A Modeling methodology for NoSQL Key-Value databases

Gerardo ROSSEL, Andrea MANNA
Universidad de Buenos Aires.
Facultad de Ciencias Exactas y Naturales.
Departamento de Computación. Buenos Aires, Argentina.
grossel@dc.uba.ar, amanna@dc.uba.ar

In recent years, there has been an increasing interest in the field of non-relational databases. However, far too little attention has been paid to design methodology. Key-value data stores are an important component of a class of non-relational technologies that are grouped under the name of NoSQL databases. The aim of this paper is to propose a design methodology for this type of database that allows overcoming the limitations of the traditional techniques. The proposed methodology leads to a clean design that also allows for better data management and consistency.

Keywords: *NoSQL, Key-Value Store, Conceptual modeling, NoSQL, Database design, Big Data*

1 Introduction

The need for analysis, processing, and storage of large amounts of data has led to what is now called Big Data. The rise of Big Data has had strong impact on data storage technology. The challenges in this regard include: the need to scale horizontally, have access to different data sources, data with no scheme or structure, etc. These demands, coupled with the need for global reach and permanent availability, gave ground to a family of databases, with no reference in the relational model, known as NoSQL or "Not Only SQL" (in some contexts also called NoSQL data stores).

There is a huge variety of NoSQL databases. They can be classified, among other things, according to the way they store and retrieve the information [1], [2]:

- Key-Value databases.
- Document databases.
- Column Families databases.
- Graph Databases.

Redis, one of the emblematic examples of key-value stores, is one of the most popular databases [3]. The development of conceptual modeling and general design methodology associated with the construction of NoSQL databases is at an early stage. There is previous work on development methodologies we can cite, like the Big Data Apache Cassandra methodology, proposed by Artem Chebotko [4].It uses the Entity Relationship Diagram as a conceptual model but it is oriented to a specific engine, Apache Cassandra. Thus, it is not generic and does not adapt to a design of key-value stores. Another proposal using a conceptual model for the design of NoSQL is described in [5]. It suggests the use of the various NoSQL databases common features to obtain a general methodology, in which an abstract data model called NOAM is used for conceptual data modeling. Such data model is intended to serve all types of NoSQL databases using a general notation.

Recently, an attempt to generate a universal modeling methodology adapted to both relational and non-relational database management systems was also presented, on the grounds of overcoming the constraints that the entity relationship model has, according to the author [6]

These efforts show that traditional methodologies and techniques of data modeling are insufficient for new generations of non-relational databases. It is necessary, then, to develop modeling techniques that adapt to these new ways of storing information. In this sense, this

paper will provide the tools to solve these limitations for key-value databases.
The rest of the paper is organized as follows: Section 2 outlines the fundamentals of the methodology proposed; Section 3 describes the main elements of the key-value databases design, which we call dataset and keeps certain correspondence with the entities of the conceptual model, to specify, in turn, the final adjustments that involve navigability between datasets and the addition of auxiliary datasets; Section 4 presents a simple design example and finally Section 5 presents conclusions and future work.

2 Methodology

The proposed design methodology has as its starting point the conceptual model, which can also be seen as a high-level description of data requirements. Conceptual modeling is usually performed using some form of entity-relationship diagram (DER) or conceptual class diagram in UML [7,8,9,10,11].

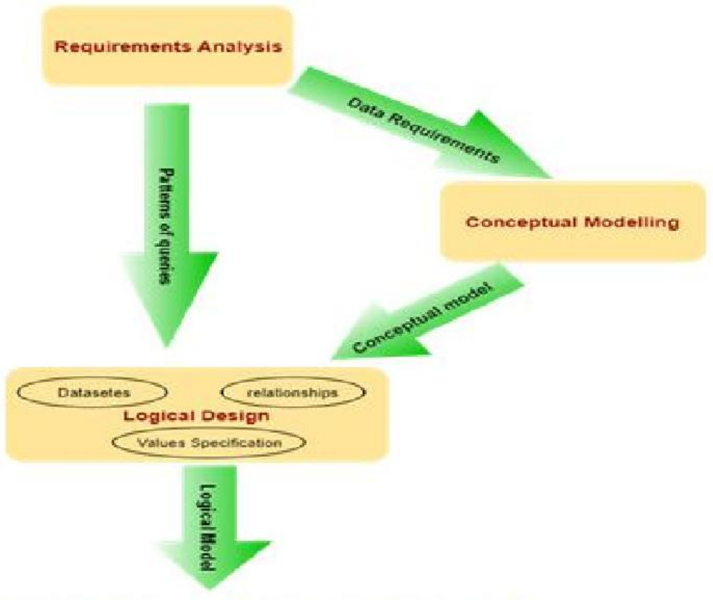

Fig. 1 Main Phases of Key Value Databases design

In traditional relational database design methodologies, conceptual modeling gave way to a logical design that was later transformed into a physical design.
It operates by transforming models from higher levels of abstraction to a model that maps directly into the structures of the database.
NoSQL databases, and in particular key-value storage systems, are generally referred to as *schema-less*, which seems to suggest that it is not necessary to make a model before the development starts. The fact that the structure of the data does not need to be defined has a priori many advantages for prototyping or exploratory development, but as data expands and the applications make use of them, the necessity to have them organized in some way arises.
In order to model and implement a key-value database correctly, it is necessary to take into account, firstly, the access patterns established in the requirements. In other words, the inputs to the logical and physical design are: the conceptual data model and the access patterns. It is assumed that during the phase of gathering requirements the feasibility of using this type of storage management system was analyzed.
From the conceptual model and access patterns, it is possible to specify the entities that are represented in the database by means the concept of datasets, as more precisely defined in section 3. Access to the values is performed using the keys. For every dataset it is necessary to realize a

correct design and keys selection. The keys design must be able to balance legibility and easy maintenance (to provide a logical structure), with access and storage efficiency. The keys also play an essential role in the implementation of distributed and scalable architectures, because they are used to distribute data across a set of servers.

Once the initial datasets have been established, the interrelations between them must be defined; indicating how the information obtained from a dataset can possibly enable access to the elements of another. In this context new datasets may arise whose role is simply to organize keys and favor navigability of the applications accessing the database. The diagram in Figure 1 illustrates the steps of the methodology proposed.

2 Datasets

Although this type of databases NoSQL is presented as a collection of key value pairs in such a way that access to the values is performed using a key, or part of it, it is also true that the data can be grouped into sets of data that represent some type of entity, relationship or event category in the real world.

These groups of data that can be inferred from the conceptual model are called datasets. All the elements (concrete elements) of a dataset share a common access mechanism, using a key with a uniform format. In addition, they present a similar scheme although there may be some differences between the elements of the same dataset. Namespaces, if any in the selected database engine, are not sufficient to differentiate the various types of datasets.

When in order to define a key, a data type is used that can adopt the same value for different datasets, a clash of keys could occur, i.e., there would be two equal keys referring to different values.

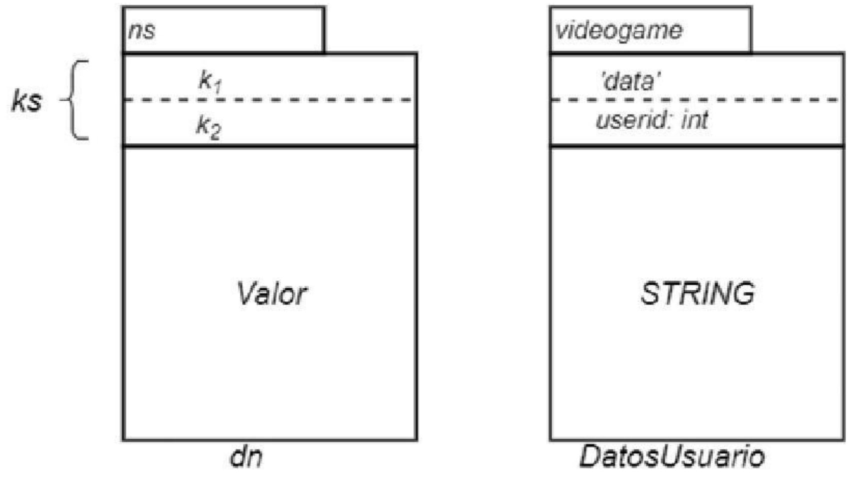

Fig. 2 Datasets Graphic Representation

For example, if you use a numeric key for customers and products, the value 2120 could be either the client 2120 or the product 2120. One way to avoid the clash of keys is to use a prefix that identifies the entity or context to which the key belongs. It is usual to use the name of the dataset as a prefix. This way a key “product:2020” would not conflict with another one “customer:2020”. The prefixes that represent different entities allow for the identification of datasets.

The methodology in this article proposes that the logical design is represented by a set of interrelated datasets, defined as a 4-tuple, namely:

$$dataset := \langle dn; ns; ks; v \rangle;$$

where a dataset has a name dn, a namespace ns, an ordered list of elements that form the key ks, and a structure or a value type$v$.
The name "dn" allows to identify the dataset and must be unique, usually if it corresponds to an entity in the conceptual model same name is used. Namespace can be a string or a null value. In database engines that do not have support for namespaces its value becomes part of the key. The set of key parts ks helps usassemble complete key to access the value by concatenating them and using a separator between them. Each $k_i \epsilon k_s$ can take either a variable of some kindor constant value. The constant value is used, as previously explained, to avoidkey collision.
In our example of products and customers, it would be:

$$ks_{product} = \{k_1, k_2\}$$

where $k_1 = 'product'$ and $k_2 = idProduct\colon int$

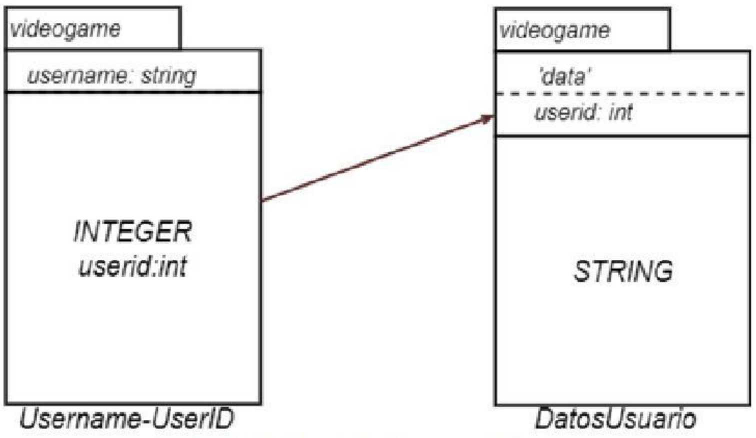

Fig. 3 Relationship between datasets

It is said that a concrete element e (a key-value pair) belongs to a *dataset* $(Ds := \langle dn; ns; ks; v\rangle)$ if its key e_k is the ordered concatenation (with a uniform separator) of any of the possible values of each $k_i \epsilon k_s$ and its value (e_v) has the structure or the type of v. The value that takes the part of key k_i for a concrete element e_j is notatede_{jk_i} and its valuee_{jv}.
A navigation interrelationship between two datasets Ds_1 and Ds_2, indicates that an element of $e_1 \in Ds_1$ (a key-value pair or element) is related to an element $e_2 \in Ds_2$ and is defined as follows:

$$Ds_1 := \langle dn_1; ns_1; ks_1; v_1\rangle$$
$$Ds_2 := \langle dn_2; ns_2; ks_2; v_2\rangle$$
$$rel := \langle (Ds_1, Ds_2), lstVinc\rangle$$
$$lstVinc := (k_2 | (k_1, k_2))^*$$

where

$$k_1 \in ks_1 \text{ and } k_2 \in ks_2$$

The ordered pair (Ds_1, Ds_2) indicates the direction of the navigability from Ds_1 to Ds_2. The list of links $lstVinc$ defined the interrelationship, ie., how to obtain, from a concrete element $e_1 \in Ds_1$, the information to assemble the key to have access to a concrete element belonging to $e_2 \in Ds_2$. If there is only onekey in the list (which would correspond to Ds_2) then its value is obtained fromthe value of e_1. In the other case, the value takes the part corresponding to k_1 in e_1, whose value must be equal to the value taken by k_2 in element e_2, indicatingthat the concrete elements in each of these parts of the key have the same value.
The interrelationship is completely defined by all the elements of the list $lstVinc$
The datasets and their relationships can be

represented in a diagram to visualize them and the navigation between them immediately. Figure 2 shows the graphic representation of a dataset, as well as a concrete example.

The interrelations that allow from the value or a part of the key of a concrete element in a dataset access to an element in another dataset is represented by directed arrows departing from the value or key part of the first dataset to the key part *k* corresponding to the set *ks* of the second one. Figure 3 shows a relationship between two datasets, the one labeled *UserName-UserID* has a numerical identifier or *userid* by each user name (used as key) that can be used in other places to access the information that corresponds to that user. The*userid* can be used, for example, to access user data such as email, address, name, and surname, etc. stored in another dataset.

This relationship example is a verycommon pattern in key-value databasesthat allows such things as renaming a user with minimal impact or easily generating reverse indexes

4Application Example

In this section, a simple partial example is presented to illustrate how a diagram is done. The example is to design a key-value database that stores the information of users and online game matches. The input to dataset design is the conceptual model, query patterns and requirements. To keep the example simple only a small part of the problem domain is shown. For the requests, it is known that every user can save a game at any time. Upon entering the game, the user can choose to continue from any state where the game was saved.

The status of the items is determined by a location in the map of the game, a group of objects and other data as the level of energy, remaining lives, etc.

From the conceptual model, the following entities and relationships are derived: users, starting status and friends (one user can be a friend of another). The established query patterns are: view the list of saved games, retrieve a saved game and view a user's friends.

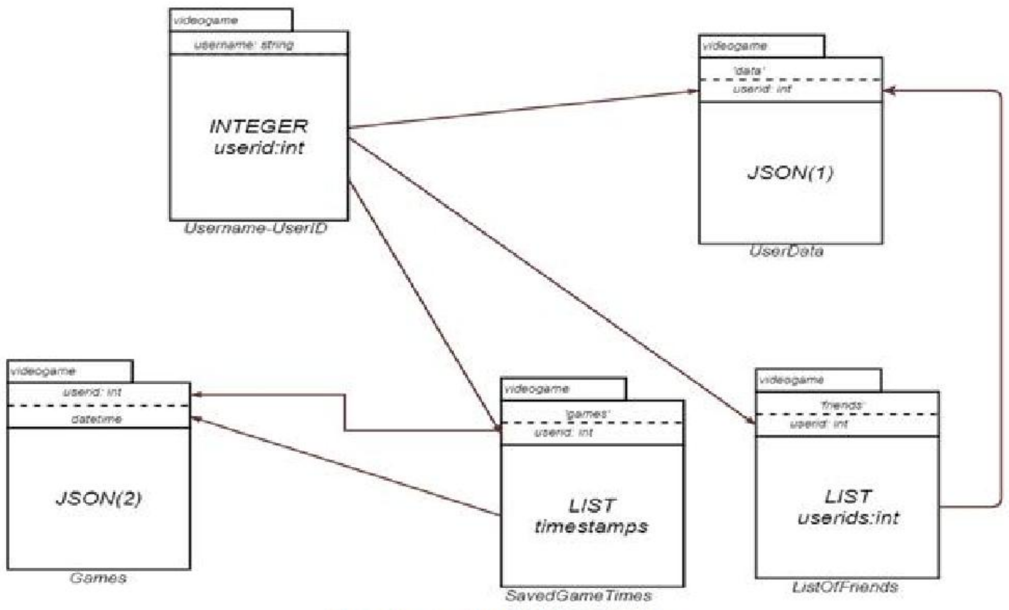

Fig. 4 Logical Design Example

The datasets arising from this simplified description represent items and users (entities of the conceptual model). One of the ways to solve the interrelations that exist in the conceptual model is to use a dataset that allows one entity to navigate to

another one. In the simplified example presented, a dataset is used to store a list of timestamps that would show the moments in which a user saved their games. To identify the user the pattern outlined in the previous section can be used, which links a user name with an identifier. Figure 4 shows the complete diagram. Labeling values as type JSON with an associated number would specify
the structure that this JSON would take in the format *JSON Schema.*

5 Conclusions

This article has presented an original methodology to support the design of NoSQL Key-Value databases. The input to the logical design incorporates query patterns along with the conceptual model. We have explained how to represent the conceptual entities and design through the concept of dataset. In addition, a new type of diagram was specified, which allows the specification of datasets and interrelations established during the design phase.
These are some highlights of the proposed methodology:

- This is, to the best of our knowledge, the first work that presents a method logy supported by specific notation to develop NoSQL databases based on key-value.
- The diagrams presented favors communication and understanding of the design decisions made during the development

The prospects for this project involve the improvement of this work, based on the experience of its use in various development projects and the production of a series of transformation patterns from the conceptual model to the logical design.

6Acknowledgment

We want to thank María Jose Constanzo of University of Buenos Aires for their assistance in the final writing of this article.

References

[1] Adam Flowler,"The State of NoSQL", 1st edition, 2016
[2] Pramod J. Sadalage, Martin Fowler,"NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", *Addison-Wesley Professional*,1st edition, 2012
[3] "DB-Engines Ranking"http://db-engines.com/en/ranking (consulted in June 2017)
[4] ArtemChebotko, Andrey Kashlev, Shiyong Lu, "A Big Data Modeling Methodology for Apache Cassandra", *IEEE International Congress on Big Data (BigData'15)*, pp. 238-245, New York, USA, 2015.
[5] Francesca Bugiotti, Luca Cabibbo, Paolo Atzeni, Riccardo Torlone. "Database Design for NoSQL Systems". *International Conference on Conceptual Modeling*,pp.223 - 231 Atlanta, USA, Oct 2014.
[6] Ted Hills,"NoSQL and SQL Data Modeling",*Basking Ridge, NJ: Technics Publications*, 2016
[7] Peter P. S. Chen,"The entity-relationship model: toward a unified view of data",*Proceedings of the 1st International Conference on Very Large Data Bases,* ACM, New York, NY, USA, 1975.
[8] Carlo Batini, Stefano Ceri, and Shamkant B. Navathe, "Conceptual Database Design: An Entity-Relationship Approach", *Benjamin-Cummings Publ. Co., Inc.,* Redwood City, CA, USA, 1991
[9] François Lagarde, Huáscar Espinoza, François Terrier, and Sébastien Gérard, "Improving uml profile design practices by leveraging conceptual domain models.", *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE '07)*, pp. 445-448 ACM, New York, NY, USA, 2007
[10] Antoni Olivé. "Conceptual Modeling of Information Systems.",*Springer-*

Verlag New York, Inc., Secaucus, NJ, USA, 2007

[11] Toby J. Teorey, Dongqing Yang, and James P. Fry,"A logical design methodology for relational databases using the extended entity-relationship model",*ACM Comput. Surv*. 18, 2, June 1986

Gerardo ROSSEL graduated as Ms. Sc. in Computer Science from the Faculty of Exact and Natural Sciences of the University of Buenos Aires. He has a Doctor's degree from the National University of Tres de Febrero. At present, he is an assistant lecturer at Computer Department of FCEyN UBA. He has more than 20 years of experience in software industry and is Chief Scientist of UpperSoftsoftware company. Her specific area of competences is in Software Engineer, Databases, NoSQL, Intelligent Agents and Multiagent Systems, Patterns and Software Architectures, Epistemology and History of Computer Science. He is co-author of book "*Algoritmos,Objetos y Estructuras de Datos".* He has published several papers in national and international conferences and journals. He was a member of the International Program Committeeof several international conferences.

Andrea MANNA, graduated from the Faculty of Exact and Natural Sciences of the University of Buenos Aires in 2000. She got the title of Ms. Sc. of Computer Science. At present, she is Head of practical work in the Faculty of Exact and Natural Sciences of the University of Buenos Aires. She has been working in the software industry since 1995. She is Chief Software Architect of UpperSoftSofware Company and work in software development for more than twenty years. She is co-author of book "*Algoritmos,Objetos y Estructuras de Datos".*