Inclusion of e-Commerce Workflow with NoSQL DBMS: MongoDB Document Store

Dharavath Ramesh[*] Member IEEE, Ekaansh Khosla[1]
Department of Computer Science and Engineering
Indian Institute of Technology (ISM)
Dhanbad-826004, India
ramesh.d.in@ieee.org[*], ekaanshkhosla007@gmail.com[1]

Shankar Nayak Bhukya[2]
Department of Computer Science and Engineering
Swami Vivekananda Institute of Technology
Hyderabad-500003, India
bsnaik546@gmail.com

***Abstract*—In today's open market, e-commerce has developed its strategy to process various business applications. There applications are developed with the platforms like Magento, Zen Cart, Prestashop, Spree etc., to run a successful online store. These are also used for customer e-commerce apps and require a scalable database for storing the data. The NoSQL database provides an efficient storage access and processing environment with horizontally scalable and replicable strategy over RDBMS. Because of its schema less structure, NoSQL databases provides access to the users to dynamically change the data model applications. Due to this characteristics, NoSQL databases are being used in real time analysis. In order to build a successful online and scalable store, in this study, we choose the database as Document-Oriented NoSQL database for the inclusion of e-commerce workflow. As a new finding, in this study, we will ascertain design and working strategy of e-commerce based on MongoDB, which is a widely used Document-Oriented NoSQL database to process large scale business applications.**

***Keywords*—MongoDB; RDBMS; e-Commerce; NoSQL Db**

I. INTRODUCTION

Traditional database systems and RDBMSs are used to store important data of organizations by providing back-end of a web services [2]. These systems works very well while handling a limited amount of data. However, while dealing with data analysis, social services related data and in other entity processing, relational databases becomes very expensive and complex. To magnify and overcome the problems of RDBMS, a terminology called "*No SQL (Not Only SQL)*" is modeled[3]. NoSQL database handles unstructured data, email, documents, and social media efficiently. As NoSQL database servers are easily scalable and having replicated strategy than RDBMS, it provides efficient storage access for processing very large data. The common features of *NoSQL* databases are summarized in terms of its schema less structure, no support of joining operations, high scalability, simple data modeling with simple query language. On the other hand, it has high availability of losing *ACID* properties of traditional databases. *NoSQL* uses cheap commodity server to manage large amount of data. It also uses distributed indexes for efficient data storage and has an ability to add new attributes to the data records over many servers [3]. These are developed with major internet companies such as E-bay, Facebook, Google, Amazon and Twitter which has significantly different challenges in dealing with large amount of data where the traditional DBMSs could not handle. NoSQL daabases also support functional instances such as different key features [3]. The categories of NOSQL falls into three different models; a) Column-Oriented NoSQL; contains extendable columns of closely related data but do not support table association. E.g. Cassandra, HBase; b) Key-Value NoSQL stores; corresponds to a key and data is stored as a key-value pairs. E.g. Redis, flare; c) Document Oriented NoSQL; data is organized and stored as a collection of documents but the value is stored in JSON or XML format. E.g. MongoDB, Couch DB [3].

Electronic commerce which is also known as e-commerce; is a term used for business and commercial transactions which involves the transfer of information across the Internet. In the current scenario, it is one of the important aspects of the Internet to emerge. The strategic instances make e-commerce to allow consumers to electronically exchange services and goods with no restrictions of distance and time. It also been expanded at a very high pace over the past five years and is predicted to continue with accurate exponential rates. Performing transactions electronically is a very advantageous aspect over traditional methods. When it is implemented properly, e-commerce became very cheaper, faster and flexible than the traditional methodologies of referential services. Different benchmark systems made this possibility to scale the database with accurate occurrences [7]. As *e-commerce* has many advantages over traditional methods, there are also some challenges to implement. *E-commerce* sites are very rich in content. With a wide variety of product offering leading to millions of products, tons of rich images and millions of geographically distributed shoppers, E-commerce sites and platforms has to face a big challenge of maintaining consistent availability, performance, and throughput across all channels [6]. The problem with the relational databases is that they run on a single server, so the resources available to the database for processing, storage and memory are not only limited but also fixed. When relational databases powered internal-facing applications with limited number of users, there was a need of efficient predictable workloads to handle. However, applications with thousands and millions of web and mobile customers can overwhelm relational databases with increasing and volatile workloads. The only way to maintain performance of the number of growing customers and products, especially during peak periods, there is a need to add more resources. With relational databases, this is difficult and expensive because, in this we have to replace the RDBMS server with a larger and expensive server. The integration of RDBMS and NoSQL can make the system more advantageous over traditional systems [5]. Also, if a relational database is running on a single server fails, the data becomes unavailable. Hence, NoSQL is the better solution for the shortcomings of RDBMS. NoSQL products with distributed architectures are designed to elastically scale on low-cost commodity. In NoSQL databases, data are replicated to multiple nodes. If one fails, the data are available on the other working and existing nodes. Hence, in case of the system failure data can be easily recovered. Due to its flexibility, distributed design and low cost, NoSQL has become a de-facto standard for modern, interactive e-commerce applications.

II. MONGO DB: AN OVERVIEW

It is an open-source NoSQL document-oriented database. It is scalable in nature and written in C++ libraries. This instance made the NosQL databases to follow high scale systems with faster access[8]. The main reason for migrating from relational databases is to make scaling easier. The concept of "*rows and columns*" is replaced with a model called "*document*". By using document properties and array functionalities, it is possible to represent a hierarchical relationship in terms of a single record. This made the performance evaluation of MongoDB scalable in different operational occurrences [9]. MongoDB as a NoSQL database and RDBMSs are different in terms of their implementation and operational concepts. TABLE I shows the terminology differences between RDBMSs and MongoDB.

TABLE I. FUNCTIONAL DIFFERENCE BETWEEN RDBMS AND MONGODB

Relational DBMS	**MongoDB**
Table	Collection
Column, Rows	JSON document
Index	Index
Join	Embedding & Linking across documents
Partition	Shard
Partition key	Shard key
Fixed schema	Schema less

A. Features of MongoDB

The MongoDB supports BSON (binary encoded JSON like objects) as formal data structures to store different data types. It also supports a complex query language with a special feature called *GridFS;* which is used to store and retrieve files that are more than 16MB such as audio, video etc. Because of these added features, MongoDB projects have been considered to process larger data over RDBMS.

B. Data Design: MongoDB

MongoDB database provides different collection of objectives. One of the collection of MongoDB has no pre-defined schema like RDBMS, and it stores data as documents. BSON (Binary encoding of JSON) are used to store documents. As there is no specific schema design, we can insert documents of the form of JSON (Java Script Object Notation). **Fig. 1** illustrates the existence of JSON document. It contains data structures in various complex modes such as lists.

```
{
  "first name" : "Steve" ,
   "last name" : "smith" ,
   "age" : 35 ,
   "address" : {
          " streetAddress " : " 211 street ",
          " city " : " New York ",
          " postalCode " : 10029
     }
}
```

Fig. 1. Exixtence of JSON document.

C. API of MongoDB

MongoDB has its formal query language which is made containing the fields that the desired document should match. **Fig. 2** shows different commands and their related queries.

Insert Command-> inserts the information in the collection
>db.users.insert ({ name: "Mark" , user_id: "A-123", age: 45, status: "H" })
Display Command-> displays all the information in the collection in JSON format >db.users.find().forEach(printjson)
Or
>db.users.find().pretty()
Drop Command-> drops a collection form the database
>db.users.drop()
Select Command-> selects the fields that are to be retrieve form the collection and displays them
>db.users.find ({ status : "H" , age : 45 })
Delete Command-> delete certain information from the collection
>db.users.remove ({ status : "H" })

Fig. 2. Commands of MongoDB with related queries

D. MongoDB: Explicit Architecture

MongoDB architecture is built with three components; *Configuration servers*, *Shared nodes* and *Routing services or mongos*[13]. The architectural instance is shown in **Fig. 3**. **Shard Nodes** consist of one or more nodes which are responsible for storing the data. Corresponding read and write queries are routed to the appropriate node that holds the data. A replicated node comprises of one or more servers and acts as a primary replica to give the backup for failure instances

[3]. **Configuration servers** store metadata and routing related information of MongoDB which is acting as a current primary replica [3]. **Routing services** are also called as mongos and perform tasks related to different clients. Routing server also handles different queries issued by various clients. The result of a query will be send to the related node before it sends back to the client [10]. MongoDB uses memory-mapped files to increase the performance[15]. It also supports auto-sharding instance which increases the scaling behavior across multiple nodes.

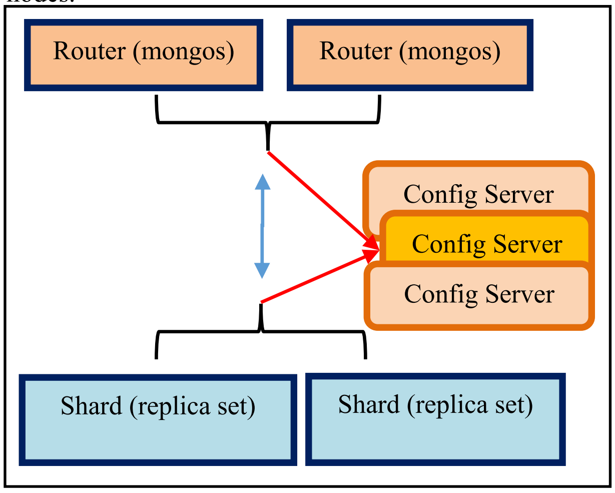

Fig. 3. Architecture of MongoDB (source [13])

III. INCLUSION AND WORKING OF E-COMMERCE WORKFLOW

The open source e-commerce market business has already got the increased popularity, platforms like osCommerce, Magento, Zen Cart, Prestashop, Spree helps in creating small to large business scale business opportunities to run a successful online store[1]. With a wide variety of product offering leading to millions of products, tons of rich images and millions of geographically distributed shoppers, E-commerce platforms and sites like EBay, Staples, Amazon etc. has to face a big challenge of maintaining consistent availability, performance, scalability and throughput across all channels. As RDBMS was not able handle the requirements of the e-commerce business NoSQL was the only solution.

- E-commerce sites are used massively during seasonal swings. NoSQL is the best suited for this type of usage pattern with its dynamic scalability.
- E-commerce functionalities like shopping carts and session information are considered as temporary data. NoSQL databases are best fit to store this data for better performance and scalability.
- Completed orders are better stored in document-oriented NoSQL database like MongoDB.
- An architecture to handle the growing user base effectively is certainly a major consideration for any e-commerce store.
- Scalable architecture is very essential for any e-commerce business: keeping the multichannel presence, increasing content and data, rules and analytics, social media content etc [6].

As the e-commerce business required rapid development, scalability, flexibility, 24x7 availability and has to deal with massive data volume across the e-commerce sites, document-oriented database like MongoDB is an excellent option to deal with all the requirements of e-commerce business. Hence, moving from Relational databases to NoSQL DBMS like MongoDB gives many benefits to e-commerce business.

- Big Data, Big medical images, etc. are easily stored with easy access
- High availability of storage and retrieval
- Enabled very rapid development and growth and cost reduction
- Performance improvement
- Simplification of code for external usage and need for external caching system [14]

Hence, e-commerce business systems are a good starting for data modeling with document-oriented database like MongoDB. As there is no pre-defined schema documents can be inserted in any form of JSON document. MongoDB has its own credentials in terms of its rich query operators and update operators which lets us to access the documents easily and to perform atomic updates on single fields of a document [4].

IV. SCHEMA DESIGN FOR E-COMMERCE USING MONGODB

A. The Product Catalog

To insert the document in the collection of products for e-commerce, a related insert command is used. The insertion of document is represented in **Fig. 4**. This model stores the details of manufacturing and shipping documents in the large product management. The developed product model has a unique identifier to identify the product catalog along with title, its description, present stock quantity, and pricing information of the concerned item.

The available products have different division of categories with their specifications. In the available scenario of Sony Xperia Z, it’s a 4G featured phone with FM receiver facility. Hence, we have to add the categories to the existing document, with the following update() operation [4].

```
db.products.update(
{sku: "123444GB2"},
{$set: { categories: ['mobile/4G', 'mobile/fm'] }}
);
```

```
Mongo_space
use e-commerce
db.products.insert({
        id : “123444GB2”,
        title : “ Sony Xperia Z Mobi_phone ”,
        description: "No.1 android phone in the current market",
        manufac_info: { model_number: “S00Z-X”,
        rel_date: new IS-ODate( “2013-04-17T09:17:16”) },
        Ship_detail: { weight: 250, width: 10, height: 15 },
        quant: 199
        price: { cost: 30000 }
    })
```

Fig. 4. Inserting the document in the product collection

After this, we fetch the item information in the category that begins with mobile-phone/fm_facility by using the following *find* operation [4]:

```
db.products.find(
{categories: /^mobile-phone\/fm_facility/}
);
```

B. The Cart

Users can save items from the inventory and keep them in their cart until they check out and pay for the items. The application must ensure that there are not more items in the cart than there are in stock at any point of time and if the users deletes items from the cart, the application updates the information to the cart inventory without losing information of any objects. Following is the *insert()* operation to create the cart [4]:

```
db.carts.insert({
  _id: "the_product_id",
  status:'act', quant: 3,
  total_avl: 4000, products_catalog: []});
```

The array of related product shows the list of products. If the inventory shows 199 items before purchase, after completion of the above operation, the inventory updates with 196 item availability. The following operation adds a product to the customer’s cart [4]:

```
db.carts.update_invent(
  {_id: "the_product_id",
    status:'act' },
 { $set1: { modi_on: IS-ODate() },
  $push: {
   products: {sku_op: "123444GB2",
    quant: 1,
    tit: "sony xperia z mobile_phone",
    price:30000 }  }
});
```

After this, we check the possibility to add a product to the customer's cart [4].

```
db.products.update_invent(
{ sku_op: "123444GB2", quant: {$gte-qn: 1} },
  { $inc: {quant: -1},
    $push: {
      in_carts: {
        quant:1,
      id: "prod_id", timestamp: new IS-ODate() } } });
```

This operation can be completed only if there are sufficient inventories available. This application detects the success and failure rate of performed operations. This functionality can be performed by calling *getLastError* function to get the update [4].

```
if(!db.runCommand({getLastError:1}).updatedExisting)
  {  db.carts-avl.updated(
    { _id: "prod_id" },
   { $pull-qn: {prod: {sku-op:"123444GB2"}} } );
```

The required operation to add the product to the user’s cart is “*roll back*” if updated Existing is not available in the resulting inventory because the operation is failed. This method ensures the application to check the available products in the inventory.

C. Check Out

Figure After clicking on the “confirm” option of the application, a new application is created with new “order” document that reflects the entire order. **Fig. 5** shows the shipment details of the products ordered by a particular customer.

```
db.orders.insert({
  created_on: new IS-ODate( “2013-05-14T09:17:16”),
    shipping: {
        customer: "John Peterson",
        address: "228 Park Street",
        city: "N-york",
        country: "US",
         tracking-onfo: {
         company-det: "kps",
          tracking_num: "32122X21SD",
          stat: "on_way",
estimated_delivery:
                new IS-ODate("2013-0517T09:14:16")},},
          payment-id: {
        method-pay: "cash on delivery",
        transac_id: "231444312XXXTD" }
        product-avl: {
         {quant: 3, sku_op:"123444GB2", tit: "Sony xperia Z ",
cost:30000, currency:"USD$"}
        }
      })
```

Fig. 5. Details of shipment window

In RDBMS, information is stored in the form of records as set of relatons/tables. By using MongoDB as NoSQL database, we create a self-contained document to update the inventory information based on purchase activities. This scenario makes the user to easily understand the application. The up-to-date information is shown by inserting the following operation.

```
db.carts.update-info(
  { _id: "prod_id" },
  { $set-flag: {stat:"executed"}
 }
   );
```

The following segment of code executes the operation and removes the related identifier form the records [4].

```
db.products.update-info(
{ "in_carts-stat.id": "prod_id" },
  {  $pull: {in_carts:
    { id: "the_product_id"}}
  },0-false, 1-true);
```

V. CONCLUSIONS

In this paper, we outline the design instances of e-commerce with NoSQL databases such as MongoDB. If we are running into scalability, performance, and availability challenges with the traditional RDBMS and looking for faster development and innovation, NoSQL technology is the best choice to adopt. Hundreds of leading retail and e-commerce companies have already made the move, deploying NoSQL to support a growing number of use cases from customer profile management personalization, to product catalogs, shopping carts and many more. As the document-oriented NoSQL database like MongoDB have many advantages over RDBMS, it may contain large amount of data with a suitable flexible data model. Also, these kind of data models such as schema design models are useful to further develop new resource adopted systems.

ACKNOWLEDGMENT

The authors wish to express their gratitude and thanks to the Department of Computer Science & Engineering, **Indian Institute of Technology (ISM)**, Dhanbad, India for providing their support.

REFERENCES

[1] How MongoDB makes custom e-commerce easy? http://blog.mongodb.org/post/31729833608/how-mongodb-makes-custom-e-commerce-easy.

[2] Yoon, J., Jeong, D., Kang, C. H., & Lee, S. (2016). Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study. *Digital Investigation*, *17*, 53-65.

[3] Khan, S., & Mane, V. (2013). SQL support over MongoDB using metadata. *International Journal of Scientific and Research Publications*, *3*(10), 1-5.

[4] Data Modeling: Sample E-Commerce System with MongoDB https://www.infoq.com/articles/data-model-mongodb

[5] Khan, S., & Mane, V. (2013). SQL support over MongoDB using metadata. *International Journal of Scientific and Research Publications*, *3*(10), 1-5.

[6] Scalable ecommerce e-commerce with NoSQL database http://echidnainc.com/scalable-ecommerce-with-nosql-databases/

[7] Lungu, I., & Tudorica, B. G. (2013). The development of a benchmark tool for nosql databases. *Database Systems Journal BOARD*, *13*.

[8] Tauro, C. J., Aravindh, S., & Shreeharsha, A. B. (2012). Comparative study of the new generation, agile, scalable, high performance NOSQL databases. *International Journal of Computer Applications*, *48*(20), 1-4.

[9] Aboutorabi, S. H., Rezapour, M., Moradi, M., & Ghadiri, N. (2015, August). Performance evaluation of SQL and MongoDB databases for big e-commerce data. In *Computer Science and Software Engineering (CSSE), 2015 International Symposium on* (pp. 1-7). IEEE.

[10] Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: reviewing some next-generation non-relational database's. *International Journal of Advanced Engineering Science and Technologies*, *11*(1), 15-30.

[11] Kristina Chodorow, Michael Dirolf, "Orielly MongoDB, The Definitive Guide", September-2010.

[12] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. (2011, November). Security issues in nosql databases. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 541-547). IEEE.

[13] MongoDB https://docs.mongodb.com/manual/sharding/

[14] MongoDB e-commerce and transactions http://www.slideshare.net/spf13/mongodb-ecommerce-and-transactions-10524960

[15] Wang, Y., Chen, H., Wang, B., Xu, J. M., & Lei, H. (2010, November). A Scalable Queuing Service Based on an In-Memory Data Grid. In *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on* (pp. 236-243). IEEE.