Design and Implementation of Falling Star
A Non-Redudant Spatio-Multidimensional Logical Model for Document Stores

Ibtisam Ferrahi[1], Sandro Bimonte[2], Myoung-Ah Kang[3] and Kamel Boukhalfa[4]
[1]Department of Computer Science, Faculty of Sciences, University Mhamed Bougara of Boumerdes, Boumerdes, Algeria
[2]TSCF, Irstea. 9 Av. Blaise Pascal, Aubiere, France
[3]LIMOS-UMR CNRS, ISIMA, Blaise Pascal University, Campus des cezeaux, 63178 Aubière, France
[4]USTHB, LSI Laboratory, Houari Boumediene University of Sciences and Technology, Algiers, Algeria

Keywords: Multidimensional Model, NoSQL Databases, Data Warehouses.

Abstract: In the context of Spatial Big Data, some NoSQL spatial DBMSs have been developed to deal with the Spatiality, Velocity, Variety, and Volume of Spatial Big Data. In this context, some works recently study NoSQL logical Data Warehouse (DW) models. However, these proposals do not investigate storing and querying spatial data. Therefore, in this paper we, propose *a new logical model for document Spatial DWs*. Moreover, motivated by the expressivity, readability and interoperability offered by UML profile, we represent our model using a UML profile. Finally, we present an implementation in document Spatial DBMSs.

1 INTRODUCTION

In the era of Spatial Big Data (Shekhar et al., 2012), more and more geo-referenced data are available via new acquisition data systems (remote sensing, social networks, sensors, etc.). In this context, some NoSQL Spatial Database Management Systems (SDBMSs) have been developed to deal with the Velocity, Variety, and Volume of data. Several NoSQL databases have been proposed that can be classified into four categories: Key-value, Extensible record, Graph, and Document. *Key-value* database is a collection of data without a schema and organized as a collection of key-value pairs. Data represented by the "value" is accessed using "key". *Extensible record* databases represent data with tables, where each row can present different attributes (different columns). *Graph database* uses graph structure with nodes, edges and properties associated to theses nodes and edges to store the data. *Graph databases* are suited for applications in which there are more interconnections between the data like social networks. *Document databases* allow storing and querying complex structured information as geo-referenced documents (tweets, images, etc.).

GeoBusiness Intelligence (GeoBI) technologies represent first citizens of systems allowing analysis of Spatial Big Data. GeoBI systems include Spatial Data Mining, Spatial OLAP, spatial statistical tools, and reporting systems. Data Warehouse (DW) and OLAP systems allow analyzing huge volume of data represented according to the multidimensional model, which defines the concept of dimension (the analysis axes) and fact (the analysis subject) (Kimball et Ross., 2002).

On one hand, Spatial Data Warehouses (SDWs) and Spatial OLAP (SOLAP) systems extend OLAP functionalities by integrating spatial data into the multidimensional analysis. On the other hand, warehousing spatial data raises several challenges due to the design, the storage and the visualization of spatio-multidimensional data.

In the context of alphanumeric data, (Chevalier et al., 2015) proposes two approaches, "1 *collection based approach"*, and "*n collections based approach"*, for warehousing and OLAPing huge volumes of complex data using document DBMSs. The "*1 collection based approach"* proposes to represent dimensions and facts in a single document collection to avoid *"join"* operations among documents. However, join operations are not natively supported by schemaless document DBMSs. The *"n collections approach"* is similar to the relational star and snowflake approaches (Kimball et Ross., 2002) where facts and dimensions are stored using different collections. This approach avoids data redundancy, but leads to implement join operators. Therefore, (Chevalier et al., 2015) highlights that for most of

Ferrahi, I., Bimonte, S., Kang, M-A. and Boukhalfa, K.
Design and Implementation of Falling Star - A Non-Redudant Spatio-Multidimensional Logical Model for Document Stores.
DOI: 10.5220/0006353803430350
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 343-350
ISBN: 978-989-758-247-9

OLAP queries the *"1 collection approach"* seems to have a better query performance than the *"n collections approach"*.

Despite of several existing geospatial applications using document SDBMSs (Zhang et al., 2014; Lutz et al., 2014;), to our knowledge no work studies the design of spatio-multidimensional models using document DMBSs.

Therefore, in this paper, based on the "*1 collection"* approach, we *propose a new logical model, called Falling Star, for SDW.*

Falling Star represents an efficient solution for storing and querying multidimensional complex geo-referenced documents. Falling Star proposes a particular logical representation of warehoused data to handle with issues related to the redundancy of the spatial data, which affects storage and computation performances (as already shown for Relational Spatial DBMSs (Siqueira et al., 2008). Moreover, motivated by expressivity, readability and interoperability offered by UML, we *propose an UML profile for the design of: (i) document databases and (ii) SOLAP applications using Falling Star based on the document databases model.*

Indeed, the benefits of the usage of UML, and its associated Computer-Aided Software Engineering tools, for designing, developing and maintaining complex information systems have been widely proved in several computer science and application domains. Finally, *we evaluate our proposal with some experiments*.

The paper is organized in the following way: Section 2 presents related work, and section 3 details the case study used all along the paper. Section 4 presents our spatio-multidimensional logical model and its UML profile representation. Section 5 shows the implementation. Performance study is discussed in Section 6, which followed by conclusion and future work.

2 RELATED WORK

In this section, we present some works that study the implementation (Sec 2.1) and the design (Sec 2.2) of spatial and/or OLAP applications using document DBMSs.

2.1 Implementation

Nowadays, among the most popular document SDBMSs, we can find *CouchBase* and *MongoDB* (Filho et al., 2015). MongoDB is a cross platform document database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas (MongoDB calls the format BSON). MongoDB supports a very rich set of spatial data types. Using GeoJSON format, MongoDB can store spatial data with a variety of geographic data structures. MongoDB natively supports topological operators such as intersection, union, etc. CouchBase stores data in a collection of documents. It provides native types for spatial data (i.e. point, line, etc.). Data can be loaded using the GeoJSON format. CouchBase provides spatial views which support *only* window queries. Therefore, MongoDB have attracted the attention of several researchers (Zhang et al., 2014; Lutz et al., 2014). (Zhang et al., 2014) propose an approach to efficiently store spatial data in the ArcGIS shape format, using MongoDB. In (Lutz et al., 2014), MongoDB is used for the provision of measured and processed massive data collected by the remote sensing. (Xiang et al., 2016) investigate MongoDB to manage planar spatial data that are still widely used in city-scale spatial applications.

Despite the existence of several works for storing and managing spatial data, the integration of spatial data in DWs with NoSQL systems remains unexplored.

Regarding to the implementation of DWs with NoSQL DBMSs, several logical designs of DWs using NoSQL DBMSs have been proposed (Chevalier et al., 2015; Dehdouh et al., 2015). In (Dehdouh et al., 2015), three approaches are proposed to map the multidimensional conceptual data model into a logical modelling adapted to the column DWs.

Regarding to the implementation of DWs with document DBMSs, (Chevalier et al., 2015) proposes two logical models named MLD0 and MLD1, both based on the *"1 collection approach";* and one logical model MLD2 based on *"n collections approach"*. MLD2 investigates data normalization (one document for facts and one document per dimension). The comparative study between these models proved that each model has its weaknesses and strengths (Chevalier et al., 2015). On one hand, MLD2 uses less disk memory, but it is quite inefficient to answer queries with joins (i.e. most of OLAP queries). On the other hand, MLD0 and MLD1 do not show significant performance differences. However, the above described works do not study the impact of storing and querying spatial data. Indeed, when spatial data is integrated in DWs, multidimensional logical models need to be adapted/extended in order to deal with spatial data. Spatial data is complex data and requires particular storage and querying methods as widely investigated in the context of relational DBMSs.

2.2 Design

The conceptual and logical design of DW and SDW using UML, have been widely proved effective (Mazon et al., 2005; Boulil et al., 2015). Indeed, the usage of UML profiles for OLAP and Spatial OLAP projects reduce design time efforts since *UML profiles allow readable, well-formed and familiar class diagrams*.

A UML profile is a set of stereotypes, tagged values, and constraints used to adapt UML elements to a specific application. Stereotypes extend the semantics of existing elements for a specific domain. Tagged values are used to add new properties to existing elements. Finally, constraints are used to specify rules to check the validity of a stereotype. UML profiles can be easily implemented in Computer-Aided Software Engineering (CASE) tools such as MagicDraw, Eclipse, etc.

Moreover, UML profiles can also be associated to automatic implementation of CASE tools to obtain high-quality, defect-free, and maintainable software products.

Therefore, UML profiles have been proposed for conceptual and relational logical spatio-multidimensional models (Boulil et al., 2015; Bimonte et al., 2013 and Cuzzocrea et Fidalgo., 2012). A UML profile for SDW is proposed in (Boulil et al., 2015). Moreover, UML profile have been used also for data warehouses integrating complex spatial data, such as networks (Bimonte et al., 2013) or trajectory data (Oueslati et al., 2014).

Regarding to NoSQL DBMSs, some works have been proposed using UML. For instance, (Gwendal et al., 2016) describes the mapping from UML/OCL conceptual model to logical model for graph DBMS. (Abdelhédi et al., 2016) details an MDA framework for column DBMS. Authors propose, in this paper, transformation rules to generate two NoSQL models: columns-oriented model and documents-oriented model. Based on the Model Driven Architecture, a method that transforms UML class diagrams into HBase based on meta-model is proposed in (Li et al., 2014).

3 CASE STUDY

In this section, we present the case study issued from the relational logical model of the Spatial SSB (Star Schema Benchmark) which we will use all along the paper to present our proposals. Spatial SSB (the only existing) is a benchmark for SDWs. It is presented using the ICSOLAP UML profile (Boulil et al., 2015). The ICSOLAP UML profile allows the conceptual representation of complex spatio-multidimensional applications. In particular, it contains stereotypes for each spatio-multidimensional element.

A *<<Fact>>* is composed of *<<Measure>>* and is associated to dimensions levels *(<<AggLevel>>)* using a *<<DimRelationship>>*. An *<<AggLevel>>* is composed of dimensional attribute and can be thematic, spatial or temporal. A *<<SpatialAggLevel>>* extends the *<<AggLevel>>* with a geometric attribute *(<<LevelGeometry>>).*

As shown in Figure 1, this SDW consists of a fact *LINEORDER* with many measures: *QUANTITY*, *REVENUE*, *TAX*, etc. Due to the lack of space, we present only *QUANTITY*. The dimensions are: *PART*, *TIME*, *CUSTOMER* and one spatial dimension with the spatial levels: *SUPPLIER*, *CITY*, *NATION* and *REGION*. Using the above SDW, it is possible to answer SOLAP queries that provide the total revenue of each supplier per year, the total revenue of each of supplier per nation and year. Due to space limitation, we do not present in Figure 1 the levels of the dimensions *PART* and *CUSTOMER*, but we show them as packages. Using this conceptual model, it is possible to answer the following queries, issued from Spatial SSB, that are representative of SOLAP operators:

Q1: Roll-up
Total of sales per year
Q2: Slice
Total of sales for the product category "MFRG#12"
Q3: Spatial Roll-up
Total of sales per region
Q4a: Spatial slice
Total of sales for the region "AFRICA"
Q4b: Spatial Slice with spatial predicate
Total of sales by suppliers whose regions are inside a rectangular window.

4 LOGICAL SPATIO-MULTI-DIMENSIONAL MODEL

4.1 Document Database Model

In this Section, we present an UML profile for logical document DBMSs, defining the stereotypes for main elements of a document database, and their associated OCL integrity constraints. The proposed

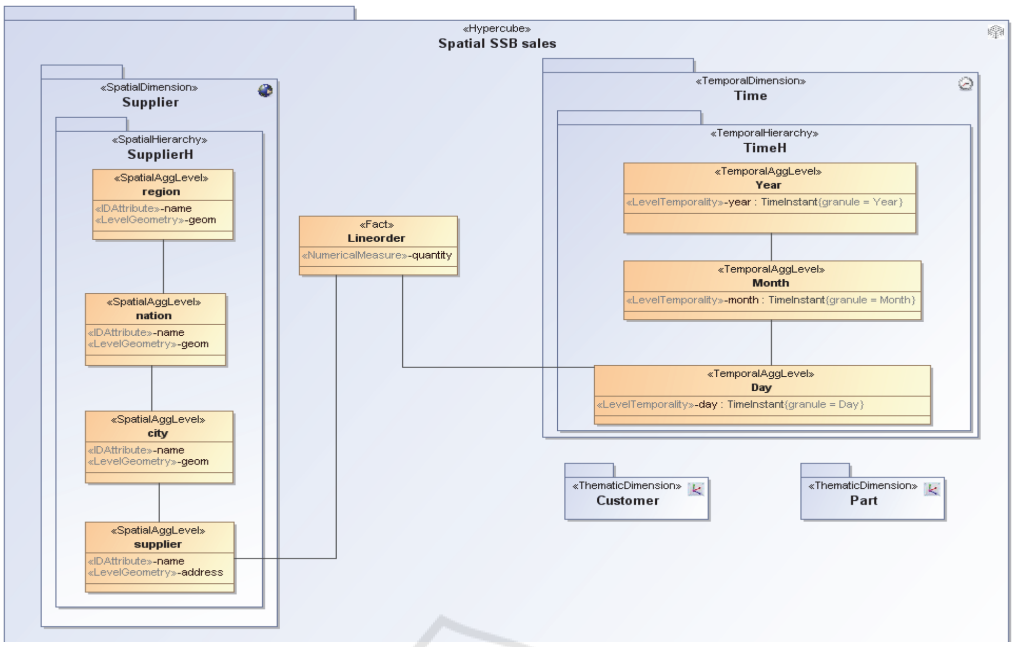


Figure 1: Case study DW: SSB conceptual model.

stereotypes are shown on Figure 2.

A <<*Document*>> is an extension of a class, and it is composed of a set of attributes. These attributes are attributes with particular data types: <<*attributeAlphanum*>> for alphanumeric data type (and no type), <<*attributeGeom*>> for spatial types and <<*attributeIDSubDoc*>> for <<*subDocument*>> data. Indeed, a document can be composed of many subdocuments, which are represented by the class stereotype <<*subDocument*>>. Moreover, a document is composed of an <<*attributeID*>> identifier attribute. When a document is associated to another one it contains an attribute <<*attributeIDlink*>> that is a pointer to the <<*attributeID*>> attribute of the other document. The association between these documents is stereotyped <<*linkingDocuments*>>.

An example of OCL constraints is:

```
(type.oclIsUndefined() = false)  and
(type.oclIsTypeOf(subDocument))
```

It states that the type of an <<*attributeIDSubDoc*>> attribute must be a subdocument.

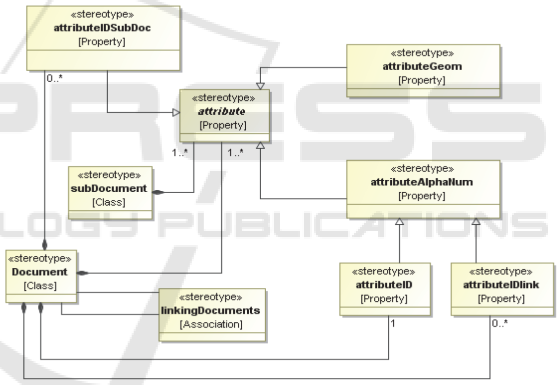


Figure 2: Document database: metamodel.

Finally, let's detail the type of the attributes of the documents. Since geometrical attributes have a particular representation in document DBMSs, we have defined three spatial data types: POINT, LINE and REGION. Other spatial type could be defined. Therefore, using the following OCL statement, we constraints an <<*attibuteGeom*>> attribute to have a spatial type:

```
(type.oclIsUndefined() = false)  and
(type.name='Point'               or
type.name='Region'               or
type.name='Line')
```

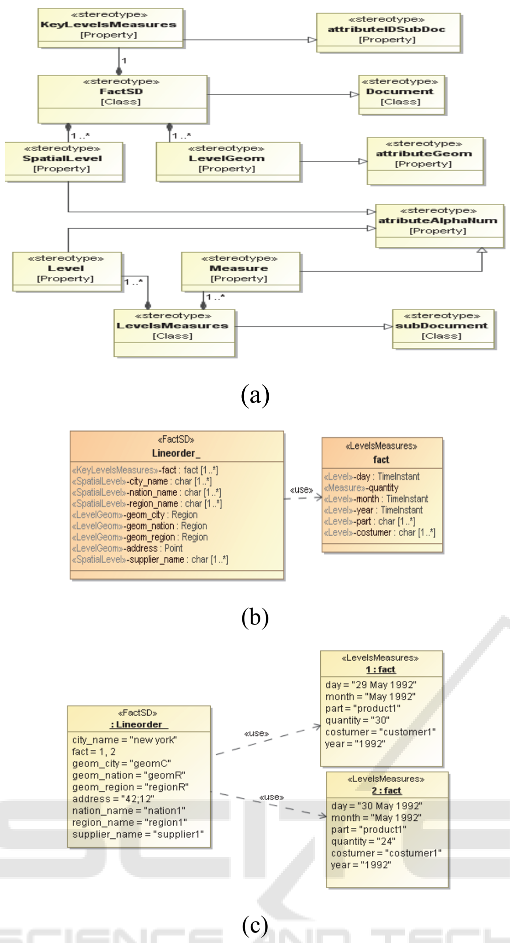

«stereotype» KeyLevelsMeasures [Property]
«stereotype» attributeIDSubDoc [Property]
1
«stereotype» FactSD [Class]
«stereotype» Document [Class]
1..*
1..*
«stereotype» SpatialLevel [Property]
«stereotype» LevelGeom [Property]
«stereotype» attributeGeom [Property]
«stereotype» atributeAlphaNum [Property]
«stereotype» Level [Property]
«stereotype» Measure [Property]
1..*
1..*
«stereotype» LevelsMeasures [Class]
«stereotype» subDocument [Class]
(a)
«FactSD» Lineorder_
«KeyLevelsMeasures»-fact : fact [1..*]
«SpatialLevel»-city_name : char [1..*]
«SpatialLevel»-nation_name : char [1..*]
«SpatialLevel»-region_name : char [1..*]
«LevelGeom»-geom_city : Region
«LevelGeom»-geom_nation : Region
«LevelGeom»-geom_region : Region
«LevelGeom»-address : Point
«SpatialLevel»-supplier_name : char [1..*]
«use»
«LevelsMeasures» fact
«Level»-day : TimeInstant
«Measure»-quantity
«Level»-month : TimeInstant
«Level»-year : TimeInstant
«Level»-part : char [1..*]
«Level»-costumer : char [1..*]
(b)
«FactSD» : Lineorder
city_name = "new york"
fact = 1, 2
geom_city = "geomC"
geom_nation = "geomR"
geom_region = "regionR"
address = "42;12"
nation_name = "nation1"
region_name = "region1"
supplier_name = "supplier1"
«use»
«use»
«LevelsMeasures» 1 : fact
day = "29 May 1992"
month = "May 1992"
part = "product1"
quantity = "30"
costumer = "customer1"
year = "1992"
«LevelsMeasures» 2 : fact
day = "30 May 1992"
month = "May 1992"
part = "product1"
quantity = "24"
costumer = "costumer1"
year = "1992"
(c)

Figure 3: Falling Star: a) metamodel, b) model, c) instances.

Moreover, since document DBMSs do not support data type, we allow the OCL constraint `type.oclIsUndefined()` for *<<attributeAlphanum >>* attributes.

4.2 Falling Star

Motivated by the good computation performance of the *1 collection* approach, and trying to the spatial data redundancy issues (Siqueira et al., 2010), we describe in this section our new document spatio-multidimensional logical model, named *Falling Star model*. The main idea is to represent all facts with the same spatial member by a single document. It means that the list of measures and non-spatial dimension members are associated to each spatial member.

Our proposed model avoids spatial data redundancy. Indeed, spatial objects are complex objects that require important storage size. For example, the representation of a country at a high resolution can use millions of points. Falling Star is based on the previous UML profile for document database described in Section 4.1.

The UML profile for this approach is represented in Figure 3a. We represent a fact with the *<<FactSD>>* stereotype which is composed of spatial levels *(<<SpatialLevel>>)* and their geometries *(<<LevelGeom>>)*. Moreover, it presents an array attribute of *<<LevelsMeasures>>* type *(<<KeyLevelsMeasures>>)*.

<<LevelsMeasures>> is a subdocument composed of a set of measures and non-spatial levels. *<<FactSD>>* is a document (it extends *<<Document>>*), and *<<LevelsMeasures>>* extends *<<subDocument>>*. *<<SpatialLevel>>* and *<<Level>>* are the name of the spatial and non-spatial levels respectively, and thus they extend *<<attributeAlphaNum>>*.
Since *<<LevelGeom>>* represents geometry, it extends *<<attributeGeom>>*. An example of OCL constraint is:

```
   self.ownedMember->select   (m   |
m.oclIsTypeOf(attributeIDlink))-
>size()=0
```

It states that a fact is not associated to other documents, then it does not contain *<<attributeIDlink>>* attributes.

As an example, our case study SDW is shown in Figure 3b. A document LINEORDER represents the facts with attributes representing spatial levels names (CITY_NAME, REGION_NAME, etc.) and geometries (GEOM_CITY, GEOM_REGION, etc.).

LINEORDER has a set of subdocuments (via the *<<KeyLevelsMeasures>>* fact attribute). FACT has *<<Level>>* attributes for all non-spatial dimensions (e.g. DAY, MONTH, YEAR, PART, etc.) and *<<Measure>>* QUANTITY.

An example of instances is shown on Figure 3c, where the *'Supplier1'* is associated to two facts for the two days *'May29, 1992'* and *'May30, 1992'*. A supplier is associated to one geometry. Let us note that geometries are not repeated for each day. Geometries are present only once for each supplier in the instance of the class LINEORDER.

5 IMPLEMENTATION

In this section, we present the implementation of our UML profile using the CASE tool *MagicDraw* and the implementation of our logical model using document DBMSs.

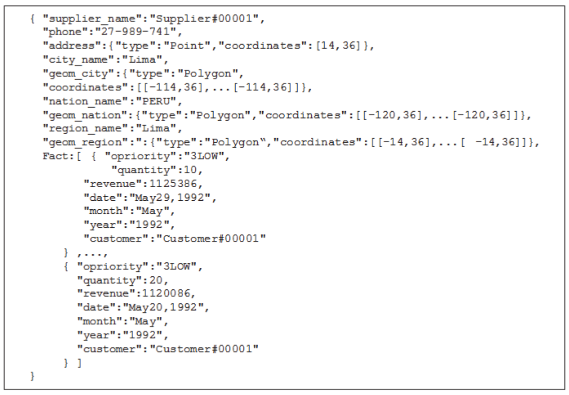
MagicDraw is a CASE tool supporting UML profiling mechanism with OCL constraints. OCL constraints are automatically checked by MagicDraw when an instance of a UML profile is defined, and an error message is displayed when necessary. By this way, MagicDraw allows users to define only models that are conform to the defined UML profile.

Since document databases are not based on a well-accepted standard yet, to show the generality of our implementation, we test it on two different document DBMSs: MongoDB and CouchBase. However, due to space limitation, we present in this paper only the implementation on MongoDB. Instead of taking a business subject and breaking it up into multiple relational structures, MongoDB can store the business subject in the minimal number of documents. MongoDB documents are composed of attribute-value pairs and have the following structure:

```
{attribute1: value1,
         attribute2: value2,
         …
         attributeN: valueN }
```

All documents are stored in collections. A collection is a group of related documents that have a set of shared common indexes. MongoDB supports search by field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Therefore, using MongoDB, we have implemented the SDW of Figure 3b as a collection composed of a set of documents. The number of documents is the number of the spatial dimension members. Each document is composed of two parts. The first one is composed of 4 pairs of key/value where key is an attribute representing spatial level and value is geometric. The second part is an array of subdocument called "***Fact***" that presents <<*LevelMeasure*>>. Each subdocument is composed of 48 attributes: 11 of which are measure attributes from the fact, and the others come from all other non-spatial dimensions levels. An example is shown in Figure 4. We can note that geometric attributes (*geom_nation, geom_region, geom_city*) are stored only once, but they are associated to two facts (one for 29 May, and another for 20 May). In this way, spatial data is not redundant. Using MongoDB documents each document has a maximum size of 16MB. This feature is important to ensure that a single document cannot use excessive amounts of RAM.

To deal with this problem, we have developed a java ETL program that calculates the size of each generated documents and splits it in several different documents that repeat the attributes of the spatial levels.



```
{ "supplier_name":"Supplier#00001",
  "phone":"27-989-741",
  "address":{"type":"Point","coordinates":[14,36]},
  "city_name":"Lima",
  "geom_city":{"type":"Polygon",
  "coordinates":[[-114,36],...[-114,36]]},
  "nation_name":"PERU",
  "geom_nation":{"type":"Polygon","coordinates":[[-120,36],...[-120,36]]},
  "region_name":"Lima",
  "geom_region":":{"type":"Polygon‟,"coordinates":[[-14,36],...[ -14,36]]},
  Fact:[ { "opriority":"3LOW",
           "quantity":10,
      "revenue":1125386,
      "date":"May29,1992",
      "month":"May",
      "year":"1992",
      "customer":"Customer#00001"
    } ,...,
    { "opriority":"3LOW",
     "quantity":20,
     "revenue":1120086,
     "date":"May20,1992",
     "month":"May",
     "year":"1992",
     "customer":"Customer#00001"
    } ]
}
```

Figure 4: Falling Star implementation in MongoDB.

Let us note that, in the same way of relational DBMS, logical models must be adapted to fit with the DBMS's implementation. We have implemented the queries of Section 3 using native MongoDB query language. For example, the query Q4b (Spatial slice with query window) is rewriting as following:

```
db.falling_star.aggregate (
{ $match : {"geom_region" :
 {  $geoWithin:{$polygon:  [[-5000,-5000],[-5000,5000],[5000,5000],[5000,5000]]}}}},
,{$unwind : "$Fact"} ,
{ $group : { _id:null,
  total: {$sum : "$Fact.revenue"} } );
```

6 EXPERIMENTS

In this section, we evaluate our proposal and compare it to existing logical models for document data warehouses. In particular, we evaluate our model with Spatial SSB data (Siqueira et al., 2010). Spatial SSB extends the SSB benchmark to support SOLAP queries with spatial predicates, such as intersection, containment queries. Motivated by the lack of benchmark for spatial big DW (Chevalier et al., 2015) (Dehdouh et al., 2015), we have modified Spatial SSB to write generated data on MongoDB using the Falling Star Schema. In particular, it generates a GeoJson file, which is loaded in MongoDB.

Data is generated using different scale factors (*sf*), namely *sf*=1, *sf*=10, *sf*=20, *sf*=50 and *sf*=100 in our experiments. The scale factor *sf*=1 generates approximately 6*10 ^{6} facts and 10,000 suppliers, for *sf*=10 we have approximately 6*10^{7} facts and 100000 suppliers and so on. We have generated for the geometries of spatial levels polygons composed of 10,000 points.

6.1 Storage Performance

We have compared our approach to the existing *"1 collection"* MLD0 model (since as previously described in Section 2.1 there is no difference between the MLD0 and the MLD1).

In particular, we have extended the logical model MLD0 presented in (Chevalier et al., 2015). MLD0 stores data in one collection, where each document presents an attribute for each level. We have added to MLD0 geometrical attribute. We can note that for each fact (each document) the geometries of spatial levels (polygons composed of 10,000 points) are repeated.

We have compared our proposal and the extension of MLD0 under two aspects: data storage and query execution time. Experiments were conducted on a virtual machine with a 6 VCPU, 32 GB of main memory, a 9 TB hard disk, Windows Server 2012, and MongoDB 3.2. Finally, we do not provide any indexes on our data, since the main goal of this work is to compare logical models.

We present in Table 1, the database size of both logical models without and with spatial data. For *sf=1* the redundant geometries of the MLD0 model are the responsible of the huge size of the database.

Table 1: Size and loading time of data by the scale factor.

	Data size without spatial data (GB)		Data size with spatial data (GB)		Data loading time (sec)	
	MLD0	Falling star	MLD0	Falling star	MLD0	Falling star
sf =1	5.5	5	320	6	1290	326
sf=10	58	51	3000	60	12679	2730
sf=20	112	102	6000	120	20784	5212
sf=50	295	250	15000	280	26413	12480
sf=100	570	500	30000	3000	45213	34320

Indeed, geometries represent 314.5 GB of a 320 GB total size. For bigger *sf,* the SDW size is bigger. Due to the lack of hard disk space, we estimate the size of the SDW generated by MLD0 from *sf=10*, thus 10, 20, 50 and 100. Therefore, we can conclude that the spatial data redundancy is the main factor of the huge size of the SDW in the MLD0 model. This is not true for the Falling Star model, where since geometries are not redundant, they do not affect considerably the total size of the SDW. We observe that MLD0 needs more disk space than Falling Star, because MLD0 repeats spatial attributes values on every document. For instance, at scale factor *sf=10* (6*10^7 facts), Falling Star needs 6GB while MLD0 needs 320 GB space. We summarize also in Table 1 data loading time by scale factor. Data is loaded into MongoDB using native instructions. By consequence, experiments also confirm that Falling Star is also better in terms of data loading time.

6.2 Query Performance

In the following, we analyse query execution time performance for 5 queries of our case study. We perform these queries only under the instance of MLD0 with *sf=1* since MLD0 is not feasible in terms of storage as shown in Sec 6.1.

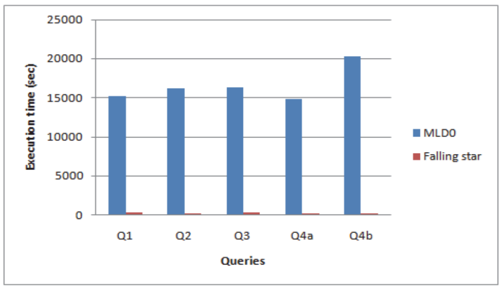


Figure 5: Execution time by model.

Figure 5 shows that Falling Star is better than MLD0 for all queries. This is explained by the small quantity of data, of Falling Star with respect to MLD0. Indeed, the redundancy of the geometries in MLD0 affects query execution time. Therefore, for query performance, Falling Star is acceptable contrary to MLD0.

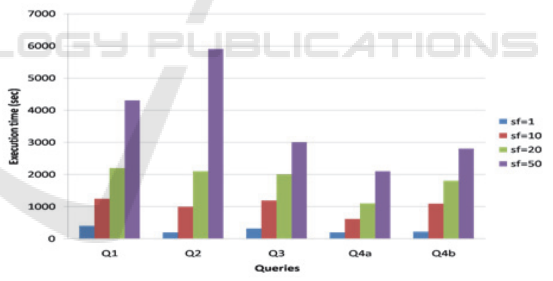


Figure 6: Execution time by SOLAP query.

Let's consider the query time of the SOLAP queries of Sec 3 for Falling Star (Figure 6). Performance is inversely proportional to the size of the SDW. However, query execution time remains acceptable for our PC configuration. Moreover, let's note that execution time of spatial slice queries (Q4a and Q4b) is better than the execution time of other queries. This is due to the fact that, contrary to alphanumeric attributes, MongoDB can directly retrieve the documents with the right spatial members since with Falling Star they are represented as normal attributes. Indeed, non-spatial levels are represented with alphanumeric attributes inside an array structure; therefore, MongoDB must load all the documents before to unwind the arrays to look for the right non-spatial members.

7 CONCLUSION AND FUTURE WORK

In the Spatial Big Data field, many academic and industrial communities propose new Spatial DBMSs, (such as *MongoDB*, *Cassandra*, etc.) to handle with the volume and the variety of these very huge georeferenced datasets. In particular, document Spatial DBMSs appear well-adapted to store complex and voluminous spatial data. Despite of important spatial analysis possibilities offered by document Spatial DBMSs, no work investigates their use in the context of Spatial OLAP and Spatial Data Warehouse.

Therefore, in this work, we focus on the logical modelling and query processing of document spatial data warehouse. We propose a new logical schema for Spatial DW using UML profile. We generate datasets of different size according to different scale factor values. We have tested our models under MongoDB. Our experimental work shows that Falling Star model is better than existing models for document data warehouses, since it explicitly takes into account spatial data.

On-going work involves comparing Falling Star with different logical implementations (relational or NoSQL) and testing our models in a distributed architecture. We will also analyze the impact of query selectivity on the performance of Falling Star model.

REFERENCES

Abdelhédi, F., Ait Brahim A., Atigui, F., Zurfluh G., 2016. Processus de transformation MDA d'un schéma conceptuel de données en un schéma logique NoSQL. INFORSID, pp 15-30.

Bimonte, S., Kang, M., Trujillo, J., 2013. Integration of Spatial Networks in Data Warehouses: A UML Profile. Int Conf. on Computational Science and Its Applications (ICCSA), pp 253-267.

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

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R., 2015. Implementation of Multidimensional Databases with Document-Oriented NoSQL. Int Conf. on Big Data Analytics and Knowledge Discovery (DaWaK), pp 379-390.

Cuzzocrea, A., Fidalgo, Robson, do N. 2012. Enhancing Coverage and Expressive Power of Spatial Data Warehousing Modeling: The SDWM Approach. Int Conf. on Big Data Analytics and Knowledge Discovery (DaWaK), pp 15-29.

Dehdouh, K., Bentayeb, F., Boussaid, O., and Kabachi, N., 2015. Using the column oriented NoSQL model for implementing big data warehouses. Int Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp 469-475.

Gwendal, D., Gerson, S., and Jordi, C., 2016. UML to GraphDB: Mapping Conceptual Schemas to Graph Databases. Int Conf, on Conceptual Modeling ER, Springer International Publishing, pp 430-444.

Kimball, R., Ross, M. 2002. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition.

Li, Y., Gu, P., Zhang, C,. 2014. Transforming UML Class Diagrams into HBase Based on Metamodel. Int Conf. on Information Science, Electronics and Electrical Engineering, pp 720-724.

Lutz, R., Ameri, P., Latzko, T., Meyer, J, 2014. Management of meteorological mass data with MongoDB. Proceedings of the 28th EnviroInfo Conference.

Mazón, J-,N., Trujillo, J., Serrano, M., Piattini, M. 2005. Applying MDA to the development of data warehouses. DOLAP, pp 57-66.

Oueslati, w., Akaichi, J. 2014. Trajectory data warehouse modeling based on a Trajectory UML profile: Medical example. Int Work-Conf. on Bioinformatics and Biomedical Engineering (IWBBIO), pp 1527-1538.

Filho, WB., Olivera, H V., Holanda, M., Favacho, A A. 2015. Geographic Data Modeling for NoSQL Document-Oriented Databases. GEOProcessing 2015 : Int Conf. on Advanced Geographic Information Systems, Applications, and Services, pp 63-68.

Shekhar, S., Gunturi, V., Evans, M. R., and Yang, K., 2012. Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing. Proceedings of the 11th ACM Int Workshop on Data Engineering for Wireless and Mobile Access -MobiDE, p. 1.

Siqueira, TLL., Ciferri, RR., Times, VC., and Ciferri, CDA. 2008. Investigating the Effects of Spatial Data Redundancy in Query Performance over Geographical Data Warehouses. GeoInfo, pp 1-12.

Siqueira, TLL., Ciferri, RR., Times, VC., and Ciferri, CDA, 2010. Benchmarking Spatial Data Warehouses », in Data Warehousing and Knowledge Discovery, vol. 6263, Springer Berlin Heidelberg, 2010, pp 40□51.

Xiang, L., Huang, J., Shao, X., Wang, D, 2015. A MongoDB-Based Management of Planar Spatial Data with a Flattened R-Tree, ISPRS Int J Geo-Information, vol. 5, nº 7, p. 119.

Zhang,X., W. Song, et L. Liu, An implementation approach to store GIS spatial data on NoSQL database, 2014, pp 1□5