IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Comparative Study of Integrity Constraints, Storage and Profile Management of Relational and Non-Relational Database using MongoDB and Oracle

V.J. Dindoliwala[1*], R.D. Morena[2]

[1]C. B. Patel Computer College, Bharthana, Vesu, Surat, India
[2]Department of Computer Science, Veer Narmad South Gujarat University, Surat, India

[*]*Corresponding Author: vaishali_1331@yahoo.co.in, Tel.: 98795 00031*

Available online at: www.ijcseonline.org

Accepted: 24/Jul/2018, Published: 31/Jul/2018

***Abstract*--** In the last decade, there is a rapid development in web technologies, social media applications and mobile applications which generates unstructured data. The way these applications deal with data has been changed extensively over the last decade. These applications collect more data and more users are accessing these data concurrently than ever before. Thus it is a big challenge for relational databases in terms of scalability and performance to handle these data which has given boost to the initiation of various NoSQL databases. Among the several NoSQL databases, MongoDB is the most popular document store database because of its sharding and aggregation framework coupled with document validations and efficient data manipulation, fine-grained locking, replication facility, administration capabilities and so on. In this paper, we have studied how integrity constraints, contents and resources are managed by MongoDB and also studied various features provided by MongoDB and compared them with the widely used Oracle database.

***Keywords*--** Relational databases, Non-Relational Databases, Integrity Constraints, Relationships, Resources, Profile

I. INTRODUCTION

MongoDB, an open source document store NoSQL database, is becoming more popular nowadays because of capability of handling high volume of data, high performance, high availability and automatic scaling. The high performance will be achieved by means of indexes, embedded data models and keys from embedded documents and arrays. High availability is achieved through the replication facility known as replica set which provides automatic failover facility. Automatic scaling is provided through automatic sharding which distributes data across a cluster of machines.

MongoDB uses a document-oriented data model which follows JSON (JavaScript Object Notation) like documents with loose structure. Documents may have hierarchical structure and grouped into heterogeneous collections that are stored into a database [1]. Like any relational databases, MongoDB has also various security mechanisms like authentication, authorization, database auditing and data encryption [7]. MongoDB provides robustness, scalability and flexibility which are not necessarily met by traditional relational database systems. But there are still some features like referential integrity, user profiles which are desirable in MongoDB. Referential integrity constraints guarantee that relationships between various data are preserved. For example, it ensures that a course should exist before students register in it. Such relationships are essential and have to be maintained in any databases. This constraint also ensures that no operations violate the integrity between various data [6]. Through resource and user profile management, we can restrict users from performing operations that exceed reasonable resource utilization. Oracle is the most popular relational database system used around for ages while MongoDB is comparatively new but it has been used by many applications.

The aim of this paper is to study how various integrity constraints like entity integrity, referential integrity and domain integrity are supported by the MongoDB and how resources, contents and profiles are managed by MongoDB and also made comparison with Oracle which motivates us towards finding the gaps in MongoDB and what further features can be added in MongoDB to make it suitable for all applications. The rest of the paper is organized as follows: Section II describes the data storage mechanism of MongoDB and Oracle. Section III is the literature review. Section IV describes the management of integrity constraints, content, resources and profile in MongoDB and also shows how they will be managed in Oracle. Section V discusses the reasons for not providing integrity constraint like referential integrity, user profile or resource management in MongoDB and it also gives the comparison of various features provided by MongoDB with Oracle. Section VI concludes the paper.

II. MONGODB VS ORACLE DATA STORAGE MECHANISM

Every database in MongoDB has a single .ns file and several data files. Each new data file will be double in size to avoid the wastage of space on disk for small databases and it also keeps large databases in mostly contiguous regions on disk. Within its data files, each database is arranged into namespaces which stores a specific collection's data. The documents and indexes for each collection have their own namespace. Metadata for namespaces is stored in the .ns file of the database. Each data file is made up of multiple extents in which data, indexes and MongoDB generated metadata are stored. The data and indexes for a collection will usually spread across multiple extents and they contained in their own set of extents. Whenever a new extent is required, MongoDB will try to use available space within current data files. If no space is found, MongoDB will create new data file [18]. While in Oracle, every database has one or more physical data files which contain all the database data including table and index data. And the logical units of database space allocation include data blocks, extents, segments and tablespaces. The Figure 1 and Figure 2 show the logical data storage structure of MongoDB and Oracle respectively.

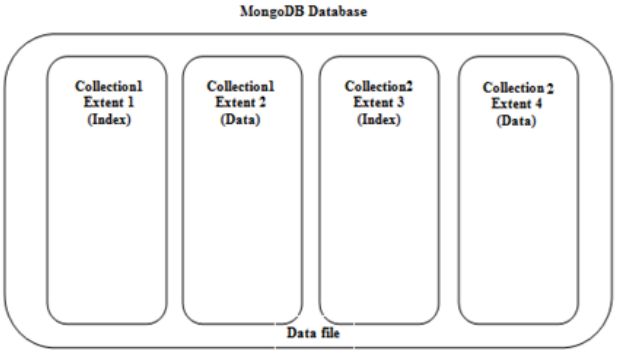

Figure 1. MongoDB Data Storage Structure

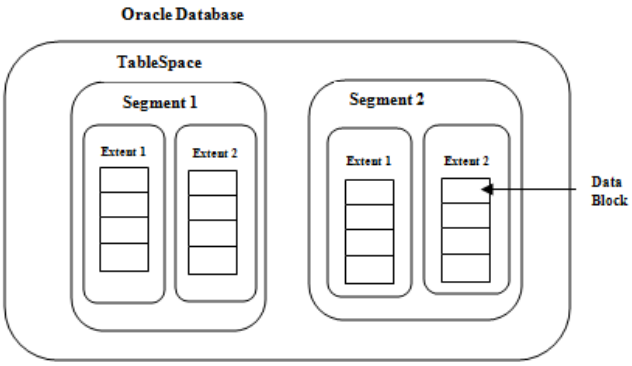

Figure 2. Oracle Data Storage Structure

In MongoDB, data are stored in the form of documents in a binary representation called BSON (Binary JSON) which extends the JSON representation to include additional types such as integer, long, date and floating point. These BSON documents contain one or more fields with one or more values [2]. Documents are similar to the concept of rows in Oracle. These documents are stored within the collection which is similar to a table in Oracle and the collection itself is stored in the database.

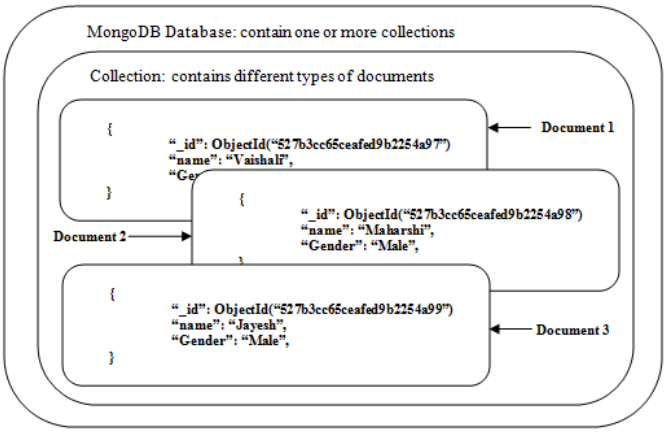

Figure 3. Document Structure in MongoDB

III. LITERATURE REVIEW

Much work has been done in comparing and determining the differences in performance of relational databases and non-relational databases. But, we have not seen any research work done about the performance comparison in terms of integrity constraints, resource or profile management in non-relational databases like MongoDB.

In their survey paper [4], A. Boicea, F. Radulescu and L. Gapin have compared MongoDB with the Oracle database. According to their research, the integrity model used by Oracle Database is ACID and MongoDB uses BASE. MongoDB offers consistency, durability and conditional atomicity. MongoDB doesn't offer integrity features such as isolation, transactions, referential integrity and revision control. They have suggested that if one requires a more complex database with relations between tables and a fix structure, one should stay with the classic Oracle Database.

In their paper [14], authors have discussed various types of data stores available with NoSQL. They have discussed about MongoDB focusing on CAP (Consistency, Availability and Partition Tolerance) theorem. They have also discussed MapReduce which can help in handling large volumes of data. And they concluded that NoSQL databases provide great opportunity where SQL databases are not useful.

K. Georgiev [5] has addressed several problems by using foreign keys and some semantic relationships between documents which are lying in the same collection or in different collections. They have implemented a verification approach which uses the MapReduce programming model in order to detect incorrect references in document oriented databases that may be caused by errors in the program code or

incomplete transactions. The proposed solution would only report the existence of issues like duplicate primary keys, references to non-existing primary keys without any suggestions about the cause for the generation of inconsistent records.

In their survey paper, Y. Li and S. Manoharan [17] have investigated performance of some NoSQL and SQL databases. They have compared CRUD operations on key-value stores implemented by NoSQL and SQL. They have concluded that for various NoSQL databases, the performance of these operations varies. They have also concluded that among the several NoSQL databases, MongoDB and Couchbase are the fastest for read, write and delete operations.

N. Jatana, S. Puri, M. Ahuja, I. Kathuria and D. Gosain [16] have also studied relational and non-relational databases for storing and retrieving huge amount of data and made comparison among them based on basic features provided by them. They have also discussed limitations of relational databases such as lack of high availability, replication, big data handling and faster update and discussed which type of NoSQL database to use to overcome these limitations.

A. Nayak, A. Poriya and D. Poojary [15] have surveyed various NoSQL databases based on data stores used and also discussed when each data store can be used which includes parameters like query language, availability, interface and consistency. They have also made comparison of NoSQL databases with relational databases in terms of advantages and disadvantages of NoSQL databases over relational databases.

B. Jose, S. Abraham and Praveen Kumar V. S. [19] have done performance analysis between MongoDB and MySQL with large number of records by performing simple query which includes only SELECT statement in MySQL and equivalent in MongoDB. They have concluded that increasing the number of records, the performance of MongoDB increases in contrast with MySQL. If number of records is small, there is not too much difference in the execution time taken by both the databases.

IV. COMPARISON OF MANAGING INTEGRITY CONSTRAINTS, CONTENTS AND RESOURCES IN MONGODB WITH ORACLE

The integrity constraints are used to enforce business rules by specifying conditions or relationships among the data. So that any operation that modifies the database must satisfy the corresponding rules without the need to perform any checking within the application. The term integrity with respect to databases includes both database structure integrity and semantic data integrity. The goal of database structure integrity is to ensure that each database object is created, formatted and maintained properly. The semantic data integrity refers to the data and relationships that need to be maintained between different types of data. After storing data in the database, it is important factor how efficiently and quickly the database system retrieves data from the database. Any database system provides options, procedures and controlling mechanism for defining and assuring the semantic integrity of the data stored within the databases. In big data environment, management of resources and user profiles are also a big challenge for managing various database resources like a database, collections, documents, files, CPU, memory, disk storage space etc. In the following sub sections, we have studied how integrity constraints, contents, resources and profiles are managed in MongoDB and also compare them with Oracle.

A. Entity Integrity Constraint

The role of a primary key in Oracle is to uniquely identify each record of the table. Similarly, in MongoDB, each document in a collection has unique identifier known as "_id" which will be treated as primary key for the document. User can also assign its own value to the "_id" field which can work as primary key. If user does not provide any value to the "_id" field then the system generates the ObjectId as the "_id". This ObjectId is the 12 bytes BSON type where the first 4 bytes represents the time in seconds since the UNIX epoch, the next 3 bytes represents the machine identifier, the next 2 bytes represents the process id and the last 3 bytes represents a random counter value. This ObjectId is generated while inserting the document in the database collection. In Oracle, one can change the value of primary key field which is not true with the "_id" field of MongoDB document. If one wants to change the "_id" field value of a document, one has to save the same document using a new "_id" and then one has to remove the old document.

B. Referential Integrity Constraint

In Oracle, the referential integrity constraint is maintained by defining a foreign key in the table which enforces the relationship between the two tables. One advantage of MongoDB is that all the data are present everywhere you need it and you can still pull the whole activity stream back as a single document. MongoDB lacks relations particularly the foreign key among the documents. If one wants to maintain relations among the documents, one has to maintain by building application level code. Still one can manage relations among the documents either using embedded document in a single document or using the referencing in MongoDB.

1) Embedded document approach: It provides strong association among documents. Through embedded approach, one can set one-to-one or one-to-many relationships among the documents [9]. Here, the related documents are going to be stored in a single document. So that using the single read, one can get all information related to particular document. The whole document will be persisted in the same collection

so, the read operation results in better performance as compared to relational databases because we do not require any join operation at all [10, 11]. This approach is generally used when a limited amount of information is going to be stored in an embedded document [12]. The problem with this approach is that, it increases the in-memory requirements. Also it may happen that some information is going to be repeated in all embedded documents which utilize more storage space which may lead to data redundancy and data inconsistency. If the data to be embedded is expected to grow larger in size, it is better to use referencing (Linking) approach to avoid the document becoming too large and to avoid the duplication of data.

2) Referencing approach: In Oracle, referencing among records is done by setting foreign keys in the table. In MongoDB, the referencing approach allows saving of "_id" field of one document in the related document as a reference. The referencing documents may be in the same collection or in a different collection. Referencing enables normalization of data and can give more flexibility than embedding approach. But in MongoDB, there is no mechanism to maintain relationships. The relations and their corresponding operations have to be taken care manually that is through the application code as no foreign key constraints and rules apply. There is no CascadeDelete mechanism which is there in Oracle. As compared to embedded approach, referencing provides weak association among documents. Also, in MongoDB, no joins are there. So to access referenced document, one has to first fetch "_id" field from the document and then one has to write second query to return the referenced data which takes additional round trip to the server which affects the performance as it requires multiple reads from multiple physical locations [9]. There is also a $lookup functionality for performing a left outer join to an unsharded collection in the same database to filter in documents from the joined collection for processing.

C. Domain Integrity Constraint

Domain integrity confirms that the column of the table must satisfy certain rules. So that performing any modification on the column value will not make database inconsistent. Oracle provides various domain integrity constraints like data types, null, not null, unique, check constraint and default value for a column of a table. MongoDB also supports a feature called document validation that can be used to enforce some validation rules on the documents structure inside a particular collection. These validation rules will be checked when a document is going to be inserted or updated within a collection. One can set the validation rules while creating the collection using the db.createCollection() with the validator option. If the validation rule is violated, an error or warning will be generated depending upon the validationAction option. Using these validation rules, one can define various domain constraints like data types, null constraint, check constraint, data ranges, requirement of mandatory fields etc. for the fields of a document [11, 12]. One can also create index on the document's fields through which one can achieve unique constraint for the document's field [2]. By default, MongoDB creates an index for the primary key of the document for faster access.

D. Resource Management

The resource management provides granular control of various database resources allocated to the users or the applications. Efficient resource management helps organizations to economize by associating servers. In Oracle, the resource manager controls the database instances' CPU utilization, limits the number of database sessions that are allowed to run concurrently within a group of users, manages each database sessions, limits the degree of parallelism for any database operation, automatically manages the workload across all the instances of the database and so on [3].

Various database resources in MongoDB include collections, single collection across databases, multiple databases across databases and a cluster that are accessed by the database users which can be controlled by creating roles in a database. We can set all these resources by creating roles using db.createRole() command. One can also give access privileges to the specific collection using db.createRole(). For example, if one wants to specify only "update" and "insert" privileges on "Student" collection of the "College" database then the command is as follows:

```
db.createRole({role:"Role1",privileges:[
{resource:{db:"College",collection:"Student"},
actions: ["update","insert"]}], roles:[]})
```

In MongoDB, to limit the size of the collection, there is a mechanism called "capped collections" which are the fixed-size circular collections which insert and retrieve documents based on insertion order which can be imagined as circular queues [13]. They offer high-throughput operations. When there is no space in a collection, it automatically removes the oldest document in the collection and makes space for new documents. These collections preserve the insertion order. So, queries do not need an index to return documents in insertion order. Thus, no extra overhead is required for indexing which provides higher insertion throughput. One can't remove documents from a capped collection. If you want to specify a maximum number of documents for this collection, you can do it by following way:

```
db.createCollection ("Test", {capped: true, size:
                              5242880, max: 5000})
```

MongoDB also supports horizontal scaling through sharding which divides the system dataset and load over the multiple servers by adding additional servers to increase the capacity as required. Each machine handles a subset of the overall workload which increases efficiency as compared to a single

high speed server. MongoDB handles sharding natively on a per-collection basis. The MongoDB partitions a collection using a shard key which exists in every document of the sharded collection. Using sharding, read and write across the shards will be more efficient. It also increases the storage capacity of the cluster by adding additional shards whenever required. [13, 18]

MongoDB also provides a mechanism for limiting and controlling the usage of system resources like threads, files and network connections on a per-process and per-user basis. The "ulimit" command refers to the per-user limitations for various resources that prevents single user from using too many system resources. The "ulimit" command will be performed on MongoDB instances. Using "ulimit" command, one can set various parameters like file size, cpu time, virtual memory, memory size and open files.

MongoDB also provides a mechanism called GridFS. It is useful when file system does not allow storing more than limited number of files and also when query is made, no need to load the whole file into memory. It stores file that exceeds 16MB. GridFS divides file in small chunks and stores across different document. The maximum size of each document is 255k. Here, two collections are maintained for the file - files and chunks. Files collection stores the metadata of the file and chunks collection stores information of each part of file in documents. [8]

E. Profile Management

Profile management is used to set the resource limit for the users of the database. For example, it restricts users from performing operations that exceed beyond the resource utilization. Oracle handles profiles for the users to limit the resource utilization by setting various profile parameters like concurrent sessions per user, CPU time limit for a session, session connect time, session idle time, number of data blocks read per session etc. MongoDB database profiler is a tool to collect server performance data. It is used for performance analysis. It provides information about operations that are executed on MongoDB instance. It collects fine grained data about queries, write operations, cursors and other database commands on a running server instance. The profiling can be enabled on a per-database or per-server instance level. Apart from this, there is no user profile management in MongoDB as any user can use any amount of data at any given time.

F. Content Management

Once we have stored data within a database, we may want faster retrieval of required data or want some more functionality than just retrieving them. For better data retrieval from the database, MongoDB uses various techniques like indexing, aggregation framework.

Indexing is the mechanism through which the speed of data retrieval can be improved. Conceptually, Indexing in MongoDB and Oracle are same. In MongoDB, index is defined at collection level and it can be created for a single field or combination of fields of a document of a collection while in Oracle, index is defined on table level and it can be created for any column of the table. In MongoDB, if there are no indexes in a database, to search a document from the collection, the full collection will be scanned to find the required document from the disk which is limited by server's disk subsystem I/O which will results in slow read operations thus affects the performance [8]. But indexes increase the read operation speed by avoiding unnecessary scanning of all documents of a collection from the storage. Thus indexes also minimize the cost of additional storage space.

The aggregation framework allows us to transform and combine documents in a collection. It groups the values from multiple documents together and can perform various operations on the grouped data and returns a single value. For doing this, MongoDB uses the aggregation pipeline and the map-reduce function which has functionality as the group by clause in Oracle.

V. DISCUSSION

NoSQL databases work on CAP theorem. According to this theorem, all these features can't be achieved at the same time. As opposed to relational databases, NoSQL databases are also ACID free. MongoDB follows the schema free structure for a collection of documents in a database which makes the code more error-prone, increases code duplication and easily creates deeply-nested structures also. The fields in a document can be added or deleted at any point of time. Users of the system are completely free to define the contents of a document at all times. They are not bound to a predefined set of tables, columns and their types as in the case with the relational databases like Oracle. This may be the one of the reason for not defining referential integrity in NoSQL databases like MongoDB. Documents can easily be modified by adding or deleting fields without any need to restructure the entire document. Also documents with old and new structure can still exist in parallel in the database collection. One can store whatever he wants irrespective of any other documents.

Again in MongoDB, by using embedded document approach for storing the related documents together, atomicity of document is achieved which satisfies requirement of data integrity in the database. One or more fields may be written in a single operation including updates to multiple sub-documents and array elements. So that any error occurs, entire operation will be rolled back and thus clients get a consistent view of the document.

As MongoDB is typically designed for the applications which have millions of concurrent users who are accessing any information continuously, resource and user profile management is a critical part for it to handle with the penalty

of performance. With relational databases, user profile can be handled as they are limited to a single server while distributed databases can scale out across multiple servers. The Table 1 gives comparison of various features provided by the MongoDB with Oracle.

Table 1. Features provided by MongoDB vs Oracle

Features Provided by	**MongoDB**	**Oracle**
Database Model	Document Store	Relational
Data Schema	Schema free, Dynamic	Predefined table structure and relationship
Query Language	JSON query language	SQL
Scaling	Horizontally scalable	Horizontally and Vertically scalable.
Entity Integrity	Provided by means of "_id" field of a document. Once it is set, one can't change it.	Provided by means of primary key.
Referential Integrity	Not provided. One can establish relationship between documents by using embedded document or referencing. But there is no mechanism for CascadeDelete. One has to take care of it by writing application code.	Provided by defining foreign keys. Also supports CascadeUpdate and CascadeDelete mechanisms.
Domain Integrity	Provided by means of document validation.	Provided.
Joins	Not provided.	Provided.
Indexing	Provided.	Provided.
Resource and profile management	Provided. But can't be set on per user basis.	Provided.
ACID properties	Follows the CAP theorem. Partial support for ACID.	Provided.
Consistency	Eventual consistent	Provided.
Normalization-Denormalization	Embedded approach provides denormalization of data while referencing provides normalization.	Provides normalization by dividing single table into smaller tables to minimize data redundancy and improves performance.
Aggregation framework	Using aggregation pipeline and Map-reduce function.	Using Group by

VI. CONCLUSION

Relational databases are widely used databases and they have good performance when limited amount of data is there. But to handle large amount of data, they will be insufficient. Again, it is the job of the developer to decide which database to use to meet the application requirements. MongoDB provides data model flexibility, scalability, high performance and availability. The horizontal scaling feature of MongoDB significantly reduces the storage cost. It is the database that enables developers to build applications faster and can give ability to enhance their applications continuously. In MongoDB, by adding some lacking features like integrity constraints or profile management, one can use it in applications with small amount of data.

REFERENCES

[1] P. Colombo, E. Ferrari, "Enhancing MongoDB with Purpose-Based Access Control", IEEE Transactions on Dependable and Secure Computing, Vol. 14, Issue. 6, pp. 591 – 604, 2015.

[2] A MongoDB White Paper, "MongoDB Architecture Guide", MongoDB 3.2.

[3] An Oracle White Paper, "Effective Resource Management Using Oracle Database Resource Manager", 2011.

[4] A. Boicea, F. Radulescu, L. Gapin,"MongoDB vs Oracle -- Database Comparison", Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), 2012, doi: 10.1109/EIDWT.2012.32.

[5] K. Georgiev, "Referential Integrity and Dependencies between Documents in a Document Oriented Database", GSTF Journal on Computing (JoC), Vol. 2, No. 4, pp. 24-28, 2013.

[6] H. Raja, "Referential Integrity in Cloud NoSQL Databases", A thesis submitted to the Victoria University of Wellington, 2012.

[7] V. J. Dindoliwala, R. D. Morena, "Survey on Security Mechanisms In NoSQL Databases", International Journal of Advanced Research in Computer Science, Vol. 8, No. 5, pp. 333-338, 2017, ISSN No. 0976-5697.

[8] S. Agrawal, J. Verma, B. Mahidhariya, N. Patel, A. Patel, "Survey on MongoDB: An Open-Source Document Database", International Journal of Advanced Research in Engineering and Technology, Vol. 6, Issue. 12, pp. 01-11, 2015, ISSN Print: 0976-6480.

[9] Z. Parker, S. Poe, S. Vrbsky, "Comparing nosql Mongodb to an sql db", proceeding of the 51th ACM Southest Conference, Article No. 5, 2013, ISBN: 978-1-4503-1901-0.

[10] Chaitanya. P, Ranjan H. P, Kiran T. S, Anitha. K, "Implementation of an Efficient MongoDB NoSQL Explorer for Big Data Visualization", International Journal of Advanced Networking & Applications (IJANA), pp. 444 – 447, ISSN: 0975-0282.

[11] A MongoDB White Paper, "RDBMS to MongoDB Migration Guide", Considerations and Best Practices, 2018.

[12] K. Bhamra, "A Comparative Analysis of MongoDB and Cassandra", A thesis presented for the degree of Master of Science, Department of Informatics, University of Bergen, 2017.

[13] Swathi N, "Making your Application Highly Available and Highly Scalable using NoSQL Database (MONGODB)", International Journal of Advanced Computational Engineering and Networking, Vol. 1, Issue. 7, pp. 40 – 43, 2013, ISSN: 2320-2106.

[14] L. Bonnet, A. Laurent, M. Sala, B. Laurent, N. Sicard, "Reduce, You Say: What NoSQL can do for Data Aggregation and BI in Large Repositories", 22nd International Workshop on Database and Expert Systems Applications, pp. 483-488, 2011, doi: 10.1109/DEXA.2011.71,

[15] A. Nayak, A. Poriya, D. Poojary, "Type of NoSQL Databases and its Comparison with Relational Databases", International Journal of Applied Information Systems (IJAIS), Vol. 5, No. 4, pp. 16-19, 2013, ISSN : 2249-0868.

[16] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, D. Gosain, "A Survey and Comparison of Relational and Non-Relational Database", International Journal of Engineering Research & Technology, Vol. 1, Issue. 6, pp. 1-5, 2012, ISSN: 2278-0181.

[17] Y. Li, S. Manoharan, "A performance comparison of SQL and NoSQL databases", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 15-19, 2013, Electronic ISBN: 978-1-4799-1501-9.

[18] K. Chodorow, "MongoDB: The Definitive Guide", 2nd edition, O'Reilly, 2013, ISBN: 978-1-449-34468-9.

[19] B. Jose, S. Abraham, Praveen Kumar V. S., "Query Performance Analysis in NoSQL and Relational Databases: MongoDB Vs MySQL", International Journal of Computer Sciences and Engineering, Vol. 6, Special Issue. 4, pp. 179-182, 2018.

Authors' Profiles

Ms.V. J. Dindoliwala, Asst. Prof., C. B. Patel Computer College, Bharthana, Surat, has got her M. Phil., M.C.A. and B.E. Electronics degree from Veer Narmad South Gujarat University, Surat and is pursuing Ph. D. from the same university. She has published 4 research papers in National and International conferences and journals. She has 9 years of teaching experience.

Dr R D Morena is working as a Professor in department of Computer Science, VNSG University, Surat. He has been associated with teaching in MCA course since last 23 years. He has obtained B.Sc. (Computer Science) and MCA degrees. He has been awarded M.Phil(Computer Sc.) in 2001 and Ph.D. (Computer Science) in 2003. His research area is Data Management. He has 51 research papers published in reputed journals & conference proceedings. He has co-authored 5 books on computer Sc. subjects. He is a member of the review committee of various national & international journals. He is a member of the Departmental Research Committee and is a registered Ph.D. guide at VNSG University. At present he is supervising 9 research students in Computer Science. He is a member of Board of Studies of Computer Science and also Information Technology.

