Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 9, No. 3, August 2018), Pages: 87-101
www.jacr.iausari.ac.ir

Apply Uncertainty in Document-Oriented Database (MongoDB) Using F-XML

Ferdos Mehrab[1], **Ali Harounabadi**[2]
1) Institute for Higher Education ACECR, Khouzestan, Iran
2) Central Tehran Branch, Islamic Azad University, Tehran, Iran
mehrabferdos@gmail.com; a.harounabadi@iauctb.ac.ir

Received: 2017/10/29; **Accepted:** 2018/02/01

Abstract

As moving to big data world where data is increasing in unstructured way with high velocity, there is a need of data-store to store this bundle amount of data. Traditionally, relational databases are used which are now not compatible to handle this large amount of data, so it is needed to move on to non-relational data-stores. In the current study, we have proposed an extension of the MongoDB architecture to support FMQL by integrating a software layer on MongoDB translating FMQL queries to corresponding F-XML queries. Flexible queries have been done in the oriented-document database (MongoDB) by using fuzzy xml commands in the document-centric (MongoDB) injected uncertainty and expressed this commands based on F-XML and by using uncertainty. We used the fuzzy properties to improve query in the document-oriented database (MongoDB) and we showed the Position of the proposed method by the implementation and evaluation of a case study.

Keywords: *NOSQL, Document-Oriented Database (MongoDB), Fuzzy Query, XML, F-XML*

1. Introduction

In the last few years, usage of NOSQL database has been spread in different sectors because of their ability to deal with new requirements of applications. NOSQL DBs present new storage architectures that provide high scalability, availability and fast retrieval requirements for managing unstructured and partially structured data [1]. Many NOSQL DBs are open source and they are cheaper per terabyte than traditional DBs. The DBs are more appropriate for the web based data [2]. According to the type of data storage, NOSQL databases are divided into four categories:

- Key-value: Key-value Databases are constituted by a simple data model and data is stored corresponding to key-values [3]. Data is organized as an associative array of entries consisting of key-value pairs. Few popular key-value data stores are Riak, Berkeley DB, Voldemort and Redis [4].
- Column-oriented: Column Oriented Databases store groups of related data into the family column and rows with many columns are associated with a row key. Each record can differ in the number of columns stored, and columns nested inside other columns are called super columns. This database type is extremely scalable and works well with more complex datasets [5]. In distinction from the conventional row representation, the column-oriented representation is much

more efficient in the execution of OLAP queries [6]. The examples of NOSQL DBMS based on column storage, we can mention to the big table, Cassandra, etc.

- Graph-oriented: A graph data model is at the heart of graph-oriented databases. In some applications, the relationships between objects are even more important than the objects themselves. Relationships may be either static or dynamic. Graph databases are also popular for implementing access control and authorization subsystems for applications that serve millions of end users. Neo4j is an example of this type of database [7].
- Documents-oriented: This model is versioned documents that are collections of other key-value collections. The semi-structured documents are stored in formats like JavaScript Object Notation (JSON) or Binary JSON (BSON). Among the NOSQL DBMS oriented- document, we can mention to the MongoDB, CouchDB, etc. [8].

Specific databases have been designed to handle such data relying on big dense network structures, especially within the NOSQL world. These databases are built to remain robust against huge volumes of data against their heterogeneous nature and the high speed of the treatments applied to them, thus coping with the so called Big Data paradigm [9].

To save this type of data as well as complex data in everyday life, one should use a new type of database to hold the complex data and are able to cope with uncertainty of this data. That is why the uncertainty of fuzzy query is used [5].

Fuzzy database is a database which is able to cope with uncertain or incomplete information using fuzzy logic [10, 11]. Fuzzy attribute is an attribute, which can get its amounts from a fuzzy set, such as age which can acquire being old, young, middle-age or such as height which can be tall, short, medium and etc. [12, 13]. The reflected answers of the queries in the fuzzy database emphasize answered with a membership degree in a domain of [0, 1] [14]. Therefore in this paper, we used document-oriented database Mongo and we have been added the agent fuzzy to MongoDB commands and proper grammar is expressed in this database. We have proposed an extension of the MongoDB architecture to support FMQL by integrating a software layer on MongoDB translating FMQL queries to corresponding F-XML queries. Flexible queries have been done in the document-oriented database (MongoDB) by using fuzzy xml commands in the document-centric (MongoDB) injected uncertainty and expressed this commands based on F-XML and by using uncertainty.

In the second part of the paper, related works are reviewed. The proposed method of the paper to present fuzzy queries with MongoDB syntax and fuzzy XML tags and a new architecture of MongoDB to mapping FMQL to F-XML is described in third section. The fourth part presents a case study. We finish the paper with a conclusion and a presentation of some future works.

2. Related Works

Recently, there have been several works on modeling fuzzy data and working on object-oriented database and relational database but in non-relational databases because of the novelty of a wide surveys have not been conducted.

In the mapping relational database to XML, [15] have proposed a process of defining syntax and implementing of mapping relational database to XML. An application solution that enables usage of the fuzzy logic constructs with XML data has been provided. Users

are given a possibility to define the arbitrary membership functions, and their computation is achieved in real time with the usage of the MATLAB software.

The practical side of this implementation and its satisfactory performance has been demonstrated. In the paper [16] focused on modeling fuzzy data with fuzzy data types in the fuzzy databases and fuzzy XML and identified several fuzzy data types, including fuzzy simple data types, fuzzy collection data types and fuzzy defined data types. For the purpose of representing and processing fuzzy data, they elaborated the explicit declarations of the fuzzy data types in the fuzzy OODB model and fuzzy XML Schema, respectively.

In mapping object-oriented database to XML, [17] have proposed a fuzzy grammar in XML to increase the accuracy in fuzzy queries. By applying this approach, a complicated fuzzy query could be handled easily and objects near to the user's request would be returned more easily. They had presented a new fuzzy grammar for fuzzy queries on the object oriented database. The proposed model of this paper is able to make the necessary influence and the priority of each attribute in query result into allocating weight to the available attributes in the query. As they had observed, the user could find all the required objects through applying the linguistic variables without dealing with the numbers through the proposed model. They had used object and class concepts to define some functions in the class without any need to receive any digital parameters. They could establish the access to the required variables of the class and carried out the operation. Their suggested approach presents a better workability in huge and heavy databases and this grammar facilitates fuzzy queries. In [18] have presented a new fuzzy grammar for fuzzy queries on the object oriented database. The proposed model of this paper has been able to make the necessary influence and the priority of each attribute in query result through allocating weight to the available attributes in the query. As they observed, the user could find all the required objects by applying the linguistic variables without dealing with the numbers through the proposed model. They used object and class concepts to define some functions in the class without any need to receive any digital parameters. They could establish the access to the required variables of the class and carried out the operation. Their suggested approach presented a better workability in huge and heavy databases and this grammar facilitates fuzzy queries.

In the field of non-relational database (NOSQL), [19] proposed a new way of modeling and traversing interconnected data that is unparalleled in the information storage. With the advent of production grade systems such as Neo4j using GDB problems could be addressed without resorting to a limiting implementation on a RDBMS. They only could offer the graph database which has a natural application for biological, semantic, network, and recommender systems that require the type of data model.

In [20] has proposed an invention which could be used in any MongoDB powered application where a schema backed validation is needed before DML operation. Also it would have great use in migration of data from traditional RDBMS to MongoDB and vice versa. When data needs to be migrated, Mongo Schema Validator will be very handy where mschema.xml can be defined as per existing RDBMS table definition. So it will be validated before the data is ported from and to MongoDB application.

In [21] had proposed a 5 phase forensic investigation framework for the document store NOSQL DBMS based on its unique features. Their framework divides the collection or acquisition phase of other more traditional frameworks into a logical evidence acquisition, preservation phase, distributed evidence identification, acquisition, and preservation phase. The logical evidence acquisition and preservation phase included the

schema analysis of the document store NOSQL DBMS. The important evidence in the DBMS such as transaction logs and data files are acquired in the following distributed evidence identification, acquisition and preservation fuzzy.

In [7] considered the current challenges with respect to health data management, a model based on the Document-based databases and Cloud Computing were presented. In order to evaluate the performance of this model compared to previous one, these two models were implemented by using real health data. Considering data-preparation, the new model is capable of storing semi-structured and unstructured data with no preparation need. The new model also has higher flexibility than the former model. The extensibility of the new model is high, while the former model has some limitations in this regard. In [22] built a high-performance OPC XML-DA server, which had achieved effective real-time management to the underlying control system, and improved the industrial on-site data access flexibility, and increased remote access to data from the on-site, and offered an effective solution for industrial real-time control. It had great significance for improving the on-site industrial management.

3. Research Methodology

In this part we explains the research method which contains the definition of FMQL, architecture of FMQL and MongoDB commands categories.

3.1 Fuzzy Mongo Query Language (FMQL)

A description of the fuzzy query language is FMQL that it provided an extension of the MQL to support the flexible queries. Mainly in the development of MQL is used to support the linguistic labels and fuzzy comparators.

We can define a language for the linguistic label, if an attribute supports a fuzzy processing. These tags easily with the symbol (#) before those features will distinguish them. These tags at distances corresponding minimum and maximum values in a database that stores a meta-knowledge have been replaced. For example, in a document-oriented database NOSQL describing the staff, such a normal fuzzy query could be addressed:

>db.employees.find ({Age: #young, Salary: #well-paid})

This query allows to find the young and well-paid staff where “well-paid” and “young” are fuzzy tags described by fuzzy sets.

3.2 Architecture

The implementation of a fuzzy query system can be tackled in two types of architecture [23]: the low coupling and the high coupling. We have chosen the low coupling where new features are integrated through a software layer above the DBMS. The generated MQL queries will be executed by the existing MongoDB engine. Thus, MQL is used as a low level language to achieve fuzzy queries.

Implementation of a fuzzy queries system can be considered as two types of architecture: low connection and high connection: Here we chose connecting low where new features are integrated through a software layer above the DBMS. The produced MQL queries by the available MongoDB engine will run. So, MQL as a low-level language is used to attain the fuzzy query.

An extension of the MongoDB database management system architecture to support flexible data retrieval (fuzzy query) relational database architecture is as following. In this architecture after MongoDB commands categories, the functions of F-XML Syntax

layer are as an interface between fuzzy query modeling in FMQL and query modeling in the Mongo database management system (Figure. 1).

The F-XML Syntax layer is a tool that allows the automatically transition from FMQL query to equipollent F-XML query. The basic idea is that linguistic tags of fuzzy queries replaced the compatible Boolean phrase with MongoDB engine system. This method is by shown thanks of the translations of the layer F-XML Syntax.

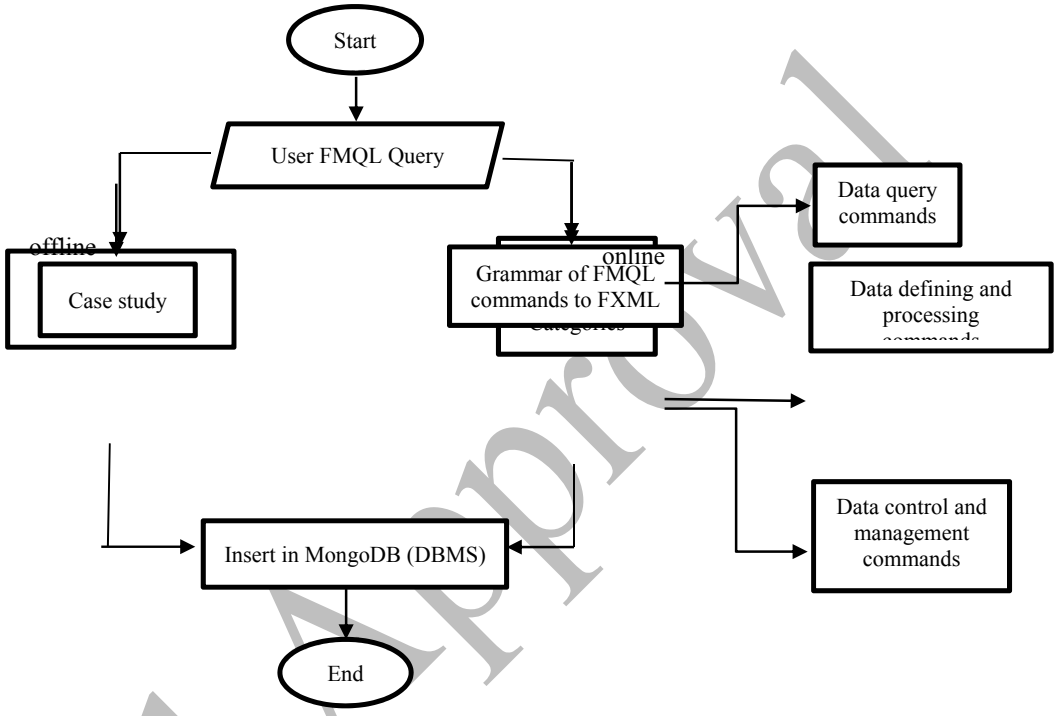

Figure 1. New MongoDB architecture to support fuzziness

3.3 MongoDB Commands Categories

Grouping of the commands in the document oriented database Mongo has the characteristics and for this reason it does not match to relational database commands category, accordingly we divided MongoDB commands to three categories as follows:

I. Data defining and processing commands such as: insert, update, drop, create, save, use, remove, sort, count distinct limit
II. Data query commands such as:Find
III. Data control and management commands such as: help, show dbs, stats

In the proposed method, uncertainty could be applied by adding a fuzzy (fuzzy label) in this database and fuzzy mapping (MongoDB) showed on F-XML. It should be noted that all orders could not be fuzzy, like control command and data management such as: help, show dbs, stats and data processing as well as part of the defining command such as use. In the following, there is a review of some of these commands are addressed.

I. Data defining and processing commands

MongoDB is a document-centric database which provides high performance, high availability and scalability. MongoDB works on the concept of the document and collection. Accordingly all commands do change and process on data they can insert and create or delete the data in documents and collection.

Some of these commands and syntax proposed in the following tables expressing the fuzzy MongoDB mapping to F-XML. It is needed to use insert () or save () to enter data in a MongoDB. If the collection does not exist in the database, MongoDB creates the collection and then puts the document in it.

To insert data into MongoDB collection, one needs to use MongoDB's insert () or save () method. In the inserted document, if we don't specify the _id parameter, then MongoDB assigns a unique Object for this document.

Here users dealing with fuzzy characteristics rather than species that can determine the exact value of each attributes, the use of linguistic variables, FMQL here by adding # to express any attribute MQL fuzzy stems. Syntax of insert with label of F-XML is expressed in Table 1.

Table 1. Syntax of Insert

Syntax / Command	MQL	FMQL	FXML
Insert	>db. COLLECTION_NAME. insert(document)	>db. COLLECTION_NAME. insert({e MongoDB xpression: #Value})	<ClassName> <Object> <AName>AValue</AName > <μ>[0,1]</μ> ------ </Object> </ClassName>

Remove () method is used in MongoDB to remove a document from the collection. If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. This is equivalent of SQL's truncate command.

MongoDB's db.collection.drop () is used to drop a collection from the database. MongoDB db.dropDatabase () command is used to drop an existing database. This will delete the selected database. If any database is selected, then it will delete default 'test' database. Syntax of Delete is expressed in the Table 2.

Table 2. Syntax of Delete

Syntax / Command	MQL	FMQL	FXML
Delete	1)>db.COLLECTION_NAM. remove(DELLETION_CRITTERIA) 2)>db.COLLECTION_NAME. drop() 3)db.dropDatabase()	1)>db. COLLECTION_NAME. remove ({expression, #Value})	<ClassName><Delete><from> <class name="CName"/> </from> <where> < AName ="AValue " ="[0,1]" /> </where></Delete> </ClassName>

To sort documents in MongoDB, you need to use sort () method. The method accepts a document containing a list of fields along with their sorting order. Order of 1 and -1 are used to specify sorting. 1 is used for ascending order while -1 is used for descending order. If you don't specify the sorting preference, then sort () method will display the documents in ascending order. Syntax of Sort is described in Table 3.

Table 3. Syntax of Sort

Syntax / Command	MQL	FMQL	FXML

Sort	>db.COLLECTION_NAME.
	find().sort({KEY:1})
		>db.COLLECTION_NAME.
		find ({expression,
		#Value}).
		sort({KEY:1})
			<xsl:sort
			select="expression"
			lang="language-code"
			data-type="text|number|qname"
			order="ascending|descending"
			case-order="upper-first|lower-first"/>

MongoDB's update () and save () methods are used to update document into a collection. The update () method updates the values in the existing document while the

save () method replaces the existing document with the document passed in save () method. Syntax of Update is expressed in Table 4.

Table 4. Syntax of Update

Syntax / Command	MQL	FMQL	FXML
Update	>db.COLLECTION_NAME. update (SELECTION_CRITERIA, UPDATED_DATA)	>db.COLLECTION_NAME. update ({expression: #Value }, {$set: { New expression: #NewValue}})	<ClassName> <Update> <Where AName=" AValue" μ="[0,1]"/> <Set AName="NewAValue" μ="New[0,1]"/> </Update></ClassName>

In MongoDB, it need not to create collection. MongoDB creates collection automatically, when some document is inserted. Collection-Name only required field when creating the collection. Creating a document could not be fuzzy. Creating syntax is expressed in Table 5.

Table 5. Syntax of Create

Syntax / Command	MQL	FMQL	FXML
Create	>db.createcollection("CollectionName ",{capped:Boolean, autoIndexID:Boolean, size:Number, max:Number})	---------	<ClassName> <attribute name="AName" type="nvarchar" unique="true/false" notnull="true/false"/> <attribute name="μ" type="decimal" notnull="true/false"/> ----- </ClassName>

II. Data query commands

In MongoDB for querying data, find () method is used which include the most workload for a user who works with the MongoDB. This command, along with several commands that are within the scope of this command is caused to widespread use of this command.

The command is express in Table 6. Find () method shows all documents in the unstructured way. FindOne () method is used to display a document.

Table 6. Syntax of Select

Syntax / Command	MQL	FMQL	FXML
Select	>db. COLLECTIO N_NAME. find()	>db. COLLECTION_NAM E. find ({expression, #Value})	<xfsql> <query> <select> <attribute name="AName" /> --- </select> <from> <class name="CName" /> </from> <where> < AName =" AValue " ="[0,1]" /> </where> </query> </xfsql> <where> < AName =" AValue " ="[0,1]" /> </where> </query> </xfsql>

III. Data control and management commands

As it explained above, the categories in MongoDB could not be fuzzy. The following cases are including data management and control commands in MongoDB:

- show

Show dbs command displays the list of databases.

- Help

This command displays a list of MongoDB commands and syntax of this command is as follows:

>db.help ()

- Statistics

Type the command db.stats() in MongoDB client to find statistics of the Mongo server. The order of the list includes the database name, number of collections and documents in the database.

4. Case Study

Since all MongoDB commands are considered in the study, the completeness of the study is confirmed. A case study in the field of banking is observed in the following.

One of the main activities is given the facility to the banks and the credit institutions. Their main challenge ensure the timely return of their principal and interest of the payment facility. Therefore, they need to have a comprehensive system to assess customer's credit to minimum the risk of their payment facility.

In this case study is studied the part of the facilities of a bank (related to the loan affairs). This case study is calculated the loan which awarded to a person based on the cases such as the age of the customer, turnover and customer relationship with the bank. Accordingly MongoDB database commands and XML were expressed by the fuzzy approach. Figure.

2 displays the membership function of the characteristics of age to determine the membership degree of age's customer in the case study of the banking system.

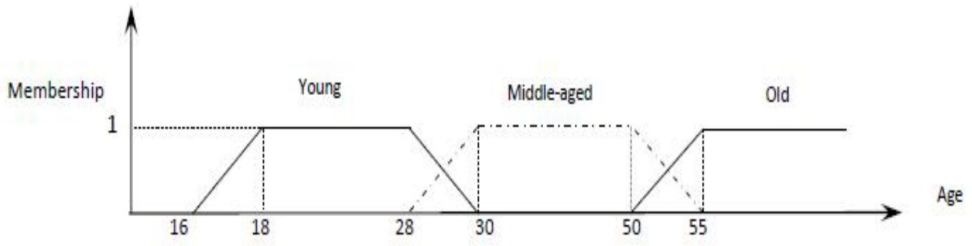

Figure 2. Fuzzy value for Age Attribute [17]

- Add the client to the bank database

The case study is implemented for the grammar of Insert in Table 7.

Table 7. Add client bank to the bank database

FUZZY MAPPING / Command	Fuzzy Mongo	Fuzzy XML
Insert	>db. customer.insert({ Customer name :"mohammad", Customer street :"saadi", Customer age: #Middle-aged, Customer account number: 025056, :Customer account balance 5523970, Customer turnover : # medium, LRFM: #low },	<bank> <Customer> <Customer name> mohammad </customer name> <Customer street> Saadi</customer street> <Customer age>Middle-aged μ=0.9 </customer <loan> <Customer account </Customer> age> number> 025056</Customer account number> < Customer accounts balance >55239700. </Customer accounts balance> <Customer turnover>medium μ=0.8 </Customer turnover> < LRFM> low μ=0.1 </ LRFM > </loan> </bank>

- Removing operation in the bank database

The case study is implemented for the grammar of Delete in Table 8.

Table 8. Removing operation in the bank database

FUZZY MAPPING / Command	Fuzzy Mongo	Fuzzy XML
Delete	>db.customer.remove({ Customer accounts turnover:#low}) > db.customer.drop()	<bank> <Delete><from> <class name= customer /> </from> <Where Customer accounts turnover =low μ=0.3/> </Delete> </bank>

- Update the customer characteristics in the bank database

The case study is implemented for the grammar of Update in Table 9.

Table 9. Update the customer characteristics in the bank database

FUZZY MAPPING / Command	Fuzzy Mongo	Fuzzy XML
Update	>db.Customer. Update ({Customer accounts balance: 5523970},{$set:{ New Customer turnover: #medium } })	<bank> <Update> <Where Customer accounts turnover =medium μ=0.4 /> <Set Customer accounts turnover =medium μ=0.8/> </Update> </bank>

- Query in the customer characteristics in the bank database

The case study is implementation to the grammar of Query in Table 10.

The used data in the case study are used and studied from the data of the Tejarat Bank dataset belongs to the year of 1390. Assessment of the case study shows, for example, when the membership degree is 0.6, if the loan ceiling is 100 million Tomans then the allowed loan for the customer will be only 60 million Tomans.

Table 10. Query in the customer characteristics in the bank database

FUZZY MAPPING / Command	Fuzzy Mongo	Fuzzy XML
Query	>db.customer.find({Customer turnover: #high})	<xfsql> <query> <select > <attribute name = Customer accounts turnover /> </select> <from> <class name="bank" /> </from> <where> <Customer accounts turnover = high μ=0.8 /> </where> </query> </xfsql>

5. Conclusions and Future Work

In the proposed method, we looked at the completeness of the work for keep the data. Therefore we showed the various commands in MongoDB by xml t o demonstrate the completeness of the work.

Unprecedented volumes of data, connected data and the needs of scalability and performance for the volume of data changed this view that the relational databases are the only data management-based approach. Compatibility and availability, and scalability are three basic concepts that will determine which data management systems are appropriate for the desired program. Although, NOSQL systems are mainly used for new programs with the characteristics of horizontal scalability, high performance and conditional compatibility, but fortunately existing programs with reengineering process have turned to the use of NOSQL systems.

The point that should be noted in this regard is that the developer has to choose the right tool to do the thing he/she wants to. This means that for many common applications, traditional databases are the best solution yet and should not be considered finished. NOSQL databases are appropriate for specific items that enhance the performance of all their software suite. One could not conclude which database is right and which database is wrong by comparison, but it should be based on functional requirements select the appropriate database. Accordingly, this database examined based on some of the features.

Table 11 investigates the proposed method with four approaches based on object-oriented, Neo4J, Cassandra and XML. In the proposed method and object-oriented approach based, Neo4J and XML-based applied on fuzzy relations but in the Cassandra approach were not applied.

The proposed method and approaches based of the non-relational database supported all types of data including structured and unstructured data, but XML and object-oriented database, able to do not process unstructured data and the big data but support structured data.

Table 11. Survey of the completeness of the proposed method

Approach / Feature	The proposed method	Cassandra-based approach [24]	based Neo4J approach [9]	Object-oriented approach [13]	XML-based approach [16]
Fuzzy relations	✓	×	✓	✓	✓
Relational operators	×	×	×	✓	✓
column oriented	×	✓	×	×	×
graph oriented	×	×	✓	×	×
Unstructured data	✓	✓	✓	×	×
Structured data	✓	✓	✓	✓	✓
ACID	×	×	✓	✓	✓
Big Data	✓	✓	✓	×	×
join	×	×	✓	✓	✓

The proposed method and approach Neo4J and Cassandra Unlike approaches on object-based and XML-based Neo4J based do not follow the ACID properties and relational operators. Neo4J and XML based and object oriented approaches used join for query, but the proposed method of approach and Cassandra approach has not been used join for query.

As mentioned above, we could not say which method is superior to another, it depends on the functional requirements and appropriate database is selected. In the proposed method applied the uncertainty in command, we could get closer to human language.

As future perspectives, we plan to (a) extend the fuzzy concept to description manipulating language dealing with data stored in a oriented document database of NOSQL, (b) Implementation of the Final Solution that allows read and write, and query flexibility to accept. (c) This database has many advanced capabilities and useful features such as database clustering, MapReduce and etc. that would be assessed in the future. (d) In this research, we express Fuzzy Mongo queries by label of F-XML and we don't use XML as the database. In future research fuzzy data would be stored in the XML database.

References

[1] B. Hindawi, H. M.El-Bakry, M. Abu-Elkheir, "Evaluating NoSQL Databases with Fuzzy Decision Making," International Journal of Computer Engineering and Information Technology, 2015.

[2] Z. Goli-Malekabady, M.k. Akbari-fatidahi, M. Sargozaei-javan, "An effective model for store and retrieve big health data in cloud computing," Computer Methods and Programs in Biomedicine, 2016.

[3] J. Liu, X.X. Zhang, "Data integration in fuzzy XML documents," Information Sciences, 2014.

[4] K. Kaur, R. Rani, "Modeling and Querying Data in NoSQL Databases," IEEE International Conference on Big Data, Silicon Valley, CA, USA, 2013.

[5] J. Bhogal, I. Choksi, "Handling Big Data using NoSQL," 29th International Conference on Advanced Information Networking and Applications Workshops, Washington, DC, USA, 2015.

[6] E.V. Ivanova, L.B. Sokolinsky, "Parallel Processing of Very Large Databases Using Distributed Column Indexes," Programming and Computer Software, 2017.

[7] V.N. Gudivada, D. Rao, V.V. Raghavan,"NoSQL Systems for Big Data Management," IEEE 10th World Congress on Services, Anchorage, AK, USA, 2014.

[8] S.D. Kuznetsova, A.V. Poskoninb, "NoSQL Data Management Systems," Programming and Computer Software, 2014.

[9] A. Castelltort, A. Laurent,"Fuzzy Queries over NoSQL Graph Databases: Perspectives for Extending the Cypher Language," Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2014.

[10] R. Akhoondi, R. Hosseini,"A Fuzzy Expert System for Prognosis of the Risk of Development of Heart Disease," Journal of Advances in Computer Research, 2016.

[11] V. Khodashenas Limouni, S.A. Gholamian, M. Taghipour Gorjikolaie, "Inter-Turn Fault Detection of PMSM Based on Fuzzy Logic and Discrete Wavelet Transform Using Unsupervised Clustering," Journal of Advances in Computer Research, 2016.

[12] A. S. Moghadam, B. Moshiri,"Fuzzy Threat Assessment on Service Availability with Data Fusion Approach," Journal of Advances in Computer Research, 2014.

[13] B. Fahimiana, A. Harounabadib, "A structure to support queries in object-oriented database using fuzzy XML tags," Management Science Letters, 2014.

[14] Z. Ma, "Fuzzy Database Modeling with XML," Springer Science and Business Media, 2005.

[15] G. Panić, M. Racković, S. Škrbić, "Fuzzy XML and prioritized fuzzy XQuery with implementation," Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 2014.

[16] L. Yan, "Modeling Fuzzy Data with Fuzzy Data Types in Fuzzy Database and XML Models," The International Arab Journal of Information Technology, 2013.

[17] F. Farazi, A. Harounabadi, M. Abbasi Dezfouli, "Presented a New Structure with Purpose Improvement Fuzzy Queries in Object Oriented Databas," International journal of Computer Science & Network Solutions, 2013.

[18] M. Pourbehzadi, A. Haroonabadi, M. Sadeghzadeh, "A new weighted fuzzy grammar on object oriented database queries," Management Science Letters, 2012.

[19] J. J. Miller, "Graph Database Applications and Concepts with Neo4j," Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, 2013.

[20] G. Prabagaren, "Systematic Approach for validating Java-MongoDB Schema," International Conference on Information Communication and Embedded Systems, Chennai, India, 2014,

[21] J. Yoon, D. Jeong, C. Kang, S. Lee, "Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study," The International Journal of Digital Forensics & Incident Response, 2016.

[22] X. Dai, Q. Xie, "Sept 16-18. An approach to build high-performance OPC-XML DA server system based on MONGODB and Comet," International Conference on Electrical and Control Engineering, Yichang, China, 2011.

[23] B. Kacem Abir, G. Touzi Amel, "Towards Fuzzy Querying of NoSQL Document-oriented Databases," The Seventh International Conference on Advances in Databases, Knowledge and Data Applications, Rome, Italy, 2015.

[24] R. Hernandez, Y. Becerra, J. Torres, E. Ayguade, "Automatic Query Driven Data Modelling in Cassandra," Procedia Computer Science, 2015.