ISSN 2007-9737

Comparison of the Response Times of MongoDB and PostgreSQL According to Type of Query in Geographical Databases

Marlen Treviño Villalobos, Leonardo Víquez Acuña, Rocío Quirós Oviedo

Instituto Tecnológico de Costa Rica,
Costa Rica

{mtrevino, lviquez, rocio.quiros}@tec.ac.cr

Abstract. A hybrid and distributed geographical database is being developed which utilizes the database engines PostgreSQL and MongoDB for later implementation in Geoserver's architecture. In the course of the study, it was necessary to establish whether there was difference between the response times of PostgreSQL and MongoDB per type of query and the use of geographical indexes, in order to appropriately select the Database Management System to be used by the Geoserver implemented Web Map Service. After a statistical analysis with 11 different types of query, the conclusion was that the type of query affects the response time of the Database Management Systems.

Keywords. Performance evaluation, SQL, relational database, NoSQL.

1 Introduction

The demand for geographical information has grown considerably in recent years [1] and citizens are among the main generators of this geographical explosion [2]. While using it, users also participate actively in the generation of geographical data, in the sense that most people have currently become a mobile sensor that registers and records large volumes of data that require higher computing capability and more advanced and efficient processing and analyzing methods [3].

As a response to the issue raised above, work has been conducted on producing hybrids between the relational (SQL) and non-relational (NoSQL) database paradigms [4]. Hybrid databases work as an abstraction layer over SQL and NoSQL databases [5].

Currently, a hybrid and distributed database that uses PostgreSQL and MongoDB database engines is being developed for future implementation in the Geoserver base architecture [6] to evaluate whether this modification enhances its performance. However, while developing the project, it became necessary to establish if there was any difference in the response times of PostgreSQL and MongoDB, according to the type of query and use of geographical indexes. The aim was to choose appropriately the DBMS (Data Base Management System) to be employed by the Web Map Service (WMS) implemented by Geoserver.

Since PostgreSQL was one of the first databases to address spatial issues, we selected it to construct the hybrid and distributed database [7]. PostgreSQL's extension, PostGIS, [8], is highly optimized for spatial queries [3] and its large quantity of spatial functions make it relevant for this research project. On the other hand, there are currently over 225 NoSQL databases [9] with only a reduced number supporting geographical data operations, among which Neo4j, CouchDB, MongoDB [10] and ArangoDB outstand in this area. We chose MongoDB because, to date, it was the only document-based NoSQL database that supports line intersection and point containment queries [3].

Lastly, both are open code DBMS and Geoserver gives them support because in its version 2.11.4, Geoserver [6] included a data connexion and publication component from MongoDB.

doi: 10.13053/CyS-24-4-3292

One of the main motivations that led us to conduct this research was the scarce number of works comparing PostgreSQL's and MongoDB's performance. Among these works, there are [11] and [12], which focus on comparing the performance in the operations insert, select, update and delete. The study of [6] also focuses on comparing geographical queries using synthetic data. However, although the synthetic data usually favor the evaluation of queries regarding numeric and string characters, the synthetic geometries generated differ from real-world data, because the randomness of the generator can produce too-square polygons and unnatural landforms [13].

Another work analyzes geographical queries using real data [13], but it focuses on analyzing PostgreSQL, MongoDB and Neo4j. Finally, [14] published the basis for the present article, with the restriction that its analysis was based on descriptive statistics. Here, this analysis is improved by carrying out a statistical analysis that uses real spatial data corresponding to the Huetar Norte Region of Costa Rica, focusing only in evaluating the query operation and both DBMS of interest.

In this work, eleven types of queries were analyzed, using or not the geographical index. The results obtained allow us to conclude that the type of query influences the response time of both DBMS.

2 Methodology

2.1 Approach

The present research applied a quantitative approach that permits the assessment of DBMS response times in data processing in eleven different reading operations.

2.2 Type and Level of the Investigation

This was an experimental study. Experiments to obtain the data were conducted in the laboratory. In addition, it is a prospective research from the point of view of data collection planning, as the data were obtained specifically for this article. Moreover, because of the number of times the variable was measured, this is a longitudinal study, with eleven query types with two thousand requests analyzed for each DBMS. The study is also analytical, since the behavior of the DBMS was analyzed in order to detect possible relationships between them. Lastly, it is an explanatory research aimed at establishing cause-effect relationships between the variables analyzed based on the results obtained through the experiment.

2.3 Collection of Information

Valid SQL queries were defined and executed on MongoDB and PostgreSQL DBMS using equivalent databases, in order to evaluate the compatibility of different DBMS by comparing the results obtained [15]. Eleven database queries were generated, with different access patterns and levels of complexity, related to geographic information. The aim was to evaluate the behavior of the engines in simple data recovering transactions, as well as in filter application and geographical processing.

Next, each query was executed with 2000 requests and all measurements were executed 10 times, aiming at minimizing the response time variations caused by the process allocation of the operating system. The queries utilized in the tests developed are described in Table 1.

The tool JMeter was used for data collection [16]. JMeter allows the measurement of the behavior of the DBMS, according to the process described above. This application has a simple graphical user interface that offers ample load generation capability, is open code and implemented in Java [17].

Likewise, it is an environment that allows controlling the variables, that is, the operations are designed and managed by the testing team and the database used corresponds to a real project data sample. The component Summary Report, provided by the tool, was used to visualize and evaluate the test results in a table. The data this component shows are [18]:

– Label: sample label.

– #Samples: number of threads used.

– Average: average response time in milliseconds for a set of results.

Table 1. Types of queries

Query number	Type of query	Description
Query 1	Basic retrieval of the geographic table	Selects all the attributes of the towns (type of geographic data: points).
Query 2	Information retrieval using the function *within*	This is a geographic query that uses the function *within*. First, it determines the delimiting square of the geometry by using the R tree index, then it loads the geometry. This approach improves performance greatly [24]. That is, it takes two geometries as entry parameters and returns the number one if the first geometry is within the second one. Specifically, two towns are selected (geometry: points) that are within the district of Florencia (geometry: polygon).
Query 3	Information retrieval with condition on clause *where*	Selects all the attributes of the San Carlos districts data set (geometry: polygon). The aim of including conditions in the *where* clause in an SQL query is to filter those tuples that meet certain characteristics represented by those conditions [25].
Query 4	Information retrieval using the *intersect* function between two geometries of the type polygon	The *intersect* function returns a geometry that represents the intersection of the set of points of the geometries [26]. The query specifically selects the districts that intersect the protected area of Arenal.
Query 5	Information retrieval using the functions *within* and *intersect*	Selects the touristic attractions (geometry: points) found in the districts (geometry: polygons) that intersect the protected area of Arenal
Query 6	Information retrieval using the *near* function	Given a point in a geospatial query, the *near* function in MongoDB returns the documents from the closest to the furthest. The $near operator can specify a GeoJSON point or an inherited coordinate point [27]. The restriction of this function is that it necessarily requires the use of a geospatial index. In the case of PostgreSQL, the equivalent function is *ST_DWithin*, which delivers true if the geometries are *within* the specified distance between them [26]. It uses indexes if available. In addition, the data should be ordered by distance by means of the *ST_Distance* function. It selects the towns (geometry: points) that are located 0.10 meters from the town of Quesada ordered from greatest to least distance.
Query 7	Information retrieval using the *intersect* function between two geometries of the type line	Selects all the roads that intersect a specific river
Query 8	Information retrieval using the *intersect* function between a line-type geometry and a polygon-type geometry	Selects all the rivers (geometry: line) that intersect the district of Quesada (geometry: polygon).
Query 9	Information retrieval using the *intersect* function between a point-type geometry and a polygon-type geometry	Selects all the towns (geometry: point) that intersect the district of Florencia (geometry: polygon).
Query 10	Information retrieval using the *intersect* function between a point-type geometry and a line-type geometry	Selects all the rivers (geometry: line) that intersect a specific town (geometry: polygon).
Query 11	Information retrieval using the *intersect* function between two point-type geometries	Selects all the touristic attractions that intersect a specific town

- Max: maximum time it takes to a thread to access a page.
- Performance: measured performance of the requirements per second/minute/hour.
- Kb/sec: measured performance in Kilobytes per second.
- Mean in bytes: mean response size of the server (in bytes).

The data set used for test execution corresponded to the Huetar Norte Region in Costa Rica and included the geometric structures based on points, lines or polygons that correspond to vector data. For this, six files (shapefile) were selected [19] from the web site IDEHN (Infraestructura de Datos Espaciales de la Región Huetar Norte) [20].

These are: Caminos de Costa Rica; Ríos de la Región Huetar Norte de Costa Rica, which are represented as lines; Poblados de la Región Huetar Norte de Costa Rica and Atractivos turísticos de Costa Rica, which correspond to points, and Distritos de la Región Huetar Norte de Costa Rica and Áreas Silvestres Protegidas de Costa Rica, which appear as polygons. Then, these geographic data layers were converted to GeoJSON format [21] by means of the QGIS tool [22]. In both DBMS, the data layers were used with the coordinate reference system WGS84 [23] associated to SRID (Spatial Reference System Identifier) number 4326. Next, the data were manually uploaded to each DBMS, so the data had to be read, processed and stored on a data disk.

The river layer contains 204152 geographical records; the road layer, 222965; the town layer, 200662; attractions, 100509; districts, 10037 and protected areas, 533 geographical records.

2.4 Test Environment

All tests were performed using an Intel core i7 processor machine with the Ubuntu 16.04 LTS 64-bit operating system with 16 GB RAM. The DBMS versions are MongoDB Server 3.4.10 [28], and PostgreSQL 9.6 [7].

The tool Apache JMeter 3.2 was used for performance evaluation and comparison [16]. The drivers used for connection with each DBMS were: PostgreSQL 42.1.4 JDBC, MongoDB 2.11.3 Java.

2.5 Statistical Analysis

The analysis was performed per type of query, evaluating for each case the criterion of normality, using the Anderson-Darling statistical test, see [29] and [30], and the homogeneity of variance, by means of the Levene test, for the quantitative variable response time. These tests represented input for performing the combined analysis of variance (ANOVA) of a factor for independent samples, per DBMS and type of query.

Lastly, the descriptive statistics were calculated for the response time per DBMS and type of query. All the tests were verified at α=0.05 significance threshold.

3 Results and Discussion

Queries 1 and 3 were executed without geographical index only, either because of their simplicity or because a sequential search of the table is performed.

Query 6 was executed with geographical index only, because MongoDB does not allow the execution of the near function without using this data structure.

Based on the characteristics of the hardware used and the excessive response time, queries 4, 5, 7 and 8, which involved the intersection of polygons with polygons; an intersect with a within; intersection of lines with lines and lines with polygons, which could not be executed in none of the DBMS, without utilizing a geographical index. Queries 5 and 8 could not be executed in MongoDB using a geographical index either. Finally, query 10 could not be executed in MongoDB without geographical index.

3.1 Queries without Geographical Index

Query 1

For query 1, the quantitative variable response time for the DBMS MongoDB and PostgreSQL was normal, with p-values of 0.6464 and 0.5232, respectively. As for the variance homogeneity, the p-value was 0.1217. In the ANOVA (see Table 2), the p-value calculated for the DBMS was lower than the level of significance, therefore the null

Table 2. ANOVA for response time without geographical index

Dependent variable: Response time

	Sum of squares	DF	F	Pr(>F)
DBMS – Query 1	154708	1	52194	<2e-16
Residual – Query 1	53	18		
DBMS – Query 2	746544	1	48030	<2e-16
Residual – Query 2	280	18		
DBMS – Query 3	6321894	1	5576884	<2e-16
Residual – Query 3	20	18		
DBMS – Query 9	1744101	1	3383	<2e-16
Residual – Query 9	9280	18		
DBMS – Query 11	1345311	1	176099	<2e-16
Residual – Query 11	138	18		

hypothesis of equal means was rejected and the existence of significant differences in the response times between the DBMS was admitted. In addition, figure 1 shows that MongoDB’s response time was lower than PostgreSQL’s.

Query 2

The response time for query 2 behaved normally with a p-value of 0.1414 for MongoDB and of 0.1565 for PostgreSQL. The data showed homogeneity of variances with a p-value of 0.4483. Significant differences in response times between the DBMS were determined through the ANOVA (see Table 2). Similarly, figure 1 shows that PostgreSQL’s response time was lower than MongoDB’s.

Query 3

The response time in query 3 meets the assumption of normality, because the p-value calculated was of 0.3464 for MongoDB and of 0.1724 for PostgreSQL. Likewise, the homogeneity of variances between the two groups was corroborated, with a p-value of 0.1036.

Through the ANOVA (see Table 2), significant differences were determined in the response times between the DBMS. Figure 1 shows that MongoDB’s mean response time was lower than PostgreSQL’s.

Query 9

The response time data in query 9 were normal with a p-value of 0.2911 for MongoDB and of 0.3835 for PostgreSQL. The Levene test for equality of population variables was statistically significant with p-value of 0.2714.

The response times of MongoDB and PostgreSQL were compared through the ANOVA (see Table 2) and significant differences were revealed. Figure 1 shows that PostgreSQL presented lower response time than MongoDB.

Query 11

The criterion of normality was satisfied with a p-value of 0.2005 for MongoDB and of 0.5052 for PostgreSQL. Both groups evaluated complied with the homogeneity of variance with a p-value of 0.2341. The DBMS showed significantly different response times, which can be corroborated with the ANOVA results in Table 2. Figure 1 shows that MongoDB’s mean response time was lower than PostgreSQL’s.

3.2 Queries with Geographical Index

Query 2

The quantitative variable response time showed normal behavior. For the DBMS MongoDB, p-value

Table 3. ANOVA for the response time with geographical index

Dependent variable: Response time				
	Sum of squares	**DF**	**F**	**Pr(>F)**
DBMS – Query 2	154708	1	52194	<2e-16
Residual – Query 2	53	18		
DBMS – Query 4	746544	1	48030	<2e-16
Residual – Query 4	280	18		
DBMS – Query 6	6321894	1	5576884	<2e-16
Residual – Query 6	20	18		
DBMS – Query 7	20161706	1	6585004	<2e-16
Residual – Query 7	55	18		
DBMS – Query 9	7337813	1	144684	<2e-16
Residual – Query 9	913	18		
DBMS – Query 10	29084	1	4497	<2e-16
Residual – Query 10	116	18		
DBMS – Query 11	1050557	1	242507	<2e-16
Residual – Query 11	78	18		

was of 0.0765 and for PostgreSQL, the p-value was 0.1512. As for homogeneity of variance, the p-value was 0.4909.

According to the variance analysis (Table 3) the null hypothesis of equal means was rejected, and significant differences between the response times of both DBMS were accepted. Figure 2 shows that PostgreSQL had lower response time than MongoDB.

Query 4

The normality test produced a p-value of 0.8479 for MongoDB and of 0.2873 for PostgreSQL, therefore the null hypothesis was accepted. In like way, the hypothesis of homogeneity of variance was accepted with a p-value of 0.1610. The ANOVA results (see Table 3) led to the conclusion that there were significant differences in the response times between both DBMS. The average response time for PostgreSQL is lower than for MongoDB (see figure 2).

Query 6

Both MongoDB's and PostgreSQL's response times present normality, with p-values of 0.6549 and 0.1461, respectively. In addition, these two groups of data present variance equality with a p-value of 0.1376.

A significant difference in the response times between both DBMS can be deduced from Table 3. Figure 2 shows that the mean response time of MongoDB is lower than for PostgreSQL.

Query 7

With a p-value of 0.1570 for MongoDB and 0.1134 for PostgreSQL, the response time for both DBMS was catalogued as normal. The data presented homogeneity of variance with a p-value of 0.8876. The ANOVA (see Table 3) led to reject the null hypothesis of equal means for there was a significant difference between the times responses of MongoDB and PostgreSQL.

Figure 2 shows that PostgreSQL presented lower response time.

Query 9

The criterion of normality test for the response times of MongoDB and PostgreSQL produced p-values of 0.9338 and 0.1367, respectively, meaning that the DBMS satisfied this criterion. The homogeneity of variance test was also accepted with a p-value of 0.1078. The analysis of variance (see Table 3) revealed a significant difference in

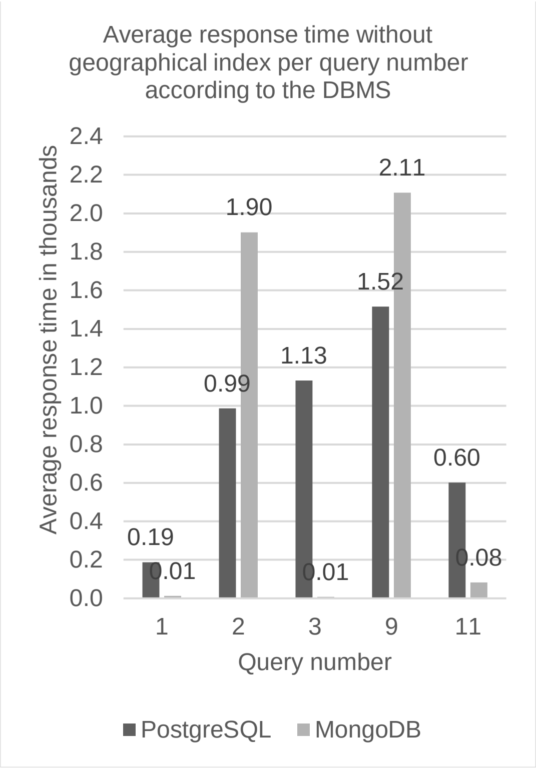

Fig. 1. Average response time without geographical index per number of query according to the DBMS

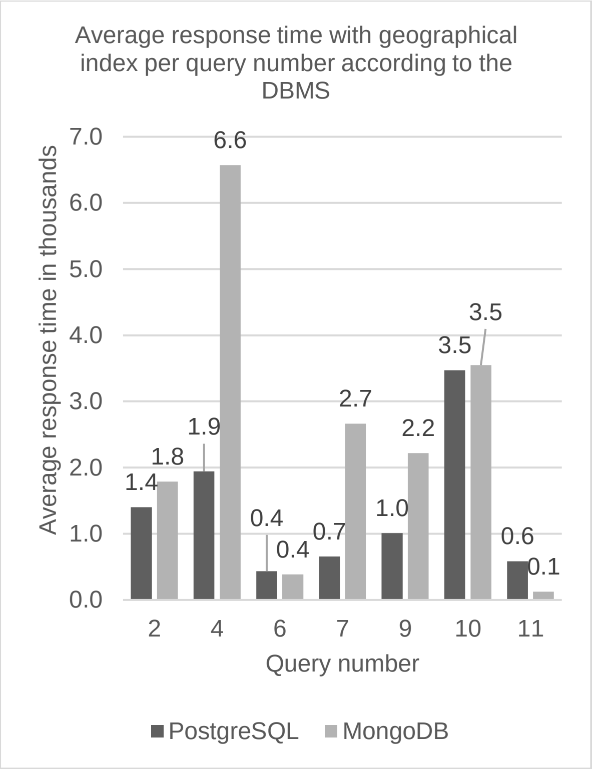

Fig. 2. Average response time with geographical index per query number according to the DBMS

the response times of both DBMS and figure 2 shows PostgreSQL's response time is lower than MongoDB's.

Query 10

Normality of the variable was verified for each DBMS; p-value was 0.1307 for MongoDB and 0.6134 for PostgreSQL. Similarly, homogeneity of variance of the two groups analyzed was demonstrated with a p-value of 0.2803. Table 3 indicates a significant difference between the response times of both DBMS, where PostgreSQL had the lowest response time.

Query 11

The response time satisfied the normality assumption, for p-value was of 0.1131 for MongoDB and of 0.1640 for PostgreSQL. Homogeneity of variances between both groups was demonstrated, obtaining a p-value of 0.2038.

A significant difference in the response times of the DBMS was determined through the ANOVA (see Table 3). Figure 2 shows that MongoDB's mean response time was lower than PostgreSQL's.

4 Conclusions

Based on the response time, using the MongoDB DBMS is preferable when the types of query to be executed have a non-geographic descriptive data filtering or their return implies a sequential scan of the table. It is also preferable when the query requires a single geographic processing between point-type geometries in intersect or near functions, as applicable.

The use of PostgreSQL is recommended for higher level processing queries involving intersect or within functions and the combination of both.

PostgreSQL is also recommended for more complex types of geographic data, such as lines, polygons and their variations, and in the implementation of geographic operation queries supported by this DBMS only.

Currently, there are some limitations on the use of NoSQL databases over the SQL databases, since the geo-functions implemented in the SQL databases only admit basic operations, while the relational databases offer greater variety [31].

According to this research, to choose which DBMS is to be used for certain queries, multiple variables should be evaluated, such as: type of geometry, size of the geographical data, type of query, use and type of geographical index.

We intend to deepen this study in the future with the implementation of other types of geographical indexes involving enhanced use of the resources, and also determine when to use a DBMS in function of the size of the geographical data.

References

1. **Moreno-Jiménez, A. (2004).** Nuevas tecnologías de la información y revalorización del conocimiento geográfico. *Scripta Nova: Revista Electrónica de Geografía y Ciencias Sociales,* Vol. 8, No. 170
2. **Ruiz, E. (2010).** Consideraciones acerca de la explosión geográfica: Geografía colaborativa e información geográfica voluntaria acreditada, *GeoFocus. Revista Internacional de Ciencia y Tecnología de la Información Geográfica,* No. 10, pp. 280–298.
3. **Agarwal, S. & Rajan, K.S. (2016).** Performance analysis of MongoDB versus PostGIS/PostGreSQL databases for line intersection and point containment spatial queries. *Spatial Information Research,* Vol. 24, pp. 671–677. DOI:10.1007/s41324-016-0059-1
4. **Colorado-Pérez, M.A. (2017).** NoSQL, ¿es necesario ahora?. *Tecnología Investigación Y Academia,* Vol. 5, No. 2, pp. 174–179.
5. **Goyal, S., Srivastava, P.P., & Kumar, A. (2015).** An overview of hybrid databases. *International Conference on Green Computing and Internet of Things*, pp. 285–288. DOI:10.1109/ICGCIoT.2015.7380474.
6. **The PostgreSQL Global Development Group (2018).** *PostgreSQL 10.3 Released! The World's Most Advanced Open Source Database.* https://www.postgresql.org/.
7. **PostGIS. (2018).** *PostGIS—Spatial and Geographic Objects for PostgreSQL*.
8. **NoSQL. (2018).** http://nosql-database.org.
9. **Ramírez-Arévalo, H.H. & Herrera-Cubides, J.F. (2013).** Un viaje a través de bases de datos espaciales NoSQL. *Redes de Ingeniería*, Vol. 4, No. 2, pp. 57–69. DOI: 10.14483/2248762X.5923.
10. **Jung, M.G., Youn, S.A., Bae, J., & Choi, Y.L. (2015).** A Study on Data Input and Output Performance Comparison of MongoDB and PostgreSQL in the Big Data Environment. *8th International Conference on Database Theory and Application (DTA),* pp. 14–17. DOI:10.1109/DTA.2015.14.
11. **Politowski, C. & Maran, V. (2014).** *Performance entre PostgreSQL e MongoDB, X Escola Regional de Banco de Dados.* pp. 1–10.
12. **Santos, P.O., Moro, M.M., & Davis, C.A. (2015).** Comparative Performance Evaluation of Relational and NoSQL Databases for Spatial and Mobile Applications. **Chen Q., Hameurlain A., Toumani F., Wagner R., Decker H. (eds).** *Database and Expert Systems Applications. Globe´15, DEXA´15.* Lecture Notes in Computer Science, Vol. 9261. DOI:10.1007/978-3-319-22849-5_14.
13. **Orellana-Cordero, M., Vele-Zhingri, C.A. (2018).** *Análisis de rendimiento para bases de datos geográficas: PostgreSQL vs MongoDB. de COMPDES.*
14. **Slutz, D.R. (1998).** Massive stochastic testing of SQL. *24th Very Large Data Base Conference (VLDB).*
15. **Apache Software Foundation. (2017). Apache** *JMeter.* http://jmeter.apache.org/.
16. **Patel, B., Parikh, J., & Shah, R. (2014).** A Review Paper on Comparison of SQL Performance Analyzer Tools: Apache JMeter and HP LoadRunner. *International Journal of Current Engineering and Technology,* Vol. 4, No. 5, pp. 3642–3645.
17. **Díaz, F.J., Banchoff-Tzancoff, C.M., Rodríguez, A.S., & Soria, V. (2008).** Usando Jmeter para pruebas de rendimiento. *XIV Congreso Argentino de Ciencias de la Computación*.
18. **ESRI. (1998).** *Shapefile Technical Description.* ESRI White Paper.
19. **Instituto Tecnológico de Costa Rica. (2017*). IDE Región Huetar Norte*. http://www.idehn.tec.ac.cr/.
20. **Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., & Schmidt, C. (2008).** *The GeoJSON format specification*. Rapport Technique, pp. 67.
21. **QGIS. (2018).** *Bienvenido Al Proyecto QGIS!.* http://www.qgis.org/es/site/.

22. **Mongodb. (2018).** *Geospatial Queries - MongoDB Manual 3.6.* https://docs.mongodb.com/manual/geospatial-queries/.
23. **Schmid, S., Galicz, E., & Reinhardt, W. (2015).** WMS performance of selected SQL and NoSQL databases. *International Conference on Military Technologies,* pp. 1–16. DOI:10.1109/MILTECHS.2015.7153736.
24. **Suárez, M.J. & Tuya, J. (2005).** Coverage Measurement for SQL Queries. *IEEE Latin America Transactions*, Vol. 3, No. 1, pp. 49–55. DOI: 10.1109/TLA.2005.1468662.
25. **Ramsey, P. (2005).** *Postgis Manual.* Refractions Research Inc., Vol. 17.
26. **Membrey, P., Plugge, E., & Hawkins, D. (2011).** *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Apress*.*
27. **MongoDB. (2017).** *The MongoDB 3.4 Manual*. http://docs.mongodb.com/v3.4/.
28. **Anderson, T.W. & Darling, D.A. (1954).** A test of goodness of fit. *Journal of the American Statistical Association,* Vol. 49, No. 268, pp. 765–769. DOI: 10.1080/01621459.1954.10501232.
29. **Anderson, T.W. & Darling, D.A. (1952).** Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *The Annals of Mathematical Statistics,* Vol. 23, No. 2, pp. 193–212. DOI: 10.1214/aoms/1177729437.
30. **Schmid, S., Galicz, E., & Reinhardt, W. (2015).** *Performance investigation of selected SQL and NoSQL databases.* AGILE/15.

Article received on 22/10/2019; accepted on 23/06/2020.
Corresponding author is Marlen Treviño-Villalobos.