


RESEARCH

Open Access



Automatic schema suggestion model for NoSQL document-stores databases

Abdullahi Abubakar Imam^{1,2,3*} , Shuib Basri^{1,2}, Rohiza Ahmad^{1,2}, Junzu Watada^{1,2} and María T. González-Aparicio⁴

*Correspondence:
aiabubakar3@gmail.com;
abdullahi_g03618@utp.edu.my

¹ Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Seri Iskandar, Perak, Malaysia
Full list of author information is available at the end of the article

Abstract

New generation databases also called NoSQL (Not only SQL) databases are highly scalable, 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 programmers' 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 adaptation of exploratory and experimental approaches of research. Also, few mathematical 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, traditional 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 robustness. 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. This is as a result of issues like traditional modeling mindset, increasingly complex datasets, 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. Nevertheless, 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 others 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 experiments. 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 foundation from the earlier described studies especially in the aspects of relationship prioritization 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 notations to handle highly complex datasets. Next, NoSQL modeling guidelines were produced, categorized and prioritized to guide modelers on best modeling practice. To effectively implement these solutions, substantially high level of expertise is required; thus inspired the creation of the proposed model with which, schemas can be automatically generated rather than writing from scratch. To initiate the process, programmers 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 schema suggestion model for NoSQL document-model databases. This model balances between consistency, availability and scalability. It also accepts parameters such as list of entities, CRUD operations etc. which are used in the production of NoSQL database schemas.

New formulations for calculating cluster availability which is categorized as Parallel, Serial and Partial. This availability options help in defining consistency levels.

New translatable algorithms which are compatible with any of the available programming languages.

Benchmark between the security of NoSQL database and read–write performance of queries using both the schema suggested by the proposed model and the schema produced using the conventional methods.

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 theoretical 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 programmers 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 databases](#)” section explains the overview of the NoSQL databases. While the related literature 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

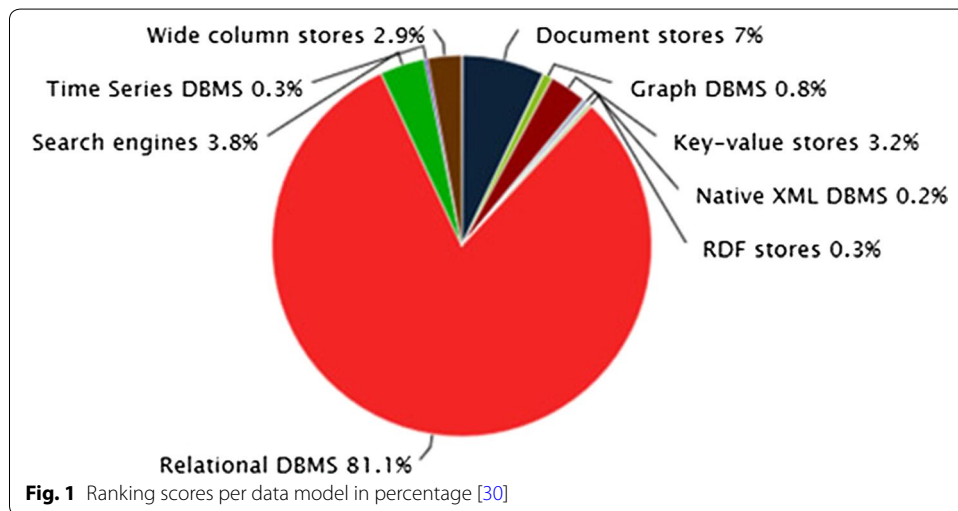
Not only SQL (NoSQL) databases are powerful databases that have recently penetrated large industries to supplement Relational Databases. These databases differ from relational databases in many significant ways, such as their avoidance of tables for storage structure and SQL as a query language. In addition, join operations are indirectly performed 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 factor 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 databases [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 contributes 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 document-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 predefined 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-store
Max size of data-base	100 s of GB	100 s of TBs	100 s of TBs	100 s of TBs
Storage concept	Tables, rows, col- umns	Tables, entities, partitions	Tables, rows, cells, column families	Collections, docu- ments
Language for query	SQL	Subset of OData queries	None	Extended subset of SQL
Transaction support	Rows and tables in a database	One partition for all entities	One row for all cells	All documents in one collection
Secondary indexes	Yes	No	No	Yes
Triggers/stored proc	T-SQL based	None	Java based	JavaScript based
Valuing	Unit of throughput	1 GBs of storage	GBs of storage + VMs per hour	Unit of throughput



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	DynamoDB	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 such as 1:M [7]. It is also considered imperative to model document-store databases with similar 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 conceptual 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 → 1:1, (ii) one-to-many → 1:M, and (iii) many-to-many → 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 document-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 cardinalities 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 addition, 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. However, 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 attention 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 inconsistencies 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].

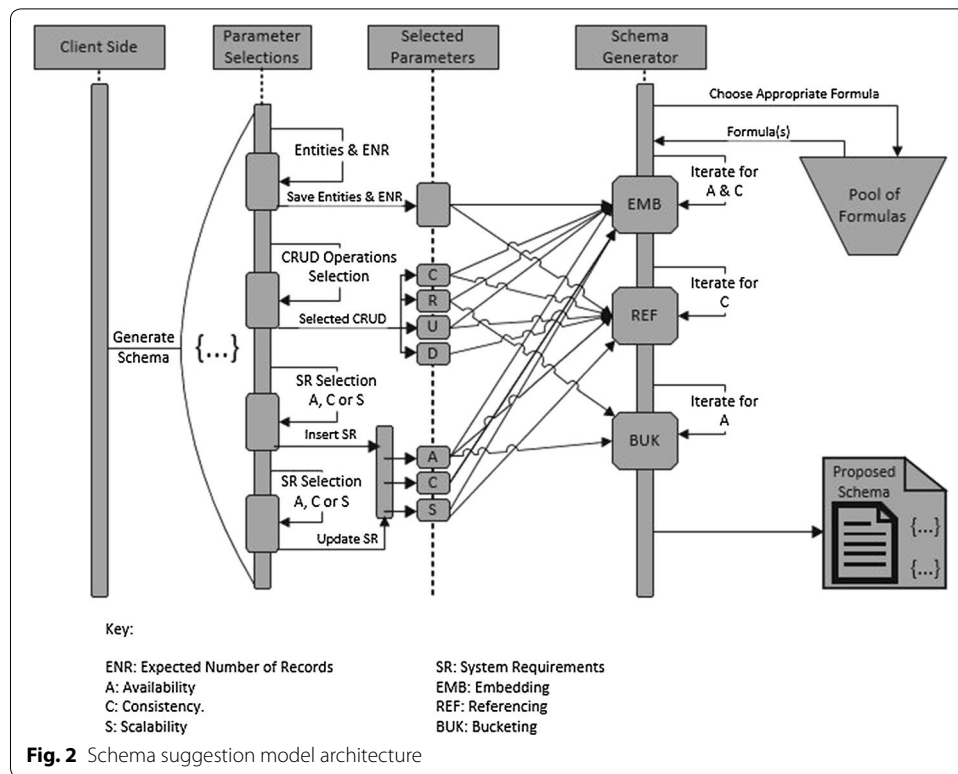
On the other hand, some technical experts such as from mongoDB [6] and JSON [9] companies explained how good data modeling relationships can be achieved as their contribution towards best modeling practice. These approaches are however focused on the NoSQL database functionalities of which they set to 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 created several opportunities for data modelers, it does not provide any working model, or tool 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 candidate 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 novice programmers is presented. The model focuses on balancing between critical significant factors of data modeling such as consistency, availability, and scalability. The



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 layers 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 generation cardinalities [11]. The following algorithms explain the logic.

Schema generator algorithms

The step-by-step procedure of the proposed model is presented in this section. The model follows this procedure in order to generate any desired schema

Schema Generator Algorithm.

Input: E = List of entities and ENR = Expected number of records per entity.
 CRUD Operations Cr (C , R , U , D).
 System Requirements SR (A =Availability, $C_{L,M,H}$ = Consistency 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  $E.size \neq \emptyset$ : then
     $SR \leftarrow$  get preferred requirement
     $Cr \leftarrow$  get selected CRUD operation
    while  $i < E.size$  do
         $ENR \leftarrow$  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
                 $pg \leftarrow newPg[i]$ ;
                 $Tpg = ROUND(ENR / RecordsPerPage)$ ;
                foreach (range(1,  $Tpg$ ) as  $pg$ )
                    if ( $pg=1$  or  $pg=Tpg$  or ( $pg \geq newPg$  and  $pg \leq newPg$ ))
                         $pg \leftarrow newPg (+ or - 1)$ ;
                    end;
                end;
            end;
            //var a  $\leftarrow ENR + ROUND((ENR / 100) * 20) /$ 
        end;
    end;
else
    return null;
end;

```

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  $C_1, C_2, C_3 \dots C_n$ 
    if availability is = 100% ( $C_{x1}C_{x2} \dots C_{xn}$ ) 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  $C_1, C_2, C_3 \dots C_n$ 
    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  $C_1, C_2, C_3 \dots C_n$ 
    if availability is = 75% (Formula as in Eq(3)) then
      Av = True;
      return Av; set i = 4; exit();
    else
      repeat
    end;
end;
end;

```

Next, the components that assist the entities mapping module are presented, they include formulas for calculating availability and storage for storing selected values.

Model components

The components that are used in the semantic mapping module are highlighted in this section. We started by describing the storage area and then followed by formulas.

Central storage area

This is a centralized area where the selectable parameters and values which are supplied to the non-selectable parameters are saved. These records are important and used for the availability computations, systematic and semantics mappings. The structure of this storage is described below:

- Project_ID: this is one of the important attributes that uniquely identifies a project. The proposed model is design in such a way that one user can have multiple NoSQL modeling projects and as such project identifier is required to distinguish between projects.
- User_ID: since the model can be further centralized to handle projects from different users, unique identification need to be extended to all users of the model. This will help to avoid interchanging schemas between users.
- Entities: entity names are saved here as they are being provided by modelers.
- ENR (Expected Number of Records): this attribute saves ENR for each entity and are considered when choosing between referencing and embedding a document.
- CRUD: Create, Read, Update and Delete (CRUD) operations choices are saved under CRUD attribute. The first level iteration process is controlled by this attribute like *C* for create.
- SR (System Requirements): the choice of SR such as consistency, availability, or scalability is housed under this attribute.
- Relations: if entities are related, this attribute is used to keep the type and the chain of the relationship like parent → child → grand-child etc.
- Con_active: not all entities are related in some cases, this indicates whether an entity is active or not.
- Flag: as iteration takes place, this attribute keeps the mappings and computational status. This is achieved by using digits *0* as completed and *1* as not attended.

Formulas

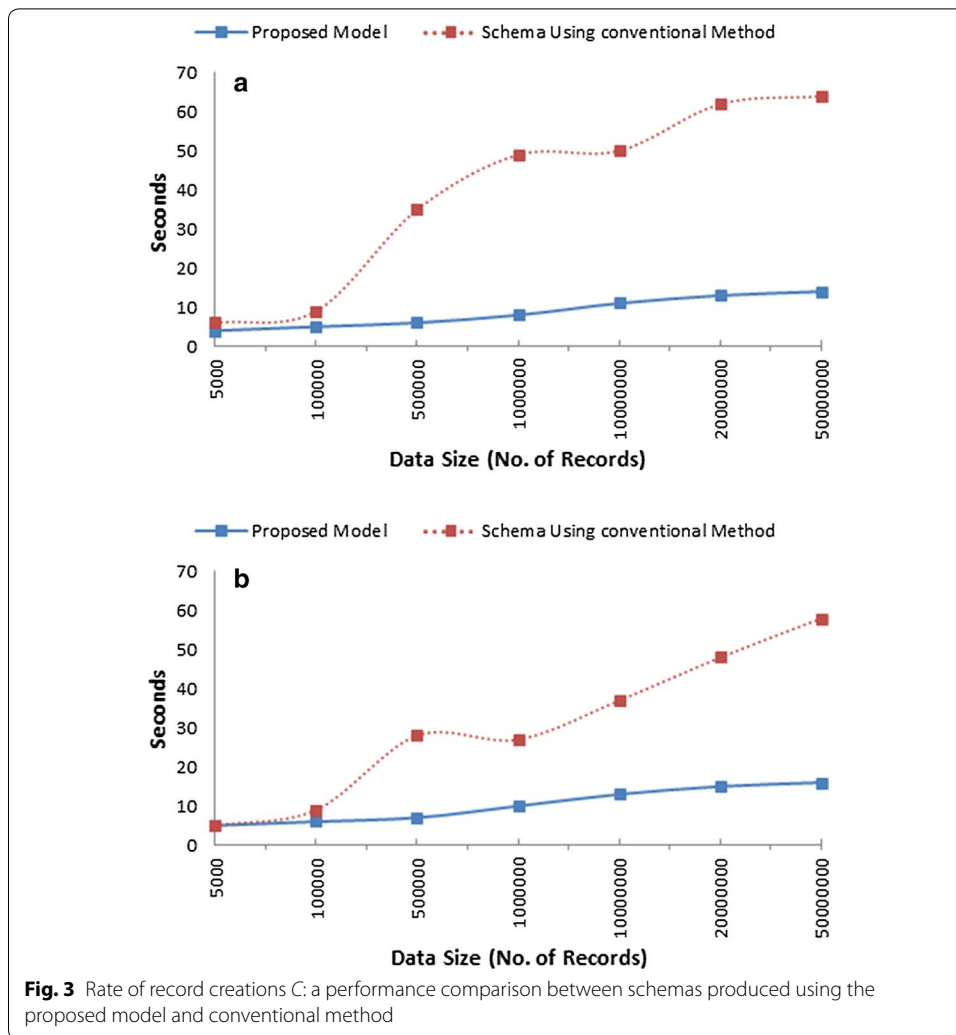
Mathematical formulas are also introduced to standardize the computation process. This is to provide accurate mappings of entities, especially while checking document or cluster availability for different level of consistency requirements. The formulas are described as follows.

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

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.

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

Contrariwise, if the choice of consistency is low *C_L*, Eq. 2 is applied. Level of consistency 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 cluster *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.



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

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 available 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 experiment 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 categorized as follows based on the aforesaid factors, starting with performance in the context of query-writing speed.

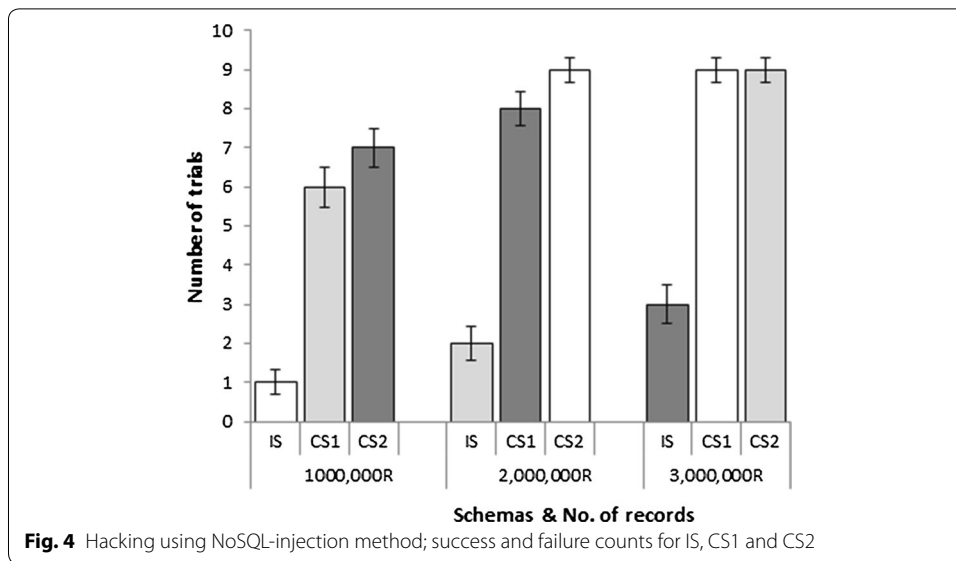
Results on performance (query-writing speed)

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 considerably 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



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 example 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 encouraged 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 collection 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 (represented 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 3,000,000 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 successful NoSQL-injection. 0 represents failed hacking attempt while 1 represents successful 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 perform 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 margin 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 guidelines 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 varies across different levels of programmers, which is why some little but inconclusive difference (based on error margin) is noticed between CS1 and CS2.

With such recorded significant improvement in the aspects of performance and security 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 practice. This is not only for accurate mapping of entities but also alleviating modeling difficulties 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 mathematical 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. Thereafter, 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 programmers to model NoSQL database is considerably reduced, which again simplified the modeling process. In addition, the schemas generated through the proposed model proved to be much more secure than those generated through the conventional methods. This improvement in NoSQL database modeling practice may continue to attract researchers' interest in the near future.

Abbreviations

SQL: Structured Query Language; NoSQL: Not only Structured Query Language; DB: databases; IS: Improved Schema; CS1 & 2: Competitor Schema 1 & 2; ENR: Expected Number of Records; CRUD: Create, Read, Update, and Delete; SR: system requirement; ACID: Availability, Consistency, Isolation, and Durability.

Authors' contributions

Following the International Committee of Medical Journal Editors (ICMJE) guidelines, AAI, SB, RA, JW, MTG have made substantial contributions to conception and design, or acquisition of data, or analysis and interpretation of data; been involved in drafting the manuscript or revising it critically for important intellectual content; given final approval of the version to be published. Each author should have participated sufficiently in the work to take public responsibility for appropriate portions of the content; and agreed to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved. All authors read and approved the final manuscript.

Authors' information

AAI is a Ph.D. student at the Petronas University of Technology, Malaysia. He received his B.Sc. and M.Sc. degrees in Information Technology from Islamic University in Uganda (IUIU), and Petronas University of Technology, Malaysia. He is a lecturer at the department of computer science, Ahmadu Bello University, Zaria-Nigeria, after working with IUIU as a system developer and United Nations AFRI-Uganda as a database and network administrator. Imam is a recipient of research grant from ministry of education Malaysia, and also 2 best paper awards from international conferences. He has published many journal papers, conferences and a book chapter. His area of interest includes but not limited to, big data analytics, SQL, NOSQL and distributed database, software engineering, heterogeneous mobile databases.

Junzo Watada. He received the B.Sc. and M.Sc. degrees in electrical engineering from Osaka City University, Japan, and the Ph.D. degree from Osaka Prefecture University, Japan. After being a system engineer of basic systems and application systems at Fujitsu Co. Ltd, he joined the Academia. Currently he is a professor of Petronas University of Technology and an emeritus professor of Waseda University, after contributing as a professor at Graduate School of Information, Production & Systems, Waseda University, Japan. Dr. Watada is a vice president of Forum of interdisciplinary Mathematics, and a fellow and honorary member of Japan Society of Fuzzy Theory and Intelligent Informatics. He is a recipient of Henri Coanda Medal Award from Inventico in Romania in 2002. Dr. Watada is the Principal Editor, a Co-Editor and an Associate Editor of several international journals including *IDT journal*, *ICIC Express Letters*, *Information Sciences*, *International Journal of Systems and Control Engineering*, *JACIII*, and *Fuzzy Optimization & Decision Making*. His professional interests include soft computing, meta heuristic, model building tracking system, knowledge engineering and management engineering. He is a senior member of the IEEE.

Rohiza Ahmad obtained her B.Sc. (Computer Science) from the University of Tennessee at Martin, USA in 1987, M.Sc. (Computer Science) from the University of North Texas, USA in 1989 and Ph.D. from the University of Malaya in 2009. Currently, she is a Senior Lecturer at the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia. Her research interests are in the area of database management, mobile database application as well as technology acceptance.

Shuib Basri received his bachelors degree in IT (BIT(Hons)) From Universiti Utara Malaysia, Masters degree in Software System Engineering (MSSE) from Melbourne University, Australia, and Ph.D. degree in Software engineering from Dublin City University, Republic of Ireland. He joined the Academia, after being an IT engineer of basic and application systems at several IT based companies for several years. Currently he is a Senior Lecturer of Petronas University of Technology. His research interest include Software Quality, Software Process Improvement, Agile method, Big Data analytics, knowledge engineering and management engineering and several more area in software engineering. He is a member of ISO WG24, IEEE and the IEEE Computer Society.

Maria Teresa Gonzalez-Aparicio. She received the B.Sc. and the Ph. D. degrees in Computer Science from Oviedo University, Spain. Currently, she is an Associate Lecturer in Computing at the Department of Computer Science, Oviedo University, Spain. She is involved in a National Spanish Research project supported by the Ministry of Economy and Competitiveness. She has published in international journals and conferences. She was part of the program committee of an

international conference. She was a reviewer of an international journal. Her research interests include NoSQL databases, transactions in cloud and services computing and software testing. She is a member of the IEEE.

Author details

¹ Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Seri Iskandar, Perak, Malaysia. 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.

Acknowledgements

This paper/research was fully supported by Ministry of Higher Education Malaysia, under the Fundamental Research Grant Scheme (FRGS) with Ref. No. FRGS/1/2018/ICT04/UTP/02/04. Any opinions, findings, and conclusions stated in this paper are those of authors and do not necessarily reflect those of the MOHE.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The data that support the findings of this study are available from PETRONAS university of Technology and CyberSecurity Malaysia but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of PETRONAS university of Technology and CyberSecurity Malaysia.

Funding

Not applicable.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 1 September 2018 Accepted: 20 November 2018

Published online: 06 December 2018

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 Jt Conf 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 systematic mapping study. *Expert Syst Appl*. 2017;80:14–27.
5. Varga V, János 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 & Document DB; 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, González-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. *Geoinformatica.* 1997;1(1):29–58.
24. Borges KA, Davis CA, Laender AH. Omt-g: an objectoriented data model for geographic applications. *Geoinformatica.* 2001;5(3):221–60.
25. Kusters G, Pagel B-U, Six H-W. Gis-application development with geoooo. *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-nosecurity-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 database model category. *DB-Engines*, 2017. https://db-engines.com/en/ranking_categories. Accessed 21 Apr 2017.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- ▶ Convenient online submission
- ▶ Rigorous peer review
- ▶ Open access: articles freely available online
- ▶ High visibility within the field
- ▶ Retaining the copyright to your article

Submit your next manuscript at ▶ springeropen.com
