NoSQL Database Design Processes - A Comparative Study

Luciano Marrero[1], Verena Olsowy [1], Pablo Thomas[1]

[1] Instituto de Investigación en Informática LIDI
Facultad de Informática - Universidad Nacional de La Plata – Argentina
Centro Asociado a la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)
526 e/ 10 y 11 La Plata Buenos Aires
{lmarrero, volsowy, pthomas}@lidi.info.unlp.edu.ar

Abstract. NoSQL databases have emerged as a response to scalability problems presented by relational databases when used in Big Data contexts. These databases do not have a standard process for designing data schemas, but have emerged as solutions directly at the physical level. As NoSQL databases have gained popularity, different NoSQL design processes and/or methodologies have been proposed, which are necessary to understand the semantics of the stored data. This paper presents a comparative study of NoSQL database design processes.

Keywords: NoSQL, Design NoSQL, Database Design Processes.

1 Introduction

For several years now, there have been software applications that have pushed relational databases to their performance limits. Examples of these applications are social networks, such as Twitter or Facebook, online shopping applications, such as Amazon or eBay; Internet of Things (IoT) applications, Smart Cities; or the use of Big Data, among others. These applications need to manage large volumes of data that are often distributed across multiple servers, must ensure adequate response times and high availability in contexts of a high number of concurrent requests. In these scenarios, relational databases have shown different scalability problems. In response to this problem, a new generation of database management systems, known as NoSQL, has emerged.

NoSQL is not only an alternative to relational databases, but is an umbrella term for various strategies for storing unstructured data. Initially, these databases emerged at the implementation level (physical level), and consequently without a defined process for their design [5, 6].

Traditional relational database design and construction methodologies have been extensively studied, applied and refined for decades. However, the principles and/or rules that apply to a relational data model are not appropriate for a NoSQL database. This is because they have a different implementation, therefore, their design process must also be different.

This paper proposes a comparative analysis of NoSQL database design processes.

Starting from section 2, the paper is organized as follows: first, the essential aspects of NoSQL databases are presented, section 3 presents the selected design processes, section 4 presents a comparative analysis, and finally, section 5 expresses conclusions and future work.

2 Main features of NoSQL databases

NoSQL differs from traditional relational database management systems in several respects; it does not have a structured query language (SQL) as its core language, does not require a fixed, tabular structure, does not support JOIN operations, does not fully guarantee ACID properties (atomicity, consistency, isolation and durability), and is generally suitable for horizontal scalability [6, 7, 8].

NoSQL proposes a system called "BASE (Basically Available, Soft State, Eventual Consistency)". Through these properties basic availability is achieved, meaning that the system will be available most of the time. With soft state the system becomes more flexible in terms of consistency and with eventual consistency it is guaranteed that the system will eventually become consistent [6, 7, 8]. There are four main types of storage for NoSQL Databases.

Key/Value Storage: Simple in implementation, they store data as a set of "key/value" pairs. The key represents a unique identifier that can return an arbitrary complex object of information, called a value. For example, Redis and DynamoDB, among others, implement this type of storage. [13, 14].

Documentary Storage: the central concept of this type of storage is the document. A Documental NoSQL Database stores, retrieves and manages documents. These documents encapsulate and encode data in some standard format (XML, YAML, JSON, BSON). For example, MongoDB and CouchDB, among others, are implementations of Document Databases.[9][10]

Column Family Storage: In this type of storage, data is organized by columns, rather than rows. For example, Cassandra and HBase, among others, use this type of storage. [15][16].

Graph Storage: the database is represented under the concept of a graph, allowing the information to be stored as nodes and their respective relationships with other nodes by means of edges. Graph theory is applied to traverse the structure. They are useful for storing information in contexts where there are numerous relationships between their data. Neo4j and OrientDB, among others, implement this type of storage. [11][12].

3 Design processes for NoSQL databases

In recent years, methodologies for the design of NoSQL databases have emerged. In [5] a literature review was conducted on this topic. As a result of this review, only three processes for the design of NoSQL databases have been identified [1], [2], [3, 4], which are analyzed in the present work.

3.1 **Mortadelo: Automatic generation of NoSQL stores from plataform-independent data models [1]**

Mortadelo is described as a model-based NoSQL database design process, where, starting from a conceptual data model, independent of any database type, a possible implementation for a specific NoSQL database system is autonomously generated. In addition, this process allows the final design to be configured according to the needs of each context.

This process needs to operate with well-defined models, in particular, it needs as input a metamodel in which the conceptual data model and the queries that will retrieve and update the represented information are specified. In addition, it allows certain general annotations to be made, e.g. the update rate that an entity is likely to have. This metamodel is called Generic Data Metamodel (GDM) and describes its components (entities, relations or references and queries) by means of a textual notation that has its own syntax.

In this process, starting from a case study, represented through a GDM, logical models are proposed for two categories of unstructured data storage, column family, where the Cassandra database engine [15] is used, and documentary, where the MongoDB database engine [9] is used. For this purpose, a set of predefined rules and algorithms are applied to transform an instance of a conceptual model into a concrete logical model for a NoSQL database type.

Finally, using another set of rules, they generate concrete code transformations for two database engines, Cassandra (column family storage) and MongoDB (document storage), i.e. the physical schema for a specific database engine is generated.

3.2 **NoAM (NoSQL Abstract Model): Data Modeling in the NoSQL World [2]**

A design process is proposed that has a conceptual phase, a logical phase, which is independent of the database type, and a final phase that considers the specific characteristics of a NoSQL database engine. NoAM is based on the following main activities:

A. Conceptual data modeling from Domain Driven Design (DDD) resulting in a UML diagram. There is no mention of how to realize this diagram.
B. On the UML diagram of the previous point, aggregates are identified. An aggregate is a group of related objects, representing an atomic unit of access and manipulation.
C. Implementation of the NoAM model based on the identification of aggregates.

The process begins with database design, building a conceptual representation of the data of interest, in terms of entities, relationships and attributes. Next, aggregations are identified. This activity may be driven by data access patterns, as well as by scalability and consistency needs. Specifically, aggregates must be designed as the units in which atomicity must be guaranteed. Each aggregate should include all the data required by a relevant data access operation. On the other hand, aggregates should be as small as possible. Small aggregates reduce concurrency collisions and meet performance and scalability requirements.

In this approach, NoAM is used as an intermediate model between aggregates and NoSQL databases. In NoAM, the unit of data access and distribution is modeled by a block, which represents a maximal unit of data for which atomic, efficient and scalable access operations are provided. NoSQL systems provide efficient, scalable and consistent operations on blocks and, in turn, this choice propagates such qualities to operations on aggregates.

Finally, we discuss how a NoAM data representation can be implemented in a specific NoSQL database engine (e.g., MongoDB [9]).

3.3 Modeling NoSQL databases: From Conceptual to Logical Level Design [3,4]

This approach proposes a common conceptual level model for several types of NoSQL databases and a NoSQL data specification language to represent a logical level data model, independent of any physical level representation. In addition, different validation rules have been proposed with respect to the conceptual model through the evolution of a case study.

This conceptual model has a common set of constructs, relationships and a set of meaningful properties of relationships to unify the conceptual level representations of different NoSQL databases. This model consists of three interrelated layers: Collection, Family and Attribute. The Attribute layer is the base layer of the conceptual model and the AT construct types are groups of all possible same instance types and elementary in nature.

The family layer is the middle layer of the conceptual model and can contain numerous types of FA constructs. This layer can be decomposed into multiple levels according to the designers' preferences.

The collection layer is the top layer of the conceptual model. The semantically related families of the top layer are assembled to form a column.

From a higher level, the database can be viewed as a set of columns.

The constructs of this model are connected to each other by distinct relationships. These relationships can be of two types: type relationship between layers and type relationship within the layer. These relationships have several properties, such as multiplicity, order, modality, availability, conditional participation and consistency.

A specification language is proposed to transform a conceptual data model into a logical model and then into a corresponding physical model for a NoSQL database engine.

Finally, a set of validation rules is proposed for the NoSQL model obtained, these rules are divided into three groups: for structural validation, for constraint validation and for consistency validation.

4 Comparative analysis

The three works present a design process for NoSQL databases. These three processes propose an approach which starts with a conceptual modeling stage, continues with a logical model and culminates with a physical model specific to a particular NoSQL database engine.

All three processes present details that should be taken into account when using and/or applying each of them.

In [1], it is required to define a metamodel called GDM that integrates a high-level conceptual model and the queries that will impact the database. Defining queries at an early stage can be prone to major changes at later stages when there are changing requirements. In addition, examples are presented for specific NoSQL databases, Cassandra for columnar family and MongoDB for documents. In the case of NoSQL databases with key-value storage, the possibility of applying this design process following the same rules as for column-family and document NoSQL databases is mentioned. For problems where the complete conceptual model and the main queries that will impact the database are available, it is a NoSQL database design process through which a physical model can be obtained for two types of NoSQL database engines, column-family (Cassandra) and document (MongoDB). For key-value NoSQL database types it is mentioned that it is an applicable process, following the same guidelines as for column-family and document engines, but no example is specified and with respect to graph-oriented NoSQL database types, no details of its application have been provided.

In [2], the starting point is a UML diagram [x] that is generated from a DDD (Domain-Driven Design) of which no details are given as to how it was done. On the UML model generated, the design of aggregates is carried out. This design requires knowledge of how the data will be retrieved and modified, which makes the process difficult, as it is generally difficult to establish the exact way in which the data will be manipulated at an early stage of the design. However, for documentary NoSQL database types, which have flexible schemas, or for key-value NoSQL databases, where the key must be clearly defined and is the only way to access the data, this design process can be adequately applied. For column-family NoSQL databases, e.g. Cassandra and/or graph-oriented databases, e.g. Neo4j [11], there is not enough detail to draw a conclusion.

In [3, 4] the design of a conceptual model specifically created for the proposed approach that has a set of layers or phases is required. In this conceptual model the existing relationships between the data is not explicit, but is at the layer level, something that can make it difficult to read and interpret. Subsequently, a logical and physical model is defined based on process-specific templates and finally a set of 18 rules that can be applied to obtain a final physical model must be analyzed. Compared to [1] and [2] this approach has formalisms and technicalities that make it a design process that demands a lot of attention and discipline from the designer. However, it provides an overview that supports changes dynamically and has important features with respect to data availability and replication, something that is not clear in [1] and [2].

In summary, in the case of using a NoSQL database engine that implements column family storage (e.g. Cassandra) or document storage (e.g. MongoDB), the design process proposed in [1] could be considered, this is because, in the examples presented, it is a complete process in its definition and its implementation is clearer and simpler compared to [2, 3, 4].

For graph-oriented NoSQL database engines, the design process defined in [3, 4] could be used, as it is the only one that considers this type of storage. For key-value NoSQL database engines, the design process in [1] could be used, if it is possible to generate the GDM metamodel. In the case of having only the conceptual model, one could consider the design process [2] or [3, 4] if one wants to use the phased conceptual model defined in that approach.

5 Conclusions and future work

This paper focuses on the analysis of three processes for the design of NoSQL databases. A literature review was conducted in [5], which identifies three design processes or methodologies that are presented or classified as applicable to more than one type of NoSQL databases [1], [2], [3, 4].

First, Mortadelo: *Automatic generation of NoSQL stores from platform-independent data models* [1] was presented. This approach describes a process based on models that need as input the conceptual data model and the queries that will impact the final database. Subsequently, the derivation is made to two logical models according to the type of database to be used (column family and documentary). Finally, a set of algorithms are applied, according to the logical model generated, to create the physical model corresponding to the NoSQL database engine to be used.

Secondly, *NoAM: (NoSQL Abstract Model): Data Modeling in the NoSQL World* [2] was presented. This approach proposes a phased design process. In the first phase the conceptual data model is made, and a set of aggregates are identified, in the second phase a logical model is proposed and finally, in the third phase the physical model is generated according to the NoSQL database engine to be used.

Thirdly, the approach *Modeling NoSQL databases: From Conceptual to Logical Level Design* [3, 4] was presented. In this approach, a conceptual model was presented, which is generated based on a set of constructs and/or layers. Then, through a set of templates, the corresponding transformations are proposed to obtain a logical and physical model according to the NoSQL database engine to be used.

Finally, a comparative analysis is proposed where it is suggested which design process or processes are more convenient according to the type of NoSQL database to be used.

As future work, we intend to apply the design processes analyzed to different case studies and different types of NoSQL databases.

References

1. Alfonso de la Vega, DiegoGarcía,Saiz Carlos Blanco,Marta Zorrilla, Pablo Sánchez. Mortadelo: Automatic generation of NoSQL stores from platform-independent data models.Future Generation Computer Systems. Volume 105, April 2020, Pages 455-474.
2. Paolo Atzeni, Francesca Bugiotti, Luca Cabibbo, Riccardo Torlone. Data Modeling in the NoSQL World. HAL open science. https://hal.archives-ouvertes.fr/hal-01611628.
3. Shreya Banerjee, Anirban Sarkar. Modeling NoSQL Databases: From Conceptual to Logical Level Design. 3rd International Conference on Applications and Innovations in Mobile Computing (AIMOC – 2016) At: Kolkata, India.
4. Shreya Banerjee, Anirban Sarkar. Logical level design of NoSQL databases.2016 IEEE Region 10 Conference (TENCON).
5. Luciano Marrero, Verena Olsowy, Fernando Tesone, Pablo Thomas, Leandro Corbalán, Juan Fernández Sosa, Patricia Pesado: Un Estudio de Procesos de Diseño de Bases de Datos NoSQL. XXVIII Congreso Argentino de Ciencias de la Computación - CACIC 2022. ISBN: 978-987-1364-31-2. http://sedici.unlp.edu.ar/handle/10915/149452.

6. Pesado P., Thomas P., Delía L., Marrero L., Olsowy V., Tesone F.: Análisis de performance en Bases de Datos NoSQL y Bases de Datos Relacionales. XXVI Congreso Argentino de Ciencias de la Computación (CACIC 2020). ISBN 978-987-4417-90-9. http://sedici.unlp.edu.ar/handle/10915/114202.
7. Pesado P., Thomas P., Delía L., Marrero L., Olsowy V., Tesone F., Fernandez S. J.: Un estudio comparativo de bases de datos relacionales y bases de datos NoSQL. XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019). ISBN 978-987-688-377-1. http://sedici.unlp.edu.ar/handle/10915/91403.
8. Migani Silvina, Vera Cristina, Lund María Inés. NoSQL: modelos de datos y sistemas de gestión de bases de datos. XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste). http://sedici.unlp.edu.ar/handle/10915/67258.
9. MongoDB. https://www.mongodb.com/es. Abril de 2023.
10. CouchDB. https://couchdb.apache.org/. Abril de 2023.
11. Neo4j. https://neo4j.com/. Abril de 2023.
12. OrientDB. https://orientdb.org/. Abril de 2023.
13. Redis. https://redis.io/. Abril de 2023.
14. Amazon DynamoDB. https://aws.amazon.com/es/dynamodb/. Abril de 2023.
15. Apache Cassandra. https://cassandra.apache.org/_/index.html. Abril de 2023.
16. Apache HBase. https://hbase.apache.org/. Abril de 2023.