*Raf. J. of Comp. & Math's., Vol. 15, No. 1, 2021*
*DOI: 10.33899/csmj.2021.168252*

# Data Modeling and Design Implementation for CouchDB Database

**Shaymaa A. Razoqi**
*shymaa.raazoqi@uomosul.edu.iq*
*Computer Science Department*
*College of Education for Pure Science*
*University of Mosul, Mosul, Iraq*

**Received on: 20/07/2020** **Accepted on: 07/10/2020**

## ABSTRACT

In the modern database environment, new non-traditional database types appear that are Not SQL database (NoSQL). This NoSQL database does not rely on the principles of the relational database. Couchdb is one of the NoSQL Document-Oriented databases, in Couchdb the basic element was a document. All types of databases have the same conceptual data model and it was deferent in the logical and physical model, this mean UML class diagram can be used in the NoSQL design at a conceptual level, that is, it can be used to design a Couchdb database. In this research, we suggest a method to model and implement the conceptual level of the Couchdb database from the UML class diagram in using simple way depending on the association types. Depending on the types of relationships between classes, we can have more than one database model to choose from and find the most suitable for the system to be designed. A medical clinic database was proposed to implement the transfer steps according to the proposed method. Three database models were designed and implemented to study the suitability of the proposed transfer method.

**Keywords:** Class Diagram; Database Modeling; Document Database; Unified Modeling Language.

# نمذجة البيانات وتنفيذ التصميم لقاعدة بيانات CouchDB

***شيماء أحمد رزوقي***
*قسم علوم الحاسوب، كلية التربية للعلوم الصرفة*
*جامعة الموصل، الموصل، العراق*

**تاريخ استلام البحث: ٢٠٢٠/٠٧/٢٠** **تاريخ قبول البحث: ٢٠٢٠/١٠/٠٧**

## الملخص

في بيئة قاعدة البيانات الحديثة، ظهرت أنواع من قواعد البيانات غير التقليدية الجديدة وهي قاعدة بيانات لا تعتمد SQL (NoSQL). لا تعتمد قاعدة بيانات NoSQL على مبادئ قاعدة البيانات العلائقية. قاعدة بيانات Couchdb هي إحدى قواعد بيانات NoSQL الموجهة نحو الوثيقة، العنصر الأساسي في Couchdb هو الوثيقة. جميع انواع قواعد البيانات لها نفس نموذج البيانات المفاهيمي ولكنها تختلف عن بعضها البعض في النموذج المنطقي والفيزيائي، هذا يعني انه يمكن استخدام مخطط الفئات للغة النمذجة الموحدة UML Class Digram في تصميم قواعد بيانات NoSQL في المستوى المفاهيمي، اي انه يمكن ان تستخدم في تصميم قاعدة

بيانات CouchDB. في هذا البحث تم اقتراح طريقة لنمذجة وتنفيذ المستوى المفاهيمي لقاعدة بيانات CouchDB باعتماد مخطط الفئات وباستخدام اسلوب مبسط يعتمد على انواع علاقات الارتباط. بالاعتماد على انواع العلاقات بين الفئات، سوف يتكون لدينا اكثر من نموذج بيانات للاختيار بينها وايجاد الاكثر ملائمة للنظام الذي يتم تصميمه. تم اقتراح قاعدة بيانات عيادة طبية من أجل تنفيذ خطوات التحويل حسب الطريقة المقترحة، وتم تصميم ثلاث نماذج لقواعد بيانات وتنفيذها لدراسة مدى ملائمة طريقة التحويل المقترحة.

**الكلمات المفتاحية:** مخطط الفئات ، نمذجة قاعدة البيانات ، قاعدة بيانات الوثيقة، لغة النمذجة الموحدة.

## 1. Introduction

The NoSQL database design stages consist of 3 steps: conceptual, logical, and physical designs like as the traditional relational database stages. NoSQL databases are separated into four kinds. The Key value data stores, column-oriented data stores, document data stores and graph data stores. Each kind has its different tools to work with[1]. Document-based database stores and organizes data as document collections, rather than structured tables with standard fields for each record. With this database, users can add any number of fields of any length to the document as implemented in Couch DB and Mongo DB[2]. CouchDB ("Cluster of Unreliable Commodity Hardware DataBase") is an open-source, NoSQL, web disposed of the database that depends on JavaScript as its query language and JSON documents to store the data. CouchDB does not utilize tables absolutely to store data or deal with relationships. Instead, a database is a set of independent documents, maintaining their designs and their independent data.[3] Sometimes the CouchDB database is designed as a collection of documents linked by relationships to reduce redundancy and maintain data consistency. Relationships in CouchDB can be a reference and embedded [4]

When starting to design a database, the first step is always the data modeling process at the conceptual level. Data modeling plays an important role in NoSQL environments. The conceptual data model does not rely on a specific data model in database design, such as a relational database, Therefore, it can also be applied when designing a NoSQL database[5]. The data modeling procedure includes generating a diagram that represents the meaning of the data elements and the relationships that link them. Thus, detail expansion is one of the important aspects of data modeling, and this type of representation has little assistances for building NoSQL databases[4]. The Unified Modeling Language (UML) is assisting in describing and designing software systems. The primary driver behind all of this is that programming languages are not high enough level stripping to facilitate design discussions. Class diagrams are the fundamental of the UML. The class diagram depicts the types of objects in the system and the relationships that exist between them[6]. The Class diagram uses several types of relationships between objects when modeling a database. These relationships represent when designing a document database with two types of relationships, reference document and embedded document, several different modeling methods may develop for the same database, depending on the understanding of the business and management requirements of each data manager. To address this issue, this paper proposes a method for modeling a CouchDB database to represent association relationships.

The designer uses the NoSQL database when using Relational database is unable to meet the needs of the application or database user requirements. When designing a relational database, this requires defining the entire data structure from the beginning of

the design, by specifying the size and type of data for each field in the table. This process makes the design complex and inflexible, especially with constantly changing data models. As dealing with medical clinics data and reviewing patient records when information changed after add or drop attributes through structure altering. While the use of CouchDB NoSQL databases are semi-structured databases and rely on documents instead of table records, and since the document consists of an unlimited number of key-value pairs, and each document is considered an independent unit that is not dependent on the structure of other documents. The rest of the paper is planned as follows: Section 2 presents related works. Section 3 explores the concepts of Unified Modeling Design Language. Section 4 shows a data model for NoSQL documents oriented, and Apache CouchDB data model, the proposed l method presents in Section 5. Finally in Section 6, presents the conclusion of the research and future works

## 2. Related Works

The topic of data modeling is very important. when talking about databases, we find that most of the research focuses on the Mongo type and a few of them deal with CouchDB, so this research focused on CouchDB databases as one of the types of document databases of NoSQL databases widely spread at present. The approach used in this paper is how to convert the conceptual UML database model into best CouchDB models. Several previous studies dealt with designing base models, but most of them adopted the general design method that relies on a data model in general. Kwangchul SHIN [5] proposed a design method of NoSQL document database, Peter Chen's framework is used as a basic conceptual data model. The design method is applied to the e-commerce business area to study the applicability of it. In this study all associations, which are "<<reference>>" or "<<embedded>>" that depends on query forms, are designated into "<<reference>>" excluding Product's inheritance relationship designated into "<<embedded>>". Before that in 2016, Viorica VARGA [7] in his paper proposed a "Formal Concept Analysis (FCA)" approach for modeling conceptual level document databases. The paper proposed a "Relational Concept Analysis (RCA)grounded" approach to design the NoSQL database in a conceptual document. The approach depends on mapping from Entity-Relationship to a semi-structured data schema presented for "One-to-One", "One-to-Many" and "Many-to-Many" relations types. Also, Harley VERA [4] provided a standard modeling diagram for document-oriented document databases by building compact graphs of a conceptual statement template that are clear and intuitive. The study provided general techniques and did not specify a specific modeling engine for the NoSQL document database.

## 3. Unified Modeling Language

The Unified Modeling Language (UML) is a set of graphical symbols, supported by one typical program that assists in describing and designing software systems. UML defines, in its current state, a coding and identification model. The things you see in the models are the graphical structure of the modeling language. [6]. Static view is one of the data modeling methods used to describe the conceptual and internal parts for the applications, the name came from being that it does not describe the behavior of the system depending on time. The main components of a static view are the classes and the relationships that they link together in the class diagram[8]

The class diagram is slightly used but is also subject to the largest set of modeling concepts. Classes are depicted as rectangles. Lists of features and operations are

displayed in separate booths. Compartments can be suppressed when full details are not required. A class may appear on many charts. Its features and operations are often suppressed in all shapes except for one scheme[8]

The relationships in the class diagram are represented by connecting lines that connect the rectangles of the diagram. The types of relationships are distinguished by the pattern of the line or the end forms of the connecting lines[8]. Relationships among classes can be of two types: associations and generalizations. Generalization relationships also define as Inheritance relationships. This relationship joins the items hierarchically from top to down, and the arrowhead always goes to the parent. Association or usage relationships shows in Figure1 can be aggregation( is part of ) or composition( is entirely made of ) [6]

The properties' multiplicity is a reference to the number of entities that may fill the property. The common signs can be:

- 1 (A person must have exactly one profile.)
- 0 .. 1 (A person may or may not have one car.)
- * (The patient does not have surgeries, or there is a limit to the number of surgeries he underwent)

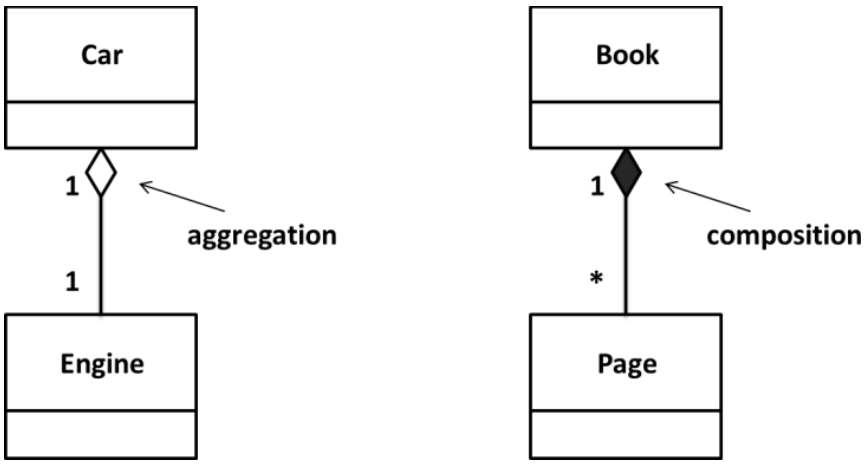



**Figure 1.** Association Types

## 4. Data Model for NoSQL

A conceptual data model can only express data in itself independently of specific techniques has been used differently to share data, reference data model, and so on. The NoSQL database design stages consist of conceptual, logical, and physical design[5], as:

- conceptual data modeling is the method of developing a conceptual diagram of a database of user requirements. The NoSQL database conceptual diagram is independent of how a high level of database structure is implemented.
- Logical data modeling is the method of evolving a NoSQL logical diagram from a conceptual diagram. The logical diagram defines the managed data structures in the NoSQL database. It rests on the data models used in the NoSQL database but is separating from the NoSQL database software.
- Physical data modeling is the method of developing a NoSQL database diagram. The physical diagram describes the data structure applied in a specific NoSQL database program.

The document database is concerned with ensuring large data storage and good query performance rather than giving attention to high reading and writing performance simultaneously[9] It is used for applications in which data is changed occasionally[11], In the document database, documents are stored in collections, the data is stored in the JSON, BSON, or XML document format. Where any number of fields can be added to the database without keeping blank spaces for other documents in the collection.[1]. In general, the format of the data is stable and the structure is flexible and called a semi-structured database. Each document in the document store databases has a unique key to represent that document, that is used in database applications to handle that document[10]. Apache CouchDB is an example of a document store. It uses JSON to store data. [12].

### 4.1. Apache CouchDB Data Model

The Apache Foundation introduced the CouchDB project and is completely written in Erlang. Perfectly suited for simultaneous applications Erlang has been chosen as a programming language through its lightweight operations and functional programming model.[13]

The data in CouchDB is structured into documents. Each document can contain any number of features. CouchDB supports String, Number, Boolean, and Array data types, and every feature can contain any types, lists, or even objects. Documents are stored and accessed as JSON objects[14].

Update operations in CouchDB are accomplished on complete documents. The CouchDB query form consists of two methods, one of which is an HTTP query API, and the second method is viewed that are created with MapReduce functions[14]. Each CouchDB document has a unique key Id and each document has also a revision Id. The revision Id is rewritten by CouchDB every time a document is updated.

The document is a set of key-value pairs. Values can be numbers, string, Boolean, lists, or objects. CouchDB is a simple container for a group of documents and may not create any required relationship between them [2]. Thus, The concepts of normalization do not apply to CouchDB because it is a non-relational database.[4]

Relationships in CouchDB can be a reference and embedded. In an embedded relationship, the large document has embedded one or multiple documents within it according to the cardinality of the relationship, figure2.[4]. While, In reference relationship, an arrow must use from the current document directed to the referenced document, note that the current document can reference many documents, the bidirectional arrow is used to represent the many to many reference relationship, figure3.[4]

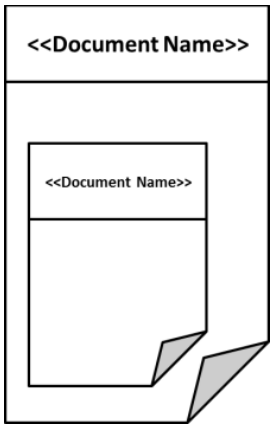



**Figure 2.** Document Embedded Relationship

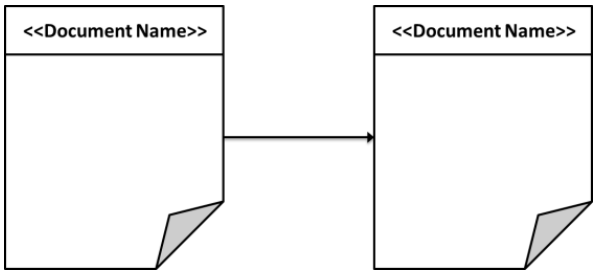



**Figure 3.** Document Reference Relationship

On 2018 Paola Gomez et al [16] suggest several criteria that can be adapted to measure the model quality of a document-oriented database, they are called Structural Metrics. The suggested criteria are based on the structure of schema in the Document-Oriented NoSQL model. Nesting depth is one of these structural metrics that will be used in this study. It is based on the fact that "the cost of accessing information increases as the depth of the embedded information increases." The reason is due to the increase in the complexity of accessing the information hidden in the lower levels of the document structure, especially if this information is not required continuously with the information at the higher level in the document structure. There are several formulas for measuring Nesting depth, including Collection Depth and Global Depth[16]:

$$colDepth(\alpha) = max\ (depth(p_i)) \quad : p_i \text{ is a valid child path of collection } \alpha \qquad (1)$$

$$depth(p) = n \quad \text{number of nodes embedded in path } p \qquad (2)$$

$$globalDepth(x) = max\ (colDepth(\alpha_i)) \quad : \text{for each collection } \alpha \text{ in schema } x \qquad (3)$$

From (1),(2), and( 3), increasing the value of $n$ leads to an increase in the value of the Collection Depth and thus increasing the value of Global Depth, and it becomes more expensive.

## 5. The Proposed Conceptual Modeling Mechanism

The approach used in this paper is how to convert the conceptual UML database model into multiple CouchDB models choose the most suitable ones according to the proposed application and convert it into a physical database model. To create a CouchDB database that fits the requirements of large databases, we will first define the conceptual data model in the first step described in paragraph 5.1 and use the class to achieve this step. In the second step explained in paragraph 5.2, a method was proposed to convert the class diagram into a logical CouchDB database model by following several steps and rules. In which the components of the class are converted into the components of the CouchDB database in a way that achieves the highest level of efficiency. Paragraph 5.3 provides a practical application of what is described in paragraphs 5.1 and 5.2 and determines the best design model by adopting standards

### 5.1 Design UML for Database

I first defined the source UML class diagram(*umlCD*) and the target generic CouchDB logical model. A class diagram *umlCD* defined with[15]:

- The class diagram name(*umlCD.N*),
- The set of classes(*umlCD.Sc*), Each class( *c* ) in *Sc* has:
  - name (*c.N*),
  - a set of attributes (*c.SA*),
  - and a class primary key (*c.Pk*),

Where for each attribute ***a*** in ***c.SA***, ***a.N*** is the attribute name, and ***a.T*** is the attribute data type. The primary key (*c.Pk*) is a special attribute of a class and it has a name and data type.

- and the set of links(*umlCD.Sl*) between classes in Class Diagram. Each link(*l* ) in ***Sl*** has:
  - name( ***l.N***),
  - relationship type denoted by link type (*l.LT*),
  - and a set of n couples( *l.LP*). Each couple in *LP* contained (*c and cl*), *c* is a linked class and *cl* is the cardinality placed next to *c*. In generalization link *cl* can hold a null value if no cardinality is specified to class *c*.

### 5.2 Convert UML Class to CouchDB Logical Model

CouchDB is the top-level container that owns all the elements, we defined it as its name *(couchDB.N)* and set of document collections *(couchDB.Sdoc*).
Each document collection( *doc*) in *couchDB.Sdoc* has:

- name( *doc.N),*
- the set of fields (*doc.SF*), *SF* can contain any number of fields, each field (*f*) has:
  - name (*f.name*) and
  - data type(*f.type*), the type can be atomic or complex.
- and *doc.Id* is a special field of *SF* it has a name and a type is used to identify each document in the database.

This section introduced the method of converting the UML class diagram (conceptual level) of the database into a logical model of the CouchDB database. We suggested four steps:
Step1: each umlCD is Converted to into CouchDB where umlCD.N=couchDB.N.
Step2: each class *c* in *Sc* is Converted to document collection *doc* in *Sdoc*. Where *doc.N = c.N* And *doc.Id= c.Pk* .
Step3: each attribute *a* in *SA* (in class *c* ) Converted to *f* in *SF* (in document *doc* ) where *f.name* and *f.type* is the same as *a.name* and *a.type* .
Step4: different kinds of relationships amongst documents are defined as Reference and Embedded documents. For each link between two classes *c1* and *c2*, there are three kinds of conversion :

1. Embedded Relationship: link *l* is Converted by embedding the document collection *doc2*( where *doc2.N=c2.N*) in the document collection *doc1*( where *doc1.N=c1.N*) ,then *doc2* was in complex *doc1.SF* .
2. Reference Relationship: link *l* is Converted into reference field *fl* referencing *doc2* ( where *doc2.N=c2.N*), then *fl.N= Ref.doc2.N* and *fl.type* is the same as *c2.Pk* type, and then added to the fields of *doc1* ( where *doc1.N=c1.N*) such as *fl* was in *doc1.SF*.
3. New document: link *l* between *c1* and *c2* which represents *doc1* and *doc2*, is Converted into a new document collection *doci*, where *doci.N =l.N*, *doci.SF={fl1,fl2}*, *fl1.N=Ref.doc1.N*, *fl1.type =doc1.Pk* type, *fl2.N=Ref.doc2.N*, *fl2.type =doc2.Pk* type.

To determine the previous conversion kinds, we suggested the following rules:
Rule1: The link between *c1* and *c2* is a Generalization ( Inheritance) relationship that is converted into an Embedded relationship between *doc1*and *doc2*(with a condition that the children document do not need individual updates separately from the parent).

- When the previous condition is not met, the link is converted to a reference relationship, and the parent document *doc.Id* is added to the children document as reference field.

Rule2:The link between *c1* and *c2* is a Composition-relationship and of type one to one or one to many (With a condition that the number of many side is not large and there is no need to independently access the included parts) is converted to an Embedded relationship between *doc1*and *doc2*.

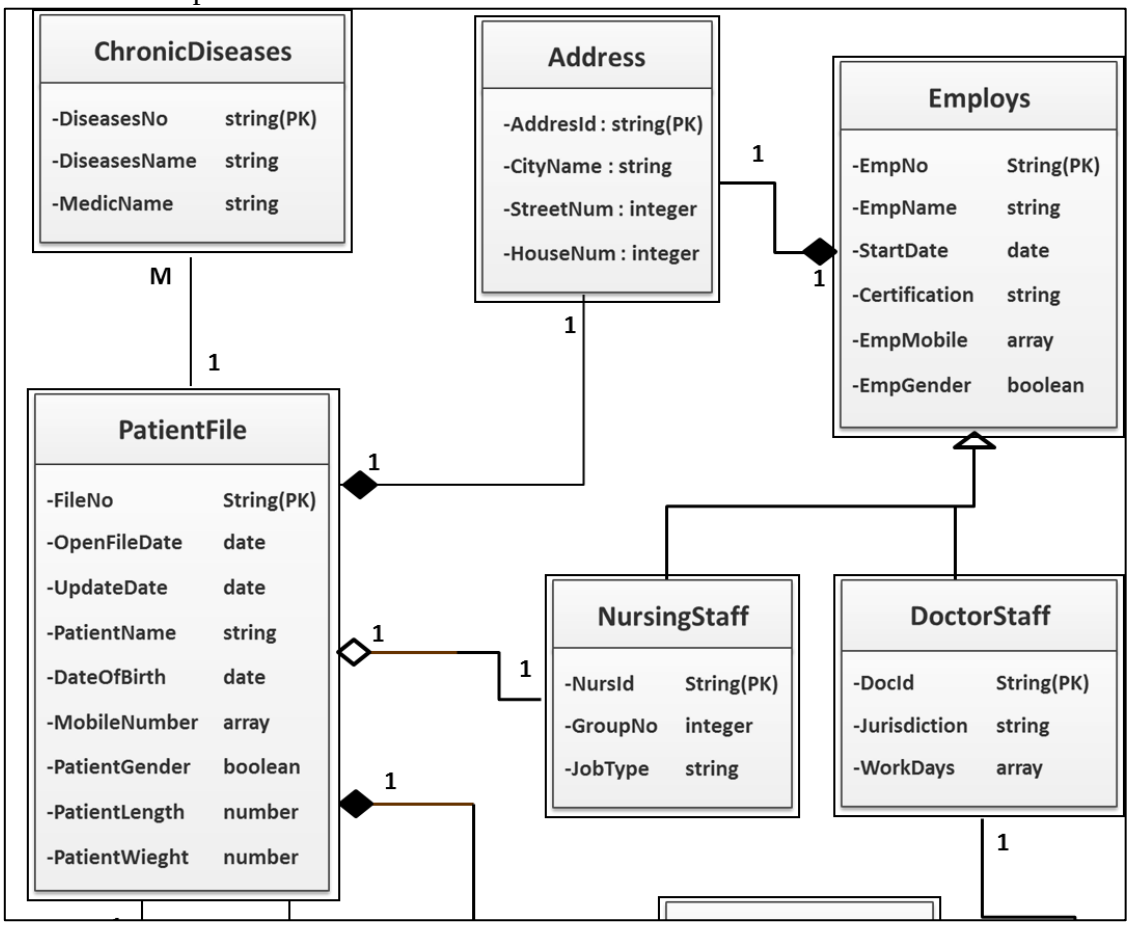
- When the previous condition is not met, the link is converted to Reference relationship, and the document *doc.Id* in the one side is added to the document in the many side as reference field.

Rule3: The link between *c1* and *c2* is an aggregation-relationship converted to a Reference relationship between *doc1* and *doc2*.

Rule4: The link of type Many to Many relationship between *c1*and *c2* converted to a new document with Reference relationship to *doc1*and *doc2*.

### 5.3 Implementation and Discussion

A clinic database was used as a model for conversion as a case study. After collecting information about the medical clinic database, the diagram was shown in Figure4, which represented the conceptual level in the UML class diagram. From Figure 4, the PatientFile class is the basic element of the database, and it is created for each patient entering the clinic by entering his personal and satisfactory information. PatientFile class had composition associated with PatientVisit class that represents the information of patient visit to the clinic.



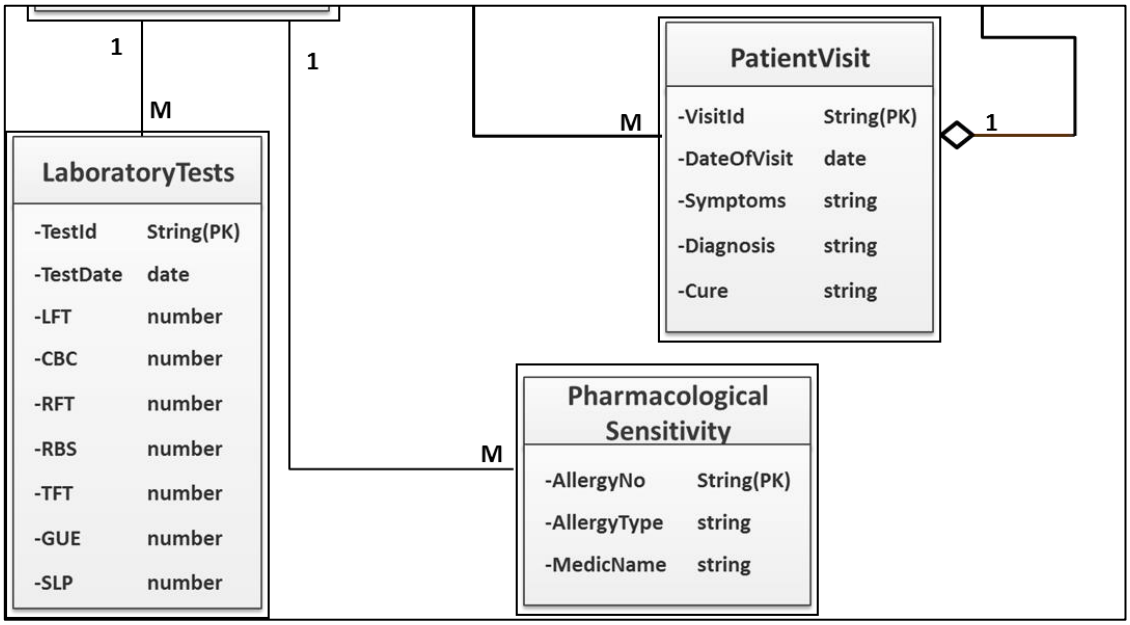



**Figure 4.** UML class diagram.

Address class had composition associated with PatientFile class and with Employs class. Employs class had two subclasses NursingStaff and DoctorStaff. In the same time, NursingStaff had aggregation associated with PatientFile class that for register patient information to the file. DoctorStaff had aggregation associated with PatientVisit class to wrote information about the patient visit during a test. Also, each of ChronicDiseases, PharmacologicalSensitivity, and LaboratoryTests classes had an associated with PatientFile class.

The four steps outlined in paragraphs5.1 and 5.2 have been implemented to convert the UML class to CouchDB database, the classes have been converted to document collections. The document will be created against each class, and the document includes all the simple and complex attributes of the corresponding class in pairs of (key-value). After creating the corresponding documents for each class and identifying the components of each document, relationships are created between the documents depending on the nature of the relationships within the classes and the number of common elements from each document in the relationship( cardinality). The relationship between Employs and each of NursingStaff and Doctor has been transformed into an embedded relationship between documents according to rule 1. As the nature of the relationship is an inheritance relationship, and then all information must be included in one document for ease and speed of access to information. Also, the relationship between Address and each of Employs and PatientFile converted into the embedded relationship between these documents according to rule 2, where the relationship between them is a compassion relationship of 1:1 type and address data is not requested independently of patient or employee data. The relationship between PatientFile and NursingStaff is a weak relationship converted into a reference relationship between these documents because the relationship is an aggregation and according to Rule 3, it is possible to access the nurses' information without the need for patient data. Thus, including the nurse’s information within the patient’s information makes it difficult to independently access and update that information. this also applied to the relationship between DoctorStaff and PatientVisit is a weak relationship that

converted into a reference relationship between these documents, the same former also relied on Rule3.

The relationship between PatientFile and PatientVisit is a strong relationship that is supposed to be converted to an embedded relationship, According to rule 2 and the fact that the number of common elements in the many side of the relationship increases unstably. but it does not fulfill the condition since PatientVisit is frequently duplicated leading to problems in the storage process and need independently access to that the relationship converted into a reference relationship between these documents.

As in the previous paragraph, when converted the 1: M relationship you must know the value of the M or at least the estimating values of it because this affected on the conversion method to the CouchDB model. To convert the relationship between PatientFile and each of (ChronicDiseases, LaboratoryTests, or PharmacologicalSensitivity) must know the values of m, if the number of elements is very small, then the sub-documents can be included within the parent document to speed access, and this method is illustrated in Figure 5 where each of the three subdocuments is embedded into the patient document. But if the number of elements is large will increase the cost and complexity relying on Global Depth and Collection Depth. Where the increased value of M will increase the value of Depth(PatientFile) and therefore increase the value of Collection Depth and Global Depth because of Max function in both formulas. The model in Figure 5 depended on the embedding of the documents inside the PatientFile, This proposal was good in terms of speed of data retrieval, but it caused problems in dealing with the included data when needed it independently.

To avoid that when m is increasing, it is preferable to create separate documents and use reference relationship to refer to them, as shown in Figure 6 where each of three subdocuments is created separately and referenced by the patient document. The model in Figure 6 solved the problem of accessing data independently by creating separate documents, but it led to a slow process of fully retrieving data.

The documents that need independent modifications and large growth were studied and documents that do not need independent access or major modifications, Figure7 which is the model that was approved. Depending on the nature of the database under study, the drug sensitivity data and the chronic disease data are frequently or rarely updated, while the laboratory examination is always changing and increasing. The three models shown in the figures5,6 and 7 were implemented and converted to physical models using Apache CouchDB 2.3 application as (clinc1,clinc2, clinc3 ) databases and in similar environments to compare the storage space for each of them. The results are represented in the diagram, Figure8 using data sample for 100 patients.

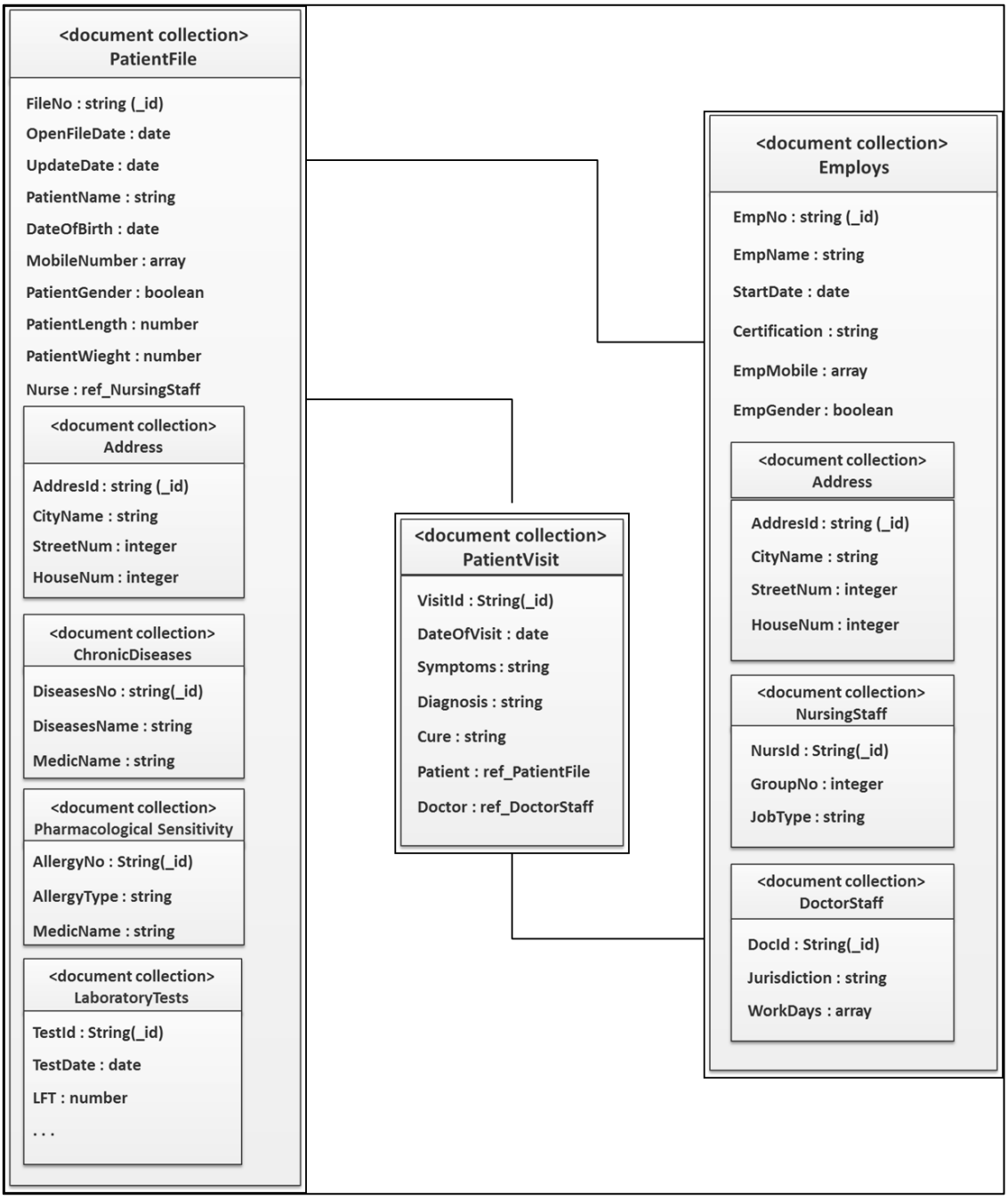



**Figure 5:** Used embedded documents (clinc1)

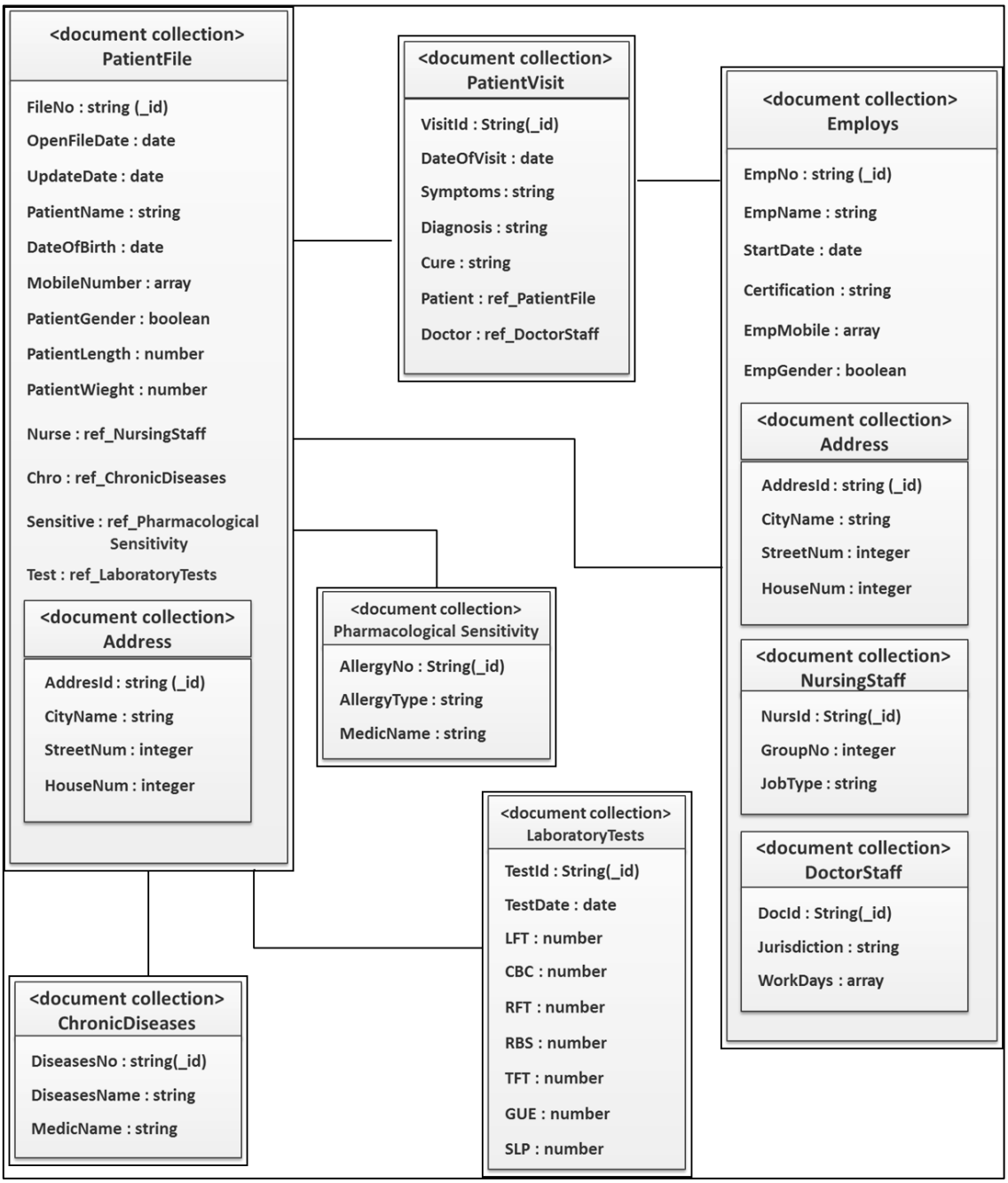



**Figure 6:** Used reference relationships (clinc2)

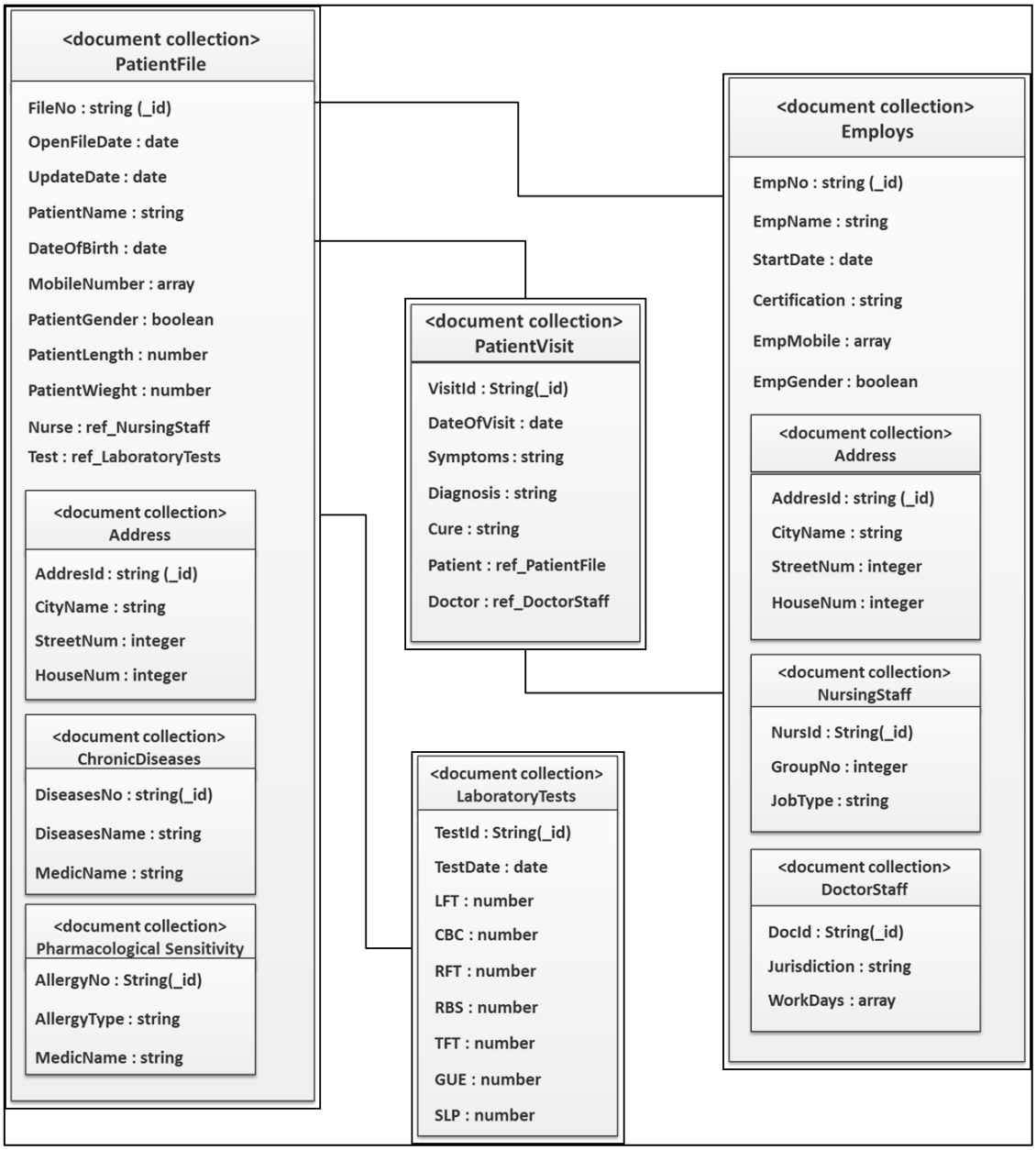



**Figure 7:** Used embedded with ChronicDiseases and PharmacologicalSensitivity (clinc3)

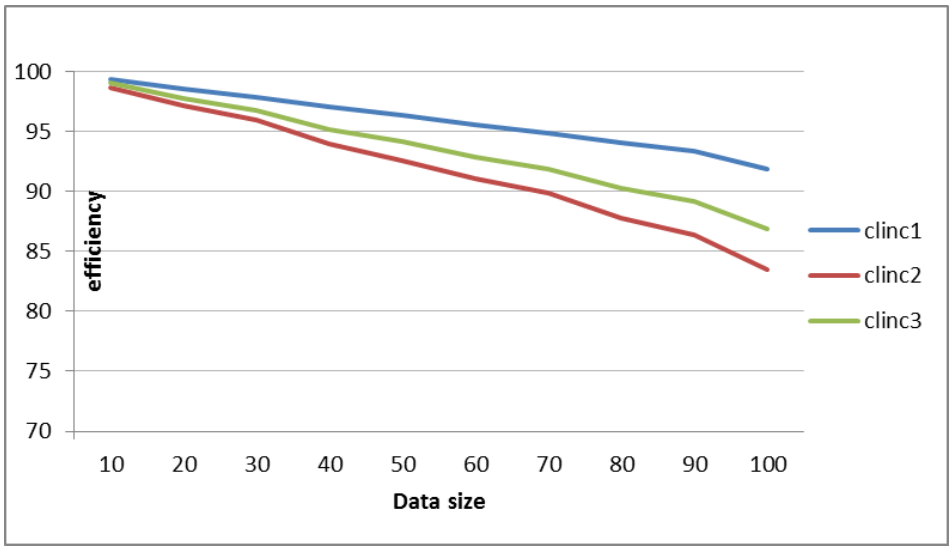



**Figure 8:** An increase in the amount of data

## 6. Conclusion

Several different modeling methods may develop for the same database, depending on the understanding of the business and management requirements of each data manager. The main goal of this research paper is to find a simplified method for modeling and designing the CouchDB database. CouchDB databases are a type of document-oriented NoSQL. The modeling process is not just a brief art, but rather it is a science. I first need to study and analyze the design of the database that you will be built, Secondly, we need to define the nature of the relationship between the parts of the database, how they relate to each other, the extent of the frequency and independence of the part. After that, we then determine the type of relationship when the physical design.

This research paper presented a simplified method for converting the UML class diagram model into a CouchDB logical model by following simple steps to obtain the best efficiency, speed of access and flexibility in dealing with the parts of the database. UML class diagram links convert into CouchDB using two types of relationships between documents, embedded and reference, each one of the two relationships has its benefits and used. The conversion steps are illustrated using an example of a clinic database.

After clarifying the above, we suggest that we build a programmatic method for applying the proposed conversion steps in this research as future works, in addition to that we propose to study other types of NoSQL databases**.**

## 7. Acknowledgment

First and foremost, we want to acknowledge the University of Mosul, and we are very thankful to our colleagues for helping to complete this research. Well also thank for Dr Adnan Mohammad, who has provided us assistance in the preparation of the clinic database that owns.

## ***REFERENCES***


[1] CHAUHAN, Divya; BANSAL, K. L. Using the advantages of NOSQL: a case study on MongoDB. International Journal on Recent and Innovation Trends in Computing and Communication, 2017, 5.2: 90-93.

[2] TAURO, Clarence JM; PATIL, Baswanth Rao; PRASHANTH, K. R. A comparative analysis of different nosql databases on data model, query model and replication model. In: Proceedings of the International Conference on ERCICA. 2013.

[3] SHARMA, Sugam, et al. A brief review on leading big data models. Data Science Journal, 2014, 14-041.

[4] VERA, Harley, et al. Data modeling for NoSQL document-oriented databases. In: CEUR Workshop Proceedings. 2015. p. 129-135.

[5] SHIN, Kwangchul; HWANG, Chulhyun; JUNG, Hoekyung. NoSQL database design using UML conceptual data model based on Peter Chen's framework. International Journal of Applied Engineering Research, 2017, 12.5: 632-636.

[6] FOWLER, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.

[7] VARGA, Viorica; JÁNOSI-RANCZ, Katalin Tünde; KÁLMÁN, Balázs. Conceptual design of document NoSQL database with formal concept analysis. Acta Polytech. Hungarica, 2016, 13.2: 229-248.

[8] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. The unified modeling language reference manual. Addison Wesley.1999.

[9] MASON, Robert T. NoSQL databases and data modeling techniques for a document-oriented NoSQL database. In: Proceedings of Informing Science & IT Education Conference (InSITE). 2015. p. 259-268.

[10] PORE, Supriya S.; PAWAR, Swalaya B. Comparative Study of SQL & NoSQL Databases. International Journal of Advanced Research in Computer Engineering & Technology, 2015, 4.5: 1747-1753.

[11] AGGARWAL, R.; ARORA, R. Modeling and Querying Data in MongoDB. International Journal of Scientific & Engineering Research, vol, 2013, 4: 141-145.

[12] KAUR, Sawinder; KAUR, Karamjit. Visualizing Class Diagram using Orientdb NOSQL Data–Store. *International Journal of Computer Applications*, 2016.

[13] PADHY, Rabi Prasad; PATRA, Manas Ranjan; SATAPATHY, Suresh Chandra. RDBMS to NoSQL: reviewing some next-generation non-relational database's. International Journal of Advanced Engineering Science and Technologies, 2011, 11.1: 15-30.

[14] COUCHDB, Apache. Apache CouchDB. URL https://couchdb. apache. org.

[15] ABDELHEDI, Fatma, et al. MDA-based approach for NoSQL databases modelling. In: International Conference on Big Data Analytics and Knowledge Discovery. Springer, Cham, 2017. p. 88-102.

[16] GÓMEZ, Paola; RONCANCIO, Claudia; CASALLAS, Rubby. Towards quality analysis for document oriented bases. In: International Conference on Conceptual Modeling. Springer, Cham, 2018. p. 200-216.