UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS DOIS VIZINHOS
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

HERON CARLOS GONÇALVES

MANIPULANDO DADOS ESPACIAIS EM BANCO DE DADOS NOSQL: UMA VISÃO CENTRADA NO USUÁRIO

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2021

HERON CARLOS GONÇALVES

MANIPULANDO DADOS ESPACIAIS EM BANCO DE DADOS NOSQL: UMA VISÃO CENTRADA NO USUÁRIO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Prof. Dr. Anderson Chaves Carniel

DOIS VIZINHOS
2021

4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

HERON CARLOS GONÇALVES

MANIPULANDO DADOS ESPACIAIS EM BANCO DE DADOS NOSQL: UMA VISÃO CENTRADA NO USUÁRIO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Data de aprovação: 30/dezembro/2021

Anderson Chaves Carniel
Doutorado
Universidade Federal de São Carlos

Ives Renê Venturini Pola
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco

Rafael Alves Paes de Oliveira
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

DOIS VIZINHOS
2021

Proceedings XXII GEOINFO, November 29 - December 02, 2021, São José dos Campos, SP, Brazil. p 167-178

Spatial Data Handling in NoSQL Databases: A User-centric View

Heron Carlos Gonçalves[1], **Anderson Chaves Carniel**[2]

[1]Federal University of Technology - Paraná
Dois Vizinhos – PR – Brazil

[2]Department of Computer Science – Federal University of São Carlos
São Carlos – SP – Brazil

heroncarlos67@gmail.com, accarniel@ufscar.br

Abstract. *Spatial data handling is a core aspect in several advanced applications due to the popularity of storage and retrieval of spatial information. NoSQL databases have been widely used to manage massive volumes of data and have added some specialized support for handling spatial data. However, it is a challenging task to analyze the spatial support provided by NoSQL databases and their possible spatial extensions. In this paper, our goal is to overcome this challenging task by presenting a systematic review of the literature. This allows us to distinguish popular NoSQL databases employed by spatial applications and compare them based on a user-centric view. That means our study helps users to select a NoSQL database according to their needs. It is possible since we correlate the characteristics of NoSQL databases and their spatial extensions with typical spatial application requirements.*

1. Introduction

Spatial data handling has been widely required by modern and advanced applications that manage geometric and geographic phenomena to improve and enrich different types of tasks, such as information retrieval, data analysis, and user experience. Specialized data types like *points*, *lines*, and *regions* are often employed by these applications in order to represent specific geometric and geographic phenomena [Güting 1994]. The instances of these data types called spatial objects are then manipulated by using *spatial operations*, such as *geometric set operations*, *topological relationships*, and *numerical operations*. Further, applications can process different types of *spatial queries* [Carniel 2020], such as *range queries* and *k-nearest neighbors queries*.

Increasingly, applications have managed large spatial datasets. This leads to the interest in specialized data management systems that provide efficient functionalities to store, handle, process, and retrieve large volumes of spatial objects. Examples of such systems are *spatial extensions* designed for parallel and distributed data processing frameworks based on Hadoop and Spark, such as GeoSpark (now called Apache Sedona) and SpatialHadoop. More spatial extensions are compared and analyzed in [Castro et al. 2018, Castro et al. 2020].

NoSQL databases also play an important role in this context [Davoudian et al. 2018]. They provide the needed foundation for storing and handling massive data by using different types of data models, such as *key-value*,

document-oriented, *column-oriented*, and *graph-oriented*. Further, they can be integrated with parallel and distributed data processing frameworks to provide *big spatial data* solutions. Hence, the focus of this paper is on NoSQL databases.

Due to the importance of spatial data handling, NoSQL databases have also incorporated some support for dealing with spatial data. Further, many approaches have been proposed in the literature to incorporate spatial data processing in these systems (see Section 3). However, the choice of the best NoSQL for a given spatial application is a complicated task since spatial applications can have different characteristics. For instance, there are applications focused on executing ad-hoc spatial queries or requiring interoperability among different architectures [Castro et al. 2020].

This motivates us to understand and compare characteristics of NoSQL databases that have some support for spatial data handling from a *user* point of view. This allows us to correspond NoSQL databases with the requirements of spatial applications. Unfortunately, there is a lack of studies on the literature that conducts this *user-centric* comparative analysis. We cite two main limitations of existing studies. First, there are studies that compare NoSQL based on performance evaluations only. Hence, they focus on the *system-centric* view. Second, several studies have a limited comparison scope in terms of the number of spatial operations and spatial extensions.

Our paper fills this gap by conducting a systematic review of the literature that permits us to identify NoSQL databases with spatial extensions and analyze these systems from the user-centric point of view. The contributions of this paper are detailed as follows.

- A systematic review of the literature that presents a comprehensive study on the spatial data handling in NoSQL databases.
- A user-centric comparison of the spatial support provided by popular NoSQL databases and their extensions proposed in the literature. We identify the main characteristics and limitations of existing studies.
- A correlation of spatial application requirements and the compared NoSQL databases. This helps users to select a NoSQL database according to their needs.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents our systematic review and compares the identified NoSQL databases and their spatial extensions. Section 4 correlate these NoSQL databases with typical requirements of spatial applications. Finally, Section 5 concludes the paper.

2. Related Work

There are several studies in the literature that conduct comparisons of NoSQL databases with support for spatial data handling. We can group them as follows: (i) studies that compare characteristics of NoSQL databases, and (ii) studies that empirically analyze the performance of NoSQL databases.

Concerning the first group, the studies discuss how NoSQL databases fulfill some spatial features required by spatial applications. In general, these studies establish a set of criteria that are used to check whether a NoSQL database satisfies them. This means that these studies conduct qualitative comparisons. However, these comparisons have a limited scope. For instance, distinct spatial operations are not taken into account and spatial extensions for NoSQL databases are not deeply compared. In [Guo and Onstein 2020],

the authors conduct an extensive qualitative comparison on NoSQL databases that attempt to indicate the most suitable NoSQL database in terms of storage and processing of spatial queries. Another example of qualitative comparison is the study in [Nassif et al. 2020], which describes the characteristics of three families of NoSQL databases and how they are used to manipulate spatial data.

As for the second group, the studies conduct extensive experimental evaluations to analyze the performance of NoSQL databases when processing different types of spatial queries. In some cases, the studies also include relational databases in their experiments as well. For instance, in the study [Baralis et al. 2017] the authors compare relational and NoSQL databases by using the Database-as-a-service model on Azure. Another example is the study in [Makris et al. 2021], which compares the performance of the MongoDB and PostgreSQL/PostGIS to process spatial queries like range and distance-based queries. The studies of this group are *system-centric* views since the internal structure and algorithms of NoSQL databases are stressed in tests focusing on evaluating the runtime performance of operations.

On the other hand, this paper aims to conduct a *user-centric* view of NoSQL databases with respect to spatial data handling. For this purpose and differently from related work, we select NoSQL databases based on a *systematic review* of literature that also considers the development of spatial extensions for them. We also do not face the same drawbacks since our comparison criteria are based on spatial features commonly required by applications. It allows us to go further and correlate how NoSQL databases fulfill the usual requirements of spatial applications. Based on them, we point out limitations and future research opportunities for NoSQL databases.

3. A Systematic Review of NoSQL databases with Support for Spatial Data

In this section, we present a systematic review that aims to pick existing and relevant studies on NoSQL databases that provide some support for spatial data handling. This is conducted by employing a well-defined and reproducible methodology (Section 3.1) that enables us to identify which NoSQL databases are usually studied and extended in the literature to deal with spatial data. For this, we consider different comparison criteria based on a user-centric view whose underlying motivation and importance are discussed in Section 3.2.

3.1. Methodology

To gather relevant studies on spatial data management in NoSQL databases, we have formulated the following search string:

("nosql" OR "nosql database" OR "nosql document" OR "nosql key-value" OR "nosql column" OR "nosql graph") AND ("spatial data" OR "geographical data" OR "GIS" OR "spatial database" OR "spatial operation")

We employed this search string in the search engines IEEE[1], Science Direct[2], Springer[3], ACM DL[4], and Google Scholar[5] from April 10, 2021, to May 14, 2021. As a

[1]https://ieeexplore.ieee.org/Xplore/home.jsp
[2]https://www.sciencedirect.com
[3]https://link.springer.com
[4]https://dl.acm.org
[5]https://scholar.google.com

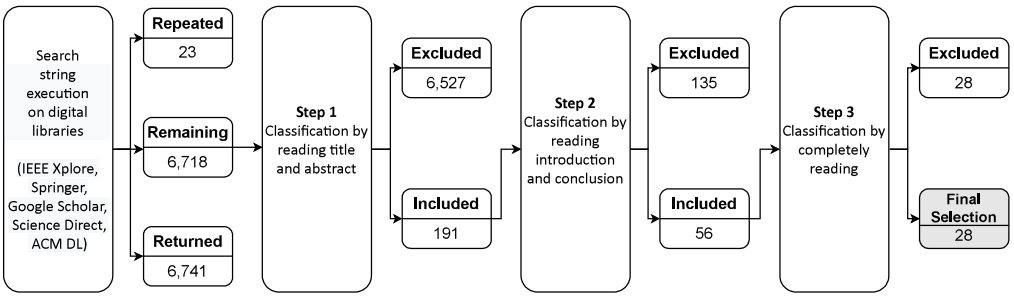

Figure 1. The steps employed by our systematic review to select relevant studies that aims to explore spatial data handling in NoSQL databases.

result, we gathered 6,741 studies. Our inclusion criteria focused on studies that (i) apply NoSQL databases in spatial applications, (ii) compare and discuss the support of spatial data management in NoSQL databases, and (iii) propose spatial extensions for NoSQL databases. We have excluded studies that only mention spatial data and do not discuss the role of spatial operations.

The main focus of this paper is to discuss and compare the support for spatial data handling in NoSQL databases. Since spatial extensions are known to provide this kind of support, we are interested in analyzing them here. To this end, we have incrementally applied three steps in our methodology, as depicted in Figure 1. In Step 1, we have classified the 6,718 studies according to the inclusion criteria by reading the title and abstract. Further, we have initiated a classification that categorize a study as an (i) *application*, (ii) a *comparison*, or (iii) an *extension*. Note that they match our inclusion criteria and that one study may belong to more than one category at the same time. In Step 2, we have read the introduction and conclusion to refine our classification and excluded those studies that either do not belong to a category or refer only to application descriptions. Step 3 aimed to select only spatial extensions or novel NoSQL databases focused on spatial data. It was made by completely reading the research paper and excluding studies that mainly conduct experimental evaluations of NoSQL databases or provide a discussion on some characteristics of these databases (as discussed in Section 2). As a result, we obtained 28 studies that either propose extensions for NoSQL databases or introduce novel NoSQL databases with spatial support.

3.2. Comparing NoSQL databases: A User-centric View

In this section, we compare the spatial support provided by popular NoSQL databases and how studies in the literature have extended them to deal with spatial data. Our systematic review allowed us to identify the six most popular NoSQL databases employed by the studies. The popularity was measured by counting the number of times that a NoSQL database was employed by the studies obtained in the second step of our systematic review. Here, we have excluded the NoSQL databases mentioned less than or equal to 2 times. Section 3.2.1 provides an overview of these popular NoSQL databases, while Section 3.2.3 compares their available spatial extensions.

Table 1. An overview of the NoSQL databases with some spatial support.

NoSQL	Data model	Latest version	Has native spatial support?	Available spatial extensions in the literature
Redis	key-value	6.2.4	✓	
MongoDB	document	4.4.5	✓	[Xiang et al. 2016]
CouchDB	document	3.1.1	✓	
Cassandra	column	3.11.10		[Wei et al. 2014, Ben Brahim et al. 2016]
HBase	column	2.3.4		[Van and Takasu 2015, Zhang et al. 2015, Zhang et al. 2016, Wang et al. 2017, Kokotinis et al. 2017, Jo and Jung 2017, Jo and Jung 2018, Zhang et al. 2018, Zheng et al. 2019]
Neo4j	graph	4.3.2	✓	

3.2.1. General Characteristics

Table 1 presents an overview of popular NoSQL databases identified by our systematic review. They are: (i) Redis, (ii) MongoDB, (iii) Apache CouchDB, (iv) Cassandra, (v) HBase, and (vi) Neo4j. This overview presents (i) the underlying NoSQL data model, (ii) the latest version of the NoSQL, (iii) whether the NoSQL has native support for spatial data handling, and (iv) the list of available extensions proposed in the literature. Each item is described as follows.

NoSQL data models. There are four main data models of NoSQL databases: (i) key-value stores, (ii) document-oriented databases, (iii) (wide-)column stores, and (iv) graph databases. Key-value stores are simple data models that represent information by using pairs where each pair consists of a value associated with a key. This principle is extended by document-oriented databases, which store collections of key-value pairs that are usually represented by JavaScript Object Notation (JSON) objects. Column stores deal with tables, rows, and columns that can be dynamically structured as needed. Finally, graph databases represent information by using nodes and the relationship between nodes by using edges. Table 1 shows that there is some native spatial support in these data models, including the interest in extending them in research papers.

Latest version. The version control of NoSQL databases indicates the evolution of features provided by them. Some of these systems have added some native spatial support recently only. For instance, Neo4j has introduced 2D and 3D points in its version 3.4. In this paper, our comparison considers the versions indicated in Table 1.

Native spatial support. Similarly to the support for alphanumerical data, NoSQL databases can provide support for data handling in their internal structures. This means that applications can store and manage spatial objects in such databases without third-party extensions. This kind of native support is not provided by Cassandra and HBase,

Table 2. Comparing the native support of the NoSQL databases for spatial data handling. Here, we do not consider third-party extensions proposed in the literature (see Table 3).

NoSQL	Spatial data types	Representation of spatial objects	Topological relationships	Distance-based operations	Spatial indexing methods
Redis	simple points	GeoJSON	✓[a]	✓	sorted sets with geohash
MongoDB	simple and complex points, lines, and regions		intersect, within	✓	2d index based on geohash, 2dsphere index
CouchDB	simple points			✓	index on two numeric fields
Neo4j	simple points			✓	space filling curves over an underlying generalized B^+-tree

[a] It offers functions for processing some specific types of spatial queries based on topological relationships.

which are NoSQL databases based on column stores. A comparison of the native spatial support of NoSQL databases is conducted in Section 3.2.2.

Available spatial extensions in the literature. Since Cassandra and HBase do not provide native support for spatial data handling, the majority of available spatial extensions proposed in the literature are for these NoSQL databases. Our systematic review also reveals that there is an increasing focus on extending HBase. The main reason is that this is often used as the underlying storage of Hadoop and Spark systems. Their focus can be different in terms of storage or spatial query processing by using spatial index structures. In this sense, we compare them in Section 3.2.3.

3.2.2. Native Spatial Support in NoSQL Databases

Table 2 compares the native spatial support of the NoSQL databases Redis, MongoDB, Apache CouchDB, and Neo4j. The comparison criteria consider common spatial representations and operations required by spatial database applications [Güting 1994]. We check whether the NoSQL database provides (i) spatial data types, (ii) representations of spatial objects, (iii) spatial operations with a focus on topological relationships and distance-based operations, and (iv) spatial indexing methods.

Spatial data types. Here, we list the spatial data types of the NoSQL databases that are available to represent spatial information by using geometric data types like points, lines, and regions (polygons). Spatial data types can be simple or complex. The compared NoSQL databases provide support for simple points, which means that they are able to represent single instances of spatial information by using latitude and longitude coordinates. However, the creation of lines and regions can lead to complex structures. For instance, regions can be formed by relating nodes storing point objects in Neo4j. The native support for complex objects, including lines and regions, is provided by MongoDB.

It enables us to represent a large variety of spatial information.

Representation of spatial objects. The instances of spatial data types can have different types of textual and binary representations. For instance, the Well-known text (WKT) and Well-known binary (WKB) are specified by [OGC 2011] to represent a vector geometry object in a textual and binary format, respectively. Applications can use these formats to load, transfer, and visualize spatial objects. Redis, CouchDB, and Neo4j do not have a specific format to represent spatial objects since they only handle simple points. Hence, they use their default visualization and loading methods to handle points (e.g., based on CSV files). On the other hand, MongoDB employs GeoJSON, which is a JSON-variant representation for spatial objects. Its binary format can be stored as Binary JSON (BSON) objects, which allows MongoDB to manage internal structures efficiently.

Spatial operations. Commonly, spatial objects are handled, manipulated, and retrieved by using different types of spatial operations. There are classes of operations commonly provided by spatial databases and GIS [Güting 1994]. The first one is related to topological relationships, which express the particular relative position of two spatial objects. The definition of topological relationships is widely studied in the literature [Egenhofer and Franzosa 1991, Egenhofer and Herring 1994, Schneider and Behr 2006] since they are used as conditions in spatial queries [Carniel 2020]. A common spatial query is the range query, which returns all spatial objects intersecting a search object with a particular shape (e.g., circle or rectangle). Unfortunately, the compared NoSQL databases provide very limited support for them. This means that they do not enable us to process ad-hoc spatial queries. While Redis offers some functions to process spatial queries with notions of topological relationships, MongoDB has functions that implement two topological relationships (i.e., *within* and *intersect*). The second class refers to distance-based operations, which can be deployed to implement the k-nearest neighbors (kNN) query. Given a set of spatial objects and a search object, this type of query returns the k closest spatial objects to the search object. All studied NoSQL databases provide this kind of support. Other classes of operations include numerical operations (e.g., area, length), geometric set operations (e.g., union, intersection, difference), and general geometric operations (e.g., convex hull). The compared NoSQL databases do not provide support for these operations, limiting their applicability in spatial applications.

Spatial indexing methods. The processing of spatial queries is often optimized by employing spatial index structures. Such structures aim to reduce the search space by avoiding access to spatial objects that certainly do not belong to the final answer of the spatial query. Examples of spatial index structures include the R-tree and its variants. Spatial indexing is widely studied in the literature (see [Gaede and Günther 1998] for a survey) and specific implementations are provided for different types of systems (e.g., [Carniel et al. 2020]). The compared NoSQL databases provide some structures to improve the performance of spatial queries. In general, a common strategy is to employ geohash, which represents a spatial object by using alphanumerical data. This representation is based on the subdivision of space in buckets so that it is possible to apply space-filling curves like the Hilbert or Z-order curves or well-known indexing methods like the B+-tree. It is interesting to note that classical hierarchical tree structures like the R-tree and its variants for spatial data are not considered by these NoSQL databases.

Table 3. Comparing existing spatial extensions for popular NoSQL databases.

NoSQL	Extension	Storage of spatial objects	Types of spatial queries	New spatial indexing methods
MongoDB	[Xiang et al. 2016]	-	range queries	Flattened R-tree
Cassandra	[Wei et al. 2014]	points	range and kNN queries	KR^+-index
Cassandra	[Ben Brahim et al. 2016]	geohash	based on numeric range queries	-
HBase	[Zhang et al. 2015]	WKT, WKB, Shapefiles	range queries	based on grids
HBase	[Zhang et al. 2016]	-	range and kNN queries	based on the Hilbert curve
HBase	[Wang et al. 2017]	polygons as column families with WKT	range queries	based on Z-order curve
HBase	[Zheng et al. 2019]	rectangles and geohash	range and kNN queries	based on space filling curves

3.2.3. Available Spatial Extensions

In our systematic review, we have identified 28 available spatial extensions for NoSQL databases, which also include novel systems. In this section, we discuss the spatial extensions of popular NoSQL databases only (i.e., Section 3.2.1), resulting in 12 extensions. Most of them are focused on providing novel spatial indexing methods to improve spatial query processing. Since the goal of this paper is to analyze the spatial data handling in NoSQL databases, we consider only those approaches that distinguish themselves. They are shown and compared in Table 3, which takes into account the (i) storage of spatial objects, (ii) types of spatial queries, and (iii) proposed spatial indexing methods.

Storage of spatial objects. The spatial extensions differ in how to represent and store the spatial information in their corresponding NoSQL database. We can identify three main approaches. The first approach is to simply deal with the geohashes of spatial objects since geohash is an alphanumerical representation of a complex object. In this case, the extensions [Wei et al. 2014, Zheng et al. 2019] do not need a specialized storage method but sophisticated algorithms for processing spatial queries. The second approach is to deal with points only since coordinate pairs can be stored separately [Wei et al. 2014]. Finally, the third approach refers to the use of textual or binary representations of spatial objects, such as WKT, WKB, or shapefiles. The extensions based on this approach [Zhang et al. 2015, Wang et al. 2017] allow users to employ different spatial data types in their applications. Other extensions [Xiang et al. 2016, Zhang et al. 2016] do not focus on the storage of spatial objects since their goal is to provide solutions for improving the performance of spatial queries.

Types of spatial queries. Although there are several different types of spatial queries, spatial extensions make efforts to efficiently process range and kNN queries only. The main reason is that they are common types of spatial queries employed in experimental

Table 4. Checking how NoSQL databases and their spatial extension fulfill the spatial application requirements.

NoSQL	Type of spatial support	1	2	3	4	5	6
Redis	native	partial				✓	✓
MongoDB	native + extension	partial	✓	✓	partial	✓	✓
CouchDB	native					✓	✓
Cassandra	with extensions					✓	✓
HBase	with extensions		✓	partial		✓	✓
Neo4j	native	partial		partial	partial	✓	✓

evaluations conducted in the literature [Carniel 2020]. We highlight the Cassandra extension proposed in [Ben Brahim et al. 2016] since it extends range queries to propose other types of spatial queries like *around me* and *in my path*. Such queries are based on numeric filters in the coordinate pairs coded by the geohash of spatial objects.

New spatial indexing methods. Almost all studies propose spatial indexing methods that are implemented in the corresponding NoSQL database. This means that their main focus is on quickly processing specific types of queries by using spatial index structures. Space-filling curves such as the Z-order and Hilbert curves are widely employed in this context [Zhang et al. 2016, Wang et al. 2017, Zheng et al. 2019] since they attempt to define an ordering of access to spatial objects. This aspect is interesting in NoSQL databases due to the availability of indexing methods for alphanumerical data that are based on sorting properties (e.g., the B-tree). Other extensions strive to adapt well-known spatial index structures to a particular NoSQL database. For instance, in [Xiang et al. 2016], the classical R-tree is flattened into a collection of documents in MongoDB so that it is possible to insert, delete, and retrieve spatial objects by using these documents. In the compared studies, only the study in [Ben Brahim et al. 2016] does not deal with spatial indexing methods because it was interested in proposing new types of spatial queries for Cassandra.

4. Correlating Spatial Application Requirements with NoSQL databases

Table 4 checks whether a NoSQL database with its extension fulfill six different types of requirements commonly required by spatial applications defined in [Castro et al. 2018, Castro et al. 2020]. These requirements are viewed as a set of *guidelines* that can help users to choose the NoSQL database that better fits their needs. This table is fulfilled by considering the comparisons and discussions previously reported in this paper. The requirements are detailed as follows.

1. Focus on executing ad-hoc spatial queries. This guideline refers to the design of spatial queries without a specific format. Hence, a NoSQL database should have a broad collection of spatial operations or at least have the possibility of including more operations by using existing functionalities. Unfortunately, the compared NoSQL databases do not offer some common types of spatial operations, such as geometric set operations and topological relationships based on the 9-intersection model [Schneider and Behr 2006]. Redis, MongoDB, and Neo4j partially fulfill this guideline since they provide some spatial operations.

2. Focus on the interoperability among different systems. This guideline considers that the spatial application usually requires communication with other systems. In this case, a NoSQL database that can represent and store spatial object using well-known textual or binary formats fulfill this requirement. MongoDB with its native support for GeoJSON and the spatial extensions for HBase fulfill this guideline.

3. Focus on characteristics based on well-known standards. This guideline refers to the common requirement of using well-established concepts, techniques, and operations in spatial applications. This aspect is relevant for the development based on standards that are usually employed in the literature and industry, such as the OGC standards [OGC 2011]. Only MongoDB makes use of such standards when defining their spatial data types. Other NoSQL databases partially adopt some standards. For instance, Neo4j employs well-known coordinate reference systems in their underlying storage.

4. Focus on spatial data visualization. This guideline relates to the intrinsic need for graphically visualizing spatial objects in spatial applications. NoSQL databases do not focus on visualization but on providing storage and access methods for spatial data. However, NoSQL databases can be integrated with other systems to enrich the analysis of spatial queries. In the documentation of Neo4j and MongoDB, such perspectives are mentioned and explored. Hence, we indicate that they partially fulfill this guideline.

5. Focus on efficiently processing spatial queries. This guideline checks whether the NoSQL database provides mechanisms to reduce the elapsed time required to process spatial queries. Our systematic review was unable to find a complete system-centric comparison of the compared NoSQL databases. However, we consider that there are several performance evaluations of these databases in the literature that focus on improving this aspect (as reported by the spatial extensions). Hence, we consider that the compared NoSQL databases fulfill this requirement to deal with specific types of spatial queries, such as range and kNN queries.

6. Focus on providing extensibility. This guideline relates to the possibility of extending a NoSQL database to improve its management of spatial data. In this paper, we have identified spatial extensions proposed in the literature indicating the efforts of researchers in this topic. Further, there are other third-party extensions, such as those available in GitHub (which goes beyond the scope of this paper). Hence, we have marked that the compared NoSQL databases can be extended in some way to include new features.

5. Conclusions and Future Work

In this paper, we have conducted a systematic review of the literature on spatial data handling in NoSQL databases. This allowed us to compare, analyze, and discuss the spatial support provided by NoSQL databases popularly employed by spatial applications. The compared NoSQL databases included Redis, MongoDB, CouchDB, Cassandra, HBase, and Neo4j. Among them, MongoDB distinguishes itself by providing more spatial data types and spatial operations than other NoSQL databases.

Our systematic review also permitted us to identify existing spatial extensions for these NoSQL databases. Such extensions are usually proposed for NoSQL databases without native spatial support. It demonstrates that there is an increasing interest in providing and improving spatial data handling in different types of NoSQL data models.

We also identified how the spatial data support provided by the compared NoSQL databases and their extensions fulfill common requirements of spatial applications. It is applicable for users that need to understand and pick a NoSQL database that best fits their needs. In addition, we have indicated their problems and limitations that lead to the identification of open research topics in this area.

Future work topics include the following items. First, we aim to extend this work by searching for spatial extensions available in the public repositories of GitHub. For instance, GeoCouch[6] is a spatial extension for Couchbase and CouchDB. Second, we aim to analyze research papers mentioned by the spatial extensions in order to comprehend their use and advances. Finally, future studies can propose solutions to solve discussed problems and limitations.

References

Baralis, E., Dalla Valle, A., Garza, P., Rossi, C., and Scullino, F. (2017). SQL versus NoSQL databases for geospatial applications. In *IEEE Int. Conf. on Big Data*, pages 3388–3397.

Ben Brahim, M., Drira, W., Filali, F., and Hamdi, N. (2016). Spatial data extension for cassandra NoSQL database. *Journal of Big Data*, 3(1):1–16.

Carniel, A. C. (2020). Spatial information retrieval in digital ecosystems: A comprehensive survey. In *Int. Conf. on Management of Digital EcoSystems*, pages 10–17.

Carniel, A. C., Ciferri, R. R., and Ciferri, C. D. A. (2020). FESTIval: A versatile framework for conducting experimental evaluations of spatial indices. *MethodsX*, 7:1–19.

Castro, J. P. C., Carniel, A. C., and Ciferri, C. D. A. (2018). A user-centric view of distributed spatial data management systems. In *Brazilian Symp. on GeoInformatics*, pages 80–91.

Castro, J. P. C., Carniel, A. C., and Ciferri, C. D. A. (2020). Analyzing spatial analytics systems based on hadoop and spark: A user perspective. *Software: Practice and Experience*, 50(12):2121–2144.

Davoudian, A., Chen, L., and Liu, M. (2018). A survey on nosql stores. *ACM Comput. Surveys*, 51(2).

Egenhofer, M. J. and Franzosa, R. D. (1991). Point-set topological spatial relations. *Int. Journal of Geographical Information Systems*, 5(2):161–174.

Egenhofer, M. J. and Herring, J. R. (1994). Categorizing binary topological relations between regions, lines and points in geographic databases. In *The 9-Intersection: Formalism and Its Use for Natural-Language Spatial Predicates*.

Gaede, V. and Günther, O. (1998). Multidimensional access methods. *ACM Comput. Surveys*, 30(2):170–231.

Guo, D. and Onstein, E. (2020). State-of-the-art geospatial information processing in nosql databases. *ISPRS Int. Journal of Geo-Information*, 9(5).

Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399.

[6] `https://github.com/couchbase/geocouch`

Jo, B. and Jung, S. (2017). Quadrant-based MBR-tree indexing technique for range query over HBase. In *Int. Conf. on Emerging Databases*, pages 14–24.

Jo, B. and Jung, S. (2018). Quadrant-based minimum bounding rectangle-tree indexing method for similarity queries over big spatial data in HBase. *Sensors*, 18(9):1–18.

Kokotinis, I., Kendea, M., Nodarakis, N., Rapti, A., Sioutas, S., Tsakalidis, A. K., Tsolis, D., and Panagis, Y. (2017). NSM-tree: Efficient indexing on top of NoSQL databases. In *Int. Workshop of Algorithmic Aspects of Cloud Computing*, pages 3–14.

Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., and Anagnostopoulos, D. (2021). MongoDB vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica*, 25(2):243–268.

Nassif, E. H., Hicham, H., Yaagoubi, R., and Badir, H. (2020). Assessing nosql approaches for spatial big data management. In *Int. Conf. on Artificial Intelligence and Symbolic Computation*, pages 49–58.

OGC (2011). OpenGIS implementation standard for geographic information - simple feature access - part 1: Common architecture. Open Geospatial Consortium. `https://www.ogc.org/standards/sfa`.

Schneider, M. and Behr, T. (2006). Topological relationships between complex spatial objects. *ACM Trans. on Database Systems*, 31(1):39–81.

Van, L. H. and Takasu, A. (2015). An efficient distributed index for geospatial databases. In *Int. Conf. on Database and Expert Systems Applications*, pages 28–42.

Wang, Y., Li, C., Li, M., and Liu, Z. (2017). HBase storage schemas for massive spatial vector data. *Cluster Computing*, 20:3657–3666.

Wei, L.-Y., Hsu, Y.-T., Peng, W. C., and Lee, W.-C. (2014). Indexing spatial data in cloud data managements. *Pervasive and Mobile Computing*, 15:48–61.

Xiang, L., Huang, J., Shao, X., and Wang, D. (2016). A MongoDB-based management of planar spatial data with a flattened R-tree. *ISPRS International Journal of GeoInformation*, 5(7):1–17.

Zhang, C., Chen, X., Feng, X., and Ge, B. (2016). Storing and querying semi-structured spatio-temporal data in HBase. In *Int. Conf. on Web-Age Information Management*, pages 303–314.

Zhang, C., Zhu, L., Long, J., Lin, S., Yang, Z., and Huang, W. (2018). A hybrid index model for efficient spatio-temporal search in HBase. In *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 108–120.

Zhang, N., Zheng, G., Chen, H., Chen, J., and Chen, X. (2015). HBaseSpatial: A scalable spatial data storage based on HBase. In *IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications*, pages 644–651.

Zheng, K., Zheng, K., Fang, F., Zhang, M., Li, Q., Wang, Y., and Zhao, W. (2019). An extra spatial hierarchical schema in key-value store. *Cluster Computing*, 22:6483–6497.