schema to NoSQL Graph Data Base. In *Proc. ISDA*. Vellore, India, 410–420.
[29] Takeyuki Shimura, Masatoshi Yoshikawa, and Shunsuke Uemura. 1999. Storage and Retrieval of XML Documents Using Object-Relational Databases. In *Proc. DEXA*. Florence, Italy, 206–217.
[30] Chao Zhang, Jiaheng Lu, Pengfei Xu, and Yuxing Chen. 2018. UniBench: A Benchmark for Multi-model Database Management Systems. In *Proc. TPCTC*. Rio de Janeiro, Brazil, 7–23.