Schema Proposition Model for NoSQL Applications

Abdullahi Abubakar Imam[1,2,3], Shuib Basri[1,2](✉), Rohiza Ahmad[1,2], and María T. González-Aparicio[4]

[1] Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Seri Iskandar, Perak, Malaysia
aiabubakar3@gmail.com, {shuib_basri, rohiza_ahmad}@utp.edu.my
[2] SQ2E Research Cluster, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Seri Iskandar, Perak, Malaysia
[3] Ahmadu Bello University, Zaria, Nigeria
[4] Computing Department, University of Oviedo, Gijon, Spain
maytega@uniovi.es

Abstract. This paper proposes a schema proposition model for physical and logical modeling of NoSQL (Not Only SQL) document-store databases. Unlike the traditional databases which are rigid in design, the NoSQL databases are flexible, scalable, and low-latent making them to house big data efficiently. However, issues such as dataset complexity coupled with programmers' low competency are making database modeling and designing very challenging, especially in balancing the significant parameters like availability, consistency and scalability. The proposed model aims to address this balancing by abstractly proposing schemas to programmers with respect to user defined system requirements, Create, Read, Update and Delete (CRUD) operations, etc. Exploratory and experimental approaches were adopted in this research. System requirement formulas were introduced to compute the expected performance. A mini system was developed to validate the proposed model by comparing the manually generated schema to the schema generated through the proposed model. Results obtained indicated a significant improvement in the schema generated by the proposed model.

Keywords: NoSQL databases · Database modeling · Big data
Schema proposition · Document-store databases

1 Introduction

In database technologies and big data research, new concerns been debated increasingly are the storage capabilities of the non-conventional applications. Traditional databases predominantly occupied all aspect of data storage for decades. They provide engines to centrally control data, eliminate inconsistencies and control redundancy [1]. The emergence of big data with its untraditional characteristics has necessitated the invention of flexible, scalable and latent databases called NoSQL [2–4]. The term

© Springer Nature Switzerland AG 2019

fathey.m.ye@gmail.com

NoSQL comes from “Not only SQL”, which means NoSQL are not introduced to replace SQL or traditional databases, but to fill the gap created due to the continuous expansion in data size, complexity, variety and variability [5].

NoSQL databases have since been widely accepted and adopted by many big organizations such as Google, Amazon, Facebook among others [6]. However, the dynamism nature of today’s data makes it difficult for programmers to model NoSQL databases appropriately for reasons such as low competency, traditional modeling mindset and complex datasets [7–9]. These have attracted the attention of both the industry and academia to search for viable solutions to help NoSQL data modelers. Nonetheless, some of these solutions are vendor specific [6], while others focused on a different scope of research [10].

In our earlier contributions to the same research scope, we conducted two different studies which were industrially motivated, theoretically grounded, and empirically validated. The first paper proposed new cardinality notations and styles [11], while the second paper presented modeling guidelines for NoSQL document-store databases [12]. The outcome of these studies laid down a solid foundation for the current study in the aspects of semantic mapping of entities and relationship prioritization. They also contributed in understanding and prioritizing system requirements in relations to user requirements.

In this paper, we proposed a schema proposition model for NoSQL document-store databases. This model aims to simplify further the NoSQL modeling process. As mentioned earlier, our first solution to the aforementioned problem was to establish cardinality standards for modeling NoSQL databases. Second, we produced, categorized and prioritized NoSQL modeling guidelines to guide programmers on how NoSQL databases can be best modeled. These solutions require considerably high level of expertise thus motivated the invention of the proposed model. Programmers need not have to write schemas from scratch. They only have to feed some parameters into the proposed model, and the rest is handled by the model. The main contributions of this paper are outlined as follows:

- Automatic schema proposition model for NoSQL document store databases. This model accepts parameters like system requirements (availability, consistency or scalability), CRUD operations, entities and their expected number of records etc. to produce a modeled schema at its output.
- New ways to calculate cluster or document availability. Cluster availability is categorized as Series, Parallel and Partial.
- An algorithm which can be directly translated into any programming language for the proposed model.

With such model, programmers can supply parameters with associated data to produce an initial schema based on solid theoretical and empirical foundations. It is important to note that, the proposed model aims to propose schema which can be used at the initial stage of database design only. This means the schema generated by the proposed model require expert review and enhancement before the commencement of system development. It’s not meant to be 100% copy-and-paste functional schema.

This paper is organized as follows: Sect. 2 discusses the related literature with respect to NoSQL data modeling. Section 3 presents the architecture, algorithms and

fathey.m.ye@gmail.com

components of the proposed model. Section 4 shows the results of the proposed model with comparisons. Section 5 concludes the paper with future focus.

2 Literature Review

NoSQL document-store databases are highly flexible [13]. They are based on a flexible model that allows schemas to be written and managed by the client side application developers [5, 14]. However, this may lead to incorrect or inappropriate schema design especially in modeling the relationships between datasets and entities [5, 6]. In efforts to address these challenges, several researches were conducted.

In the work of [5], conceptual modeling was proposed using Formal Concept Analysis (FCA). This was proposed to assist developers model the document based databases. It adopted three (3) types of relationships from relational databases which are (i) one-to-one ➔ 1:1, (ii) one-to-many ➔ 1:M, and (iii) many-to-many ➔ M:M relationships. These relationships were directly inherited from relational database and applied onto document-store databases. This method reveals the effectiveness of the aforesaid relationships when applied to document-store databases. However, the types of data stored in document-stores are much more complex and bulkier than the one stored in relational databases. Therefore require more detailed and deeper cardinality breakdowns. Also, foreign keys are not directly supported in document-store databases. In addition, other contributing factors to document-store modeling such as embedding and referencing are not considered in this research despite their importance to NoSQL database modeling practice.

On the other hand, a study of patterns and techniques was conducted by Arora and Aggarwal [15] This study aims to model data using different NoSQL database categories. However, this approach does not have a modeling engine for schema propositions. While in [6, 16], data modeling was proposed for MongoDB using JSON and UML Diagram class. These approaches were developed for MongoDB only which is considered vendor specific.

On the contrary, optimizing storage efficiency and retrieval performance for geographic data was given less attention in recent year despite its great importance. This is why techniques such as Modeling Technique for Geographic Applications [17] (OMT-G), GMOD [18], data modeling (GISER) [19] and object-oriented analysis (GeoOOA) [20] for geographic IS and OMTEXT was proposed to minimize data inconsistency in NoSQL databases.

In a similar concept, some contributors, such as technical experts from JSON [9] and mongoDB [6] explained some ways to achieve relatively good data modeling relationships. These approaches are however sort of proprietary, focusing on the functionalities of the database in which they set to promote. There is a need to have a generalized approach which can be adopted by at least one category of NoSQL databases [5, 7, 21].

As can be noticed, these approaches focus more on spatial data, which is in most cases, managed by graph databases or are concerned with fundamental principles only, leaving out the automated process of modeling. Generating schemas automatically not only simplify modeling process but also makes it more accurate and secure.

fathey.m.ye@gmail.com

3 Proposed Model

The proposed model is presented in this section. The model aims to minimize data modeling hardship faced by programmers with these parameters such as availability, consistency and scalability. The model is presented in three sections, namely, model architecture, algorithms and model components.

3.1 Model Architecture

The architecture of the proposed model is represented in Fig. 1. The rectangles used in the architecture represent phases or layers, arrow-line to indicate flow direction, curly brackets to signify document schema. Other symbols are labeled as shown in the diagram.

The architecture consists of four interconnected phases, with presentation layer (Client side) being the first. The user is expected to select from the selectable parameters, and supply values to the non-selectable parameters which require system peculiar inputs like entity name. Starting from phase 3, the model takes over and proceeds with semantic mappings of the entities based on the new generation cardinalities [11] and modeling guidelines [12]. The following algorithms explain the logic.

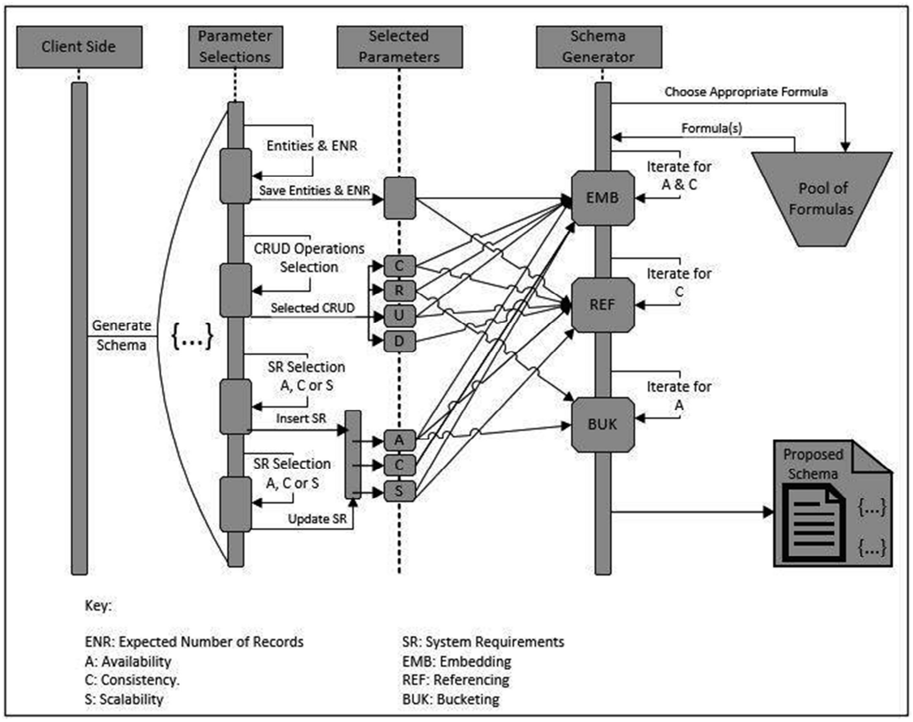

Fig. 1. Schema Proposition Model Architecture

fathey.m.ye@gmail.com

3.2 Schema Generator Algorithms

This section presents the step-by-step procedure which the proposed model follows in order to generate a desired schema.

```
Algorithm Schema Generator.
Input:  List of entities E and Expected number of records
        per entity ENR.
        CRUD Operations Cr (C, R, U, D).
        System Requirements SR(A=Availability,$C_{L,M,H}$= Con-
                              sistency S=Security).
Output: Schema for NoSQL document-store databases.
Definitions: $C_{L,M,H}$(Low, Medium and High Consistency)

begin
variables (E(E + ENR), CRUD, SR, i)
if E.size != Ø: then
  SR ← get preferred requirement
  Cr ← get selected CRUD operation Ø
  while i < E.size do
    ENR ← get ENR of an entity
      for each item in E do
        if SR = $C_H$ and (Cr = C or U) then
          if item(ENR)[i]>0 and item(ENR)[i]<1000,000 then
            embedding is preferred;
          else if item(ENR)[i] > 100,000,000 then
            referencing is preferred; but…
            call RollBack(series);
          end;
        else if SR = $C_L$ and (Cr = C or U) then
          referencing is preferred; but…
          call RollBack(parallel);
        else if SR = $C_m$ and (Cr = C or U) then
          referencing is preferred; but…
          call RollBack(partial);
        end;
        //Availability section (retrieval)
        if SR = A and (Cr = R) then
          pg ← newPg[i];
          Tpg = ROUND(ENR / RecordsPerPage);
          foreach(range(1, Tpg) as pg)
            if(pg=1 or pg=Tpg or(pg>=newPg and pg<=newPg))
              pg ← newPg (+ or - 1);
            end;
          end;
        end;
        //var a ← ENR + ROUND((ENR / 100) * 20) /
      end;
  end;
else
  return null;
end;
```

fathey.m.ye@gmail.com

```
Algorithm RollBack(method).
Input: Series, Parallel, Partial.
Output: Cluster availability.

Definition: $C$ = cluster and $_x$ is the cluster counter

begin
if method = series then
  Av = False
  for (i = 1; i <= 3; i++)
    Check availability of $C_1$, $C_2$, $C_3$ ... $C_n$
    if availability is = 100% ($C_{x1}C_{x2}...C_{xn}$) then
      Av = True;
      return Av; set i = 4; exit();
    else
    repeat
  end;
else if method = parallel then
  Av = False
  for (i = 1; i <= 3; i++)
    Check availability of $C_1$, $C_2$, $C_3$ ... $C_n$
    if availability is = 50% ($1-(1-C_x)^2$) then
      Av = True;
      return Av; set i = 4; exit();
    else
    repeat
  end;
else if method = partial then
  Av = False
  for (i = 1; i <= 3; i++)
    Check availability of $C_1$, $C_2$, $C_3$ ... $C_n$
    if availability is = 75% (Formula as in Eq(3)) then
      Av = True;
      return Av; set i = 4; exit();
    else
    repeat
  end;

  end;
```

Next, we present the components that make up the mapping module which includes selections storage and formulas.

3.3 Model Components

The components for the semantic mapping module are presented here, starting with selections storage structure then followed by formulas.

fathey.m.ye@gmail.com

Selections Storage. This storage is used to save user selected parameters and values supplied to the parameters. These records are vital for the computation, semantic and systematic mappings. The storage structure is described as follows:

- projectID: this attribute serves as a unique identifier of a particular project. The model is designed to handle multiple modeling processes for one user, so, a unique Id is required to differentiate between projects that belong to a single user.
- UserID: in case the model repository is centralized, this ID will be used to differentiate between users, so that successfully generated schemas cannot be interchanged between users.
- Entities: this attribute holds the names of all entities.
- ENR: this stands for expected number of records, which are used to calculate the possibility of either embedding or referencing a particular document.
- CRUD: this attribute house the selected CRUD operations. The model will iterate based on the saved selections such as *C* for create.
- SR: this stands for system requirements. The user selects a priority requirement such as availability, consistency or scalability.
- Relations: this attribute saves the relationship between entities.
- Active: this indicates if an entity is related to other entities or not.
- Flag: this attribute saves the computational status of all entities, where the values 0 and 1 represent not attended and completed respectively.

Formulas. We introduced some mathematical formulas to provide accurate systematic mappings, especially when checking cluster or document availability. These formulas are presented as follows.

$$C_s = A_x A_y \tag{1}$$

Equation 1 is the formula for highest level of consistency. This means all nodes (availability *A* of cluster $_x$ and cluster $_y$) must be available for an operation to take place, otherwise roll back function is called to retry or terminate the operation if the number of trials are more than 3 times or any number defined by the user.

$$C_L = 1 - (1 - A_x)^2 \tag{2}$$

If the choice of consistency is low $\mathrm{C_L}$, then parallel operation is allowed. Which means if cluster $_x$ fails to report, cluster $_y$ can take over and cluster $_x$ is updated upon its next availability.

$$C_{m_{c,n}} = \sum_{i=0}^{n} \frac{c!}{i! \times (c-i)!} \times C^{(c-i)} \times (1-A)i \tag{3}$$

On the other hand, if the system requirement is consistency *C* and the level is medium $_m$, then Eq. 3 is used to calculate the availability of each cluster $_{C..N}$. For instance, if a system has $_C$ number of clusters, under this formula, the system is

fathey.m.ye@gmail.com

considered to be available when at least $_{C-N}$ components are available, which means not more than $_N$ clusters can fail.

In the following section, we present and discuss the results of the proposed model as well as the results of the traditionally generated schemas.

4 Results and Discussion

In this section, a comparison is made between the schema generated by the proposed model and the schema produced using conventional method. One parameter (medium $_m$ consistency) was considered for this experiment. Also, C of the CRUD operations are selected to apply and test the consistency requirement using create and update queries.

By referring to Fig. 2, it can be noticed that, when data size is around 5000 to 100,000 records, the difference between the schema generated by the proposed model and the schema produced using the traditional method is insignificant with regards to the speed of data creation. However, as data size increases, the gap begins to be substantially high. The proposed model maintained its speed of data creation from around 4 s to 15 s across different sizes of records, while the traditionally generated schema performs very low, up to nearly 60 s as data size increases exponentially (10, 20 and 50 million records).

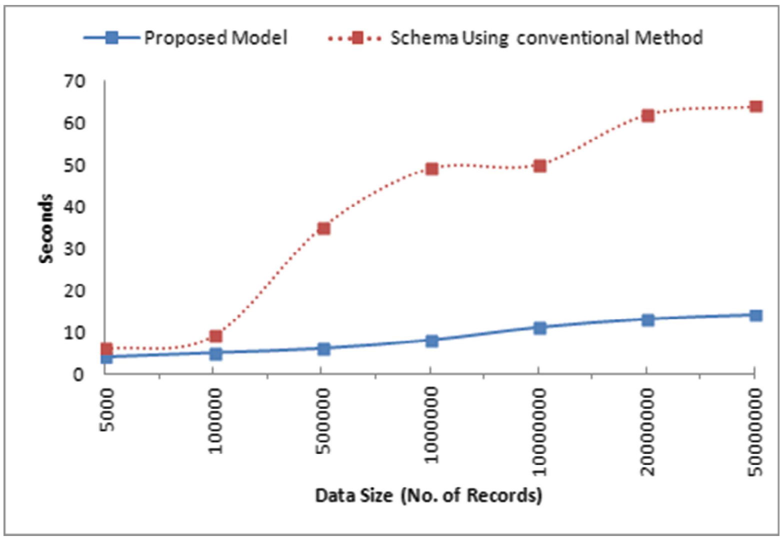

Fig. 2. Rate of record creations *C*: a proposed model and conventional method schemas comparison

fathey.m.ye@gmail.com

This significant difference can be attributed to appropriate use of embedding and referencing when modeling NoSQL document-stores. For example, if entity E_x is related to E_y and E_z, embedding should not be applied since E_y can function in the absence of E_z as defined by C_m formula (Eq. 3). In this case referencing would be the best choice here, since it allows documents to be separated and referenced using ID.

With this significant improvement in NoSQL automated modeling, we can conclude that, despite the existence of several NoSQL modeling techniques, the model that does not require much technical skills is need in practice not only to alleviate modeling difficulties but also to accurately relate entities and propose desired schema.

5 Conclusion and Future Focus

In this paper, we presented a schema proposition model for modeling NoSQL document-store databases. The model consists of a conceptual architecture, algorithms, repository and mathematical formulas which are used to determine cluster availability. The proposed model aims to minimize difficulties faced by database modelers and erroneous implementation. To assess its performance, two separate schemas were generated. The first schema was generated using the traditional method while the proposed model produced the second schema. Both the schemas were implemented using MongoDB database. Records were added in batches to each database simultaneously. Results show a significant decrease in time taken to save records into the database that was generated using the proposed model. In addition to this performance increment, the modeling time is considerably reduced, and modeling process is simplified. This advancement may continue to be the focus of research in the near future.

References

1. Garcia-Molina, H., Ullman, J.D., Widom, J., Özsu, M., Valduriez, P., Connolly, T., Begg, C., Elmasri, R., Navathe, S.B., Lin, M., Tsuchiya, M., Member, S., Mariani, M.P., Sharma, M., Singh, G., Virk, R.: Database systems: a practical approach to design, implementation, and management. Int. J. Comput. Appl. Technol. **49**(4), 90–107 (2010)
2. Chouder, M.L., Rizzi, S., Chalal, R.: Enabling self-service BI on document stores. In: Workshop Proceedings EDBT/ICDT 2017 Joint Conference Venice, Italy (2017)
3. Atzeni, P., Bugiotti, F., Rossi, L.: Uniform access to NoSQL systems. Inf. Syst. **43**, 117–133 (2014)
4. Enríquez, J.G., Domínguez-Mayo, F.J., Escalona, M.J., Ross, M., Staples, G.: Entity reconciliation in big data sources: a systematic mapping study. Expert Syst. Appl. **80**, 14–27 (2017)
5. Varga, V., Jánosi, K.T., Kálmán, B.: Conceptual design of document NoSQL database with formal concept analysis. Acta Polytech. Hungarica **13**(2), 229–248 (2016)
6. William, Z.: 6 Rules of Thumb for MongoDB Schema Design. MongoDB, (2014). [Online]. https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1. Accessed 23 Jan 2017

fathey.m.ye@gmail.com

7. Atzeni, P.: Data modelling in the NoSQL world: a contradiction? In: International Conference on Computer Systems and Technology – CompSysTech 2016, June, pp. 23–24 (2016)
8. April, R.: NoSQL Technologies: Embrace NoSQL as a relational Guy – Column Family Store. DBCouncil (2016). https://dbcouncil.net/category/nosql-technologies/. Accessed 21 Apr 2017
9. CrawCuor, R., Makogon, D.: Modeling Data in Document Databases. United States: Developer Experience and Document DB (2016)
10. Mior, M.J., Salem, K., Aboulnaga, A., Liu, R: NoSE: schema design for NoSQL applications. In: IEEE Transactions on Knowledge and Data Engineering From 2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016, vol. 4347, pp. 181–192 (2016)
11. Imam, A.A., Basri, S., Ahmad, R., Abdulaziz, N., González-aparicio, M.T.: New cardinality notations and styles for modeling NoSQL document-stores databases. In: IEEE Region 10 Conference (TENCON), Penang, Malaysia (2017)
12. Imam, A.A., Basri, S., Ahmad, R., Aziz, N., Gonzlez-aparicio, M.T., Watada, J., Member, S.: Data modeling guidelines for NoSQL document-store databases. Computer Science Education. SI on Advancing Theory About the Novice Programmer. Taylor & Francis (2018). Under Review
13. Han, J., Haihong, E., Le, G., Du, J.: Survey on NoSQL database. In: Proceedings - 2011 6th International Conference on Pervasive Computing Applications, ICPCA 2011, pp. 363–366 (2011)
14. Alhaj, T.A., Taha, M.M., Alim, F.M.: Synchronization wireless algorithm based on message digest (SWAMD) for mobile device database. In: 2013 International Conference on Computer, Electrical and Electronic Engineering Synchronization, pp. 259–262 (2013)
15. Arora, R., Aggarwal, R.: Modeling and querying data in mongodb. In: International Journal of Scientific and Engineering Research (IJSER 2013), vol. 4, pp. 141–144 (2013)
16. Kaur, K., Rani, K.: Modeling and querying data in NoSQL databases. In: IEEE International Conference on Big Data, pp. 1–7 (2013)
17. Borges, K.A., Davis, C.A., Laender, A.H.: Omt-g: an objectoriented data model for geographic applications. Geoinformatica **5**(3), 221–260 (2001)
18. De Oliveira, J.L., Pires, F., Medeiros, C.B.: An environment for modeling and design of geographic applications. Geoinformatica **1**(1), 29–58 (1997)
19. Shekhar, S., Coyle, M., Goyal, B., Liu, D.-R., Sarkar, S.: Data models in geographic information systems. Commun. ACM **40**(4), 103–111 (1997)
20. Kosters, G., Pagel, B.-U., Six, H.-W.: Gis-application development with geoooa. Int. J. Geogr. Inf. Sci. **11**(4), 307–335 (1997)
21. Mior, M.J.: Automated schema design for NoSQL databases. In: Proceedings of 2014 SIGMOD PhD Symposium – SIGMOD 2014, Ph.D. Symposium, pp. 41–45 (2014)

fathey.m.ye@gmail.com