



# Evaluating Redundancy and Partitioning of Geospatial Data in Document-Oriented Data Warehouses

Marcio Ferro<sup>1,3</sup>(✉) , Rinaldo Lima<sup>2</sup> , and Robson Fidalgo<sup>1</sup>

<sup>1</sup> Federal University of Pernambuco, Recife, Brazil  
`{mrcf,rdnf}@cin.ufpe.br`

<sup>2</sup> Federal Rural University of Pernambuco, Recife, Brazil  
`rinaldo.jose@ufrpe.br`

<sup>3</sup> Federal Institute of Alagoas, Viçosa, Brazil

**Abstract.** A Geospatial Data Warehouse (GDW) is a repository of historical and geospatial data used in the decision-making process. These systems manage large volumes of data, and their dimensions are usually denormalized to increase query performance. Many studies have analyzed the impact of geospatial data redundancy on a relational GDW. However, to the best of our knowledge, no previous study performed a similar analysis considering the NoSQL scenario. In this context, to design a scalable document-oriented GDW (DGDW) with low storage cost and low query response time, it is important to identify which geospatial fields should be normalized (referenced) or denormalized (embedded), as well as how the documents should be partitioned among collections. In this study, we exhaustively evaluated 36 DGDWs in the MongoDB document-oriented database with different levels of geospatial redundancy and different approaches to partitioning documents among collections. Our experimental results indicate that both the normalization of low-selectivity geospatial fields and the partitioning of documents into homogenous collections provide better query performance and lower storage space. The performance evaluation presented in this paper provides strong evidence that can help guide the creation of a DGDW.

**Keywords:** Geospatial Data Warehouse · NoSQL · Experimental evaluation

## 1 Introduction

A Data Warehouse (DW) is a repository of historical data used in the decision-making process [8]. Recent studies have proposed the use of NoSQL databases to implement DW [1–4, 6, 10, 11, 16, 20–22]. The main reason for this is to achieve

---

This work was supported by Fundação de Amparo à Pesquisa do Estado de Alagoas (FAPEAL).

© Springer Nature Switzerland AG 2019  
C. Ordonez et al. (Eds.): DaWaK 2019, LNCS 11708, pp. 221–235, 2019.  
[https://doi.org/10.1007/978-3-030-27520-4\\_16](https://doi.org/10.1007/978-3-030-27520-4_16)

high scalability with low hardware costs, where the data is distributed and processed by machines in a cluster. However, given the growing volume of data in the Big Data scenario, it is important to identify how DW schemas should be modeled to achieve good performance and low storage cost.

A DW can store geospatial objects, which consist of a descriptive component (e.g., name or address) and a geospatial component (e.g., vector geometry or raster surface) [7]. In this paper, the geospatial component is restricted to geometric data. Therefore, this study uses “geospatial data” and “geospatial documents” as synonyms for geometric data. Given the unconventional nature of these data, the design of a Geospatial DW (GDW) provides several challenges for storing and processing large volumes of data [9]. Studies in GDW using a relational database [12, 17] indicates how fact tables, conventional dimensions, and geospatial dimensions can be modeled to ensure good performance. However, conducting this type of study of a GDW with a NoSQL database to support large volumes of data is still an unexplored topic. In this context, we present the following research problem: *how should document-oriented GDW (DGDW) schemas be modeled to achieve both low storage cost and high query performance?*

In this study, we analyze the query performance and the storage cost of DGDW schemas that have: (i) several levels of geospatial data redundancy; and (ii) partitioning of documents among different collections. For that, we use MongoDB, because it is the most popular NoSQL database used in business applications with geospatial data support [5]. We also define a UML-based notation to represent the DGDW schemas. Using this notation, we exhaustively generated 36 DGDW schemas that have different models for (i) fact tables and their dimensions, (ii) conventional dimensions and geospatial dimensions, and (iii) geospatial dimensions and themselves. Finally, we evaluate these 36 schemas based on query execution time and data volume. In this context, our key contributions are: (1) a performance evaluation that identifies the level of redundancy and partitioning of documents into collections that provides good performance and low-cost storage; and (2) a UML-based notation that can model DGDW schemas in the MongoDB database.

The organization of this paper is as follows: Sect. 2 presents related work, Sect. 3 introduces the UML-based notation and the normalization/partitioning approaches used to model the 36 GDW schemas of our experiment, Sect. 4 details our experiment, Sect. 5 discusses the results, and finally, Sect. 6 presents the paper's conclusions.

## 2 Related Work

A study on the impact of geospatial data redundancy on relational GDW (RGDW) is presented in [18], which compares two schemas containing normalized and denormalized geospatial data. The data redundancy cost (in query performance and storage consumption) in the schema with denormalized geospatial data was found to be higher than the cost of having a greater number of joins to perform in the schema with normalized geospatial data. A similar study [12]

proposed that geospatial dimensions with only one relationship should be denormalized in the conventional dimension. This study also investigated whether the complexity of the geospatial data (i.e., point vs. polygon) influences the performance of the queries, and identified that the denormalization of geospatial data of low complexity (i.e., points) improves query performance. Although both studies [12, 18] examined the impact of data redundancy on RGDW, no analysis was performed to determine whether selectivity of a geospatial field must influence its normalization/denormalization. It is worth noting that the geometry of a geospatial object is not a good factor to define whether it has low or high selectivity, and therefore, whether the data should be normalized or not. For example, considering a GDW containing costs or revenues for a particular country's embassies, its geospatial fields for country, state, city, and address have high selectivity, regardless of the type of geometries, because there is only one embassy per country. That is, each piece of geospatial data is related to only one embassy.






Reference [13] discusses the use of a structure in cloud computing to provide horizontal scalability in an RGDW. In that study, the use of multiple machines connected in a cluster increases both the storage and processing capacity, factors that contribute to improved RGDW performance. However, the authors did not address the use of a NoSQL database.

In short, to the best of our knowledge, no previous study has analyzed the use of NoSQL databases in GDW. However, some studies address the use of NoSQL for conventional DW. Among those studies, we highlight: [21] which presents a methodology based on a key-value database to model a DW to support Big Data applications; references [1, 4, 6, 11, 16, 20, 22] investigate the use of column-oriented databases; references [2–4, 22] investigate the use of document-oriented databases; and [10] proposes a graph-oriented approach. Among these studies, a performance analysis between relational DW and NoSQL DW [1] shows that NoSQL databases have better performance in a cluster. Reference [1] also shows that the use of NoSQL databases is more acceptable when it is intended to achieve horizontal scalability.

### 3 DGDW Design

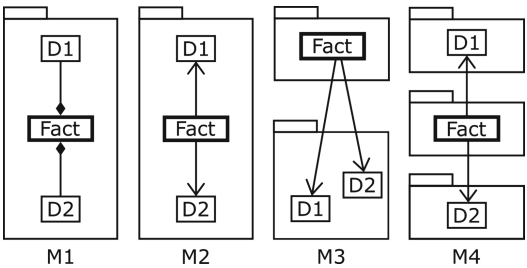
Given the popularity and expressiveness of the Unified Modeling Language (UML) class diagram, we adopted its package, class, composition, and association constructors to represent, respectively, the following constructors: collection (i.e. a collection of documents), document type (i.e., documents having the same field structure), embedded documents (i.e., denormalized documents), and referenced documents (i.e., normalized documents). We highlight that a document type can correspond to a fact table (i.e., Fact Document Type, graphically represented with a thick line) or a dimension table (Dimension Document Type, graphically represented with a thin line). Figure 1 illustrates our notation for specifying the different relationship forms (i.e., normalized/referenced vs. denormalized/embedded) and different partitioning forms (i.e., homogeneous

collection vs. heterogeneous collection) in DGDW. That is, our notation allows us to represent relationships between: (i) fact table and dimensions (Fig. 2); (ii) conventional and geospatial dimensions (Fig. 3); and (iii) geospatial dimensions and themselves (Fig. 4).

	Collection
	Fact Document Type
	Dimension Document Type
	Embedded Documents
	Referenced Documents

**Fig. 1.** Notation used to model DGDW schemas.

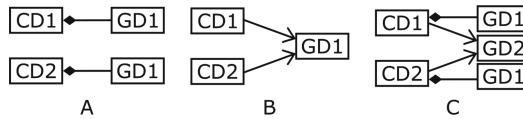
Figure 2 introduces four models for facts and dimensions. In model M1, there is a single collection of documents. Each fact is stored in a single document, which has a set of embedded documents, one for each dimension (D1 and D2 represent dimensions). Model M2 also has a single collection of documents. However, in this collection, facts and dimensions are normalized into distinct documents and referenced by an identifier. Model M3 also normalizes facts and dimensions into separate and referenced documents, but this model partitions these documents into two collections: one to store the facts, and another to store the dimensions. Model M4 differs from the M3 only in the number of collections, because there is one collection for each dimension.



**Fig. 2.** Relationships between facts and dimensions.

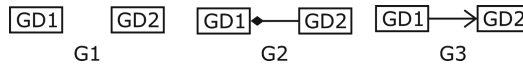
Figure 3 shows distinct forms of modeling relationships between the conventional and geospatial dimensions. We define each conventional or geospatial

dimension as, respectively, a conventional document type (CD) or a geospatial document type (GD). Thus, model A represents a GD embedded/denormalized into the CD, i.e., if a geospatial document needs to be used by more than one CD, it must be replicated for each CD. Model B represents a reference from a CD to a GD. In this model, if a geospatial document needs to be used by more than one CD, there will be no redundancy. Model C is a hybrid representation of the two previous models, in which a geospatial field with low-selectivity (e.g., “*nation*”:*aNation*) is mapped to a GD and is normalized/referenced by a CD, whereas a geospatial field with high-selectivity (e.g., “*address*”:*anAddress*) is mapped to a GD and is denormalized/embedded into a CD.



**Fig. 3.** Relationships between CD and GD.

Figure 4 presents three forms of representing the relationships between GD and themselves. In model G1, there is no relationship between GDs, because these are related only to the CD. In model G2, the GD is embedded/denormalized into other GD, so that the GD with higher selectivity embeds/contains the GD with lower selectivity (e.g., “*city*”: *aCity*  $\supset$  “*nation*”: *aNation*  $\supset$  “*region*”: *aRegion*). Model G3 normalizes the GD into separate and referenced documents.



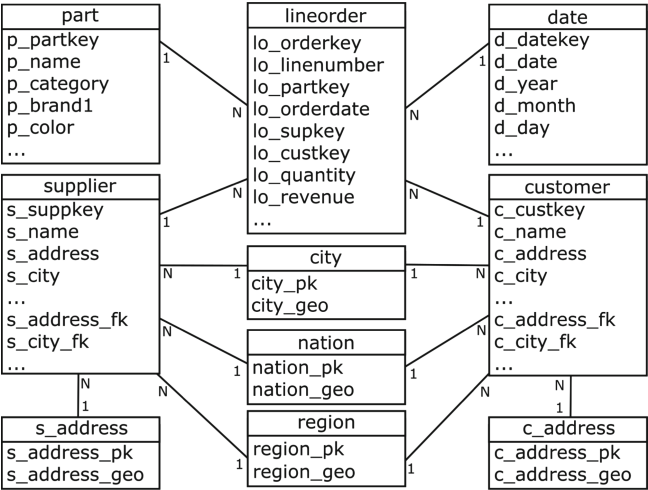
**Fig. 4.** Relationships between GD.

## 4 Experimental Evaluation

To the best of our knowledge, there is no benchmark for DGDW. Therefore, based on the models defined in Sect. 3, we used the RGDW schema presented in Fig. 5 to generate the DGDW schemas that were evaluated in our experiment. The RGDW is based on the SSB Benchmark [14] and adapted to support geospatial data [19]. We generated the RGDW with a scale factor of 1, because this factor was sufficient to identify which DGDW schemas are prohibitive or have both higher performance and lower storage cost. The RGDW was stored in PostgreSQL 10.3 with PostGIS 2.4.3. As can be seen in Fig. 5, the RGDW has a fact table called *lineorder*; four conventional dimensions called *part*, *supplier*, *date*, and *customer*, as well as five geospatial dimensions called *c.address*,

**Table 1.** RGDW used in numbers.

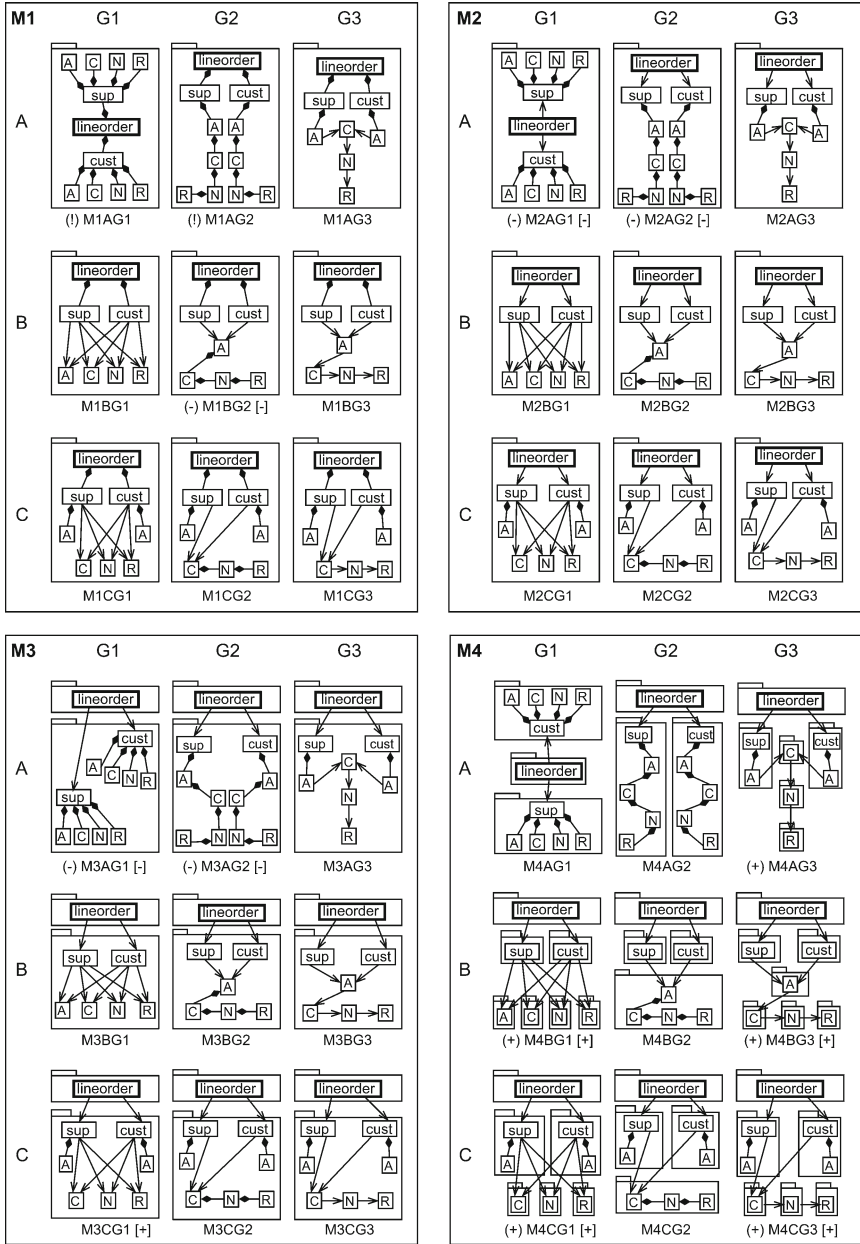
Table	Records	Columns	Size in MB
Lineorder	6,001,171	17	981.0
Customer	30,000	12	27.0
Date	2,556	17	0.3
Part	200,000	9	27.0
Supplier	2,000	11	1.4
c_address	30,000	2	2.8
s_address	2,000	2	0.2
City	250	2	7.7
Nation	25	2	5.7
Region	5	2	2.3



**Fig. 5.** RGDW schema used to generate the DGDW schemas.

*s\_address*, *city*, *nation*, and *region*. Table 1 displays the number of records, number of columns, and the table size, which corresponds to 1.03 GB, in total.

We used Pentaho Data Integration 8.0 to exhaustively transform the RGDW into 36 DGDWs (i.e.,  $\{M1, M2, M3, M4\} \times \{A, B, C\} \times \{G1, G2, G3\}$ ) (Fig. 6). The name of each DGDW schema is a concatenation of the model names presented in Sect. 3 (i.e.,  $M1 + A + G1 = M1AG1$ ,  $M1 + A + G2 = M1AG2$ , and so on). Although *part* and *date* are not shown in Fig. 6, these dimensions were included in the experiment and are related to *lineorder* as just like *supplier* and *customer*. We decided to highlight only the CD (i.e., *supplier* (sup) and



**Fig. 6.** DGDW schemas. Symbols used to highlight the schemas discussed in the Experimental Results Section: (+) Best volume size results; (-) Worst volume size results; [+] Best query performance results; [-] Worst query performance results; (!) Schemas discarded from the experimental evaluation.

*customer* (cust)) in Fig. 6 that are related to the GD (i.e., *c\_address/s\_address*<sup>1</sup> (A), *city* (C), *nation* (N), and *region* (R)). Furthermore, we created indexes for only the fields used to reference documents. It is important to highlight that the evaluation of sophisticated optimization techniques (e.g., spatial indexes, index selection, or materialized views) is another research problem and, for this reason, is outside scope of our work.

The 36 DGDW schemas used in our experiment have GD that contains points (i.e., *s\_address\_geo* and *c\_address\_geo*) and polygons (i.e., *city\_geo*, *nation\_geo*, and *region\_geo*). In order to perform a rigorous evaluation of these schemas, we defined 6 queries<sup>2</sup> (i.e., Q1, Q2, Q3, Q4, Q5, and Q6), which cover the different detail levels for projection, selection (conventional and geospatial), and grouping. These queries aggregate data from a set of customers whose address lies within a polygon determined by a random user click, simulating the *point-to-polygon* operation [15]. These queries also have increasing complexity, covering all geospatial dimensions, and simulating analytical operations (i.e., drill down, roll up, slice, and dice). In this sense, queries Q1 and Q2 aim to evaluate geospatial selection and conventional grouping, but Q2 extends this evaluation because it includes a conventional selection. Queries Q3, Q4, and Q5 aim to evaluate two levels of conventional groupings and explore geospatial selection from incrementally larger areas. Finally, Q6 differs from Q5 because it performs an *intersect* operation instead of *within*. Q6, therefore, selects a greater quantity of facts from the DGDW, being more complex than the previous queries. Although the schemas shown in Fig. 6 have two CDs (i.e., *sup* and *cust*), we decided to perform all queries on the *cust* CD, because it contains more documents than the *sup* CD ( $|cust| = 30,000 > |sup| = 2,000$ ), maintaining the rigor of our analysis. The query statements are as follows:

- Q1** How many customers made purchases and have an address within 10 miles of a given geospatial point, grouped by year?
- Q2** How many customers bought a product from a specific category and have an address within a 10-mile radius of a given geospatial point, grouped by year?
- Q3** What is the sum of revenues from customers whose address lies within a polygon corresponding to a given city, grouped by year and brand of products?
- Q4** What is the sum of revenues from customers whose address lies within a geospatial polygon corresponding to a given nation, grouped by year and brand of products?
- Q5** What is the sum of revenues from customers whose address lies within a polygon corresponding to a given region, grouped by year and brand of products?

<sup>1</sup> The *c\_address* and *s\_address* dimensions will be mentioned as *address*, since they have the same document type (i.e., the same field structure).

<sup>2</sup> The queries are available in <https://github.com/mrcferro/gdw>.



**Q6** What is the sum of revenues from customers whose address intersects a polygon corresponding to a given region, grouped by year and brand of products?

We perform the queries five times for each DGDW schema shown in Fig. 6. The test environment was composed of four computers. The central node was an Intel Xeon with 1 TB HD, 8 GB RAM, and a 1 Gbit/s network, and three other nodes were equipped with an Intel Core i3 processor having 500 GB HD, 4 GB RAM, and a 1 Gbit/s network. The computers used a CentOS 7.1 operating system and MongoDB 3.6.4 in sharding mode, which distributes the data across the machines. This setup favors horizontal scalability with low cost, by adding more hardware to the cluster.

The experimental evaluation investigated each schema shown in Fig. 6, analyzing the data volume and the arithmetic mean of five execution times for each query. Furthermore, we took special care to control the experiments. Each computer node was used exclusively for the proposed assessment and no other background processes were allowed to run while executing the queries. We rebooted all machines before executing the queries for each schema, cleaning any cached data used by the previous query.

## 5 Experimental Results

During the transformation from the RGDW to the 36 DGDWs, we noted that the M1AG1 and M1AG2 schemas would each need approximately 16 TB of storage. These schemas were therefore discarded from the experimental evaluation, because their data volume was much larger than that supported by the test environment ( $\approx 2.5$  TB). This high amount of storage was a consequence of considerable data redundancy, which follows the successive nesting of documents (i.e.,  $Fact \supset CD \supset GD$ ).

Table 2 presents the results of the experimental evaluation, showing the following columns: *Schema*, the name of the schema; *Size*, the disk space required to store each schema (in GB); *Q1* to *Q6*, the average of the five execution times for each query in seconds, followed by its respective standard deviation;  $Avg(Q_{1-6})$ , the average of the execution times for the six queries, also in seconds. The table is sorted in ascending order on column  $Avg(Q_{1-6})$ .

The schemas were evaluated based on data volume (*Size* column), and the average runtime of the 6 queries ( $Avg(Q_{1-6})$  column). Figure 7 depicts the volume vs. runtime relationship, showing each schema by its identifier (M1, M2, M3, and M4). Among the 34 schemas in Fig. 7, 18 have very similar results, with volume and runtime below 3 GB and 400 s, respectively. Figure 8 shows these 18 results at a larger scale.

Due to the great number of schemas, we will discuss the features of the top 5 (best) schemas and the bottom 5 in order to have some insight regarding the two ends of the results spectrum. Section 5.1 discusses the results based on data volume while Sect. 5.2 takes the average query execution time into account.

**Table 2.** Results of the experimental evaluation. Size in GB and query execution time in seconds. Symbols used to highlight the DGDW schemas: (+) Best results in volume size; (−) Worst results in volume size; [+] Best results in query performance; [−] Worst results in query performance; (!) Schemas discarded from experimental evaluation.

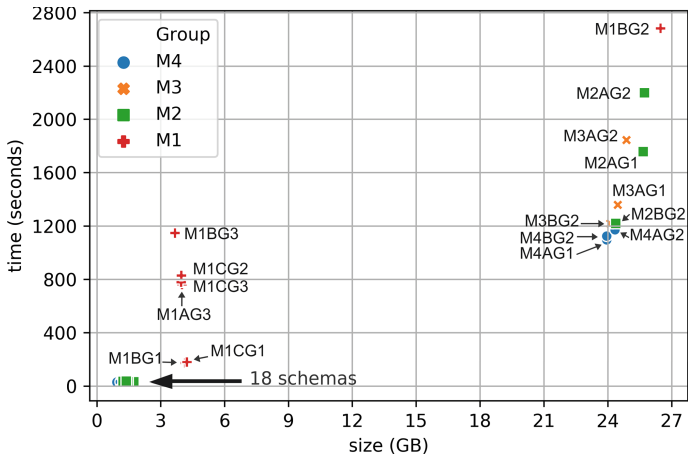
Schema	Size	Q1	Q2	Q3	Q4	Q5	Q6	$Avg(Q_{1-6})$
M4CG1	0.95(+)	0.59(0.00)	0.59(0.00)	1.33(0.00)	12.87(0.02)	61.12(0.07)	61.00(0.08)	22.92[+]
M4CG3	0.94(+)	0.57(0.00)	0.59(0.00)	1.33(0.00)	12.94(0.03)	61.76(0.08)	61.84(0.12)	23.17[+]
M4BG1	0.95(+)	0.58(0.01)	0.60(0.02)	1.36(0.02)	13.06(0.05)	62.26(0.16)	62.19(0.15)	23.34[+]
M4BG3	0.96(+)	0.59(0.01)	0.60(0.01)	1.36(0.02)	13.18(0.03)	62.49(0.16)	62.64(0.19)	23.48[+]
M3CG1	1.28	0.68(0.00)	0.69(0.00)	1.48(0.00)	13.67(0.03)	64.85(0.20)	65.01(0.14)	24.40[+]
M3CG3	1.15	0.67(0.01)	0.69(0.00)	1.47(0.00)	13.68(0.05)	65.19(0.14)	65.19(0.10)	24.48
M3BG3	1.20	0.69(0.00)	0.71(0.01)	1.51(0.03)	13.87(0.04)	65.28(0.12)	65.50(0.07)	24.59
M3BG1	1.14	0.70(0.00)	0.73(0.03)	1.53(0.02)	14.03(0.01)	65.57(0.21)	65.81(0.13)	24.73
M4AG3	0.94(+)	0.70(0.01)	0.65(0.03)	1.59(0.04)	15.64(0.03)	75.26(0.16)	75.51(0.08)	28.22
M3AG3	1.19	0.77(0.01)	0.72(0.03)	1.66(0.01)	15.71(0.07)	75.34(0.23)	75.77(0.43)	28.33
M2CG1	1.49	5.32(0.11)	5.33(0.21)	6.20(0.10)	18.92(0.09)	71.80(0.29)	71.53(0.13)	29.85
M2CG3	1.74	5.48(0.13)	5.59(0.12)	6.43(0.06)	19.22(0.33)	72.62(0.37)	70.41(0.37)	29.96
M4CG2	1.10	0.59(0.00)	0.61(0.02)	1.38(0.02)	13.72(0.01)	73.76(0.12)	90.78(0.14)	30.14
M3CG2	1.18	0.98(0.01)	0.95(0.04)	1.70(0.07)	13.65(0.04)	73.45(0.20)	90.56(0.11)	30.22
M2BG3	1.32	5.01(0.25)	5.24(0.20)	6.02(0.22)	19.04(0.40)	73.70(0.87)	73.18(1.10)	30.37
M2BG1	1.46	5.30(0.11)	5.44(0.05)	6.20(0.11)	19.14(0.14)	74.10(0.27)	73.95(0.19)	30.69
M2AG3	1.23	5.33(0.08)	5.51(0.22)	6.29(0.19)	20.91(0.35)	82.91(0.22)	83.24(0.25)	34.03
M2CG2	1.38	5.38(0.11)	5.72(0.13)	6.51(0.17)	18.73(0.06)	80.87(0.93)	98.46(0.26)	35.94
M1BG1	4.15	74.63(0.83)	56.99(0.95)	95.72(3.24)	391.71(6.75)	311.12(1.94)	102.51(1.64)	172.11
M1CG1	4.23	60.55(1.03)	59.72(0.93)	59.04(1.63)	417.80(10.98)	356.54(2.66)	119.12(1.52)	178.80
M1AG3	4.00	54.71(0.58)	53.89(0.47)	53.99(1.36)	410.78(11.61)	1978.58(33.86)	2016.48(34.39)	761.40
M1CG3	3.96	54.55(1.37)	56.11(2.42)	56.22(1.42)	413.32(6.86)	2039.98(19.72)	2050.72(18.22)	778.48
M1CG2	3.97	57.24(1.21)	56.03(1.84)	57.25(1.55)	454.27(5.40)	2159.45(17.65)	2188.14(20.80)	828.73
M4AG1	23.95	277.31(1.62)	282.22(2.92)	291.47(3.70)	439.22(8.45)	1654.81(10.47)	3631.29(5.74)	1096.05
M4BG2	23.95	296.27(0.50)	297.90(0.44)	312.31(1.14)	485.87(3.53)	1702.12(8.47)	3652.17(2.71)	1124.44
M1BG3	3.67	66.50(1.99)	52.48(2.34)	92.89(2.83)	565.32(11.80)	3013.13(49.15)	3096.65(56.61)	1147.83
M4AG2	24.34	353.25(0.88)	354.44(0.87)	364.86(3.16)	527.89(2.59)	1725.18(7.17)	3704.93(9.00)	1171.76
M3BG2	24.12	297.31(7.38)	299.04(6.18)	314.91(6.62)	674.98(11.82)	1907.40(58.06)	3814.66(12.06)	1218.05
M2BG2	24.37	360.39(10.07)	373.56(8.05)	397.82(6.87)	589.98(24.62)	1823.89(43.51)	3772.91(24.40)	1219.76
M3AG1	24.47(−)	299.21(7.13)	293.12(2.03)	329.84(7.88)	691.96(31.36)	1847.49(57.95)	4682.02(192.67)	1357.27[−]
M2AG1	25.66(−)	508.84(6.84)	508.41(9.92)	533.87(7.33)	901.93(7.50)	3023.39(40.32)	5061.43(36.19)	1756.31[−]
M3AG2	24.88(−)	309.38(3.61)	306.38(2.68)	347.55(4.29)	832.18(8.78)	3664.03(150.74)	5604.45(43.73)	1843.99[−]
M2AG2	25.72(−)	509.16(1.35)	504.56(4.62)	556.20(3.47)	1177.88(58.13)	4196.18(107.57)	6247.27(127.26)	2198.54[−]
M1BG2	26.48(−)	455.58(1.49)	441.85(3.47)	502.59(3.79)	1325.18(25.19)	5635.84(66.01)	7725.80(105.64)	2681.14[−]
M1AG1	≈16000(!)	−	−	−	−	−	−	−
M1AG2	≈16000(!)	−	−	−	−	−	−	−

## 5.1 Data Volume

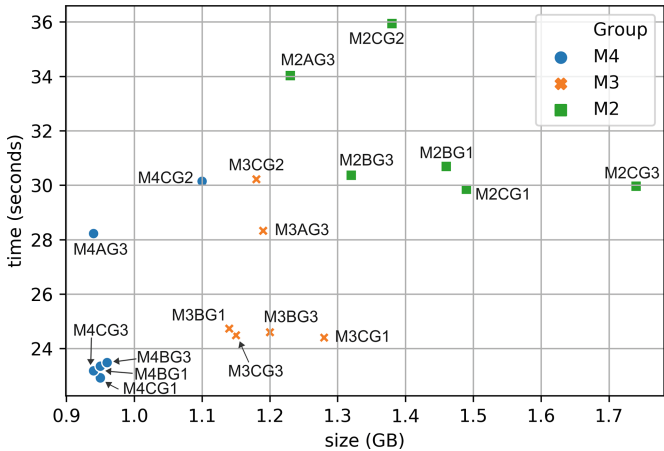
The top 5 and bottom 5 schemas are indicated in Fig. 6 by the symbols (+) and (−) at the left side of its name, respectively. In Table 2, these symbols are also found in the *Size* column.

**Top Schemas.** The five schemas with the lowest storage cost are M4CG3 (0.94 GB), M4AG3 (0.94 GB), M4BG1 (0.95 GB), M4CG1 (0.95 GB), and M4BG3 (0.96 GB). In Fig. 8, these schemas used a data volume of less than 1 GB.

The schemas with low storage costs tend to normalize their GDs. For example, all GDs of the M4BG1 and M4BG3 schemas were normalized. Except for the *address* GD, all other GDs of M4CG3, M4AG3, and M4CG1 schemas are also normalized. It is easy to see that the normalization of GDs with low selectivity



**Fig. 7.** Volume vs. execution time for the 34 evaluated schemas.



**Fig. 8.** Volume vs. execution time for the 18 schemas having similar results.

geospatial fields (i.e., *city*, *nation*, and *region*) strongly contributes to reducing the DGDW storage cost.

Analyzing the five schemas with regard to document partitioning shows that partitioning documents into specific and homogeneous collections also contributes to reduced data volume. This happens because the storage volume of the index structure is smaller in schemas holding the same document type in the collections. That is, all schemas in the M4 group (which have a collection for each dimension, i.e., homogeneous collections) have a lower volume than their corresponding schemas (e.g.,  $M4CG1 < M3CG1 < M2CG1$ ) (c.f., Table 2).

**Bottom Schemas.** The five schemas with the highest storage costs are M1BG2 (26.48 GB), M2AG2 (25.72 GB), M2AG1 (25.66 GB), M3AG2 (24.88 GB), and M3AG1 (24.47 GB). These schemas are identified in Fig. 7 as having volumes greater than 24 GB.

The results show that geospatial document redundancy increases data volume. The combination of the model A with G1 and G2, and the model B with G2 (i.e., schemas that have names ending in AG1, AG2, and BG2) results in high geospatial data redundancy when GDs that contain geospatial fields of low-selectivity (i.e., *city*, *nation*, and *region*) are denormalized into CDs. As shown in Table 2, the denormalization of geospatial fields with low-selectivity contributes strongly to raising the storage cost of the DGDWs.

Furthermore, the M1 group schemas have the highest data volumes. Indeed, the M1AG1 and M1AG2 schemas were removed from the experiment because they would have a size of approximately 16 TB. The M1 group also has higher volumes than those of its corresponding schemas (e.g.,  $M1BG2 > M2BG2 > M3BG2 > M4BG2$ ). Although the M1 group has schemas with homogeneous collections (c.f., M1AG1 and M1AG2 in Fig. 6) which contribute to reduced data volume, the denormalization of CDs (and their GDs) into fact documents strongly increases the data volume of a DGDW, which makes it impractical and not feasible.

## 5.2 Query Runtime

The top 5 and bottom 5 schemas are depicted in Fig. 6 by the symbols [+] and [-], respectively, to the right of their names. In Table 2, these symbols are also found in the  $Avg(Q_{1-6})$  column.

**Top Schemas.** The five schemas with the lowest average query runtimes are M4CG1 (22.92 s), M4CG3 (23.17 s), M4BG1 (23.34 s), M4BG3 (23.48 s), and M3CG1 (24.40 s). These schemas are depicted in Fig. 8 with query runtimes close to 24 s.

The top 5 schemas have their low-selectivity geospatial fields normalized into GDs (i.e., *city*, *nation*, and *region*). That is, the computational cost of joins to perform the queries is less than the cost of high data redundancy in the document-oriented database. However, the denormalization of the GD that contains a high-selectivity geospatial field (i.e., *address*) into CDs (i.e., M4CG1, M4CG3, and M3CG1 schemas) positively influences the query runtimes, by reducing the number of query joins (i.e., they do not need to perform joins to obtain *address*). Both the normalization of low-selectivity geospatial fields and the denormalization of high-selectivity geospatial fields contribute positively to the performance of the DGDWs.

Regarding schema partitioning, it can be seen in Fig. 6 that, except the M3CG1 schema, the best schemas all have one collection for each dimension (i.e., these schemas partition the fact documents, CDs, and GDs into specific and homogeneous collections). The M4 schemas also have better performance than their

corresponding M3 and M2 schemas (e.g.,  $M4CG3 < M3CG3$  and  $M4CG3 < M2CG3$ ). The partitioning of documents into homogeneous collections therefore contributes to improved DGDW performance.

**Bottom Schemas.** The five schemas with the highest average query runtimes are also the schemas with highest storage cost: M1BG2 (2681.14 s), M2AG2 (2198.54 s), M3AG2 (1843.99 s), M2AG1 (1756.31 s), and M3AG1 (1357.27 s). These schemas are depicted in Fig. 7, having query runtimes greater than 1300 s.

These schemas have high data redundancy, because their GDs that contain low-selectivity geospatial fields (i.e., *city*, *nation*, and *region*) are denormalized into CDs. When performing queries, selection operations and joins among dimensions require more processing power to manipulate the large volumes of data, greatly increasing the query runtimes.

Regarding the partitioning of these schemas, we noticed that they have heterogeneous collections (i.e., more than one dimension stored in the same collection). Therefore, there is a loss of query performance in collections that contain different document types, and the more heterogeneous the collection, the worse the performance. This finding can be shown by comparing, in Table 2, the results of corresponding schemas that have different partitioning models (e.g., M2BG2 vs. M3BG2).

## 6 Conclusion

In this paper, we investigated the use of a document-oriented NoSQL database for a GDW project. We sought to identify which level of geospatial data redundancy and document partitioning among collections provides low storage cost and good query performance. We proposed a UML-based notation for specifying collections, documents, and relationships between documents in DGDW. Using our notation, we exhaustively transformed one RGDW schema in 36 DGDW schemas. The 36 DGDWs contain different levels of redundancy in their conventional and geospatial data, as well as different forms of partitioning documents among collections. Next, we performed an experimental evaluation based on a computational infrastructure that can be scaled horizontally with the addition of new nodes in a cluster.

Considering the evidence obtained by experimental analysis, we identified that schemas with low-selectivity geospatial fields normalized into GDs have low storage cost and good query execution performance. Also, the denormalization of GDs that contain low-selectivity geospatial fields into CDs strongly contributes to an increase in the size of the schemas, as well as reduced query performance. This observation resembles that identified in [18]. However, in addition to the normalization of its high-selectivity geospatial fields, partitioning documents into collections also influences the volume and performance of the DGDW. We also identified that the denormalization of high-selectivity geospatial fields into CDs contributes to improved query performance because it reduces the number of joins in the queries and does not substantially affect the data volume.

Partitioning documents into homogeneous collections is another factor that positively influences the DGDW. The indexes structure of the collections has less volume and higher performance when the collections have the same document type (i.e., documents with the same fields structure). Thus, modeling the DGDW dimensions into distinct collections reduces data volume and increases system performance.

Therefore, the normalization of low-selectivity geospatial fields into GDs, the denormalization of high-selectivity geospatial fields into CDs, and the modeling of dimensions into distinct collections contribute to achieving a high level of performance at a low cost of storage for the DGDW schema.

For future studies, we will evaluate the performance of the best schemas based on volume size or query performance (i.e., M4CG1, M4CG3, M4BG1, M4BG3, M3CG1, and M4AG3) with higher data volumes (e.g., at scale factors of 10 and 100). Following this, we will compare the performance of these schemas with their corresponding schemas in a relational database.

## References

1. Almeida, R., Bernardino, J., Furtado, P.: Testing SQL and NoSQL approaches for big data warehouse systems. *Int. J. Bus. Process. Integr. Manag.* **7**(4), 322–334 (2015). <https://doi.org/10.1504/IJBPM.2015.073656>. <https://www.inderscienceonline.com/doi/abs/10.1504/IJBPM.2015.073656>
2. Chavalier, M., Malki, M.E., Kopliku, A., Teste, O., Tournier, R.: Document-oriented data warehouses: models and extended cuboids, extended cuboids in oriented document. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), pp. 1–11, June 2016. <https://doi.org/10.1109/RCIS.2016.7549351>
3. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R.: Document-oriented models for data warehouses. In: Proceedings of the 18th International Conference on Enterprise Information Systems, ICEIS 2016, vol. 1, pp. 142–149, December 2016
4. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R.: Implementing multidimensional data warehouses into NoSQL. In: 17th International Conference on Enterprise Information Systems, Proceedings 1, ICEIS 2015, pp. 172–183 (2015)
5. DB-Engines: DB-engines ranking, September 2018. <https://db-engines.com/en/ranking>. Accessed 20 Sept 2018
6. Dehdouh, K., Bentayeb, F., Boussaid, O., Kabachi, N.: Using the column oriented NoSQL model for implementing big data warehouses. In: 21st International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 469–475 (2015)
7. Fidalgo, R.N., Times, V.C., Silva, J., Souza, F.F.: GeoDWFrame: a framework for guiding the design of geographical dimensional schemas. In: Kambayashi, Y., Mohania, M., Wöb, W. (eds.) *DaWaK 2004*. LNCS, vol. 3181, pp. 26–37. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30076-2\\_3](https://doi.org/10.1007/978-3-540-30076-2_3)
8. Kimball, R., Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, Hoboken (2011)
9. Lee, J.G., Kang, M.: Geospatial big data: challenges and opportunities. *BigData Res.* **2**(2), 74–81 (2015). <https://doi.org/10.1016/j.bdr.2015.01.003>. <http://www.sciencedirect.com/science/article/pii/S2214579615000040>, visions on Big Data

10. Liu, Y., Vitolo, T.M.: Graph data warehouse: steps to integrating graph databases into the traditional conceptual structure of a data warehouse. In: 2013 IEEE International Congress on Big Data, pp. 433–434, June 2013. <https://doi.org/10.1109/BigData.Congress.2013.72>
11. Luo, W., Liu, B., Watfa, A.K.: An open schema for XML data in hive. In: 2014 IEEE International Conference on Big Data (Big Data), pp. 25–31, October 2014. <https://doi.org/10.1109/BigData.2014.7004409>
12. Mateus, R., Siqueira, T., Times, V., Ciferri, R., Ciferri, C.: How does the spatial data redundancy affect query performance in geographic data warehouses? *J. Inf. Data Manag.* **1**(3), 519 (2010)
13. Mateus, R.C., Siqueira, T.L.L., Times, V.C., Ciferri, R.R., de Aguiar Ciferri, C.D.: Spatial data warehouses and spatial OLAP come towards the cloud: design and performance. *Distrib. Parallel Databases* **34**(3), 425–461 (2016). <https://doi.org/10.1007/s10619-015-7176-z>
14. O’Neil, P.E., O’Neil, E.J., Chen, X.: The star schema benchmark (SSB). *Pat* 200(0)50 (2007)
15. Rigaux, P., Scholl, M., Voisard, A.: *Spatial Databases: With Application to GIS*. Morgan Kaufmann Publishers Inc., San Francisco (2001)
16. Scabora, L.C., Brito, J.J., Ciferri, R.R., Ciferri, C.D.d.A., et al.: Physical data warehouse design on NoSQL databases OLAP query processing over HBase. In: International Conference on Enterprise Information Systems, XVIII. Institute for Systems and Technologies of Information, Control and Communication-INSTICC (2016)
17. Siqueira, T.L.L., de Aguiar Ciferri, C.D., Times, V.C., de Oliveira, A.G., Ciferri, R.R.: The impact of spatial data redundancy on SOLAP query performance. *J. Braz. Comput. Soc.* **15**(2), 19–34 (2009)
18. Siqueira, T.L.L., Ciferri, R.R., Times, V.C., de Aguiar Ciferri, C.D.: Investigating the effects of spatial data redundancy in query performance over geographical data warehouses. In: Proceedings of the 10th Brazilian Symposium on Geoinformatics, pp. 1–12 (2008)
19. Siqueira, T.L.L., Ciferri, R.R., Times, V.C., de Aguiar Ciferri, C.D.: Benchmarking spatial data warehouses. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) *DaWaK 2010*. LNCS, vol. 6263, pp. 40–51. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15105-7\\_4](https://doi.org/10.1007/978-3-642-15105-7_4)
20. Song, J., Guo, C., Wang, Z., Zhang, Y., Yu, G., Pierson, J.M.: HaoLap: a hadoop based olap system for big data. *J. Syst. Softw.* **102**, 167–181 (2015)
21. Tria, F.D., Lefons, E., Tangorra, F.: Design process for big data warehouses. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA), pp. 512–518, October 2014. <https://doi.org/10.1109/DSAA.2014.7058120>
22. Yangui, R., Nabli, A., Gargouri, F.: Automatic transformation of data warehouse schema to nosql data base: comparative study. *Procedia Comput. Sci.* **96**, 255–264 (2016)