Using a DBMS based on various data models in solving problems of cartography and geoinformatics

Alexey A. Kolesnikov[1*] and *Pavel M.* Kikin[2]

[1]Siberian State University of Geosystems and Technologies, 630108, Plakhotnogo str., 10, Novosibirsk, Russian Federation
[2]Peter the Great St. Petersburg Polytechnic University, 195251, Polytechnicheskaya str., 29, St. Petersburg, Russian Federation

Abstract. An increasing number of database management systems are expanding their functionality to work with various types of spatial data. This is true for both relational and NoSQL data models. The article describes the main features of those data models for which the functions of storing and processing spatial data are implemented. A comparative analysis of the performance of typical spatial queries for database management systems based on various data models, including multi-model ones, is carried out. The dataset on which the comparison is performed is presented in the form of three blocks of OpenStreetMap vector data for the territory of the Novosibirsk region. Based on the results of the study, recommendations are made on the use of certain data models, depending on the available data and the tasks to be solved.

1 Introduction

Today, most geographic information systems use the relational model as the main data storage model. Moreover, it can be implemented both in the format of the geographic information system itself, and using any third-party relational database management system [1]. Relational databases have a number of undeniable advantages [2], but alternative storage models (NoSQL) have received significant development, which also have their advantages [3]. These positive aspects are considered in sufficient detail [4], but they do not always affect the side of storage and processing of spatial data. At the same time, many of the NoSQL DBMS add and extend functions for working with spatial data. The main idea of the research described in the article was to compare the performance of DBMSs based on different data storage models when performing basic spatial queries.

2 Materials and Methods

* Corresponding author: alexeykw@yandex.ru

The study consisted of the following blocks: formation of a set of vector spatial data, selection of NoSQL DBMS that allow making queries to spatial data, compiling a list of spatial queries that are identical in meaning, their implementation and formulation of conclusions.

A set of vector data for experiments is objects (point, linear and areal) from OpenStreetMap to the territory of the Novosibirsk region. To assess the effect of the number of objects on the final productivity, subsets were created for the territory of the city of Novosibirsk and its Leninsky district. The number of objects of each type for the formed subsets is presented in Table 1.

Table 1. Count of objects in sets.

	Region	City	District
Point (POI)	12102	8631	986
Lines (roads)	104615	34886	2129
Polygons (landuse)	23070	4843	513

The relational database PostgreSQL with PostGIS and pgRouting modules was taken as the base DBMS with which the study compared NoSQL solutions [5]. This choice was made based on the popularity of this software according to the DB-Engines service in the implementation of data warehouses for various geographic information systems.

2.1 Overview

The main data storage models (including spatial) NoSQL fall into the following categories:
- document based;
- graph;
- wide column;
- key-value;
- search engine.

The main advantages of NoSQL solutions (valid in the case of storing spatial data) include:

• dynamically modified data schema;

• increased processing speed compared to the relational model;

• more flexible scalability, also in comparison with the relational model [6].

At the same time, the disadvantages of NoSQL DBMS when working with spatial data are also present and among the most significant it should be noted:

• fewer types of spatial data available for storage (assessed according to ISO / IEC 13249: 2016) compared to the most popular relational DBMS;

• in most cases, there are only basic functions for working with spatial data (searching for intersections, objects in an area, at a distance, etc.), and more complex tasks are meant to be solved by using external software libraries;

• fewer types of spatial indexes compared to relational DBMSs, which does not always allow for potentially higher data processing speed;

• complexities of integration with common GIS as the main data store, as opposed to relational DBMS, which can completely replace file storage;

• difficulties with storing and processing three-dimensional data, consisting in the lack of functionality for taking into account the height data of the NoSQL DBMS when performing spatial queries and geometric calculations;

• difficulties with the use of custom coordinate systems in a NoSQL DBMS lie in the absence or insufficiently complete implementation of the most common formats for describing coordinate systems.

But, as already mentioned, the listed disadvantages of NoSQL DBMS (including multi-model ones) are compensated for by the wider possibilities of distributed storage and processing, and thereby increasing the processing speed, as well as additional functionality in the case of individual tasks, which was planned to be confirmed by the described study.

Next, let's consider the features of the main NoSQL storage models and their implementations that support working with spatial data.

2.2 Document-based Databases

Document-oriented DBMSs operate with the concept of a document, which is a collection of sets of attributes (a key and its corresponding value). Values can in turn be nested documents or arrays. Documents of the same type may or may not have common attributes because there is no hard-coded schema. Such DBMS in many cases allow the developer to reflect the entities of the subject area on the entities of the database without introducing additional entities that have to be entered in relational DBMS, that is, one-to-one and one-to-many relationships are displayed without additional tables and fields for communication [7]. The features of document-oriented DBMS include:

• predominant use of the GeoJSON format for storing spatial data;

• own query language (although there are translators from the SQL language);

• basic set of geometric feature types;

• storage of complex structures with a simpler data schema compared to relational;

• potentially faster execution of basic spatial queries (under conditions of large data volumes, correct setting of indexes and distributed storage scheme);

• initial implementations of distributed storage and processing functions, including spatial data.

2.3 Graph-oriented Databases

Graph DBMSs are initially focused on relationships between objects, and these relationships can have different characteristics. The main advantage of graph databases is their versatility in the form of storing relational, documentary and semantic data [8]. The features of graph DBMS include:

• own query language (allowing to use complex semantic connections of the database, but requires mastering query constructs and is poorly translated from SQL);

• the need to transform spatial objects with different types of localization into a graph structure ;

• faster execution of queries for analyzing network structures (provided that the architecture and indexes are configured correctly).

2.4 Wide-column Databases

Columnar databases store data not in rows (in the case of relational DBMSs), but in columns. They are like a separate table of one column, which stores only its values. In addition, all data in a columnar database is usually stored in a sorted form [9]. The features of columnar DBMS include:

• own query language (usually similar in structure to SQL);
• the need to work out the optimal structure of the database;
• basic set of geometric feature types;
• high write speed of grouped data;
• faster execution of basic spatial queries (provided that the data grouping within the columns is configured correctly);
• implementations of distributed storage and processing functions, including for spatial data.

2.5 Key-Value-Pair Datastorage

A pair-based database stores data as a collection of key-value items, in which the key is the unique identifier of the item. The values of both keys and values can be either a regular set of characters or a complex compound object. Databases using key-value pairs provide a high degree of parallelism and horizontal scaling that is often not possible with other database models [10]. The features of the "key-value" DBMS include:
• own query language;
• the need to work out the optimal structure of the database;
• basic set of geometric feature types;
• faster execution of basic spatial queries (provided that data grouping is configured correctly);
• fast execution of basic operations due to in-memory mechanisms (subject to sufficient hardware resources).

2.6 Search Engines

Search engines were initially focused on working with textual information, but now their functionality has expanded and allows you to work with spatial objects [11]. The features of search engines include:
• own query language;
• basic set of geometric feature types;
• faster execution of basic spatial queries in combination with attribute and metadata (provided that the indexes are configured correctly).

2.7 Software

After analyzing the popularity, functionality when working with spatial data, integrability with geoinformation systems for the experiment, the following DBMS were selected in each category:
• document based – MongoDB;
• graph – Neo4j;
• wide column – Cassandra;
• key-value – Redis;
• search engine – ElasticSearch.
Table 2 shows the correspondence between the used types of spatial representation of geographic objects and specific types of spatial data in the selected DBMS. Mark “Not used” means that this type of spatial objects is present in the specified DBMS, but was not used in the analyzed queries, “Not applicable” - this type of spatial objects is not implemented in the specified DBMS (but, possibly, it is implemented using additional modules or scripts).

Table 2. Data types in DBMS relative to the types of spatial representation of objects

	PostGIS	MongoDB	Neo4j	Cassandra	Redis	ElasticSearch
Point (POI)	Point	Point	Point	Point	Point	Point
Lines (roads)	LineString	LineString	LineString	LineString (not used)	Not applicable	LineString
Polygons (landuse)	Polygon	Polygon	Polygon (not used)	Not applicable	Not applicable	Polygon

For each category of DBMS, we selected those queries that PostgreSQL DBMS also allows to implement. That is, the execution time of certain spatial-attribute queries (then executed in other DBMSs) executed in PostGIS was taken as a reference point. In addition, two versions of multi-model DBMS were used [12]. The queries themselves are listed in the results tables in the third section of the article. The unit was the execution time of a query in the PostgreSQL DBMS. The results tables show a value that shows how many times faster or slower a similar query was executed in the specified NoSQL DBMS. That is, a figure less than one indicates that the query was executed slower than in a relational DBMS, more than one - respectively, vice versa. All experiments were carried out on the same hardware.

3 Results

MongoDB showed improved performance when processing the largest dataset for the Novosibirsk region, with the exception of searching for features within a polygonal region with complex geometry (table 3). In the case of this DBMS, it should be noted that it takes a lot of time to correctly (without losing or distorting the semantics of objects) loading data into the database.

Table 3. MongoDB Query Performance.

	Region	City	District
Search for objects of a certain type in a polygon	1,5	1,4	1
Search for intersections (formation of point objects) between objects of two specified types and based on the values of the semantic characteristics	1,7	1,6	1,2
Search for objects of a certain type based on the values of the semantic characteristic at a specified distance from a specified point	2,6	2,2	1,8

The specificity of the graph database allowed Neo4j to show better performance (table 4) when searching for the shortest path in the case of the largest dataset.

Table 4. Neo4j Query Performance.

	Region	City	District
Search for the nearest point object with the specified criteria by semantic characteristics	1,3	1,2	1,1
Find the shortest path (set of line features) between two specified points	1,7	1,3	1

Slower execution of queries of the Cassandra DBMS (table 5) is due to the need to work out the optimal database structure for the selected sets and set up the grouping of data in columns.

Table 5. Cassandra Query Performance.

	Region	City	District
Search for intersections (formation of point objects) between objects of two specified types and based on the values of the semantic characteristics	0,9	0,9	0,8
Search for objects of a certain type based on the values of the semantic characteristic at a specified distance from a specified point	1,4	1,1	1,1

The high speed of the Redis DBMS (table 6) is explained by the use of mechanisms for storing and processing data in RAM, but this limits the effective amount of data (for large geographic information systems) and the types of possible spatial queries.

Table 6. Redis Query Performance.

	Region	City	District
Calculating the distance between two specified points	4,3	3,8	3,3

ElasticSearch also uses the mechanisms of using RAM for storing and processing data, which explains, in general, higher rates when executing queries (table 7), but the possible queries themselves are very limited in functionality.

Table 7. ElasticSearch Query Performance.

	Region	City	District
Search for objects of a certain type in a polygon	1,4	1,4	1,3
Search for objects of a certain type based on the values of the semantic characteristic at a specified distance from a specified point	1,7	1,1	1,1

As an additional alternative, the results of queries (tables 8 and 9) were analyzed in the OrientDB and CosmosDB multi-model DBMSs. Performance is close to PostgreSQL, but

the ability to separate data across different storage models allows you to expand the possible functionality of the data warehouse.

Table 8. OrientDB Query Performance.

	Region	City	District
Calculating the distance between two specified points	1,1	1,1	0.9
Search for intersections (formation of point objects) between objects of two specified types and based on the values of the semantic characteristics	2,0	1,8	1,1
Find the shortest path (set of line features) between two specified points	0,9	0,6	0,6

Table 9. CosmosDB Query Performance.

	Region	City	District
Calculating the distance between two specified points	0,9	0,8	0.9
Search for intersections (formation of point objects) between objects of two specified types and based on the values of the semantic characteristics	0,8	0,8	1,1
Find the shortest path (set of line features) between two specified points	1,9	1,6	1,6

4 Conclusions

Based on the results of the comparative analysis, the following conclusions were formulated:

• relational DBMS, in the current version, are definitely better suited for working with spatial data in small (in terms of data volume and functionality) projects;

• each storage model has advantages for a specific use case (especially noticeable with very large amounts of data, streaming data, data heterogeneity);

• most NoSQL solutions are focused on working with web interfaces and there are difficulties in integrating them into desktop GIS;

• NoSQL DBMSs are initially better suited for distributed storage and processing, but they take time to master pipelines and set up storage;

• performance strongly depends on the setting of the DBMS, indexes (and their type), the structure of the database, regardless of the model used;

• calculations based on spatial data in a NoSQL DBMS can be faster at the expense of the accuracy of the coordinates obtained (this mode is available in a number of DBMSs).

In further research, it is planned to formulate recommendations on the use of specific data models depending on the problem being solved, certain types of indexes depending on

the spatial data used and the most frequent queries, as well as ways to increase performance in multi-model DBMS.

Acknowledgments: This research was prepared within the grant for carrying out major scientific projects in priority areas of scientific and technological development under the subprogram "Fundamental scientific research for the long-term development and ensuring the competitiveness of society and the state" of the state program of the Russian Federation "Scientific and technological development of the Russian Federation," project "Social and economic development of Asian Russia on the basis of synergy of transport accessibility, system knowledge of natural resource potential, and expanding area of interregional interactions", agreement with the Ministry of Science and Higher Education of the Russian Federation № 075-15-2020-804 (internal number 13.1902.21.0016).

References

1. Mabele Bangou Creole Passover. *Fundamentals of the geographic information database of the specially protected natural areas of the Republic of Congo*, «Proceedings of the Higher Educational Institutions. Izvestia vuzov «Geodesy and aerophotosurveying» **64** 5, 596–607, (2020)
2. K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi, F. Ismaili. *Comparison between relational and NOSQL databases*, 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 0216-0221, (2018) doi: 10.23919/MIPRO.2018.8400041.
3. W. Ali, M. U. Shafique, M. A. Majeed, A. Raza. *Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics*, Asian Journal of Research in Computer Science, **4(2)**, 1-10, (2019) doi: 10.9734/ajrcos/2019/v4i230108
4. C. Divya, K. L. Bansal. *Using the Advantages of NOSQL: A Case Study on MongoDB,* International Journal on Recent and Innovation Trends in Computing and Communication **5.2**, 90–93. (2017)
5. T. Grippa, S. Georganos, S. Zarougui, P. Bognounou, E. Diboulo, Y. Forget, M. Lennert, S. Vanhuysse, N. Mboga, E. Wolff, *Mapping Urban Land Use at Street Block Level Using OpenStreetMap, Remote Sensing Data, and Spatial Metrics*. ISPRS Int. J. Geo-Inf., **7**, 246, (2018) doi: 10.3390/ijgi7070246
6. D. Guo, E. Onstein *State-of-the-Art Geospatial Information Processing in NoSQL Databases*, ISPRS International Journal of Geo-Information, **9(5)**, 331, (2020) doi: 10.3390/ijgi9050331
7. V. Varga, K. Janosi-Rancz, B. Kálmán, Conceptual Design of Document NoSQL Database with Formal Concept Analysis, (2016)
8. A. S. Dubrovin, O. V. Ogorodnikova, E. G. Tsarkova, E. A. Andreeva, T. N. Kulikova, *Analysis and visualization in graph database management systems*, Journal of Physics: Conference Series **1902**, 012059 (2021)
9. M. Wan, C. Wu, J. Wang, Y. Qiu, L. Xin, S. Mullender, H. Mühleisen, B. Scheers, Y. Zhang, N. Nes, M. Kersten, Y. Huang, J. Deng, J. Wei, *Column store for GWAC: A high-cadence, high-density, large-scale astronomical light curve pipeline and distributed shared-nothing database*, Publications of the Astronomical Society of the Pacific, **128,** 114501, (2016)
10. S. Yamaguchi, Y. Morimitsu, *Improving Dynamic Scaling Performance of Cassandra,* IEICE Transactions on Information and Systems, **E100.D**, 4, 682-692, (2017), doi: 10.1587/transinf.2016DAP0009

11. A. Voit, A. Stankus, S. Magomedov, I. Ivanova, *Big Data Processing for Full-Text Search and Visualization with Elasticsearch*, International Journal of Advanced Computer Science and Applications(IJACSA), **8(12)**, (2017), doi: 10.14569/IJACSA.2017.081211
12. I. Holubová, S. Scherzinger, *Unlocking the potential of nextGen multi-model databases for semantic big data projects*, In Proceedings of the International Workshop on Semantic Big Data (SBD '19), 6, 1–6. (2019) doi: 10.1145/3323878.3325807