Geographic Data Modeling for NoSQL Document-Oriented Databases

Wagner Boaventura Filho, Harley Vera Olivera, Maristela Holanda, Aleteia A. Favacho

Department of Computer Science
University of Brasília
Brasília, Brazil
{wagnerbf, harleyve}@gmail.com, {mholanda, aleteia}@cic.unb.br

***Abstract*—The integration of Geographical Information Systems (GIS) with relational databases (RDBMS) and the search for interoperable standards among geospatial systems have been featured focuses on the agendas of academia, industry, and the spatial data user community in for some time now. Subsequently, in database technologies, some of the new issues increasingly debated are non-conventional applications, including NoSQL (Not only SQL) databases, which were initially created in response to the needs for better scalability, lower latency and higher flexibility in an era of bigdata and cloud computing. These non-functional aspects, which are very common in the treatment of spatial data, are the main reason for using NoSQL database. However, currently there are no systematic studies on the data modeling for NoSQL databases, especially the document-oriented ones. Therefore, this article proposes a NoSQL data modeling standard, introducing modeling techniques that can be used on document-oriented databases, including geographical features. In addition, to validate the proposed model, a study case was implemented using geographic information on changes in the land use of Brazilian biomes.**

Keywords–NoSQL; GIS; Data modeling; Document-oriented database.

I. INTRODUCTION

Huge amounts of spatial data are produced daily. They are generated by satellites, telescopes, sensor networks, and provide information that is growing exponentially. The management of this data is currently performed in most cases by relational databases with spatial extensions that provide centralized control of data, redundancy control and elimination of inconsistencies [1]; but, some of these factors restrict the use of alternative database models. Consequently, certain limiting factors have led to alternative models of databases that are used in these scenarios. Primarily, motivated by the issue of system scalability, a new generation of databases, known as NoSQL, is gaining strength and space both in information systems. The NoSQL databases emerged in the mid-90s, from a database solution that did not provide an SQL interface. Later, the term came to represent solutions that promoted an alternative to the Relational Model, becoming an abbreviation for Not Only SQL.

The purpose, therefore, of NoSQL solutions is not to replace the Relational Model as a whole, but only in cases in which there is a need for scalability and bigdata. In the recent years, a variety of NoSQL databases has been developed mainly by practitioners looking to fit their specific requirements regarding scalability performance, maintenance and feature-set. Subsequently, there have been various approaches to classify NoSQL databases, each with different categories and subcategories, such as key-value stores, column-oriented and graph databases, oriented-document. MongoDB [2], Neo4j [3], Cassandra [4] and HBase [5] are examples of NoSQL databases. This article only applies to NoSQL document-oriented databases, because of the heterogeneous characteristics of the each NoSQL database classification.

Nonetheless, data modeling still has an important role to play in NoSQL environments. The data modeling process [1] involves the creation of a diagram that represents the meaning of the data and the relationship between the data elements. Thus, ”understanding” is a fundamental aspect of data modeling [6], and a pattern for this kind of representation has few contributions for NoSQL databases.

Addressing this issue, this article proposes a standard for NoSQL data modeling. This proposal uses NoSQL document-oriented databases, including geographic data, aiming to introduce modeling techniques that can be used on databases with document features.

The remainder of the paper is organized as follows: Section II presents related works. Section III explores the concepts of modeling for NoSQL databases based on documents, introducing the different types of relationships and associations. Section IV shows the proposal model in the context of NoSQL databases based on documents. Section V presents the study case to validate the proposal model. Finally in Section VI, we present the conclusion of the research and future works.

II. RELATED WORKS

Specifically for geographic data, the current literature suggests a standard of data modeling for relational databases, for example: OMT-G, the acronym for Object Modeling Technique for Geographic Applications [7], GMOD, an environment for modeling and design of geographic applications [8], GISER, a data model for geographic information systems [9], GeoOOA, an object-oriented analysis for geographic information systems [10], MODUL-R, a data model for design spatio-temporal databases [11] and OMT EXT, an explicit representation of data that depends on topological relationships and control over data consistency [12].

Katsov [13] presents a study of techniques and patterns for data modeling using the different categories of NoSQL databases. However, the approach is generic and does not define a specific modeling engine to each database.

Arora and Aggarwal [14] propose a data modeling, but restricted to MongoDB document database, describing a UML Diagram Class format to represent the documents.

Similarly, Banker [15] provides some ideas of data modeling, but limited to MongoDB database and always referring to JSON [16] format as a modeling solution.

 ISBN: 978-1-61208-383-4

As one can see, none of these approaches refers to spatial data, and nor do they present a graphical model for use in any NoSQL document-oriented database.

III. Data Modeling for Document-Oriented Database

An important step in database implementation is the data modeling, because it facilitates the understanding of the project through key features that can prevent programming and operation errors. For relational databases, the data modeling uses the Entity-Relationship Model [1]. For NoSQL, it depends on the database category. The focus of this article is NoSQL document-oriented databases, where the data format of these documents can be JSON, BSON, or XML [17].

Basically, the documents are stored in collections. A parallel is made with relational databases, the equivalent for a collection is the record (tuple) and for a document it is the relation (table). Documents can store completely different sets of attributes, and can be mapped directly to a file format that can be easily manipulated by a programming language. However, it is difficult to abstract the modeling of documents for the entity relationship model [6].

A. Modeling Paradigm for document-oriented database

The relational model designed for SQL has some important features such as integrity, consistency, type validation, transactional guarantees, schemes and referential integrity. However, some applications do not need all of these features. The elimination of these resources has an important influence on the performance and scalability of data storage, bringing new meaning to data modeling.

Document-oriented databases have some significant improvements, e.g., index management by the database itself, flexible layouts and advanced indexed search engines [13]. By associating these improvements (some being denormalization and aggregation) to the basic principles of data modeling in NoSQL, it is possible to identify some generic modeling standards associated to document-oriented databases. Analyzing the documentation of the main document-oriented databases, MongoDB [18] and CouchDB [19], similar representations of data mapping relationships can be found: **References** and **Embedded Documents**, featuring a structure which allows associating a document to another, retaining the advantage of specific performance needs and data recovery standards.

B. References Relationship

This type of relationship stores the data by including links or references, literally, from one document to another. Applications can resolve these references to access the related data in the structure of the document itself [18]. Figure 1 shows three documents for Geographical Location, Municipality and Coordinates in a reference relationship.

C. Embedded Documents

This type of relationship stores in a single document structure, where the embedded documents are disposed in a field or an array. These denormalized data models allow data manipulation in a single database transaction [18]. Figure 2 shows a document of a Land Treatment with a Management embedded document.

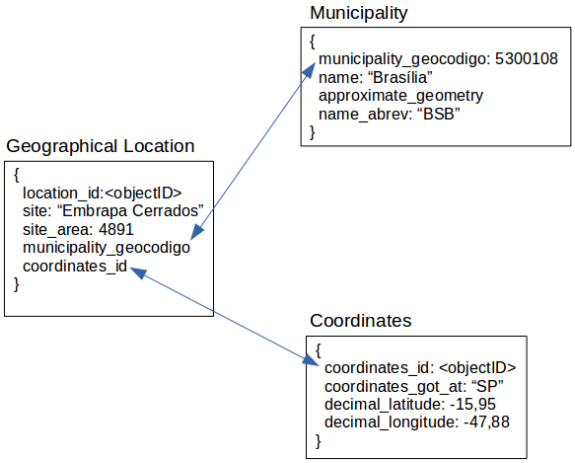

Figure 1. Example of documents referenced

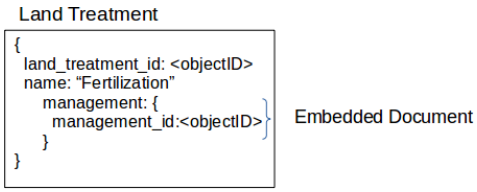

Figure 2. Example of embedded documents

IV. Proposal for Document-Oriented Databases Viewing

Unlike the traditional relational databases that have a simple form in the disposition in rows and columns, a document-oriented database stores information in text format, which consists of collections of records organized in the key-value concept, ie, for each value represented a name (or label) is assigned, which describes its meaning. This storage model is known as JSON object, and the objects are composed of multiple name/value pairs for arrays, and other objects.

In this scenario, the number of objects (or documents) in a database increases the abstraction complexity of the logical relationship between the stored information, especially when objects have references to other objects. Currently, there is a lack of solutions to conceptually represent those associated with a NoSQL document-oriented database. As described in [14], there is no standard to represent this kind of object modeling, and several different manners of modeling may arise, depending on each data administrator's understanding, which makes learning difficult for those who need to read the database model.

Therefore, this section proposes a standard for document-oriented database viewing, including for geographical data references.

Our proposal has some properties, considering the conceptual representation modeling type, such as:

- Ensuring a single way of modeling for the several NoSQL document-oriented databases.
- Simplifying and facilitating the understanding of a document-oriented database through its conceptual model, leveraging the abstraction ability and making the correct decisions about the data storage.

 ISBN: 978-1-61208-383-4

- Providing an accurate, unambiguous and concise pattern, so that database administrators have substantial gains in abstraction, understanding.
- Presenting different types of relationships between documents are defined as References and Embedded documents.
- Assisting the recognition and arrangement of the objects, as well as its features and relationships with other objects.

The following subsections present the concepts and graphing to build a conceptual model for NoSQL document-oriented databases.

A. Assumptions

Before starting the discussion about the approach of each type of the conceptual modeling representation, it is important to highlight some basic concepts about objects and relationships in a document-oriented database:

- An object (or document) describes a collection of records that have their properties organized in a key-value structure.
- Information contained in an object is described by the identifier (key) and the value associated with the key.
- Different types of relationships between documents are defined as **References** and **Embedded Documents**.
- Because NoSQL is a non-relational data database, the concepts of normalization, do not apply.
- In contrast, some concepts of relationships between objects are similar to ER modeling, such as cardinality (one-to-one, one-to-many, many-to-many).

B. Basic Visual Elements

The proposed solution for a conceptual modeling to the NoSQL document-oriented databases has two basic concepts: *Document* and *Collections*.

As noted previously, a document is usually represented by the structure of a JSON object, and as many fields as needed may be added to the document. For this proposed solution, a document is represented by Figure 3.

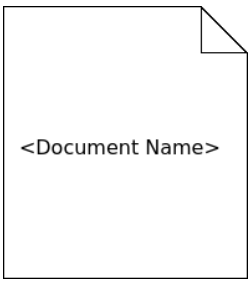

Figure 3. Graphical representation of a Document

It is also possible to store and organize the data as a collection of documents with fields and default values for each record. In this context, a collection of documents will be represented by Figure 4.

The following section presents the definitions of the relationship types and degrees for the objects features.

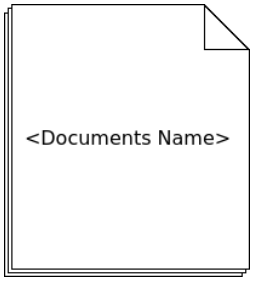

Figure 4. Graphical representation of a Collection

C. Embedded Documents 1..1

This section proposes a model that represents the one-to-one relationship for documents embedded in another document. In this case, the proposal is to use the representation of an individual document within another element that represents a Document. In Figure 5, cardinality is also suggested to specify the one-to-one relationship type.

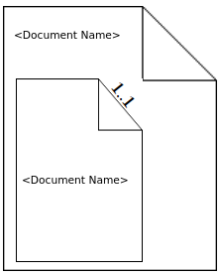

Figure 5. One-to-one relationship for embedded documents

D. Embedded Documents 1..N

A one-to-many relationship in embedded documents is represented by the Figure 6. This is the case when the notation to represent the cardinality is the same used in UML [20] and is placed in the upper right corner of the embedded documents. According to the cardinality one-to-many the larger document has embedded multiple documents within it.

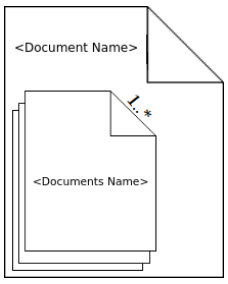

Figure 6. One-to-many relationship for embedded documents

E. References 1..1

A document can reference another, and in this case, one must use an arrow directed to the referenced document, as shown in Figure 7. One can see that the directed arrow makes the left document references to the right document. Furthermore, the cardinality of the relationship should be

specified above the arrow. The notation of cardinality is based on UML [20].

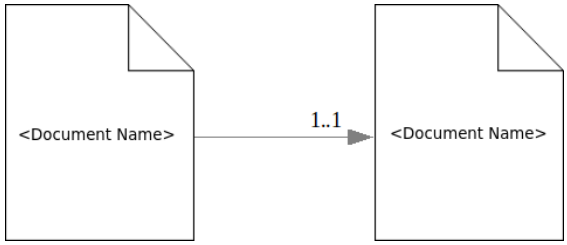

Figure 7. One-to-one relationship for documents referenced

F. References 1..N

In NoSQL, a document can reference multiple documents. To represent this relationship one should use an arrow directed to the referenced documents, as shown in Figure 8. The left document references multiple documents on the right side, by the directed arrow. Furthermore, the cardinality of the relationship is represented by the notation "1..*" as in UML [20].

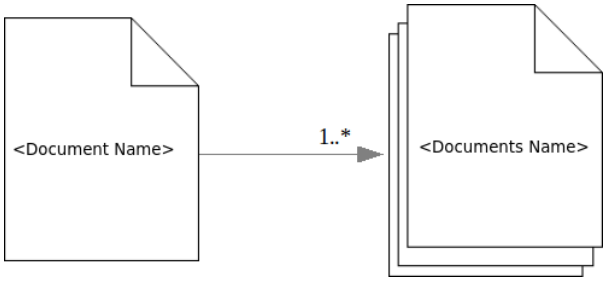

Figure 8. One-to-many relationship for documents referenced

G. Geographical Primitives

With the goal of increasing the capacity of semantic representation of geographic space, the OMT-G [7] model provides primitives to model the geometry of geographic data, so it is possible to georeference documents using these primitives, as shown in Figure 9. A conventional document differs from a georeferenced document precisely by using primitive geographics in the top right corner of the document. In this case, a point is represented by a star, a line is represented by a solid line and a polygon is represented by a square.

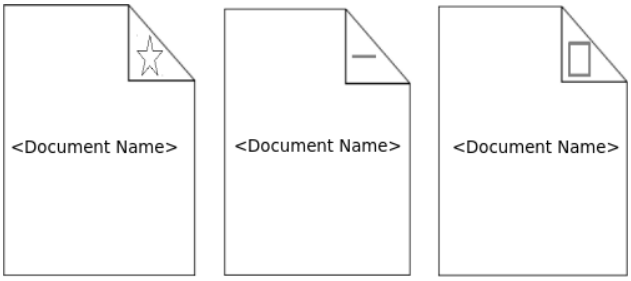

Figure 9. Geographical primitives in documents

Association of the georeference documents is represented by dashed lines [7], as shown in Figure 10. On the other hand, the association of a conventional (not-georeferenced) document with a geo-referenced document is displayed by solid lines, as shown in Figure 11. In both cases, we decided to use the same symbolism for relationship, dashed and solid lines, similar to the OMT-G model [7].

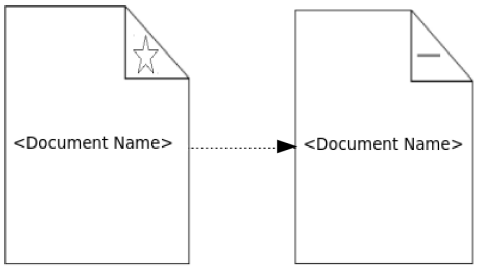

Figure 10. Spatial association

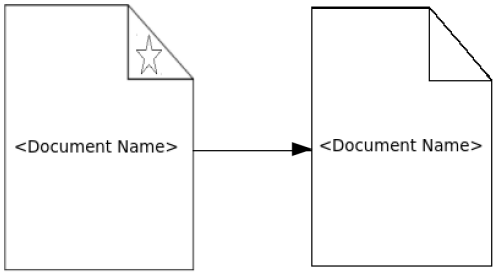

Figure 11. Simples association

Finally, Figure 12 shows the relationship between two embedded georeferenced documents.

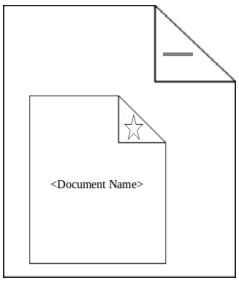

Figure 12. Embedded Document With Georeferences

V. Case Study

An example of data modeling is presented in this section to illustrate the proposed model. This conceptual model is based on the model presented in [21], which corresponds to a collection of information about changes in land use over the Brazilian biomes represented by points (latitude, longitude) where the data was collected.

Thus, the document "Location" is the document containing georeferenced data such as latitude, longitude and the collection site. A star is used as a geographical point primitive for this document in the upper right corner of the document representing the collection points, as illustrated in Figure 13.

The document "Actual type use" describes the type of real land use (cropland, pasture, secondary forest, etc). While the

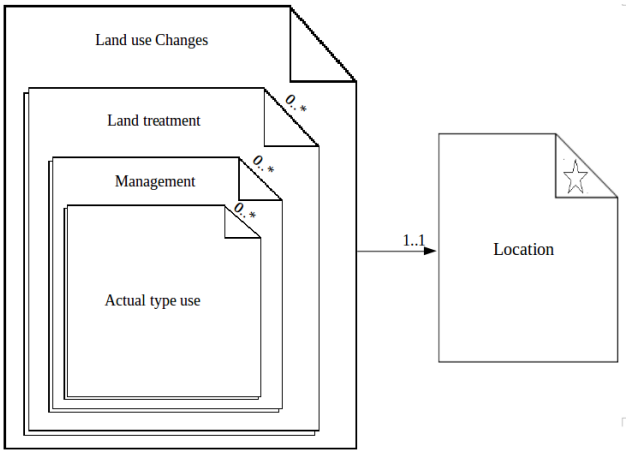

Figure 13. Modeling use case

format that is a binary representation of a JSON file. In MongoDB, data is represented as pairs of name-value elements. A field-value's pair consists of the name of the field and its value, and is always separated by a ":" character. In the Figure 14, the document "Location" is represented as a BSON's format document.

```
Document Location
{
        _id : 12003,
        Site : "Embrapa Cerrados",
        site_area : 4891,
        municipality_geocodigo: 53,
        coordinates_got_at: "SP",
        decimal_latitude: -15.95,
        decimal_longitude: -47.88
}
```

Figure 14. BSON format representation of document Location

document "Management" describes the type of land management carried out (burning, fertilizing, planting). The document "Land treatment" describes information relating to the treatment of land (accidental fire, prescribed fire, not burned, etc.). Finally, the document "Land Use Changes" presents information as species involved, predominant vegetation, removal technique.

Figure 13 shows the relationship between the documents "management" and "Actual type use" in which the cardinality is zero-to-many, which means that a document "Management" may contain zero or more documents of the type "Actual type use". Similarly, Figure 13 shows the relationship between the documents "Land treatment" and "Management" whose cardinality is also zero-to-many, meaning that a document "Land treatment" may contain zero or more documents of the type, "Management".

The document "Land use changes" and "Land treatment" have a cardinality of zero-to-many, which means that a document of "Land use changes" may contain zero or more documents of the type, "Land treatment".

Finally the document "Land use changes" reference to the document "Location" with a cardinality of one to one, means that a document "Land use changes" can be associated with one location only.

A. Implementation

Considering that the focus of our work is data modeling for NoSQL database, including spatial data features, we have a small range of NoSQL databases to work with. Although some databases provide extensions for geospatial data, as in Cassandra [22], JBoss Infinispan [23] and ArangoDB [24], only Couchbase [25] and MongoDB [2], both documented-oriented databases, offer good documentation for data modeling. The Couchbase spatial data feature is introduced as an experiment and it is not officially provided by the tool yet [26]. And although Couchbase supports many features, we decided to use MongoDB because it provides an excellent documentation and native mechanisms and indexes for geospatial information.

The MongoDB is a document-oriented and open-source database management system [27]. MongoDB uses a BSON

The biomes graphic elements and their spatial references were acquired in digital file format, known as Shape. The practical reference of this research entails installing, configuring and creating a schema in MongoDB database, which is responsible for hosting documents containing JSON collections, and storing the spatial data of Brazilian biomes. The MongoDB transactions are structured and based on a JSON format file, and the scripts created for this research follow this pattern. We created database scripts to enter and access data of the biomes location. Furthermore, it was necessary to index the geospatial columns to improve the performance of the queries operations and to provide data for the tools that are responsible for showing data graphically. The operations to insert and return data from a MongoDB document follow a specific syntax, and the result is an object in JSON format. Considering the correct installation and database configuration, and the proper data entry, the Quantum GIS tool [28], associated with add-ons, is used for graphical representation of data. Two levels of QGIS layers were added. One to represent the Brazilian map and the second containing data collections of Brazilian biomes, which were stored in MongoDB format documents. This representation was possible due to the use of components responsible for integrating Quantum GIS and MongoDB.

The graphical representation result by Quantum GIS is shown in Figure 15. The points represent the geographic location of the samples.

VI. Conclusion and Future Work

In contrast to relational database management systems, NoSQL databases are designed to be schema-less and flexible. Therefore, the challenge of this work was to introduce a data modeling standard for NoSQL document-oriented databases, in contrast to the original idea for NoSQL databeses. The objective was to build compact, clear and intuitive diagrams for conceptual data modeling for NoSQL databases. While the current studies propose generic techniques and do not define a specific modeling engine to NoSQL database, our idea was to present a graphical model for any NoSQL document-oriented database. Moreover, while other studies describe techniques based on UML Diagram Class and JSON format, for example, as a modeling solution, we have proposed a new approach

 ISBN: 978-1-61208-383-4

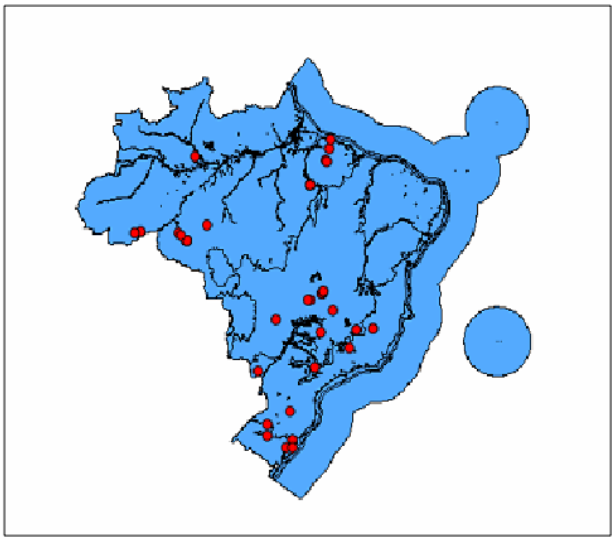

Figure 15. Plotting data in QGIS

to solve the conceptual data modeling issue for NoSQL document-oriented databases, including spatial data references.

Future work includes: verifying our model for other NoSQL database classifications, such as key-value and column; many-to-many relationship for embedded and reference documents is not covered and adjacency, connectivity, and other topological spatial concepts. These points were not developed in the present model proposed in this article.

REFERENCES

[1] R. Elmasri and S. Navathe, Fundamentals of Database Systems. Pearson Addison Wesley, 2010.

[2] MongoDB. Document database. [Online]. Available: http://www.mongodb.org/ [retrieved: Jun., 2014]

[3] J. Partner, A. Vukotic, and N. Watt, Neo4j in Action. O'Reilly Media, 2013.

[4] D. Borthakur et al., "Apache hadoop goes realtime at facebook," in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011, pp. 1071–1080.

[5] F. Chang et al., "Bigtable: A distributed storage system for structured data," ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, 2008, p. 4.

[6] R. F. Lans, Introduction to SQL: mastering the relational database language. Addison-Wesley Professional, 2006.

[7] K. A. Borges, C. A. Davis, and A. H. Laender, "Omt-g: an object-oriented data model for geographic applications," GeoInformatica, vol. 5, no. 3, 2001, pp. 221–260.

[8] J. L. De Oliveira, F. Pires, and C. B. Medeiros, "An environment for modeling and design of geographic applications," GeoInformatica, vol. 1, no. 1, 1997, pp. 29–58.

[9] S. Shekhar, M. Coyle, B. Goyal, D.-R. Liu, and S. Sarkar, "Data models in geographic information systems," Communications of the ACM, vol. 40, no. 4, 1997, pp. 103–111.

[10] G. Kosters, B.-U. Pagel, and H.-W. Six, "Gis-application development with geoooa," International Journal of Geographical Information Science, vol. 11, no. 4, 1997, pp. 307–335.

[11] Y. Bédard, C. Caron, Z. Maamar, B. Moulin, and D. Vallière, "Adapting data models for the design of spatio-temporal databases," Computers, Environment and Urban Systems, vol. 20, no. 1, 1996, pp. 19–41.

[12] G. Abrantes and R. Carapuca, "Explicit representation of data that depend on topological relationships and control over data consistency," in Fifth European Conference and Exhibition on Geographical Information Systems–EGIS/MARI, vol. 94, no. 19917, 1994, pp. 869–877.

[13] H. Scalable. Nosql data modeling techniques. [Online]. Available: http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/ [retrieved: Jun., 2014]

[14] R. Arora and R. R. Aggarwal, "Modeling and querying data in mongodb," International Journal of Scientific and Engineering Research (IJSER 2013), vol. 4, no. 7, Jul. 2013, pp. 141–144.

[15] K. Banker, MongoDB in action. Manning Publications Co., 2011.

[16] D. Crockford, RFC 4627 (Informational) The application json Media Type for JavaScript Object Notation (JSON). IETF (Internet Engineering Task Force), 2006.

[17] S. J. Pramod, "Nosql distilled: A brief guide to the emerging world of polyglot persistence/pramod j. sadalage, martin fowler," 2012.

[18] MongoDB. Data modeling introduction. [Online]. Available: http://docs.mongodb.org/manual/core/data-modeling-introduction/ [retrieved: Jun., 2014]

[19] CouchDB. Modeling entity relationships in couchdb. [Online]. Available: http://wiki.apache.org/couchdb/ [retrieved: Jul., 2014]

[20] G. Booch, J. Rumbaugh, and I. Jacobson, The unified modeling language user guide. Pearson Education India, 2005.

[21] H. V. Olivera and M. Holanda, "A gis web with integration of sheet and soil databases of the brazilian cerrado," in Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on. IEEE, 2012, pp. 1–6.

[22] Cassandra. Cassandra documentation from datastax. [Online]. Available: http://wiki.apache.org/cassandra/GettingStarted [retrieved: Dec., 2014]

[23] J. Infinispan. Get started. [Online]. Available: http://infinispan.org/documentation/ [retrieved: Dec., 2014]

[24] AragonDB. Documentation. [Online]. Available: https://www.arangodb.com/documentation [retrieved: Dec., 2014]

[25] Couchbase. Introduction. [Online]. Available: http://docs.couchbase.com/admin/admin/Couchbase-intro.html [retrieved: Jun., 2014]

[26] ——. Writing geospatial views. [Online]. Available: http://docs.couchbase.com/admin/admin/Views/views-geospatial.html [retrieved: Dec., 2014]

[27] MongoDB. The mongodb 2.6 manual. [Online]. Available: http://docs.mongodb.org/manual/ [retrieved: Jun., 2014]

[28] QGIS. A geographic information system free and open source. [Online]. Available: http://www.qgis.org/ [retrieved: Jun., 2014]

 ISBN: 978-1-61208-383-4