Data Conversion from Traditional Relational Database to MongoDB using XAMPP and NoSQL

Insha Mearaj
Department of Computer Science Engineering
Amity University Dubai
Dubai, UAE
imearaj@amityuniversity.ae

Piyush Maheshwari
Department of Computer Science Engineering
Amity University Dubai
Dubai, UAE
pmaheshwari@amityuniversity.ae

Maninder Jeet Kaur
Department of Computer Science Engineering
Amity University Dubai
Dubai, UAE
mkaur@amityuniversity.ae

Abstract— **MongoDB is a document-oriented database which helps us group data more logically. This paper demonstrates the conversion of data from a native tabular form to unstructured documents. The document and collections within needs not to be well defined prior to the creation hence adding to its flexibility. MongoDB has lots of extensive built-in-features and is highly compatible with other software systems, with extensive ways of accessing data beyond JSON query, its highly compatible Business Intelligence Connector is highly compatible which makes it compatible. High scalability is making it remarkable and popular and hence made us think about writing a paper demonstrating the data conversion. This conversion has helped us in making the most of modern data. Data was stored on the cloud as cloud-based storage is an excellent and most cost-effective solution. Our solution is highly scalable as the built-in shading solution for data handling makes it one of the best big data handling tool. The data we have used is location based in MongoDB that can directly yield data without hitches which made us work on MongoDB with the focus on the latest feature of MongoDB to support Multi-Document ACID transactions to maintain data integrity.**

Keywords— MongoDB, XAMPP, CVS, JSON, NoSQL, ACID Transactions

I. INTRODUCTION

Database is usually defined as collection of data and the system that handles data, transactions, problems of the database is Database Management System (DBMS). Databases were incepted in 1960 in order to satisfy the need of storing and finding data. They began with navigational databases which were based on linked-lists, moved on to relational databases with joins, afterwards object-oriented and without joins in the late 2000s NoSQL (MongoDB, Cassendra, HBase/Hadoop, CouchDB, Hypertable etc.) emerged and has become a popular trend [1]. On one hand there is relational database management system (RDBMS) that offers more consistency as well as powerful query capabilities and a lot of knowledge and expertise over the years [2]. On the other hand, NoSQL approach, which offers higher scalability i.e. it can run faster and support bigger loads.

NoSQL in general doesn't support complicated queries and doesn't have a structured schema. It recommends de normalization and is designed to be distributed (cloud-scale). Because of the distributed model, any server can answer any query. Servers communicate amongst themselves at their own pace. The server that answers your query might not have the latest data.

NoSQL characteristics –

- Can handle large data volume.
- Scalable replication and distribution – NoSQL automatically spreads the data into multiple servers without requiring application assistance. Servers can be added or removed from the data layer without application downtime.
- Schema-less : Data can be inserted in a NoSQL database without first defining a rigid database schema. The format of the data being inserted can be changed anytime without application disruption. This provides immense application flexibility that delivers substantial business flexibility.
- Open Source development

This research paper is based on data conversion using Mongo DB. We have demonstrated data conversion from traditional database to MongoDB document based database. Section II covers the related work done in this field. In section III the performance of the proposed concept has been verified with the aid of XAMPP and MongoDB management system. Results are followed up in Section IV presents the conclusion.

II. RELATED WORK

This research work is based on MongoDB in which relations within the applications are well defined by two main tools which are references and embedded documents. References store the relationships between data by including links or references from one document to another. Applications can resolve these references to access the related data. Having the URL link within the field definitely adds to its range of accessing data. Embedded documents captures the relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document along with a new feature.

MongoDB is derived from the word 'humongous' as the database has a characteristic of exceptional scale up capability. So big data are huge massive exponentially changing and growing data sets [3]. Traditional framework was not able to handle and analyze large amount of data. Also internet of things (IoT) is creating exponential growth in data. Big data and linked data are two faces of the same coin both representing the integral and concentric part of web based world. Each data element is connected via a URL's and are identifiable, locatable and accessible. Stronger the bond between the two. Higher will be the unsustainability and big data utility. The hottest technology with lots of opportunities to improve upon. Semantically well structured, interconnected and linking data will add intelligence to the data. The main goal is to make data more interactive, participatory and innovative. Making sense and extracting information from data for organizational benefits. So, semantic technologies can help us extracting the main value from the data. Test analytics aids in the conversion of n unstructured text to structured meaningful text by deriving patterns and extracting patterns.

Database scaling is a highly difficult task, modern applications require high scalability and data curating capacity. We have used XAMPP which is apache distribution web server solution for creating a local web server, after setting the environment the database was tested. JSON enabled the transferring data between the master server and the web application in human readable form. This improved the overall efficiency.

III. Proposed Work

The key benefits of NoSQL are speed, scalability, price, flexibility and simplicity. The main characteristics is its Non-adherence to relational database concept. The key benefits of NoSQL are speed, scalability, price, flexibility and simplicity. As an example Grolinger et al [4] identified difficulties in handling information which can be huge Map Reduce. Naheman and Wei [5] studied and compaired various BigData tools like HBase and other NoSQL databases eg Bigtable, Cassandra, CouchDB, and MongoDB etc. Laurence [6] worked on a virtualization system which does allow us to enquire and join information making use of a sql query and the result in API that is underlying to MongoDB.

Hadoop - the main frame technology being a Java based framework that supports the processing and storage of tremendously large data sets in a distributed computing environment. Even though as we all know that Hadoop is written in Java programming language, programs for Hadoop can be written in other languages like Python. Mostly Python code is translated to Java jar files using Jython. Hadoop has two major areas of concern. The first one is Hadoop which is built using Java and hence the application developers should know Java to develop the framework and to develop the map reduce technologies. Hadoop provides a framework by which Map Reduce applications can be built using python. The second is the co-sister technologies which work on top of this framework one of the example being Cassandra, Cassandra being a NoSQL database technology ideal for high-speed online transactional data, while as Hadoop being a big data analytic system.

Apache Sparke once a component of the Hadoop ecosystem is now fetching big data platform of choice for enterprises. Surveys reveal that many data analyst and data scientist preferred spark over map reduce, which is batch oriented and it does not offer itself for interactive applications and real time stream processing. As we advance towards Internet of things, we forward towards the era of sensor based things too, the sensors that are intended to send the data back to the mothership repository. The data that we deal with is mostly very complex and is deployed across various relational and non-relational systems. However the demand for analytical tools is increasing. Such tools helps us in extracting and utilizing data stored anywhere. Even for the sensor input data, the input data is tremendous. To cater the efficiency Metadata catalogues help us to relate and understand data. Machine learning is definitely automating the task of finding data in Hadoop. Some of the emerging tracks in Big Data are in the fields of Sensing and Internet of things Services, Smart City Data, Big Data Networking.

1) NoSQL comparisons

NoSQL system of designing the database is a relation less and hence schemaless, they are now constructed on a single model but and layer out to adopt different ones. There are almost a handful of different operational models for NoSQL.

Key/value, e.g. in MemchacheDB, Redis, etc.

Column, e.g. HBase

Document, e.g. MongoDB, Couchbase, etc.

Graphs, e.g. Neo4j, OrientDB.

We have used MongoDB which is document based. This technology uses deeper nesting and can handle much more complex structures.

2) Defining data model

We have created three collections in our projects. They are movies, ratings and users. The data model used in Movies is as

```
{
"_id" : ObjectId("58ab3f58ce38be146435cc76"),
            "id"  :"35",
            "title" : "20",
             "release_date": "F",
             "IMDBURL" : "homemaker",
              "genre ":”42459"
}
```

The data model used in ratings is as

```
{
"_id" :ObjectId("58ab3f58ce38be146435cc76"),
          "user" : "35",
          "movie" : "20",
          "rating" : "F",
          "timestamp" : "homemaker”
}
```

The data model for users is

```
{
"_id" :ObjectId("58ab3f58ce38be146435cc76"),
          "id" : "35",
           "age" : "20",
                    "gender" : "F",
                    "occupation" : "homemaker",
        "zip_code" : "42459"
                              }
```

3) Process to be followed for data conversion

1. **INPUT**: The input for this project is a movie.normalised_2015-11-30.sql file, which we extracted to zipped form using WinRAR file compressor and later imported the same into XAMPP phpMyAdmin.
2. **CONVERSION:** The first tool we used was XAMPP.

XAMPP Stands for Cross platform (X), Apache (A), MariaDB (M), PHP (P), and Perl (P). Using XAMPP we will take the compressed input file movie. Normalized and convert it to JSON file. But before it we will create a view of genres and movies, the view is named as moviefile. The attached snapshot covers the view creation with only the required attributes needed. The original database with five tables, we merge two tables into one view and named it as moviefile. In Moviefile view we have taken id, title, release date, IMDBURL, we dropped video attribute because the column was empty.

The code we used to create a view is as follows:

```
CREATE VIEW Moviefile AS  SELECT  id, title
AS      Name_OftheMovie,      release_date      AS
DateofRelease, IMDBURL ,Genres  FROM movies,
genres
              WHERE movies.id = genres.id
```

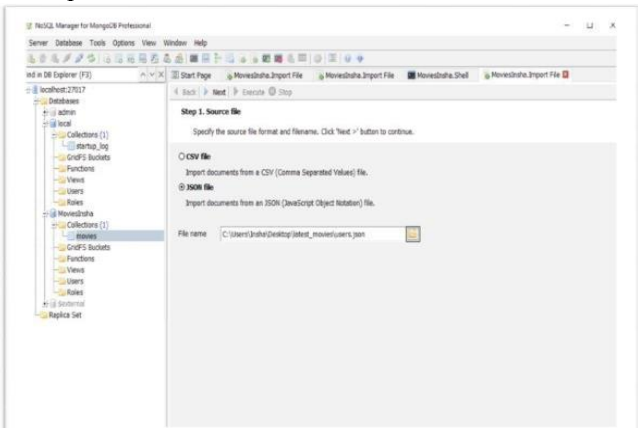
Once the View is created we export this view along with other two tables to NoSQL manager for MongoDB hence therefore Ratings, users and moviefile is exported to NoSQL manager for MongoDB. But before we export the file we are supposed to convert the view created and the SQL files users and ratings to JSON File which is later imported into NoSQL manager for MongoDB. NoSQL Manager GUI which has an embedded smart shell to work on, hence acting as a platform to work upon to maintain, update, embed, delete, interact and querying the database repository in a protocol set environment. The performance rate of NoSQL manager for MongoDB is A grade efficiency, especially the enterprise version, it's easy to use document view, easy management utility feature makes it excellent to use and implement, its SSH tunneling capacity for the connection enables us to work on a highly secure system. Map-Reduce operations editor inbuilt feature is remarkable in itself it saves all map reduce parameters on the current session in an XML format. Another point to add on is the GridFS which will divide the file onto chunks of size 255KB, with an exception of the fast chunk which as large as needed. Therefore the technological feature uses two collections one for the file chunks and the other for file metadata. Though for analyzing the data MongoDB can work very well, still the complexity of the database may increase leading to the need of a platform to hold it. This is where Hadoop can provide a powerful framework for complex analytics and measurements. During the encounter of such a scenario data can be pulled from MongoDB and is processed and taken care by Hadoop via the tool of map reduce. Nevertheless Hadoop is stronger enough that it can act as data warehouse and can hold the central control.

Data business analytics now has an option of holding Map Reduce [4] and Pig Map Reduce procedure is used to extract, transform and load data repository from one location to another. It pulls data from a place, transforms the data by using new methods and after transforming it to the desired shape it will load the data to another repository. Using this approach we can handle high tech data very easily. MongoDB is implemented in C++ [7-9] and its libraries, drivers and plugins are written in C extension for better performance and for it to be faster as for it no inter-language conversion in required.

The steps for exporting the JSON files are illustrated as below:

1.Turn on the exe files in MongoDb for the NoSQL ManagerDb Professional to work. Once both the exe console terminals are open go to NoSQL manager for MongoDB and start importing the JSON we got from XAMPP. Steps for extracting it are as below:

1. Select the JSON file type and enter the file name path, will be the first foremost step. Figure 1 represents it.



Step 1. Source file

Specify the source file format and filename. Click 'Next >' button to continue.

( ) CSV file

Import documents from a CSV (Comma Separated Values) file.

[✓ JSON file]

Import documents from an JSON (JavaScript Object Notation) file.

File name C:\Users\Insha\Desktop\latest_movies\users.json

Figure 1 – Screenshot displaying File type

2. Next to it, we have to select and adjust the JSON option, File encoding methodology is to be selected in this step.
3 The Third step is to specify the import option, we in this step highlight the new collection option

and specify the name of the collection we want to use, so the JSON is imported in the new the new collection. Once we name the collection click on the Execution button to make it run. Takes the code conversion to a different level.

4 The Fourth step is to check the import process status. In the figure below, data importing process indicates that the system is fully importing.

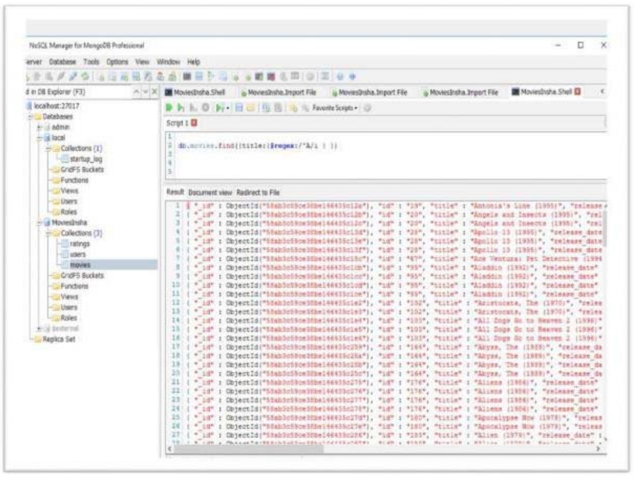
IV. Results and Discussions

MongoDB guarantees data integrity with multi-document transaction which makes MongoDB remarkable, this makes it well versed for developers to address the complete set of queries on the database. Interacting with the database constructed over on MongoDB supports multi-document due to which a global consistent view of data is maintained. This new outlook of MongoDB made it possible to transact through multiple databases with the power of multi-document ACID transaction.

We were successfully able to convert the JSON files to MongoDB format and consequently now our database is ready for interactions. Once the database is ready, we can extract information from the MongoDB multisite database verifying multi-document ACID transaction.

MongoDB Queries

1. **db.movies.find ({title:{$regex:/^A/I}})**:This extract titles starting with letter A and that is what is happening in the below query. Figure 2 is the snapshot taken from the NoSQL Manager for MongoDB Professional.

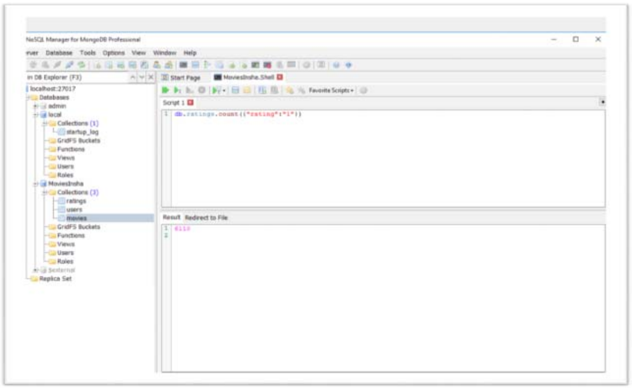


```
db.movies.find({title:{$regex:/^A/i } })
```

```
{ "_id" : ObjectId("[illegible]"), "id" : "19", "title" : "Antonia's Line (1995)", "release
{ "_id" : ObjectId("[illegible]"), "id" : "20", "title" : "Angels and Insects (1995)", "rel
{ "_id" : ObjectId("[illegible]"), "id" : "20", "title" : "Angels and Insects (1995)", "rel
{ "_id" : ObjectId("[illegible]"), "id" : "28", "title" : "Apollo 13 (1995)", "release_date
{ "_id" : ObjectId("[illegible]"), "id" : "28", "title" : "Apollo 13 (1995)", "release_date
{ "_id" : ObjectId("[illegible]"), "id" : "28", "title" : "Apollo 13 (1995)", "release_date
{ "_id" : ObjectId("[illegible]"), "id" : "67", "title" : "Ace Ventura: Pet Detective (1994
{ "_id" : ObjectId("[illegible]"), "id" : "95", "title" : "Aladdin (1992)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "95", "title" : "Aladdin (1992)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "95", "title" : "Aladdin (1992)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "95", "title" : "Aladdin (1992)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "102", "title" : "Aristocats, The (1970)", "relea
{ "_id" : ObjectId("[illegible]"), "id" : "102", "title" : "Aristocats, The (1970)", "relea
{ "_id" : ObjectId("[illegible]"), "id" : "103", "title" : "All Dogs Go to Heaven 2 (1996)"
{ "_id" : ObjectId("[illegible]"), "id" : "103", "title" : "All Dogs Go to Heaven 2 (1996)"
{ "_id" : ObjectId("[illegible]"), "id" : "103", "title" : "All Dogs Go to Heaven 2 (1996)"
{ "_id" : ObjectId("[illegible]"), "id" : "164", "title" : "Abyss, The (1989)", "release_da
{ "_id" : ObjectId("[illegible]"), "id" : "164", "title" : "Abyss, The (1989)", "release_da
{ "_id" : ObjectId("[illegible]"), "id" : "164", "title" : "Abyss, The (1989)", "release_da
{ "_id" : ObjectId("[illegible]"), "id" : "164", "title" : "Abyss, The (1989)", "release_da
{ "_id" : ObjectId("[illegible]"), "id" : "176", "title" : "Aliens (1986)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "176", "title" : "Aliens (1986)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "176", "title" : "Aliens (1986)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "176", "title" : "Aliens (1986)", "release_date"
{ "_id" : ObjectId("[illegible]"), "id" : "180", "title" : "Apocalypse Now (1979)", "releas
{ "_id" : ObjectId("[illegible]"), "id" : "180", "title" : "Apocalypse Now (1979)", "releas
{ "_id" : ObjectId("[illegible]"), "id" : "183", "title" : "Alien (1979)", "release_date" :
```

Figure 2 Result of db.movies.find({title:{$regex:/^A/I}}) query

2. The Second query is **db.ratings.count ({“rating” : “1”})** we have movies as the collection name ,ratings is one of the attributes and count is the member function which will count the number of instances of ratings. Figure 3 demonstrates the result.



```
db.ratings.count({"rating":"1"})
```

```
6110
```

Figure 3 - Result of db.ratings.count({“rating” : “1”}) query

V. Conclusion

The efforts to ensure the smooth and flexible functioning of a business greatly depends on the software capabilities of the Big Data Management Framework, Data curation, scalability and data security are the key features achieved. Through the iterations it’s clear that interactions have become easier on Big Data and even huge real time data management is possible with high understanding and accuracy. Sensor data and many more can be translated to a MongoDB format which is composed of fields and value pairs, the document in MongoDB being similar to JSON. In this paper JSON enabled the transferring data between the master server and the web application in human readable form. It was concluded from the results that MongoDB provides high performance on data persistence, support for embedded data models reduces I/O activity on database high speed interactions can include keys from embedded documents and arrays. The resultant data repository will be in document format. The JSON document we have can be very easily parsed with ease directly by JavaScript.

References

[1] K. Berg, T. Seymour, and R. Coel, “History of Databases,” *International Journal of Management and Information Services*, 17, 2013.

[2] Michael Stonebraker: SQL databases v. NoSQL databases. Commun. ACM 53(4): 10-11 (2010).

[3] Zhu Wei-ping, Li Ming-xin, Chen Huan, “Using MongoDB to Implement Textbook Management System instead of MySQL”, *IEEE*,978-1-61284-486,2011

[4] W. Naheman and Jianxin Wei, "Review of NoSQL databases and performance testing on HBase," Proceedings of International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), pp. 2304- 2309, Dec.2013

[5] R. Lawrence, "Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB,"Proceedings of International Conference on Computational Science and Computational Intelligence (CSCI), pp.285-290,Mar.2014.

[6] Dharmasiri, H.M.L and Goonetillake, M.D.J.S.,“A federated approach on heterogeneous NoSQL data stores,” in IEEE International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, December 2013.

[7] Julien Carme, R&D engineer, Atos Worldline and Francisco José Ruiz Jimenez, Technical Architect Manager, Atos Spain, “Open Source Solutions for Big Data Management,” Atos.
[8] Sanobar Khan, Prof. Vanita Mane” SQL Support over MongoDB using Metadata”,IJSRP, Volume 3, Issue 10, October 2013
[9] S. Hall (2014, Jul) MySQL vs MongoDB, jul, 2014, Available: http:// www.scriptrock.com/articles/mysql-vs-mongodb, acceseed nov. 2018.