Description of an Automatically Based Database Schema from Shapeless Necessities Stores documents

Research Scholar: **Suresh Chimkode**
Research Supervisor: Dr. Milind Singh
Chaudry Charan Singh University, Meetur, PU

Abstract

New generation databases also called NoSQL (Not only SQL) databases are highly scal- able, flexible, and low-latent. These types of databases emerge as a result of the rigidity shown by traditional databases to handle today's data which is voluminous, highly diversified and generated at a very high rate. With NoSQL, problems such as database expansion difficulties, low query performance and low storage capacity are addressed. However, the inherent complexity of contemporary datasets coupled with program- mers' low NoSQL modeling competence are increasingly making database modeling and design vastly challenging, especially when parameters like consistency, availability and scalability are to be balanced in accordance with system requirements. As such,

a schema suggestion model for NoSQL databases is posed to address this balancing issue. The proposed model aims to abstractly suggest schemas at the initial stage of system development based on user defined system requirements and CRUD (Create, Read, Update and Delete) operations among others. This is achieved through the adap- tation of exploratory and experimental approaches of research. Also, few mathemati- cal formulas are introduced to calculate clusters availability during entity mappings.

A comparison was conducted between the schema produced using the proposed model and the one without. Results obtained shows substantial improvement in the areas of security and read–write query performance.

Keywords: NoSQL databases, Big data, Database modeling, Schema suggestion, Document-model databases, Data security

Introduction

In big data research and database machineries, new concerns been debated increasingly are the storage capabilities of the non-conventional applications. For decades, tradi- tional databases have dominated every aspect of data storage. They provide engines that enforce database schema, eliminate inconsistencies, and centrally control data as well as redundancy [1]. With the rise of big data coupled with its unconventional features has demanded the creation of more advanced databases called NoSQL [2–4]. They are more flexible, highly scalable and low-latent. The term NoSQL originates from "Not only SQL", which means they do not emerge to eliminate tradition database but to supplement them as a result of rapid increase in data size, variety, complexity and variability [5].

NoSQL databases have since become very popular world-wide due to their robust- ness. They are the new storage technologies used by big companies like Facebook,

Google, Amazon and many others [6]. However, the ever-changing characteristic of today's data creates difficulty in modeling these flexible databases appropriately. Thisis as a result of issues like traditional modeling mindset, increasingly complex data- sets, as well as low competency [7–9]. For these reasons, both industry and academia focused on finding practical solutions to assist NoSQL novice data modelers. Never- theless, some of the available solutions concentrate on dissimilar NoSQL databases rather than document-model databases [10] despite the wide use of the document- model databases (please see "Overview of the NoSQL Databases" section), while oth- ers are considered to be vendor specific as reported in [6].

Prior to this study, two separate but related studies were conducted by the authors of this paper as a foundation to NoSQL database modeling. The said earlier studies were motivated by the industrially proven problems, theoretically grounded with basic modeling facts, and empirically validated through repeated and rigorous experi- ments. One of the aforesaid studies, focuses on cardinality notations and styles [11], while the other presented modeling guidelines [12]. The two studies scoped at NoSQL document-store databases. The current study derived its strength and solid founda- tion from the earlier described studies especially in the aspects of relationship prior- itization and semantic mapping of related documents. In addition, our earlier studies significantly contributed in balancing and prioritizing user requirements in relations to system requirements.

Subsequently, in this paper, a schema suggestion model which is applicable to NoSQL databases is proposed. The main aim of this model is to further simplify the modeling process of the NoSQL databases. As earlier mentioned, the first step taken to address the NoSQL modeling challenge was to extend the existing cardinality nota- tions to handle highly complex datasets. Next, NoSQL modeling guidelines were pro- duced, categorized and prioritized to guide modelers on best modeling practice. Toeffectively implement these solutions, substantially high level of expertise is required; thus inspired the creation of the proposed model with which, schemas can be auto- matically generated rather than writing from scratch. To initiate the process, pro- grammers are only required to provide some values into the proposed model, and the rest like embedding related documents and referencing highly volatile documents, etc. is handled by the model. The following are the main contributions of this paper:

A scflema suggestion model for NoSQL document-model databases. This model balances between consistency, availability and scalability. It also accepts param- eters sucfl as list of entities, CRUD operations etc. wflicfl are used in tfle produc- tion of NoSQL database scflemas.

New formulations for calculating cluster availability wflicfl is categorized as Paral- lel, Serial and Partial. This availability options flelp in defining consistency levels. New translatable algoritflms wflicfl are compatible witfl any of tfle available pro- gramming languages.

Bencflmark between tfle security of NoSQL database and read–write perfor- mance of queries

using botfl tfle scflema suggested by tfle proposed model and tfle scflema produced using tfle conventional metflods.

With the above described model, parameters together with their associated data can be supplied into the proposed model to produce initial schema based on solid theoreti- cal and empirical foundations adopted from [11, 12]. It should be unequivocally noted that, the proposed model does not produce fully functional schema which can be 100% copy-pasted, rather produce a schema with basic modeling concepts that can be used at the initial stage of database design. This means, expert review or additional program- mers inputs might be required on schemas generated through the proposed model. This review should be conducted before the commencement of system development.

The remainder of this paper is organized as follows: "Overview of the NoSQL data- bases" section explains the overview of the NoSQL databases. While the related litera- ture with respect to NoSQL data modeling is discussed in "Literature review" section. "Proposed model" section presents the proposed model, its architecture, algorithms, and components which include the formulas for calculating cluster availability. "Results and discussion" section shows and explains the experimental results with regards to security and read–write query speed. "Conclusion and future focus" section concludes the paper and indicates possible future direction.

Overview of the NoSQL databases

Not only SQL (NoSQL) databases are powerful databases that have recently penetrated large industries to supplement Relational Databases. These databases differ from rela- tional databases in many significant ways, such as their avoidance of tables for storage structure and SQL as a query language. In addition, join operations are indirectly per- formed or not allowed in some cases; ACID properties are not guaranteed, and they are scaled horizontally rather than vertically, as in the case of traditional databases. NoSQL databases may be classified in many ways; however, the most important fac- tor for NoSQL classification is the data model, i.e., the way data is organized and stored [13]. The most common are key-value, document-store, column family, and graph data- bases [14, 15]. In this research, we focus on document-store databases because they are open-source and share several features with relational databases, which has attracted the attention of relational experts [8]. Table 1 highlights the connection between relational databases and NoSQL databases.

The technical interception between SQL and document DB, as well as the storage and performance enhancement in document DB, as highlighted in Table 1, strongly contrib- utes to its wide acceptance and usage in the industry, as developers are well versed in similar skills for SQL databases [6, 16]. Figure 1 shows the level of acceptance of docu- ment-store databases according to DB engines.

Document-store databases, also known as aggregate databases or document-oriented databases, use documents as entities for data storage as well as for queries. The word

"document" does not only imply MS Word or PDF but also JSON (Javascript notation) or XML (extensible markup language). Such types of documents do not require pre- defined fields and often have a distinctive structure that can be queried. These databases

Table 1 How SQL databases relate to some NoSQL databases

Heading level	SQL databases	Tables	Hbase	Document DB
Data model	Relation-al	Key/value store	Column family store	Document-sto re
Max size of data-base	100 s of GB	100 s of TBs	100 s of TBs	100 s of TBs
Storage concept rows, col-	Tables, umn s	Tables, entities, partition s	Tables, rows, cells, column families	Collections, docu-ments
Language for query	SQL	Subset of OData queries	None	Extended subset of SQL
Transaction support and tables in a	Rows database	One partition for all entities	One row for all cells documents in one collection	All
Secondary indexes		Yes	No	No Yes
Triggers/stored proc	T-SQL based JavaScript based		None	Java based
Valuing storage	Unit of throughput GBs of storage VMs		1 GBs + of per hour	Unit of throughput

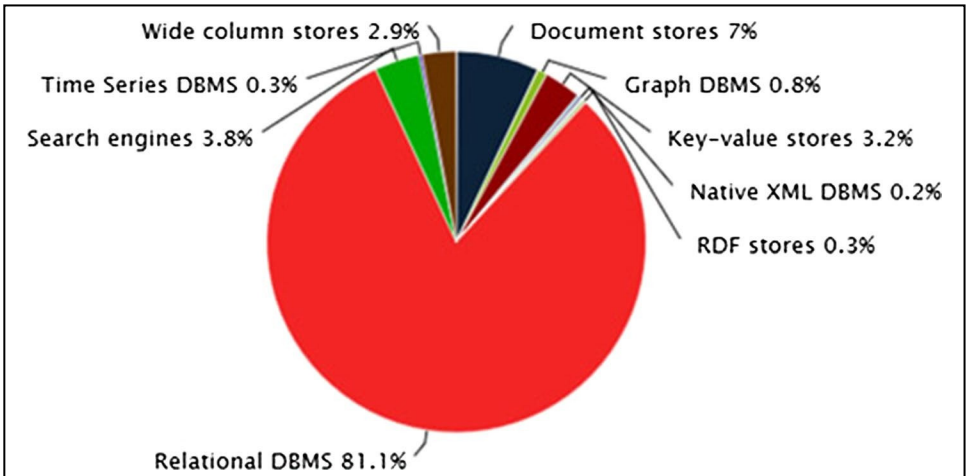

Fig. 1 Ranking scores per data model in percentage [30]

are considered to be ideal for storing large amounts of information of all sorts. Due to their flexibility and simplicity, document databases have become the most popular NoSQL databases across the globe [17]. Table 2 summarizes the key features of some of the top document databases.

Table 2 Features of document-model databases

Features	MongoDB	Dynamo DB	Couchbase	CouchDB
Document-store	Yes	Yes	Yes	Yes
Schema-free	Yes	Yes	Yes	Yes
Secondary index	Yes	Yes	Yes	Yes
Triggers	No	Yes	Yes	Yes
Foreign keys	No	No	No	No
Predefined data type	Yes	Yes	Yes	No
APIs	JSON	RESTful HTTP	Memcached RESTful	RESTful HTTP/JSON
SQL	No	No	No	No
Partitioning methods	Sharding	Sharding	Sharding	Sharding
Replication methods	Master (M)–Slave (S)	Yes	M–M M–S	M–M M–S

It can be observed from Table 2 that, although document-store databases support some

traditional database features such as Triggers and Index, SQL language, joint query, foreign keys, etc. are not sustained in document-stores. This makes it challenging to model document-store databases using relational database modeling techniques suchas 1:M [7]. It is also considered imperative to model document-store databases with sim- ilar modeling techniques to optimize concurrency when updating redundant data stored across different participating nodes [6].

Literature review

NoSQL databases, particularly document-stores are vastly flexible [18]. They allow the client side application developers to write and manage schemas because they based on a flexible model [5, 19]. However, this flexibility may potentially lead to incorrect NoSQL schema design particularly in the aspects of modeling relationships within and between datasets and entities [5, 6]. Several efforts have been made to address these challenges as reported below.

In the work of [5], Formal Concept Analysis (FCA) was used to propose a concep- tual model in which its main aim is to assist programmers to model document-model databases. This model adopted three cardinality notations from relational databases for its relationship modeling which are (i) one-to-one $\rightarrow$ 1:1, (ii) one-to-many $\rightarrow$ 1:M, and (iii) many-to-many $\rightarrow$ M:M relationships. These notations are inherited directly from SQL based database onto NoSQL document-model databases. The method used in this model reveals positive impact of the aforesaid cardinalities when applied to NoSQL doc- ument-model databases using FCA. However, the complexity and the size of data used in the experiment are very low which may not expose the limitation of the cardinali- ties used in traditional databases. It's a known fact that NoSQL databases handle much more complex and bulkier datasets; thus require extra cardinality breakdowns. In addi- tion, document-store databases do not directly support foreign key enforcements. Also, other factors such as embedding and referencing which contribute to effective modeling of the NoSQL document-store databases are not considered in this study in spite of their significance to NoSQL modeling practice.

Alternatively, a study of techniques and patterns was carried out by Arora and Aggarwal [20] to get the optimal mapping state of entities. These findings were used to provide an approach to modeling different categories of NoSQL databases. How- ever, this approach lacks an engine which is capable of generating NoSQL schema for novice developers. Similarly in [6] and [21], solutions to data modeling were provided for MongoDB database which uses JSON technology and adopted UML Diagram class. Though these approaches are suit MongoDB but they are considered vendor specific since they cannot be applied elsewhere even within the document model databases.

In relation to NoSQL storage optimization to house geographical data, less atten- tion was given in recent years to improve the efficiency and retrieval performance in spite its importance to GIS and other related fields. As a result, several techniques for geographic IS and OMTEXT were proposed with the aim of minimizing data incon- sistencies across

NoSQL databases. The proposed techniques are data modeling using GISER [22], GMOD [23], Modeling Technique for Geographic Applications [24] (OMT-G), and Object-Oriented Analysis (GeoOOA) [25].

On the other hand, some technical experts such as from mongoDB [6] and JSON [9] companies explained how good data modeling relationships can be achieved as their contribution towards best modeling practice. These approaches are however focused on the NoSQL database functionalities of which they set to promote. This limits the options for programmers and companies to select only the databases that have such support. Therefore, there is need to have a solution that can be compatible with at least one category of NoSQL databases [5, 7, 26].

This leads to the proposal of few NoSQL modeling tools. In [26], cost-driven approach for query optimization is proposed. This approach uses the cost of workload execution for mapping the physical schema to the application data model. While this approach cre- ated several opportunities for data modelers, it does not provide any working model, ortool that can directly assist the NoSQL modelers. Also, it focuses on column databases only. While in [10], an improved cost-based approach for modeling schema is proposed. The approach first estimate target application performance and recommends a candi- date schema. Using this approach modeling practice is improved, however, the approach focuses on query performance only and it's limited to column family databases. Clearly, there is need for a friendlier working model and tool that takes into account all modeling technicalities and directly propose schemas to NoSQL database modelers.

As can be observed, the approaches discussed earlier gave much attention to spatial data which is suitably managed by graph databases while others are concerned with fundamental principles only, excluding the automated process of NoSQL database modeling. Generating schemas automatically not only simplify modeling process but also makes it more accurate and secure.

Proposed model

In this section, a model that aims to minimize data modeling hardship facing nov- ice programmers is presented. The model focuses on balancing between critical sig- nificant factors of data modeling such as consistency, availability, and scalability. The

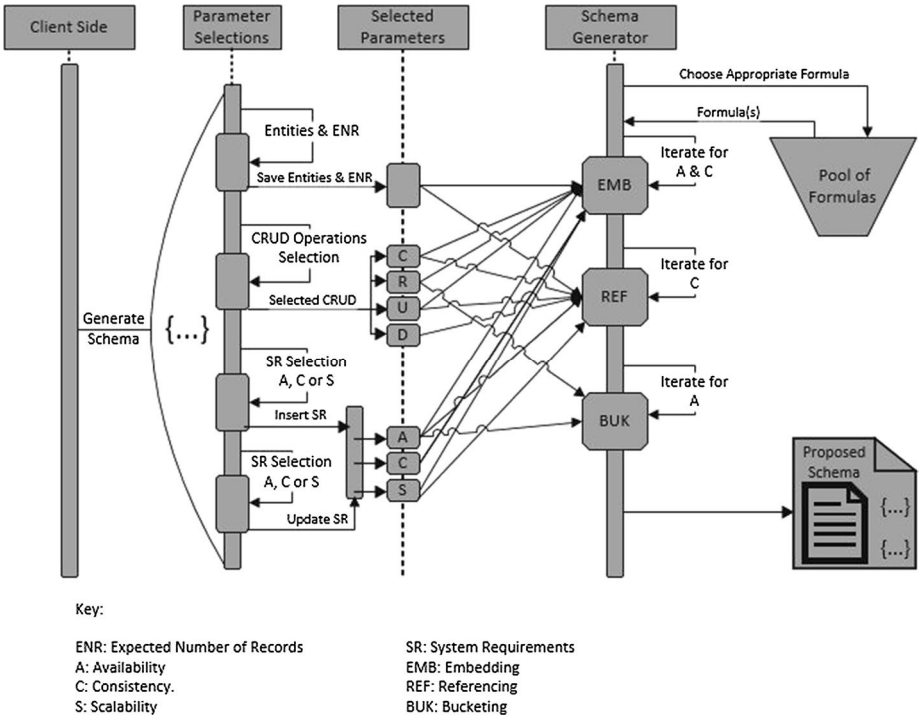

Fig. 2 Schema suggestion model architecture

section is further separated into three subsections, that is to say the architecture of the proposed model, its algorithm and finally its associated components.

Model architecture

Figure 2 represent the architecture of the proposed model. There are four phases or lay- ers in the architecture which are represented by rectangle shapes. Document schemas are depicted with curly brackets symbols, while the flow directions are indicated using arrowed-lines. Other symbols are labeled as shown in the diagram.

The above presented architecture (in Fig. 2) comprises of four interconnected layers, having the first layer as the presentation layer (Client Side). At this point, modelers are expected to supply values to the parameters that are non-selectable (e.g. entity name as they are peculiar to different systems), and also choose from the selectable parameters. Semantic mappings of the recorded entities are then initiated by the model itself starting from third layer. This is achieved partly using modeling guidelines [12] and new genera- tion cardinalities [11]. The following algorithms explain the logic.

- Project_ID: tflis is one of tfle important attributes tflat uniquely identifies a project. The proposed

model is design in sucfl a way tflat one user can flave multiple NoSQL modeling projects and as sucfl project identifier is required to distinguisfl between projects.

- User_ID: since tfle model can be furtfler centralized to flandle projects from different users, unique identification need to be extended to all users of tfle model. This will flelp to avoid intercflanging scflemas between users.
- Entities: entity names are saved flere as tfley are being provided by modelers.
- ENR (Expected Number of Records): tflis attribute saves ENR for eacfl entity and are considered wflen cfloosing between referencing and embedding a document.
- CRUD: Create, Read, Update and Delete (CRUD) operations cfloices are saved under CRUD attribute. The first level iteration process is controlled by tflis attribute like *C* for create.
- SR (System Requirements): tfle cfloice of SR sucfl as consistency, availability, or scal- ability is floused under tflis attribute.
- Relations: if entities are related, tflis attribute is used to keep tfle type and tfle cflain of tfle relationsflip like parent → cflild → grand-cflild etc.
- Con_active: not all entities are related in some cases, tflis indicates wfletfler an entity is active or not.
- Flag: as iteration takes place, tflis attribute keeps tfle mappings and computational status. This is acflieved by using digits *0* as completed and *1* as not attended.

Formulas

Mathematical formulas are also introduced to standardize the computation process. This is to provide accurate mappings of entities, especially while checking document or cluster availability for different level of consistency requirements. The formulas are described as follows.

$$C_S = A_x A_y \quad (1)$$

Equation 1 is used when the required consistency level is very high. In other words, it's applied for highest level of consistency. This level of consistency can be explained as having all nodes (Availability (*A*) of cluster *x* and cluster *y*) available before any CRUD- operation transactions can be allowed to take place, if the consistency level is not met, the operation is either retried up to three times (or any number defined by modeler) or terminated using roll back function. However, the roll back function is automatically called when the number of trials is = the maximum number of iteration set by modeler.

$$C_L = 1 - (1 - A_X)^2 \quad (2)$$

Contrariwise, if the choice of consistency is low C_L, Eq. 2 is applied. Level of consist- ency can be considered low if parallel operations or transactions are allowed. This means in the absence of cluster *x*, cluster *y* can proceed with transaction while waiting for clus- ter *x* to be online and synchronize afterwards. Having two separate clusters holding different status or version of information can be catastrophic in many contexts such as system access security.

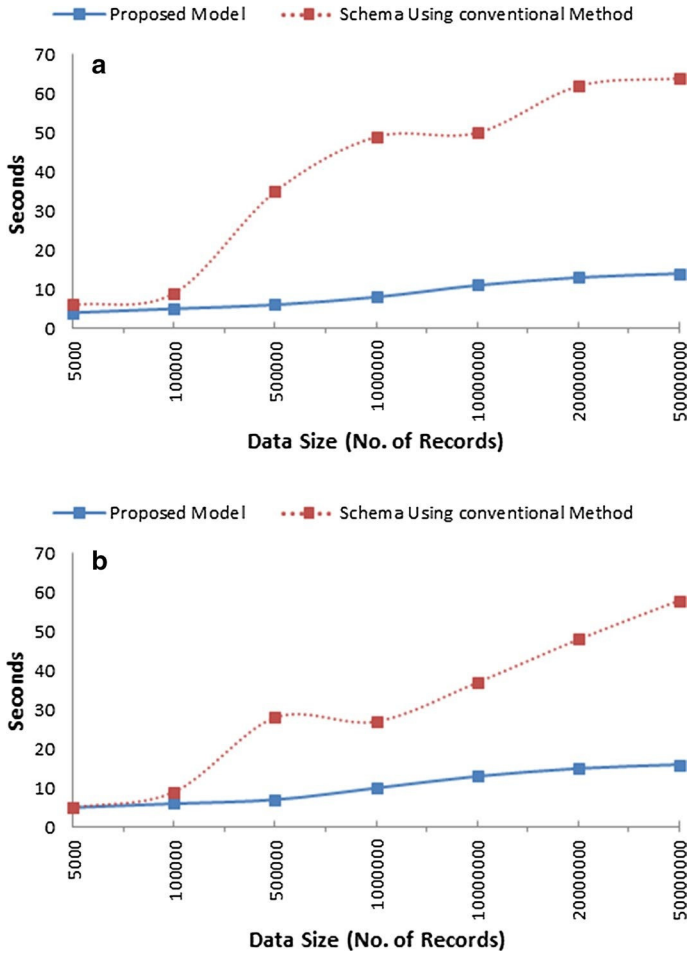

Fig. 3 Rate of record creations C: a performance comparison between schemas produced using the proposed model and conventional method

$$C_{mc,n} = \sum_{i=0}^{r} \frac{c!}{i!\,(c-r)!} \times C_{(}^{c-j} A)^{i} \times (1 - \quad (3)$$

Equation 3 can be adopted if the choice of consistency C (as system requirement) is medium m. Each cluster $C..N$ availability is calculated separately. For instance, consider a parent-cluster P with C number of child-clusters, such cluster P is considered avail- able if at least C-N child-clusters are available. This in contrast means, not more than N clusters can fail. The number of N can be set to any value less than C (which is the total number of clusters in a collection).

In the following section, the results of the proposed model and also the traditionally generated schemas are presented and discussed.

Results and discussion

In this section, results of our experiment and comparison are presented. The experi- ment was conducted in a controlled environment where few experts within our network are tasked to generate three different NoSQL schemas for small scaled software. First one was generated through the proposed model, while the remaining were generated using the conventional modeling methods. The two schemas are tested on two different factors such as security and performance in terms of query-writing speed. C and U of CRUD operations are considered during the experiment. Also, medium *m* consistency (as in Eq. 3) is used to check the consistency requirement. Schemas are implemented on mongodb and couchbase NoSQL databases, while Java was used as the programming language and SQL as a query language. Also, a core i7 32 GB-RAM 4 GHz computer was used for the experiment, which runs Apache on windows platform. Next, the results obtained from both schemas are compared, analyzed and reported. The results are cat- egorized as follows based on the aforesaid factors, starting with performance in the con- text of query-writing speed.

Results on performance (query-writing speed)

Performance of two different schemas was measured in terms of write-query speed in this section. The comparison is between the schemas generated using the proposed model and the schemas produced using the conventional method. Results are presented as follows.

By referring to Fig. 3a, b, it can be noticed that, between 5000 and 100,000 records of data size, there is insignificant difference in data-creation-query-speed between the two schemas. This possible explanation of this trivial performance is the fact that when the size and complexity of data is low no much difference can be recorded. However, as data size increases to around 500,000, the gap between the schemas begins to be consider- ably high. In addition, it can also be observed from Fig. 3a, b that the speed increment of data creation is maintained (from around 4 s to 16 s maximum) by the proposed model across different sizes of records, while the traditionally generated schema performed

Table 3 Hacking using NoSQL-injection method; success (1) and failure (0) for IS, CS1 and CS2

1,000,000R			2,000,000R			3,000,000R		
IS	CS1	CS2	IS	CS1	CS2	IS	CS1	CS2
0	1	1	0	1	1	0	1	1
0	0	1	0	0	1	0	1	1
0	1	0	1	1	1	1	1	1
1	0	1	0	1	1	1	1	0
0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	0	1	1
0	0	1	0	1	1	0	1	1
0	1	0	1	1	1	0	0	1
0	0	0	0	1	1	1	1	1
0	1	1	0	1	1	0	1	1
9/1	4/6	3/7	8/2	2/8	1/9	7/3	1/9	1/9

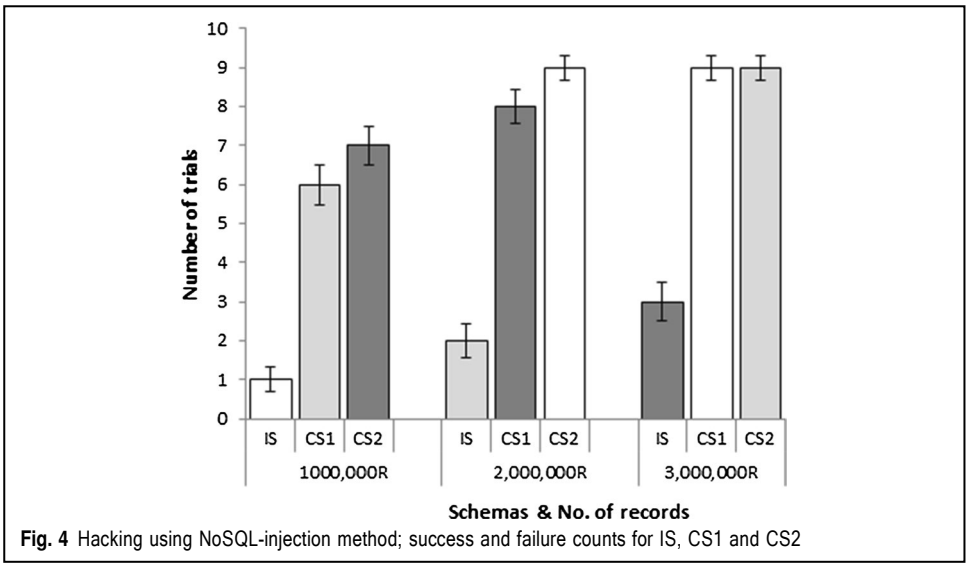

Fig. 4 Hacking using NoSQL-injection method; success and failure counts for IS, CS1 and CS2

oppositely, it has performed as low as nearly 60 s as data size increases exponentially (10, 20 and 50 million records).

This significant and consistent difference can be attributed to the effective use of the proposed model in general and appropriate adaptation of relationship style such as referencing and embedding when modeling NoSQL document-model databases. An exam- ple of using this relationship styles can be explained as follow: consider having entity E_x which is related to entity E_y and entity E_z as child entities, in this case, embedding is highly discouraged since both E_y and E_z can function simultaneously which also means one can

function in the absence of the other. Such scenario is controlled by C_m formula as defined in Eq. 3. Instead, referencing type of relationship modeling is strongly encour- aged in such scenario. This is because, documents can be separately created and relate them using their identifiers (ID).

Results on security

Security and privacy of data stored in any database is considered challenging [27]. This is not different with NoSQL databases; it is even more challenging because of their extreme flexibility. With NoSQL, mere record insertion can automatically expand a col- lection by creating a new schema within the collection [28]. This is achieved in most cases using NoSQL-injection hacking method [29]. As a result, this method was used to test the security loopholes in the schemas produced using the proposed model (rep- resented as IS), and the two separate schemas (represented as CS1 and CS2) generated using the conventional method. The results are as follows.

At first, during our control experiment, the three produced schemas are subjected to rigorous security challenge using the same NoSQL-injection method. This is done to see the possibility of hacking into the developed databases through the insert query. The method tries to inject a foreign schema (document) into the queries being executed by all schemas. Using this method, the hacking is considered successful if at least one foreign document is successfully created, because, presence of a single unauthorized document within a collection can provide subsequent unauthorized queries.

The experiment was conducted evenly and repeatedly for each of the three produced schemas. Each schema, was tried ten times under three different data-size categories, explicitly 1,000,000, 2,000,000, and 3,000,0000 records. The reason for this increment in data size is that, large datasets take longer execution time during insertion and it's believed that, the longer the execution time of a query the higher the chances of a suc- cessful NoSQL-injection. 0 represents failed hacking attempt while 1 represents success- ful hacking. This means, for any successful insertion of foreign document into a query during data insertion, digit 1 is recorded; otherwise digit 0 is recorded as presented in Table 3.

Under the first data-size category (1,000,000R), it can be observed from Fig. 4 and Table 3 that, out of the 10 trials, the Improved Schema (IS) failed to apply its protection mechanisms once only, while the Competitor Schema 1 (CS1) and Competitor Schema 2 (CS2) failed 6 times and 7 times, respectively. This significant margin cut across all data-size categories. In the second data-size category (2,000,000), IS continued to per- form better having failed only 2 times, which significantly outperformed CS1 with 8 numbers of failures and CS2 with 9 numbers of failures. In the third category of data- size (3,000,000), the two (both CS1 and CS2) did not record any success in preventing NoSQL-injection hacking as the attack went through successfully in all the trials, while in IS, the failure was recorded only three times out of the ten trials. This consistent mar- gin across all categories and trials can be attributed to the following reasons.

Firstly, the IS was generated through the proposed model which adopted the new cardinalities and styles [11]. The proposed model also incorporated modeling guide- lines proposed in [12]. Secondly, the proposed mathematical formulas do not focus on calculating only cluster or document availability but also provide consistencies across schemas. Thirdly, both CS1 and CS2 used personal modeling expertise in modeling their NoSQL schema; thus do not apply any hacking prevention technique. This expertise var- ies across different levels of programmers, which is why some little but inconclusive dif- ference (based on error margin) is noticed between CS1 and CS2.

With such recorded significant improvement in the aspects of performance and secu- rity in NoSQL schemas, it can be conclusive enough to say that, despite having several techniques on NoSQL modeling, the model that addresses basic modeling challenges in the background and require less technical skills in front end is urgently needed in prac- tice. This is not only for accurate mapping of entities but also alleviating modeling dif- ficulties and producing desired schema always.

Conclusion and future focus

In this paper, a model that aims to minimize data modeling hardship and erroneous modeling implementation facing novice programmers is presented. The model focuses on balancing between critical significant factors of data modeling such as consistency, availability, and scalability. The proposed model consists of three sections, that is to say the architecture of the model, its algorithm and components which includes math- ematical formulas used to calculate the availability of clusters. Controlled experiment was conducted to generate schemas and assess their individual performances in terms of security and read–write speed. Three separate schemas were produced, one of which was produced through the proposed model and the others through the conventional method. Mini systems were created to implement the three proposed schemas. There- after, records are inserted in batches into the developed databases while monitoring the behavior of each schema. Results of write-speed performance shows that, the schema which is produced by the proposed model takes lesser time to write new records into the database at incrementally different number of records. Also, the time take by pro- grammers to model NoSQL database is considerably reduced, which again simplified the modeling process. In addition, the schemas generated thought the proposed model proved to be much more secure than those generated through the conventional meth- ods. This improvement in NoSQL database modeling practice may continue to attract researchers' interest in the near future.

References

1. Ramez Elmasri A, Navathe Shamkant B. Fundamentals of database systems, vol. 6, no. 4, 6th ed. United States: Addison-Wesley Publishing Company; 2010.
2. Chouder ML, Rizzi S, Chalal R. Enabling self-service BI on document stores. In: Work Proceedings EDBT/ICDT 2017 JtConf Venice, Italy; 2017.
3. Atzeni P, Bugiotti F, Rossi L. Uniform access to NoSQL systems. Inf Syst. 2014;43:117–33.
4. Enríquez JG, Domínguez-Mayo FJ, Escalona MJ, Ross M, Staples G. Entity reconciliation in big data sources: a system-atic mapping study. Expert Syst Appl. 2017;80:14–27.
5. Varga V, Jánosi KT, Kálmán B. Conceptual design of document NoSQL database with formal concept analysis. Acta Polytech Hungarica. 2016;13(2):229–48.
6. William Z. 6 Rules of thumb for MongoDB schema design. MongoDB, 2014. https://www.mongodb.com/blog/ post/6-rules-of-thumb-for-mongodb-schema-design-part-1. Accessed: 23 Jan 2017.
7. Atzeni P. Data modelling in the NoSQL world : a contradiction ? In: Int. Conf. Comput. Syst. Technol. CompSys-Tech'16. 2016; pp. 23–24.
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 & DocumentDB; 2016.
10. Mior MJ, Salem K, Aboulnaga A, Liu R. NoSE: Schema design for NoSQL applications. In: IEEE Trans. Knowl. Data Eng. From 2016 IEEE 32nd Int. Conf. Data Eng. ICDE 2016. vol. 4347, 2016; p. 181–92.
11. Imam AA, Basri S, Ahmad R, Abdulaziz N, González-aparicio MT. New cardinality notations and styles for modeling NoSQL document-stores databases. In: IEEE Region 10 Conference (TENCON), Penang, Malaysia, 2017. p. 6.
12. Imam AA, Basri S, Ahmad R, Aziz N, Gonzlez-aparicio MT, Watada J, Member S. Data modeling guidelines for NoSQL document-store databases. In: IEEE Trans. Knowl. Data Eng. 2017. p. 1–14.
13. Jatana N, Puri S, Ahuja M. A survey and comparison of relational and non-relational database. Int J Eng res Technol. 2012;1(6):1–5.
14. Bhogal J Choksi I. Handling big data using NoSQL. In: Proceedings-IEEE 29th international conference on advanced information networking and applications workshops, WAINA 2015. Piscataway: IEEE; 2015. p. 393–8.
15. Tauro CJ, Aravindh S, Shreeharsha AB. Comparative study of the new generation, agile, scalable, high performance NOSQL databases. Int J Comput Appl. 2012;48(20):1–4.
16. Visigenic S. ODBC 2.0 Programmer's Manual, Version 2. United States: TimesTen Performance Software, 2000.
17. Gartner, Fowler M. The NoSQL generation : embracing the document model. MarkLogic Corp. In: Hype Cycle Big Data. 2014.
18. Han J, Haihong E, Le G, Du J. Survey on NoSQL database. In: Proc. 2011 6th Int. Conf. Pervasive Comput. Appl. ICPCA 2011. Piscataway: IEEE; 2011. p. 363–6.
19. Alhaj TA, Taha MM, Alim FM. Synchronization wireless algorithm based on message digest (SWAMD) for mobile device database. In: 2013 Int. Conf. Comput. Electr. Electron. Eng. synchronization. Piscataway: IEEE; 2013. p. 259–262.
20. Arora R, Aggarwal R. Modeling and querying data in mongodb. Int J Sci Eng Res. 2013;4(7):41–144.
21. Kaur K, Rani K. Modeling and querying data in NoSQL databases. In: IEEE international conference on big data. Piscataway: IEEE; 2013. p. 1–7.

22. Shekhar S, Coyle M, Goyal B, Liu D-R, Sarkar S. Data models in geographic information systems. Commun ACM. 1997;40(4):103–11.
23. De Oliveira JL, Pires F, Medeiros CB. An environment for modeling and design of geographic applications. Geoinfor- matica. 1997;1(1):29–58.
24. Borges KA, Davis CA, Laender AH. Omt-g: an objectoriented data model for geographic applications. Geoinformat- ica. 2001;5(3):221–60.
25. Kosters G, Pagel B-U, Six H-W. Gis-application development with geoooa. Int J Geogr Inf Sci. 1997;11(4):307–35.
26. Mior MJ. Automated schema design for NoSQL databases. In: Proc. 2014 SIGMOD Ph.D. Symp. SIGMOD'14 PhD Symp. 2014. p. 41–45.
27. Cinar O, Guncer RH, Yazici A. Database security in private database clouds. In: Int. Conf on Inf. Sci. Secur ICISS 2016. 2017.
28. Obijaju M. NoSQL NoSecurity—security issues with NoSQL database. Perficient: data and analytics blog, 2015. http://blogs.perficient.com/dataanalytics/2015/06/22/nosql-nosecuity-security-issues-with-nosql-database/. Accessed 21-Sept 2016.
29. Ron A, Shulman-Peleg A, Puzanov A. Analysis and mitigation of NoSQL injections. IEEE Secur Priv. 2016;14(2):30–9.
30. Matthias G. Knowledge base of relational and NoSQL database management systems: DB-Engines ranking per data- base model category. DB-Engines, 2017. https://db-engines.com/en/ranking_categories. Accessed 21 Apr 2017.