Moving to New Generation Databases (MongoDB Redis)

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Name: Yogesh Punia
(Roll No. 801232031)

Under the supervision of:
Dr. Rinkle Rani
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2014

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Moving to New Generation Databases (MongoDB Redis)*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. RinkleRani* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(YogeshPunia)

Roll No. 801232031

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. RinkleRani)

Assistant Professor

Computer science and Engineering Department

Thapar University, patiala

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. K. Mohapatra)

Dean (Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGMENT

No volume of words is enough to express my gratitude towards my guide, ***Dr. Rinkle Rani****, Assistant Professor, Computer Science and Engineering Department, Thapar University, who have been very concerned and have supervised the work presented in this thesis report. He has helped me to explore this vast field in an organized manner and provided me with all the ideas on how to work towards a research oriented venture.*

I am also thankful to ***Dr. Deepak Garag****, Head of Department, CSED for the motivation and inspiration that triggered me for the thesis work.*
Most importantly, I would like to thank my ***parents****,* ***sister****,* ***friends*** *and the* ***Almighty*** *for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.*

Yogesh Punia
(801232031)

Abstract

Demand for speed, manageability, distributed, open-source, agile development with schema-less databases, easier horizontal scalability leads to fixing of broken parts in MySQL or Oracle by introducing next generation databases also termed as NoSQL. These are modern web-scale databases which are introduced to handle emerging applications such as social network analysis, bioinformatics network analysis and semantic Web analysis where a wide variety of data is to be processed which needs continuous witness with a quick increase. A critical challenge these days is to have an effective management and analysis of data at a large-scale. In this thesis I have tried to make an effort to use a combination of NoSQL databases to replace traditional database like Oracle applied to information management system, comparing the traditional database system with combination of MongoDB and Redis and presents the performance comparison of these two schema. As mostly all the old legacy systems have backend as MySQL, so there is need for shifting from SQL to another database through which queries can be performed as like SQL. A conversion effort have been made in this thesis.
The change of need of databases have driven this work, to replace old existing system to with new databases so the database have been converted by fetching the information from its schema step by step and then finally successfully converting the databases that do not require real time transaction like banking system.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditional database has been serving from early 70's whether it is about storing large amount of data or fetching data from multiple tables through complex joins. However it started lacking with the rise of web 2.0 as the volume of data stored about users has piled up to layers as for instance in social networking sites like facebook. Google and Amazon found it very challenging when system crawls while millions of users lookup against tables having millions of rows of data. Much effective management and analysis of data at a large scale poses critical challenge. These days, big data has attracted a lot of attention from industry and academia [1]. This whole data comes from all around like sensors which used to gather climatic information, posts sent to social media sites, digital pictures with videos, transaction records, and GPS signals of mobile phone. These type of requirements cannot be fulfilled by existing databases like MySql and oracle so to achieve these we use combination and MongoDB, as all old systems are based on SQL , so effort has been done to convert this data into MongoDB ,that is shifting SQL to MongoDB. This data is what big data is. The paradigm of big data is shown below in figure 1.

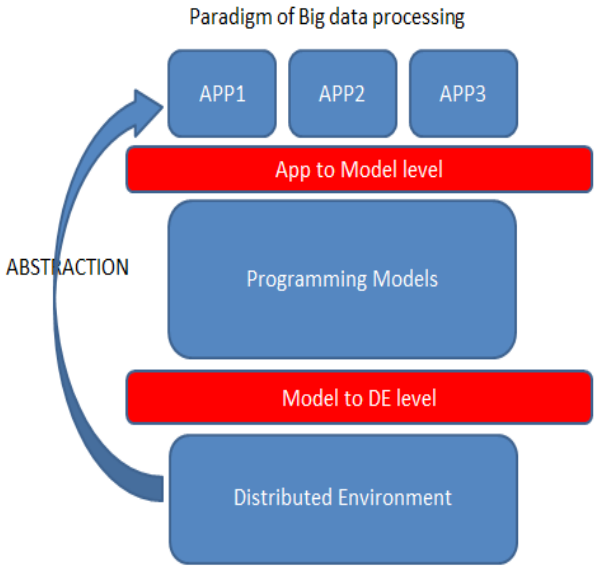

Figure 1: Paradigm of Big Data Processing[1]

1.1 NO-SQL

NoSQL is a non-relational database system, that differs from old relational database management systems in some many ways. It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users). These type of data storing may not require fixed schema, avoid join operations and typically scale horizontally. Relational database face tough competition as it does not suit well to complex programming structures consisting of complex data types and data of hierarchical nature.

Their first and foremost advantage is that, not in a way as relational databases, they handle unstructured data such as e-mails, word-processing files, social media and multimedia efficiently [2]. Complex objects consisting objects and further lists inside them cannot be mapped directly to a single row in a single table. In coding point of view it does not map well as it is difficult to handle the syntax of SQL along with the client side code for accessing SQL database. In order to overcome all these problems a range of new databases has been introduced(currently 150) having categories like Wide Column Store/Column Families, Document store, Key Value/Tuple Store, Graph Databases, Multimodal Databases, Object Databases, Grid & Cloud Database Solutions, XML Databases, Multidimensional Databases, Multi value Databases, Event sourcing and other. Dimensions of Big Data are as following:

1.1.1 Volume

The bulk amount of data produced such as 12 terabytes of data tweeted by people How much ever growing data is there 12 terabytes of Tweets created every day into better product sentiment analysis Conversion of 350 billion annual meter readings to improved predict power consumption.

1.1.2 Velocity: How fast data can be processed

The time-sensitive processes like catching fraud, big data must be used as it streams into enterprise in order to maximize its value. For example 500 million daily call details are observed in real-time in order to predict customer churn faster. All these requirements can be met by using back end that work at high speed.

1.1.3 Variety: Various types of data

Big data can be any type of data - structured and unstructured data such as text, audio, vedio, sensor data, click streams, log file. Monitor thousands of live video feeds from surveillance cameras in order to target points of interest.

1.1.4 Veracity: How accurate is data in predicting business values

Data must be verified based on both its accuracy and context. An innovative business can demand analyzing massive amounts of data in real time to assess the value of that customer quickly. It is very much necessary to identify both, the right amount and type of data that should be analyzed in real time to impact business outcomes.

1.2 Types of Big data

There are many databases which can be categorized under big data, roughly they can be summed up as key value store, column-oriented store, document store and graph databases. All the databases fall under these four categories and as the name indicates key value stores the key value with the complexity of O(1) of returning associated value, document store stores the documents, column stores the database in column and graph databases store the data in the form of nodes and edges. All the databases store data which is in bulk and needs to be accessed fast at real time with variety. As we have to store documents and tried to access the data fastly, for this purpose we use mainly two databases which are MongoDB and redis.

1.2.1 MONGODB

A Non-Relational Database MongoDB (from "hu**mongo**us") is an open source document database and the leading NoSQL database in C++ [6]. NoSQL refers to non-relational databases promises to handle large volumes of structured and unstructured data. Unlike relational databases they NoSQL databases do not require schemas to be defined beforehand. These fits perfectly fine with the agile technologies as each time a new feature is developed, the schema of the database is needed to be changed. The data-structure of non-relational database is not fixed. This allows the insertion of data without a predefined schema.

This ultimately proves helpful in making significant changes in applications in real

time, with no interruption in service, leading to faster development, reliable code integration and lessens the amount of time by database administrators. NoSQL also referred as “Not only SQL” to emphasize that these also support SQL like queries. MongoDB is a document-oriented database released in 2009. It holds a set of collections; a collection holds a set of documents. A document further is a set of key-value pairs. These documents have a dynamic schema which means that documents in the same collection do not need to have the same set of fields or structure. The key feature for which MongoDB is used is its flexibility which is understood as that the data is stored in JSON documents as shown in fig 1.1:

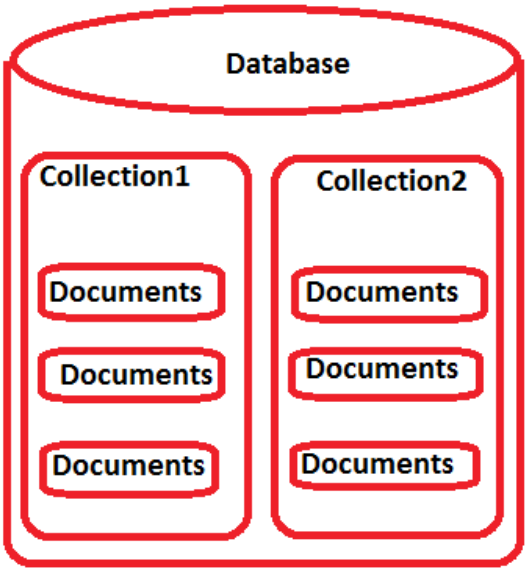

Fig 1.1 MongoDB database

1.2.1.1 Key feature of MongoDB: JSON

JSON is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML. These provides a rich data model that flawlessly maps to native programming language, and the dynamic schema makes it easier to unfold your data model as compared to a system with compelled schema RDBMS. MongoDB is chosen for evaluation because it can manage small data as well as large data efficiently.

1.2.2 Redis:Key-value Store Database

Redis (REmote DIctionary Server) is an open source, advanced key-value store. The key feature of Redis is that it can store more complex data types and atomic operations like appending to a string, incrementing the value in a hash, pushing into a list, getting the member with highest ranking in a sorted set, computing set union, intersection, difference can be defined against these data types. It is in-memory but persistent on disk which means data does not disappear on system restart. Redis holds the whole dataset in memory. In-memory data set here means that data may be swapped out when not in use for long or not needed in future much. It is also termed as data structure server since keys may contain strings, lists, hashes, sets, and sorted sets.

As Redis is well famous for speed and scalability so it is widely used with cloud storage. As cloud web applications demand fast request and retrieval of data. Whereas relational database could have been proved a bottleneck for this requirement of speed. RDBMS had fixed schema and if they had to be used in future then there may be a lot of space wastage with lots of unused columns. Whereas KVS had been really simple while designing where values may be stored anytime using a unique key. Those values could be used later using that key, also known as hash addressing.

RDBMS have various serious advantages over KVS: such as they are well tested, also have good management tools, apart from this have programming patterns available. The Key-values on the other hand are fairly newly introduced, each of them is having different approaches and these excel only in a very small segment [4]. Author in [7] has discussed about differences between the traditional SQL database and newly emerging technologies coming up with the name NoSQL like MongoDB, Redis. NoSQL had earlier misunderstood as anti-SQL but actually NoSQL is that class of database management system which is different from the traditional relational databases where data is not stored using fixed table schemas. The purpose of newly designed databases is to serve as database system for huge web-scale applications where they have proved to be more efficient.

Redis is used when data is not meant to be lost and is small enough to fit in memory. Data is to be stored and cleaned up carefully. It may begin to suffer performance problems once it exceeds the memory limit. It operates as a single process and single

threaded. Thus, a single Redis instance cannot perform parallel execution of tasks. Also this database cannot be used where durability of data is required. Data may get lost in case of any incidents, so its usage is limited.

Chapter 2

LITERATURE SURVEY

The RDBMS are not so suitable to cater to some of the critical requirement of these new generation applications such as handling large set of unstructured data or providing elastic scalability. This results in emergence of a new set of document centric or resource centric databases which are non-relational in nature [30].

Author in [6] has discussed about differences between the traditional SQL database and newly emerging technologies coming up with the name NoSQL like MongoDB, Redis. NoSQL had earlier misunderstood as anti-SQL but actually NoSQL is that class of database management system which is different from the traditional relational databases where data is not stored using fixed table schemas.The purpose of newly designed databases is to serve as database system for huge web-scale applications where they have proved to be more efficient.

MongoDB has been chosen as a NoSQL database as it is relatively new in database market. And the areas of its use have grown widely up to MTV networks, Disney Interactive Media Group, The New York Times, Git Hub, and many more.

The primary and foremost difference between these two categories is that the some of the database engines which do not support SQL in order to achieve high performance or reliability features usually provides a query language which support a subset of what SQL can do but these engines do not support the existing features of SQL like JOIN, Transaction, Limit, and non-indexed WHERE.

MongoDB was released in 2009 which stored data in dynamic schemas in JSON like documents. MongoDB revolved around four properties like flexibility, power, speed, ease of use. It supported replicated servers, indexing and sharding.

It accepted larger data 16MB as comapred to Oracle which had been supporting 4 KB.[29] Both were crossplatform database management system but MongoDB was a freely available product in comaprision to Oracle which was a licenced product. In case of distribution, both are horizontal scalable and had support for data replication.

The integrity model of Oracle supported in database is ACID where as in MongoDB, it was BASE. It provided conditional atomicity, durability and consistency. But Oracle database also supported isolation, transactions, referential integrity and revision control.

In[4] the author had explained the key-value implementation of database. He made an approach to persist the objects under race and failure conditions. As Redis is well famous for speed and scalability so it is widely used with cloud storage. As cloudweb applications demand fast request and retrieval of data. Whereas relational database could have been proved a bottleneck for this requirement of speed.

RDBMS had fixed schema and if they had to be used in future then there may be a lot of space wastage with lots of unused columns. Whereas KVS had been really simple while designing where values may be stored anytime using a unique key. Those values could be used lator using that key. Also known as hash addressing.

The author has explained about the object persistence in KVS implementation. The object attribute had been stored with following key-value pair –

ClassName :Identifier:AttributeName ->Value

Smaller scale KVS implementation include CouchDB, MongoDB, and Redis.

In[1] the author has explained the need of emerging Web2.0 post technologies for a more flexible , scalable and fast database. The era of storing data in form of tables is going to change to the storage in the form of documents in collection. Collections can be created dynamically. The approach to various schema free database is explained here.

2.1 NO-SQL

NoSQL is not only SQL that is it is a non-relational database management system, that differs from traditional relational database management systems in some significant ways. It is designed for distributed data stores where there is huge storage of data amount needed (for example social networking sites like Google, Facebook, twitter etc. which collects terabytes of data every day for their users). These type of data storing may not be satisfied with fixed schema, so join operations are avoided

and schema less system are used which follows BASE properties and data generally horizontally scalable.

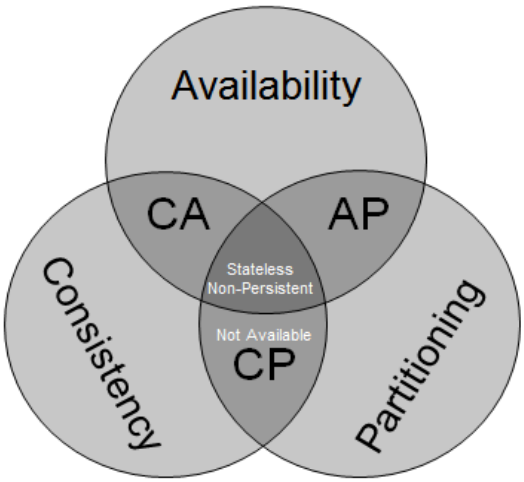

Figure 2.1 CAP Theorem[2]

NOSQL databases are based on CAP theorem, it states that there are three basic requirements which exist in a special relation when designing applications for a distributed architecture.Figure 2.1 shows the following properties:

- **Consistency** – The data should remain consistent, that is data should remain same for all after an execution of operation or transaction, for example after write or update operation all clients should see the same data.
- **Availability** –Data should be available when needed, there should be no downtime. It means that there should be sure service of availability.
- **Partition Tolerance** – The service should not be down even if there is no reliable communication between the servers, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

All three requirements cannot be fulfilled theoretically. CAP provides that any distributed system should provide any two basic requirements out of three requirements. Therefore all the current NoSQL database follow the different combinations of the C, A, P from the CAP theorem. The different combination of CAP theorem i.e. CA, CP, AP are as following:

CA – Data will be consistent and available but cannot assure the partition failure, for example single site cluster, all nodes are always in contact. When a partition occurs, the system blocks.

CP- System would be consistent and can survive bad communication network but some data may not be accessible, but the rest is still consistent/accurate.

AP - System is still available under partitioning, but some of the data returned may be inaccurate.

Data model of SQL databases is based on relationship, that is in the form of table that specifically support associated class operations and ACID transactions, but in the NSQL database fields, the mainstream data model which are in next section.

2.1.1 Mainstream NoSQL database and Data Model

Data model of SQL databases is based on relationship, that is in the form of table that specifically support associated class operations and ACID transactions, but in the NSQL database fields, the mainstream data model which are the following:

2.1.1.1 Key-value databases

In this data model a key-value pair is stored, a key is associated with its value, means a value corresponds to a key. The structure of this data model is very simple, the query performance is better than relational database, it supports mass storage and high concurrency. It also support primary key and query modify operations are performed on data. Key-value database in the market are as following:

- **Redis**-is the a Key-value memory database, when Redis run, data were entire load into memory, so all the operations were run in memory, then periodically save the data asynchronously to the hard disk. The characteristics of pure memory operation makes it very good performance, it can handle more than 100,000 read or write operation persecond. It support List and Set and various related operations. The maximum of value limit to 1GB. The main drawback is that capacity of the database is limited by physical memory, so Redis cannot be used as big data storage, and scalability is poor. Therefore, Redis is suitable for providing high performance computing to small amount of data.
- **Tokyo Cabinet- Tokyo Tyrant**-TC and TT were developed by Mikio Hirabayashi in Japan mainly for Japan's largest SNS site mixi.jp, it has been a very mature project yet. TC is a high performance storage engine, while the TT provide multi-threaded high-concurrency servers, it can handle 4-5 million times read and write operations per second. 365 While ensure the high performance of read and write

concurrent, TC uses a reliable data persistence mechanism. TC supports data structure not only Keyvalue, but also hashtable which is similar with relational database, in addition, supports simple operations like conditional query, paging and sorting. TC's main drawback is that when the amount of data grows to the billion levels, concurrent write performance will drop dramatically. Flare was developed by the second largest SNS website green.jp in Japan, Flare is stronger than TC because of the scalability has been extends. Flare added a node server before data servers in order to manage data servers at back, so user can dynamically add or remove data servers, and it also supports failover. However, Flare supports only the memcached protocol, when using Flare, you cannot use the table data structure of TC, but only can use TC's Key-value data structure.

2.1.1.2 Column-oriented database

Column-oriented database has data model as table, but doesn't support costly join operations. Data in column-oriented is of same type, having similar characteristics and good compression ratio and is stored by column where each column of data is index of the database, to reduce the input output of the system on those columns are accessed which are required by query. This data model can be used on aggregation and data warehouse. Although column-oriented database has not subverted the traditional stored by row, but in architecture with data compression, shared-nothing, massively parallel processing, column oriented database can maintain high-performance of data analysis and business intelligence processing. Column oriented databases are HBase, Yale University HadoopDB, Facebook's Cassandra, Hypertable, Google's Bigtable[9], and Yahoo's PNUTS and so on.

- **Cassandra** -is an opensource database of facebook. The schema is very flexible and does not require to design database schema at first, and add or delete field is very convenient. It support range queries, that is it can range queries for Key. It is highly scalable a single point of failure does not affect the whole cluster, and it support linear expansion. Cassandra system is a distributed database system which was composed of lots of database nodes, a write operation will be replicated to other nodes, and rea request will be routed to a certain node. For a Cassandra cluster, only to add node can achieve the goal of scalability. In addition, Cassandra also supports rich data structure and powerful query language.

2.1.1.3 Document database

The document store database are similar to key value store. The documents are stored in the JSON format.The documents are associated with nested key, means it's a further step to key/ value store with new concept of attaching documents with keys in JSON format. Documents databases like MongoDB have rich query syntax and queries are efficiently performed. Documents can be scattered over the multiple server as they are independent unit but still their locality is preserved. Writing application logic is easy as these are schema less or have simple modeling than SQL. Document database can store unstructured data , expensive join operations are not required in these databases as the document contains , whatever is required by application.

Document database is not concerned about high performance read and write concurrent, but rather to ensure that big data storage and good query performance. Typical document database are MongoDB, CouchDB.

- **MongoDB** is a database between relational databases and non-relational database, its features are: it is non-relational database, which features the richest and most like the relational database, support complex data types: MongoDB support bjson data structures to store complex data types powerful query language, it allows most of functions like query in single-table of relational databases, and also support index. High-speed access to mass data: when the data exceeds 50GB, MongoDB access speed is 10 times than MySQL. Because of these characteristics of MongoDB, many projects with increasing data are considering using MongoDB instead of relational database.
- **CouchDB -** is a flexible, fault-tolerant database, which supports data formats such as ISON and AtomPub, it provides REST-style API. To ensure data consistency, CouchDB comply with ACID properties. In addition, CouchDB provides a P2P-based distributed database solution that supports bidirectional replication. However, it also has some limitations, such as only providing an interface based on HTTP REST, concurrent read and write performance is not ideal and so on.

2.1.1.4 Graph Databases

These databases are based on graph theory that uses nodes, edges and parameters to store data in graph structure. Every node in a graph database has direct link to its

adjacent node and no indexes are required, they can store any type of graph except for few specialized graph databases such as network databases.

Graph databases are flexible and are scalable i.e. can be used over multiple machines, it doesn't provide high query language like SQL. NoSQL databases do not provide a high-level declarative query language like SQL to avoid overtime in processing.

This kind of database is designed for data whose relations are well represented as a graph (elements interconnected with an undetermined number of relations between them). The kind of data could be social relations, public transport links, road maps or network topologies, for example.

AllegroGraph is RDF GraphStore, DEX/Sparksee is high-performance graph database, FlockDB ,IBM DB2 a RDF GraphStore added in DB2 10, InfiniteGraph a high-performance, scalable, distributed graph database Neo4j OWLIM RDF graph store with reasoning, OrientDB Sones GraphDB Sqrrl Enterprise Distributed, real-time graph database featuring cell-level security OpenLink Virtuoso middleware and database engine hybrid.

All these mentioned above are mainstream databases, many companies are currently using them and others are shifting from old legacy system to new ones.

2.2 MongoDB

MongoDB is a powerful, flexible, and scalable data store. It combines the ability to scale out with many of the most useful features of relational databases such as secondary indexes, range queries, and sorting. MongoDB is also incredibly featureful it hastons of useful features such as built-in support for MapReduce-style aggregation and geospatial indexes [31].

MongoDB (from "humongous") is a schema-free, document-oriented database written in the C++ programming language. The database is document-oriented in that it manages collections of schema-less JSON-like documents. This allows data to be nested in complex hierarchies and still be query-able and index-able. Following are the main MongoDB features:

2.2.1 Data Model

A MongoDB database holds a set of collections. A collection is an equivalent term for a table, but unlike a table, a collection has no pre-defined schema. A collection holds a set of documents. A document is a set of fields. It can be thought of as a row in a collection. A document can contain complex structures such as lists, or even documents. Every document has an id.

2.2.2 API

MongoDB has its own query language named Mongo Query Language. To retrieve certain documents from a db collection, a query document is created containing the fields that the desired documents should match. For example:{name: {first: 'John', last: 'Doe'}}

MongoDB uses a RESTful API. REST (Representational State Transfer)is an architecture style for designing networked applications. It relies on a stateless, client-server, cacheable communications protocol (e.g., the HTTP protocol). RESTful applications use HTTP requests to post, read data and delete data.

2.2.3 Architecture

A MongoDB cluster is different from a Cassandra cluster. The most obvious difference is the lack of symmetry: not every node in a MongoDB cluster is the same. A MongoDB cluster is built of one or more shards, where each shard holds a portion of the total data (managed automatically). Reads and writes actions are automatically routed to the appropriate shard(s). Each shard is backed by a replica set - which just holds the data for that shard. A replica set is one or more servers, each holding copies of the same data. At any given time one server is primary and the rest are secondary servers. If the primary server goes down one of the secondary servers takes over automatically as primary. All writes and consistent reads go to the primary server, and all eventually consistent reads are distributed amongst all the secondary servers. The cluster contains a group of servers called configuration servers. Each one holds a copy of the meta-data indicating which data lives on which shard. Another group of servers is routers, each one acts as a server for one or more clients. Clients issue queries/updates to a router and the router routes them to the appropriate shard while

consulting the configuration servers. Fig 2.2 copied from the official website of MongoDB 1, depicts the MongoDB architecture.

MongoDB supports two types of replication functionality: Master-Slave replication and Replica-set replication. In both types of replication, all write operations are performed against a single server (Master or Primary). Replica-Sets provide better flexibility, allowing automatic primary promotion (if enough of the secondaryservers are available), automatic fail-over and better support for rolling upgrades. Both techniques provide data-redundancy and read-scaling (where data can be read from any of the servers in the cluster), however, in Master-Slave configuration, if a slave lags too far behind from the master, then the administrator needs to manually fix this, usually by rebuilding the slave instance.

The last piece of the MongoDB puzzle is its ability to automatically shard the data between multiple hosts. This effectively allows Mongo to scale horizontally to thousands of servers. When sharding is combined with replica-sets, the end-result is a highly scalable, redundant cluster, with no single point of failure.

2.2.4 Sharding

MongoDB supports an automated sharding/ partitioning architecture, enabling horizontal scaling across multiple nodes. For applications that outgrow the resources of a single database server, MongoDB can convert to a sharded cluster, automatically managing fail over and balancing of nodes, with few or no changes to the original application code. Sharding is the partitioningof data among multiple machines in an order-preserving manner. This makes it easier to support range queries and indexes which are inherent in MongoDB.

2.3 Redis

Redis can be described as an in-memory persistent key-value store. Redis does hold all the data in memory (more on this in a bit), and it does write that out to disk for persistence, but it's much more than a simple key-value store. The reality is that Redis exposes five different data structures, only one of which is a typical key-value structure. Example of key value pair:

set users:boxer"{name: boxer, planet: earth, likes: [space]}"

This is the basic anatomy of a Redis command., in this case set is command. Next we have its parameters. The set command takes two parameters: the key we are setting and the value we are setting it to.

get users:boxer

and this will return the value associated with key boxer.

2.3.1 Memory and Persistence

Redis is an in-memory persistent store. With respect to persistence, by default, Redis snapshots the database to disk based on how many keys have changed. You configure it so that if X number of keys change, then save the database every Y seconds. By default, Redis will save the database every 60 seconds if 1000 or more keys have changed all the way to 15 minutes if 9 or less keys has changed. Alternatively (or in addition to snapshotting), Redis can run in append mode. Any time a key changes, an append-only file is updated on disk. In some cases it's acceptable to lose 60 seconds worth of data, in exchange for performance, should there be some hardware or software failure. In some cases such a loss is not acceptable. Redis: RAM is still the most expensive part of server hardware, being IO-bound for being memory bound. Redis did add support for virtual memory.

2.3.2 Data Structures supported by Redis

2.3.2.1 Strings

Strings are the most basic data structures available in Redis.

set users:leto "{name: boxer planet: earth, likes: [space]}"

Additionally, Redis lets you do some common operations. For example strlen <key> can be used to get the length of a key's value; getrange <key> <start> <end> returns the specified range of a value; append <key> <value> appends the value to the existing value (or creates it if it doesn't exist already).

2.3.2.2 Hashes

Hashes are like strings. The important difference is that they provide an extra level of indirection: a field. Therefore, the hash equivalents of set and get are same as string.

We can also set multiple fields at once, get multiple fields at once, get all fields and values, list all the fields or delete. The benefit using the hash be the ability to pull and update/delete specific pieces of data, without having to get or write the entire value. Looking at hashes from the perspective of a well-defined object, such as a user, is key to understanding how they work. And it's true that, for performance reasons, more granular control might be useful.

2.3.2.3 Lists

Lists let you store and manipulate an array of values for a given key. You can add values to the list, get the first or last value and manipulate values at a given index. Lists maintain their order and have efficient index-based operations. This is a common pattern. ltrim is an O(N) operation, where N is the number of values we are removing. In this case, where we always trim after a single insert, it'll actually have a constant performance of O(1) (because N will always be equal to 1). This is also the first time that we are seeing a value in one key referencing a value in another.

Of course, lists aren't only good for storing references to other keys. The values can be anything. You could use lists to store logs or track the path a user is taking through a site. If you were building a game, you might use one to track queued user actions.

2.3.2.4 Sets

Sets are used to store unique values and provide a number of set-based operations, like unions. Sets aren't ordered but they provide efficient value-based operations. A friend's list is the classic example of using a set: sadd friends:boxer abc xyz

sadd friends:abc xyz

Regardless of how many friends a user has, we can efficiently tell O(1) whether userX is a friend of userY or not: sismember friends:boxer abc sismember friends:boxer vladimirn Furthermore we can see whether two or more people share the same friends: sinter friends:boxer friends:xyz and even store the result at a new key: nsinterstore friends:boxer_xyz friends:boxer friends:xyz.

2.3.2.5 Sorted Sets

The last and most powerful data structure are sorted sets. If hashes are like strings but with fields, then sorted sets are like sets but with a score. The score provides sorting and ranking capabilities. The most obvious use-case for sorted sets is a leaderboard system. In reality though, anything you want sorted by an some integer, and be able to efficiently manipulate based on that score, might be a good fit for a sorted set.

2.4 SQL

SQL is a standard language for accessing and manipulating databases.The SQL language is subdivided into clauses, which are constituent components of statements and queries. Expressions, which can produce either scalar values, or tables consisting of columns and rows of data predicates, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow. Queries, which retrieve the data based on specific criteria. This is an important element of SQL. Statements, which may have a persistent effect on schemata and data, or may control transactions.

2.5 Gaps in Literature survey

Observing the past history, it is easily being concluded that MySQL is not being partially able to fulfil the need of the rapidly emerging technologies which further forces the researchers to come out with the new NOSQL database in order to store data as structural as well as non-structural form, also it should be retrieved as fast as possible. The scalability of the database has also been on stake. Challenging MySQL , various NoSQL databases have been introduced. In order to take complete benefit out of these databases, the existing databases need to be used collectively. So there is need to experiment to use two or more databases together to provide all functionalities accordingly, for example if there is an application that require highly availability of data at faster rate, we can use database that can store data and a key value store database to retrieve data. One more problem is that as all old legacy system are based on the RDBMS , so there is need for reusability of these system, and also there is important data stored in these databases which need to be ported to new generation databases that is to new databases like MongoDB, CouchDB, Cassandra, Redis etc. So we need a shifting /porting mechanism to transfer data digitally to our new databases and same queries can be

performed.

Although these new databases can provide velocity, variety and volume but these databases follow the BASE (basically available soft state eventually consistent) properties , not ACID (atomicity, consistent ,intergrated, durable) so they cannot be used in real time transactions. As many provide Queries language for the database but still they are not up to extent to completely replace SQL.

Chapter 3

PROBLEM STATEMENT

As the gaps in literature survey is being highlighting the fact that one single database is not able to fulfil the demands at all levels, like at some place a lot of data either in form of video, audio, structural or non-structural form whereas at some levels only simple data is needed to be stored but the retrieval demand is fast. Hence, the idea which is used in order to take control on data in much amount is to switch to a more faster and scalable database which is quite capable of storing data in structural as well as non-structural form. And for fast retrieval the database may store values in RAM for fast access.

Every day, 2.5 quintillion bytes of data is created. This whole data comes from all around: like sensors which used to gather climatic information, posts sent to social media sites, digital pictures with videos, transaction records, and GPS signals of mobile phone. This data is what big data is. And here the idea is to handle big data through combination of NoSQL databases. It is pretty hard to tackle this amount of data by storing them in the form of tables in rows and columns as various times many columns either are not needed to be stored or if stored may have null values. So a lot of space is being wasted with that 'null' value. So, no more a relational structure is capable of storing big data. This gives a way to search from a non-structural way of storing data.

As shown in figure MongoDB is better than RDBMS but Redis is best in performance so, the availability, performance, and easy scalability of any database is much needed in this case and search of that kind of database or a combination of such databases is what which is the need of the hour. Also with the increase in volume, variety and velocity of this era data is expecting to be stored and managed through a modified and new generation database. Thus the work is focused on finding a way out to manage the data through either a different database or any combination of newly introduced databases. Now various databases are being introduced but key to problem

is to look for a kind of database which will solve our problem in every possible way. Any combination of such databases may also be used in order to retrieve data at a much faster rate and to store it much lesser space. A better idea is to look for any open source documents which id freely available and look from a NoSQL family databases. This will help to store data in both structural as well as non-structural way.

3.1 Objectives

- To study various SQL and NO-SQL databases.
- To propose and implement a system to port data from SQL to NO-SQL MongoDB database.
- To design and application for Client Insurance system integrating MongoDB and Key-value Redis database.
- Performing insertion retrieval, insertion and update operation on the purposed system

Chapter 4

IMPLEMENTATION

This work runs MySql and MongoDB server and fetch data from MySql server with the help of java and provide input to the MongoDB and store it, the data must be stored in the MongoDB, same as it was stored in MySql database that is with same field name and values associated to it and queries can be performed on it. To make it work some steps are to be implemented.

4.1 Algorithm: SQL-to-NoSQL(database)

Step 4.1.1 Establishing connection with SQL database.

We are using MySql database as SQL database, we make connection with the help of java drivers to connect with the MySql, we need a sql connector jar file to be added to our classpath and for making connection we have to use the following code:

```
Class.forName("com.mysql.jdbc.Driver");
Connection  conn= DriverManager.(“jdbc:mysql://localhost:3306/database
                                name","usename", "password");
```

Here com.mysql.jdbc.Driver is the package name that make connection with database and username and passwords should be provided for making connection with MySql, generally root is the username and password is first time asked by server to fill.

Step 4.1.2 Getting all the information about the database.

After successfully establishing connection with the MySql database we need to know about its information, that is how many databases it stored, what are their names and how much they store data, printing databases name etc. Following is the code in java for knowing the metadata about databases.

```
ResultSet resultSet = conn.getMetaData( ).getCatalogs( );
```

The resultSet store the array of results returned by getMetaData( ) function, metadata is data about data, and getCatalogs will return all the databases stored in the MySql

databases stored into it. As can be seen in fig 4.1.1 all the databases stored in the MySql are shown in drop down list and can be selected for the conversion.

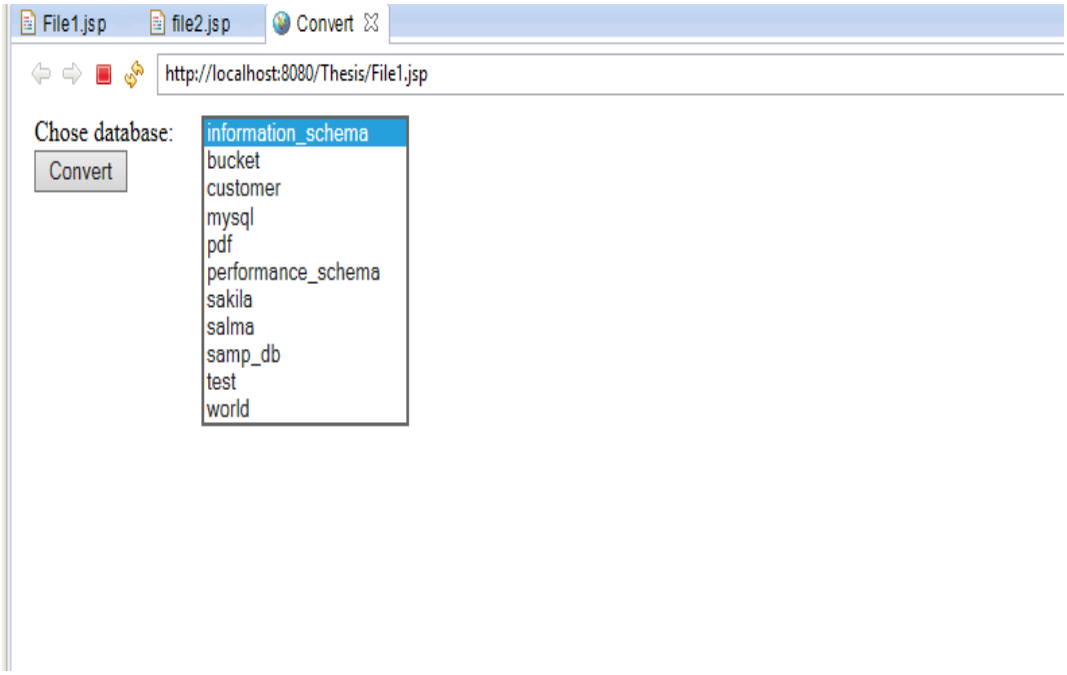

Fig 4.1.1 Information Schema of the MySql database

Step 4.1.3 Retrieving metadata about the database example number of tables, their names storage size, indexing etc.

As now we have our database we also needs its tables and columns name and other information, so it can be converted into another databases or can be stored in another databases. Following code shows how to retrieve the metadata about the tables after selecting the database name.

```
String [ ] types = { "TABLE" };
ResultSet resultSet = conn.getMetaData().getTables(null, null, "%", types);
```

We have taken array with single element TABLE , as we already database so we use getMetaData( ) to get tables from the database. We can retrieve table using the following function, reslutSet.getString(3) will return the table name of respective database , as there can be more number of tables so we have to use ResultSet.

Step 4.1.4 Making connection with NO-SQL database i.e MongoDB(our case) , using drivers example php, java, C++ etc.

For making connection with the Mongodb, we need drivers for java, so we have to download the jar file and import them to our java code and then we can connect to MongoDB using the following code.

Mongo mongo= new Mongo("localhost", 27017);

Here mongo is the client communicating with the mongo server i.e mongod , it is running currently at localhost on port number 27017.

Step 4.1.5 Creating database in Mongodb as the one retrieved from the SQL database.

Now as there is successful connection with MongoDB , so now we need to store the database into it , the following code creates the database with name db, as db is the argument passed by the SQL dataset.

Db dbase= mongo.getDB(db);

The code will create the newdatabase dbase taking db as the name, the db here is a string that is passed after selected from the given option as can be seen from the fig 4.2.2, the database in MongoDB will store the database with the same name that existed in MySql.

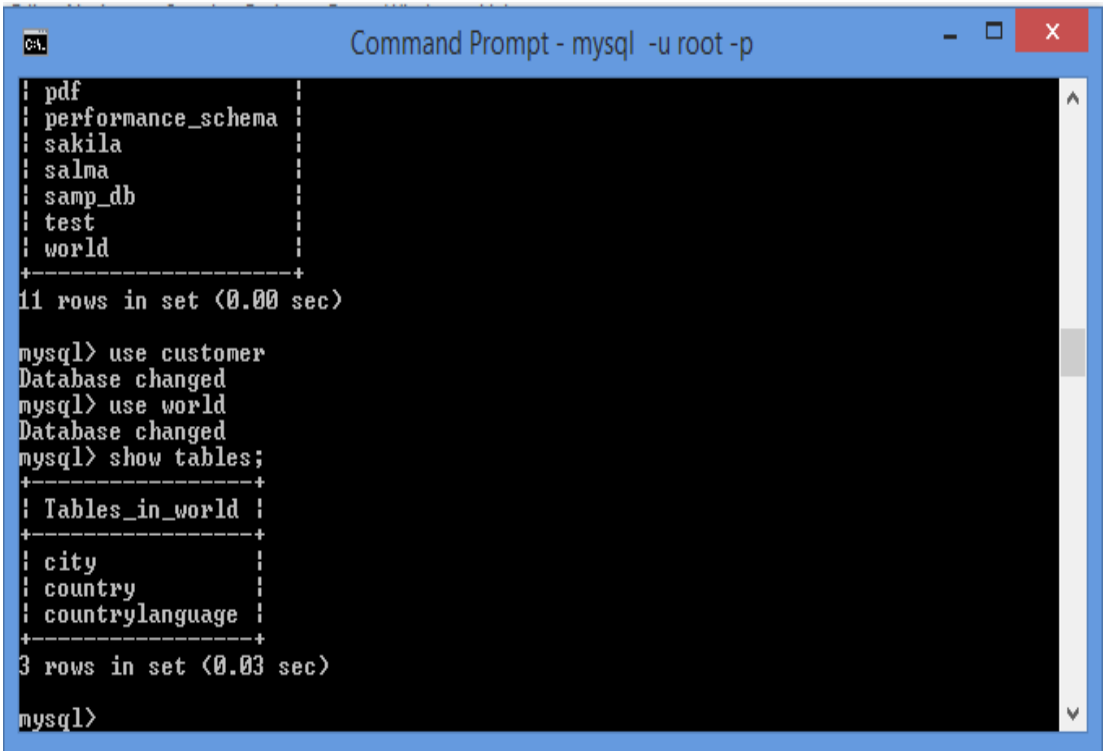

Fig 4.1.2 Database in MySql

Step 4.1.6 Retrieving data row by row from SQL database using queries and storing them in the MongoDB as documents.

Now we have database in MongoDB and we have to insert collections in it. Collection are same as table in SQL they store document instead of rows and columns with the field and value. So in order to store the database we have to use the following code (as in Fig 4.1.6. and Fig 4.1.6.1).

For every table we have to repeat the following code:

```
table = resultSet.getString(3);

DBCollection collection = mdb.getCollection(table);

Statement stmt = conn.createStatement( );

ResultSet rs = stmt.executeQuery("SELECT * FROM "+table);

ResultSetMetaData rsmd = rs.getMetaData( );
```

Now as we have access to all data stored in table we have to parse it and passed as document to be stored in the MongoDb. We can do this with following code:

```
int no= rs.getMetaData( ).getColumnCount( );

col[i] = rsmd.getColumnName(i);

BasicDBObject document = new BasicDBObject();

document.put(col[j], rs.getString(j));

collection.insert(document);
```

we are storing the Column name in col[ ] array that can hold (no+1) columns name as and receiving the value associated with it by using the function rs.getString( ) which will return the value stored at each row and line document.put(col[], rs.getString()) put the column name with its value and in last the document is inserted in collection and repeated for the each document. The following fig show the database stored in SQL:

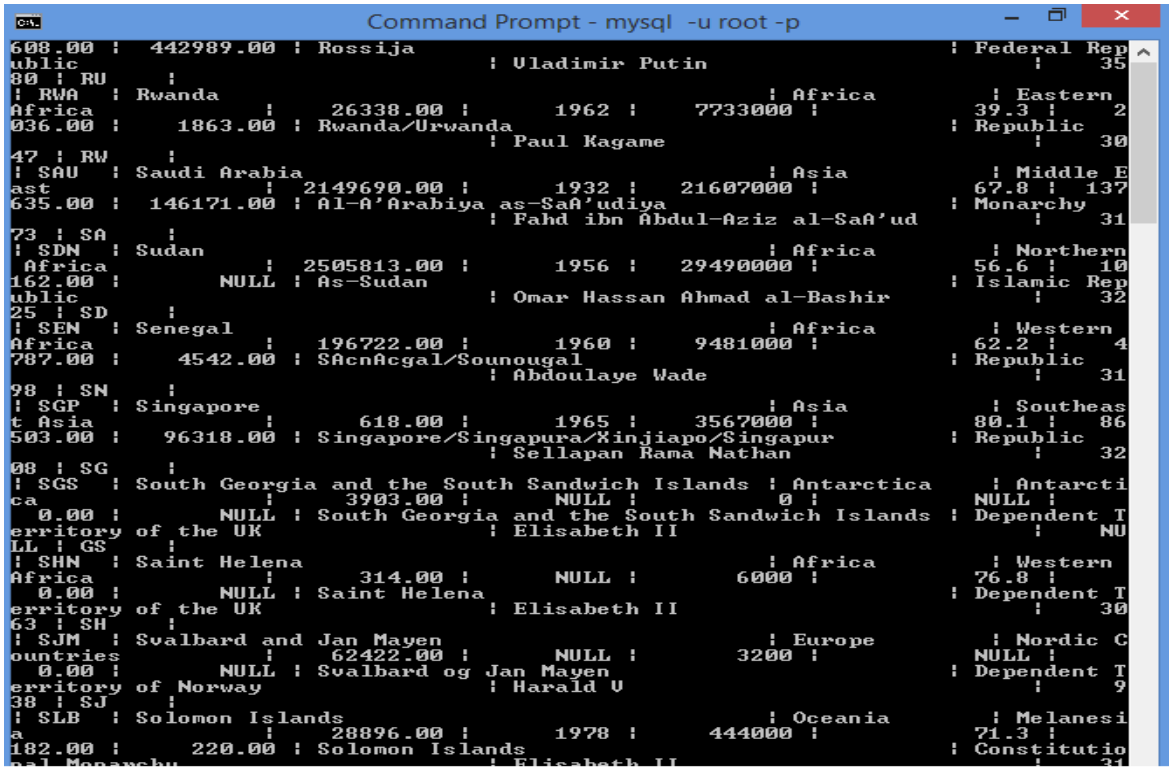

Fig 4.1.6 Converted database in MonogoDB showing Table city

The Figure 4.1.6.1 shows the converted data stored in MongoDb.

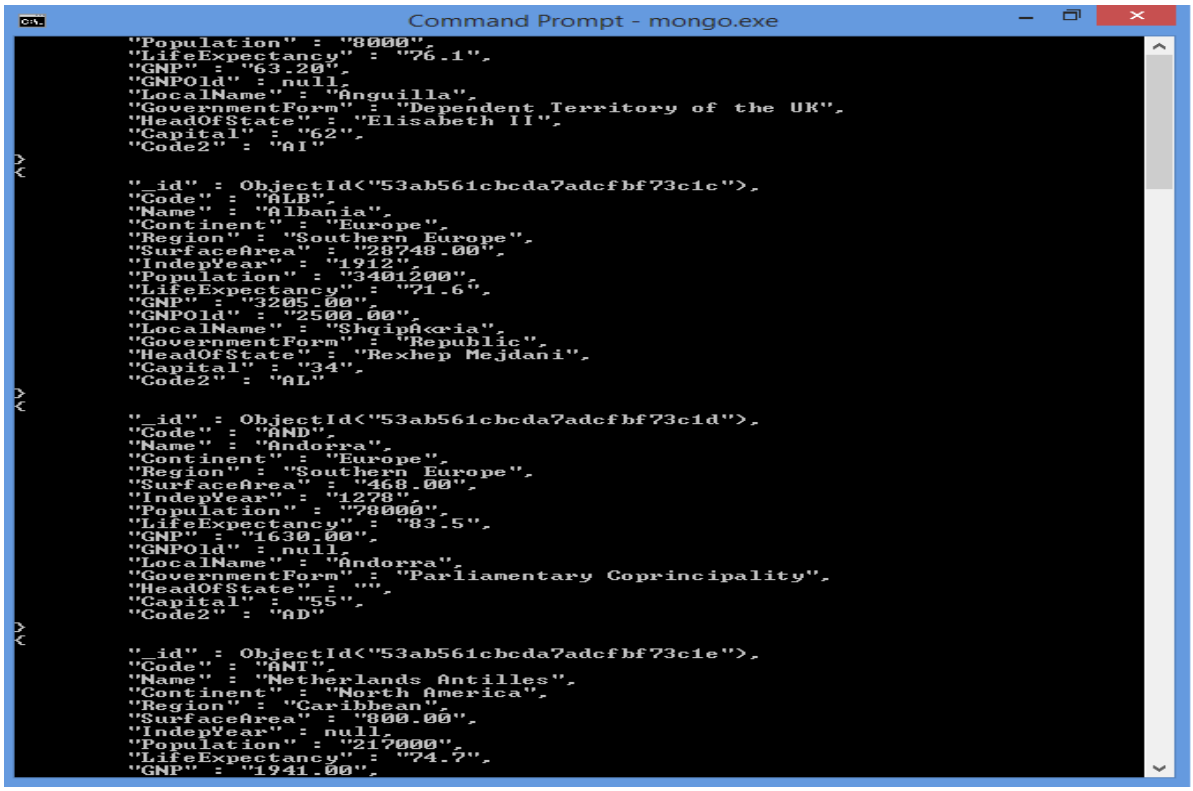

Fig 4.1.6.1 Converted database in MonogoDB showing Table country

Step 4.1.7: Repeat step for each new database.

Likewise we can convert each and every database and can store it in the document form in MongoDB.

4.2 NoSQL Data Model

The data model which is used here is based on MongoDB and Redis. Using NoSQL provides the benefit of storing data in schema less structure. As per the system's requirements; the basic information which is stored in MongoDB is about person's identification, and his current location. Insurance_ID, Total_Installments, Installments_left can be set into the basic information of Insurance with embedded documents in collection. Client's embedded documents contains the information of client personal details like name, address, insurance_id. Location is one other document which contains the location as per insurance id.

4.2 System Implementation

This work is done using Java language, MongoDB is used to store the data. This data is then made to split according to the location of the person. The categorized data according to the location is retrieved with the help of Redis and this is done for the fast access and operations on data. Thus two databases are merged in order to store and retrieve. MongoDB is used for large-scale data to be stored in schema-less structure and Redis is used for fast retrieval of data.

Table : 4.1 Experimental Set up for Implementation

Processor	Intel Core i4/i5 Processor
Processor Speed	2.5 GHz, 2.10 GHZ
Cache	3 MB
Operating System	Windows 7(64-bit)
IDE (Integrated Development Environment)	Eclipse Juno
Database	MongoDB 2.2.3, Redis 2.8.9, Oracle
RAM	2.10 GB
RAM Support	Up to 8GB
Hard Drive	320 GB

4.2.1 Establishing a Connection:

To connect Java programs with MongoDB database, drivers are installed and environment variables are set. Afterwards development kit of MongoDB is imported in the application program. Following are the basic CRUD commands used while implementation. To connect to MongoDB databases, the key code is as followed:

```
Mongo mongo =new Mongo(“DBServer”,27017)
```

connecting with the database server name and port number

```
DB db=mong.getDB(“final”)
```

open database, ‘final’ is the name of the database

4.2.2 Inserting the data

At first, Java program establishes the connection with database then passes the SQL statements to program. The data is inserted into database through java server pages (jsp). MongoDB database make use of object-oriented thinking for implementing the system, first, accordingly “insurance” to create a DBCollection object col, and then create a BasicDBObject object obj, add insurance information to the object col, and finally inserting data into database.

```
DBCollection col=db.getCollection(“insurance”);

BasicDBObject document=new BasicDBObject();

document.put("name", name);
```

4.2.3 Query the data

The data stored in MongoDB is inserted, updated and deleted using:

```
col.update(document,newdocument);

col.remove(searchQuery);
```

The data is updated and deleted by searching through the documents using cursor:

```
DBCursor cursor =table.find(searchQuery);

while(cursor.hasNext())

//processing the data cursor.Next();
```

4.2.4 Making connection with Redis

The data stored using MongoDB is splitted according to the attribute ‘Location’ and stored in Redis. To make connection with Redis following code is required:

```
static Jedis jedis = new Jedis(“DBServer”,6379);

    try{

        Jedis.connect();

        }

catch(JedisConnectionException   System.out.println("can not connect");}
```

4.2.5 Inserting data into Redis:

The data is inserted with the following code

```
jedis.set(insurance,address);
//insurance is key, address is value
```

4.2.6 Retrieving the data using Redis

The data is categorized as per location and sent to the required destination, the information is retrieved at a faster pace at the client’s demand.

```
searchQuery.put(“location”,location);
```

4.2.7 Using Eclipse:

At front end with the help of jsp (java server pages) and servlets customers are

inserting their Data. At the backend data is sent to Mongodb to store large, scalable data along with the Redis. For fast retrieval data is segregated and sent to Redis at remote locations, So that the client at their particular locations are able to get their details and left over instalments at the earliest.

Chapter 5

EXPERIMENTAL RESULTS

5.1 Implementation Steps

5.1.1 Starting Mongod.exe (Mongo server)

Installation of MongoDB is not required. Only the zipped file is to be downloaded and extracted. The data directory is needed to be configured and type the command "mongod". Mongod is ended with the alphabet 'd' because it is a daemon thread which runs at the backend. It keeps running but it is required to start it before mongo client. Fig 5.1.1 shows how to start server.

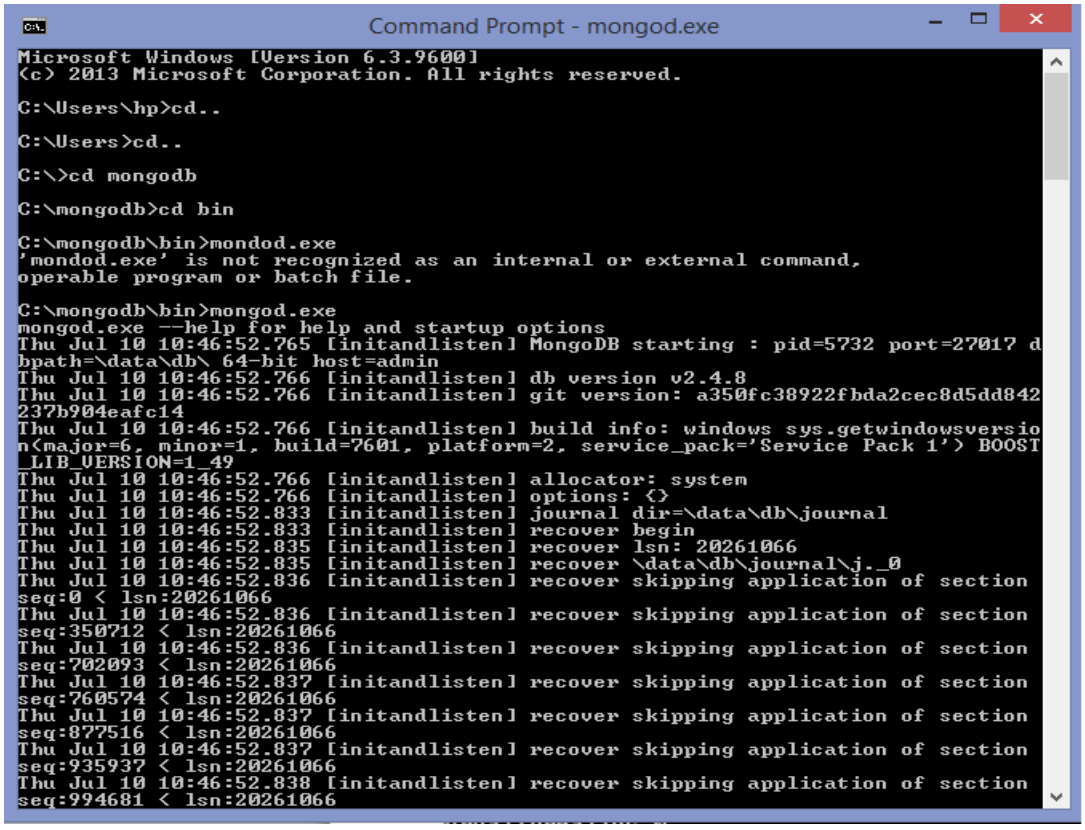

Fig 5.1.1 Starting Mongod.exe(Mongo Server)

5.1.2 Starting Mongo.exe(Client)

Mongo client is checked with the command "ping" and the "pong" keyword is

returned in order to show that the client is working. Various other commands are also run as shown in fig 5.1.2, the data is stored here in key value pair. So if a key is typed then its value is returned.

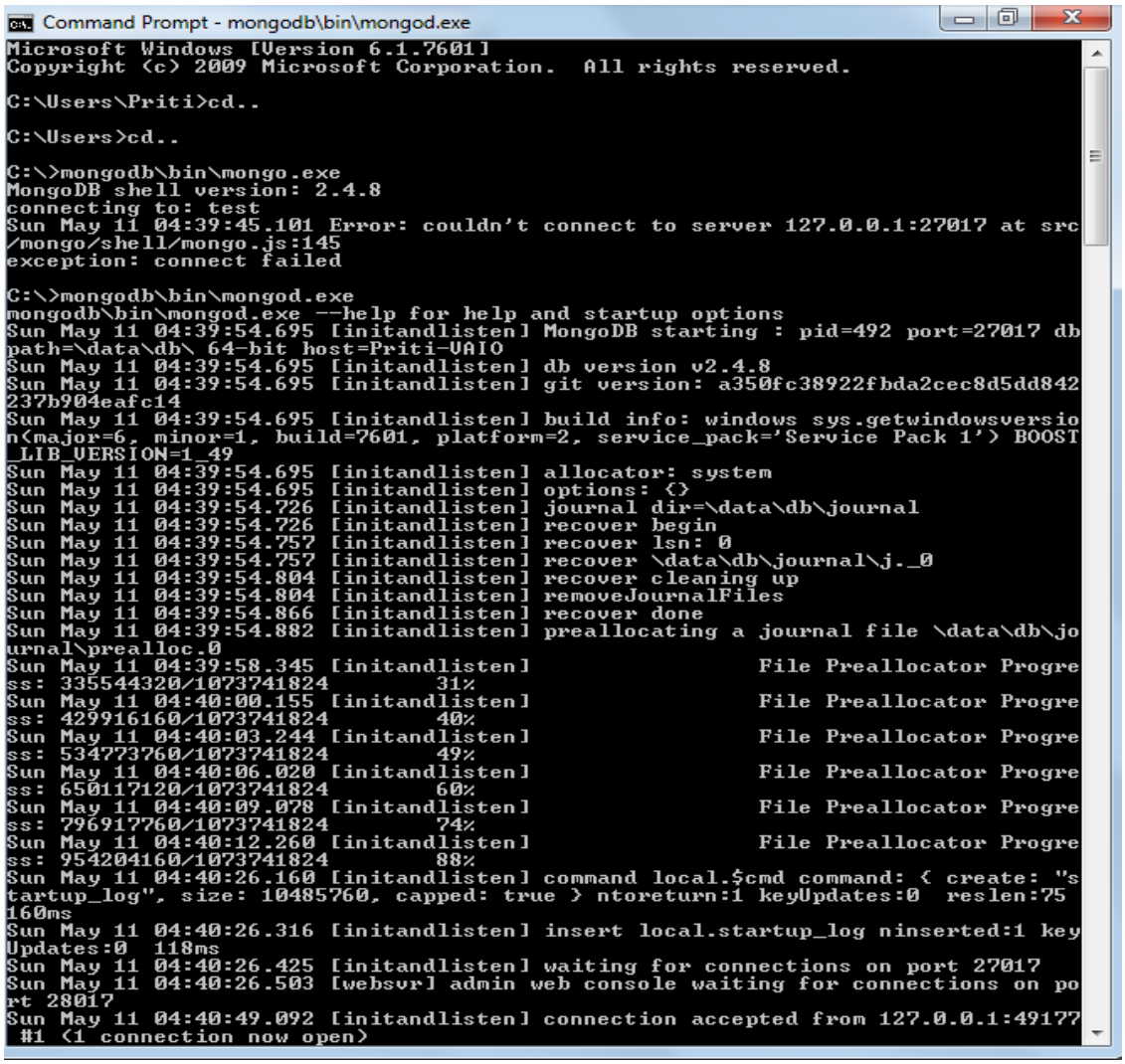

Fig 5.1.2 Starting mongo.exe(Mongo Client)

5.1.3 Creating database

Already existing databases can be checked with the command "show dbs". It list outs all the databases which are currently present.. To switch to other databases "use db_name" command is used. "show collection" command is used as shown in fig 5.1.3 to list all the tables in the current selected databases. "collection" in mongodb means "table" in SQL.

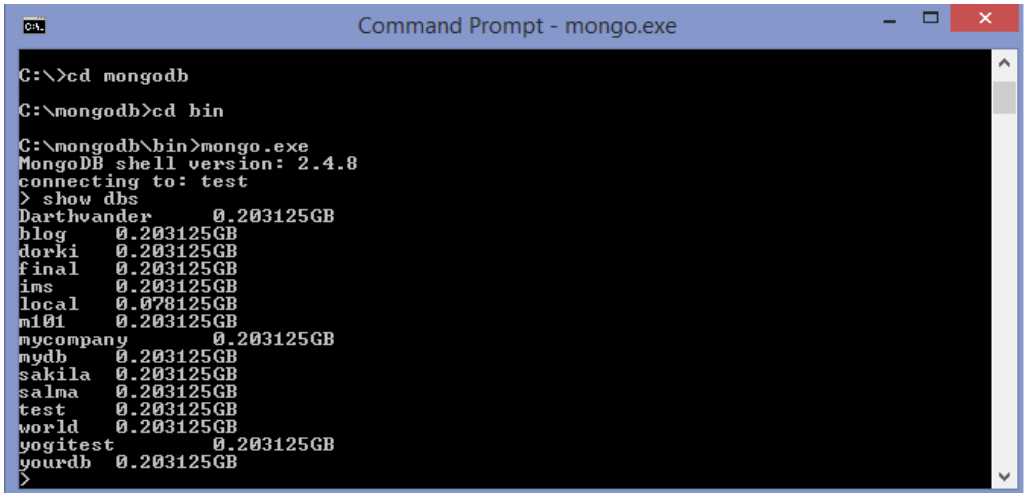

Fig 5.1.3 Creating database in MongoDB.

5.1.4 Staring Redis Server:

Redis is installed by downloading a stable version and extracting Redis binaries. Open windows services console as shown in fig 5.1.

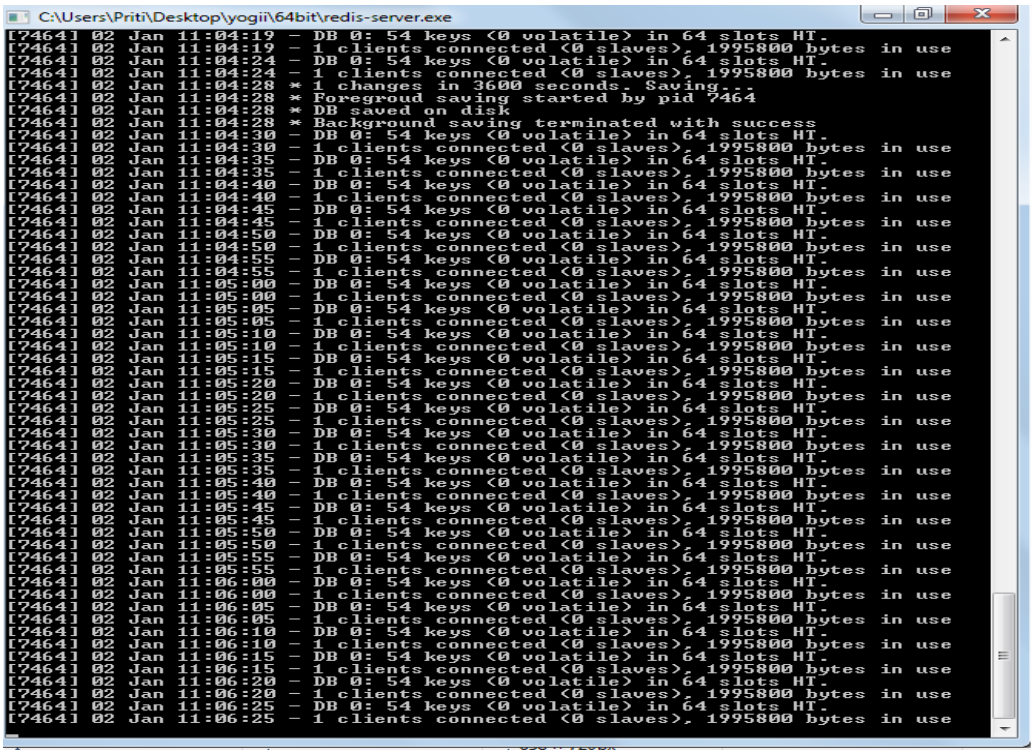

Fig 5.1.4 Starting Redis Server

From there redis service is started. It should have been istalled into c;\Program Files (x86) RedisWatcher. A config file is saved there called wather.config for additional editing.

5.1.5 Starting Redis Client:

Redis client stores data in key-value pair. First keys are set and then they are get as shown in fig 5.1.5. It stores Strings, Lists and Sets.

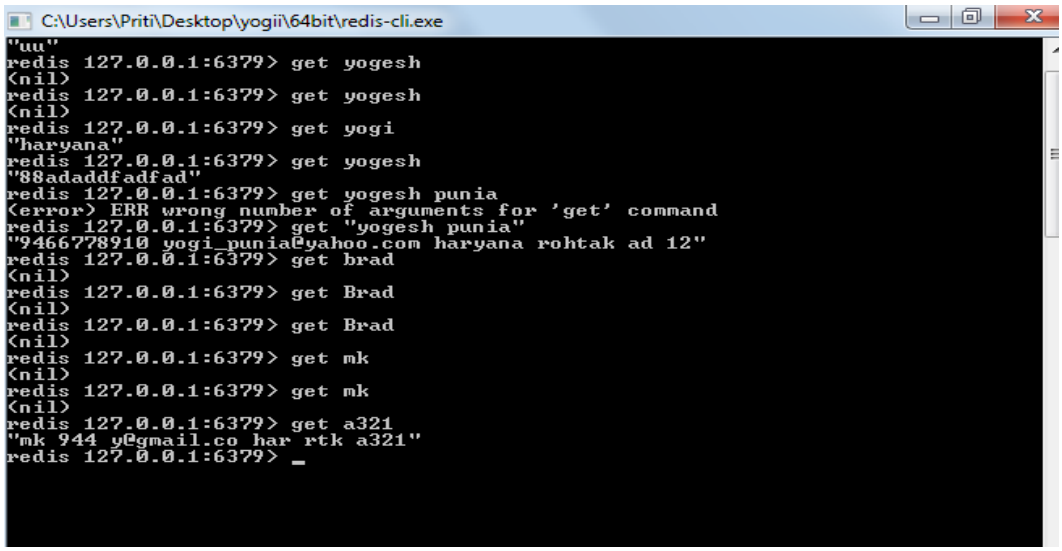

Fig 5.1.5 Starting Redis Client

5.1.6 Starting MySql server

MySql server is downloaded from Mysql.com, is freeware. There are two interface to use it with Graphical user inter face and another command prompt. The server keeps running in the background as shown in fig 5.1.6.

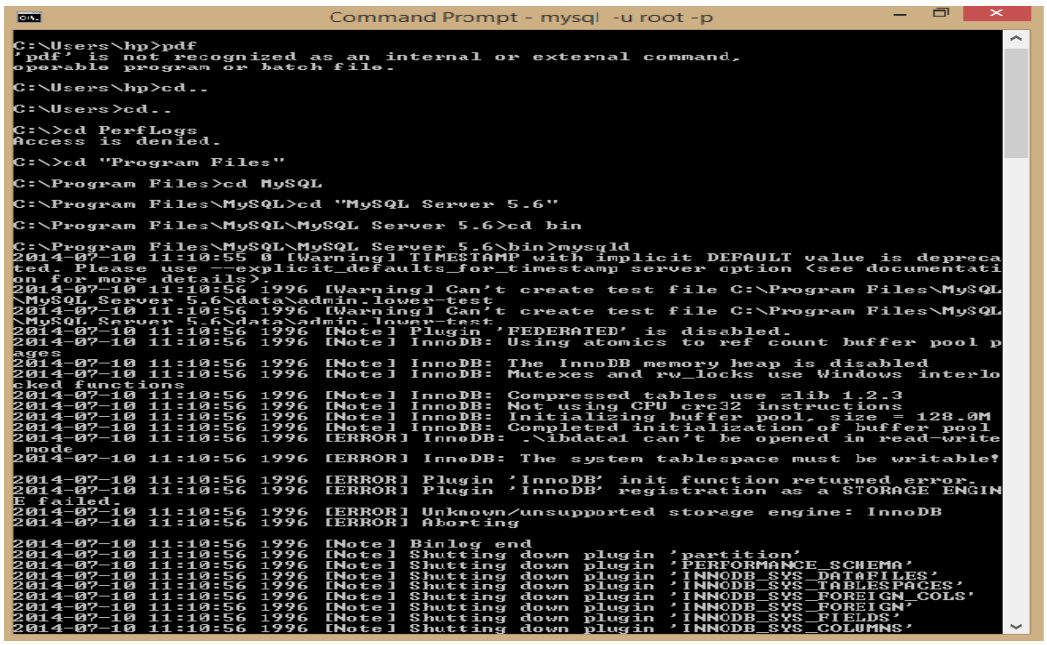

Fig 5.1.6 Starting MySql server

The command used to start the Mysql is “mysql –u root –p” and it will ask for the password and will get started by giving you the shell of mysql as can be seen in fig 5.1.6.1.

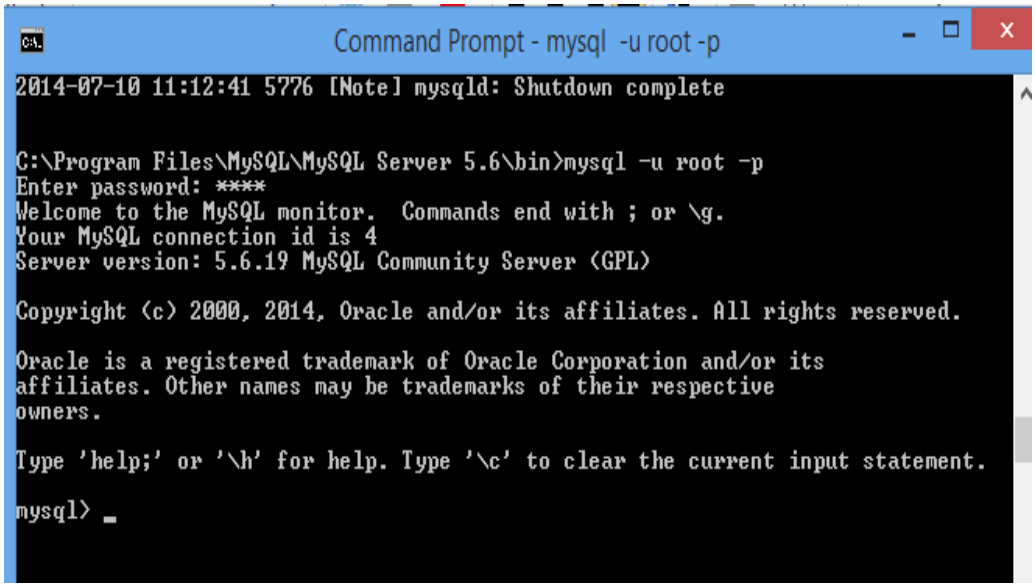

Fig 5.1.6.1 Starting Mysql

5.1.7 Inserting Values in MongoDB :

The first page of project looks like as shown below. It has various buttons which takes us to the various actions like Insert button which helps us to get the data stored into the mongodb. Show all action leads to showing out the complete list of documents stored into the database both in redis as well as in mongodb.

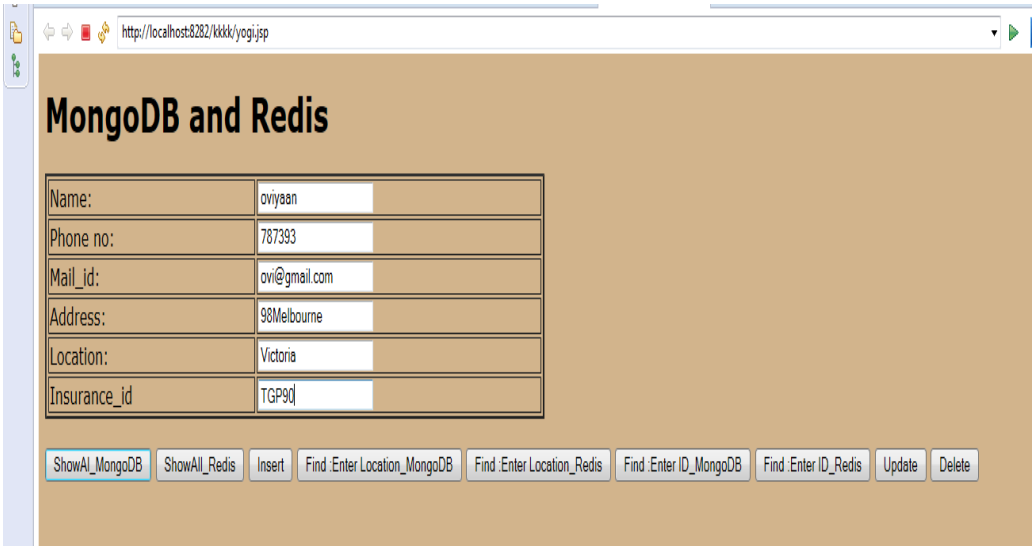

Fig 5.1.7 Inserting values in MongodB

Update and delete button helps in updating and deleting the data from the mongodb. Show by id and Show by Location button searches the data by unique insurance id of a client and by location it shows all the clients from that location.

Values are inserted and these can be checked into the database directly with the command “db.users.find()”. A unique object_id is created with every document into the collection. This is a unique row number. In fig 4.7 the values are shown as inserted into the database. Likewise a number of documents are inserted with the name, mail_id, location, address, phone number, insurance_id of different clients. In fig 5.1.7, the list inserted is shown inserted into redis too along with mongodb. This is done for the faster retrieval as compared to mongodb. CPU time is also calculated along with the insertion and searching modules in order to compare. The results of comparison are shown in further section.

5.1.8 Showing Inserted values:

The values are checked in to the database directly with the following command as shown in fig 5.1.8.1

db.users.find()

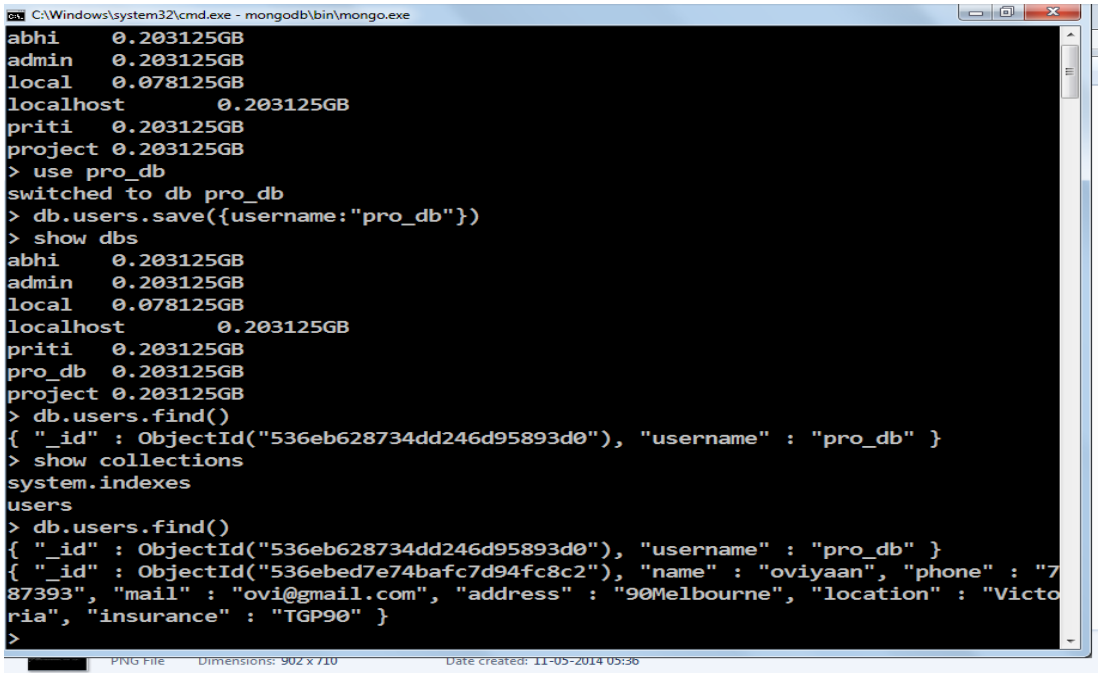

Fig 5.1.8.1 Values stored into MongodB

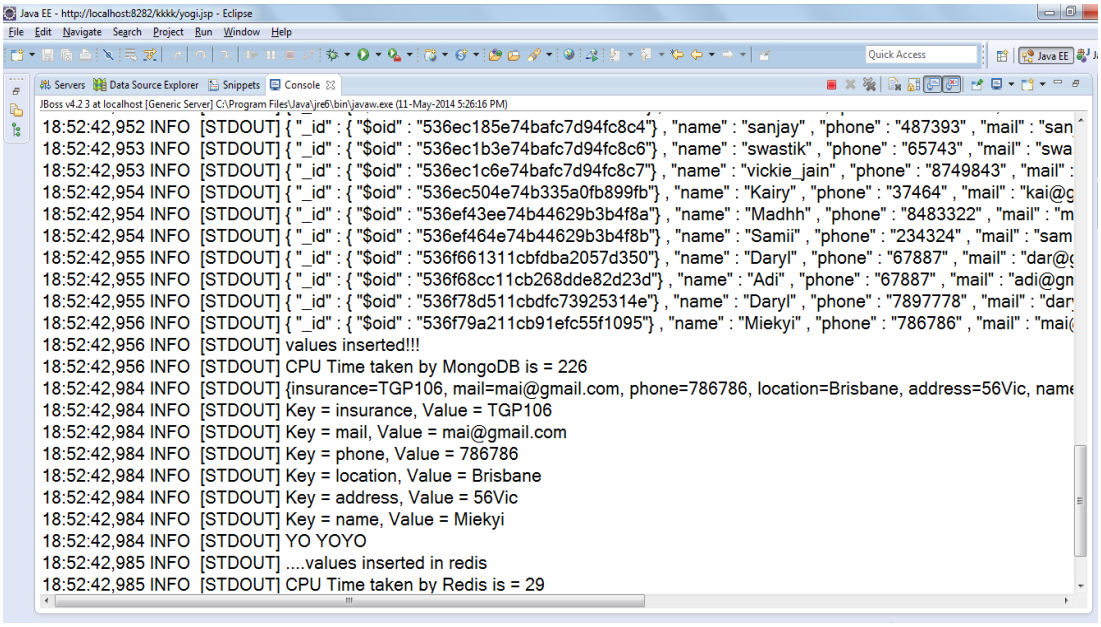

Fig 5.1.8.2 Values stored into Redis

5.1.9 Segregating values according to Location:

Enter the location like France, Victoria and all the clients are retrieved from the Redis server at a much faster rate than mongodb. The location is entered in the following figure (Fig 5.1.9).

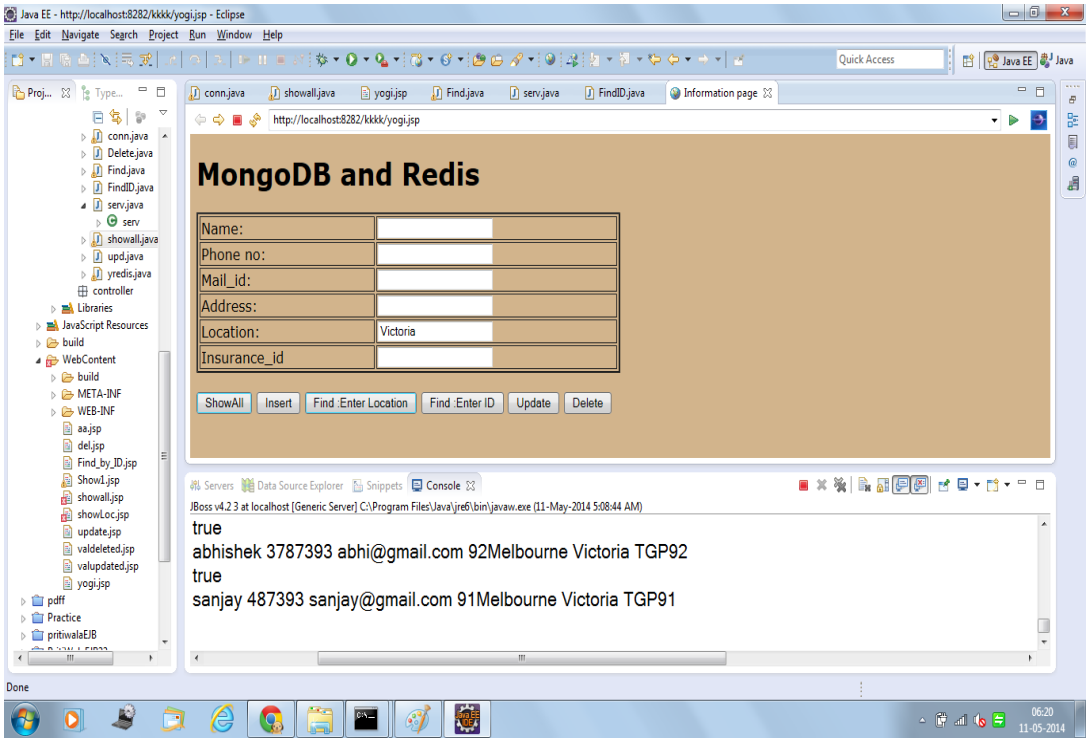

Fig 5.1.9 Search by Location in MongodB and Redis

Find_By_Loc button leads to the action of searching the information of all the clients from that particular location. The searched data is shown in the next figure (Fig 5.1.10).

5.1.10 Retrieving values according to Location:

The comparison between retrieving time of mongoDb and redis is shown in fig 5.1.11 and fig 5.1.12.

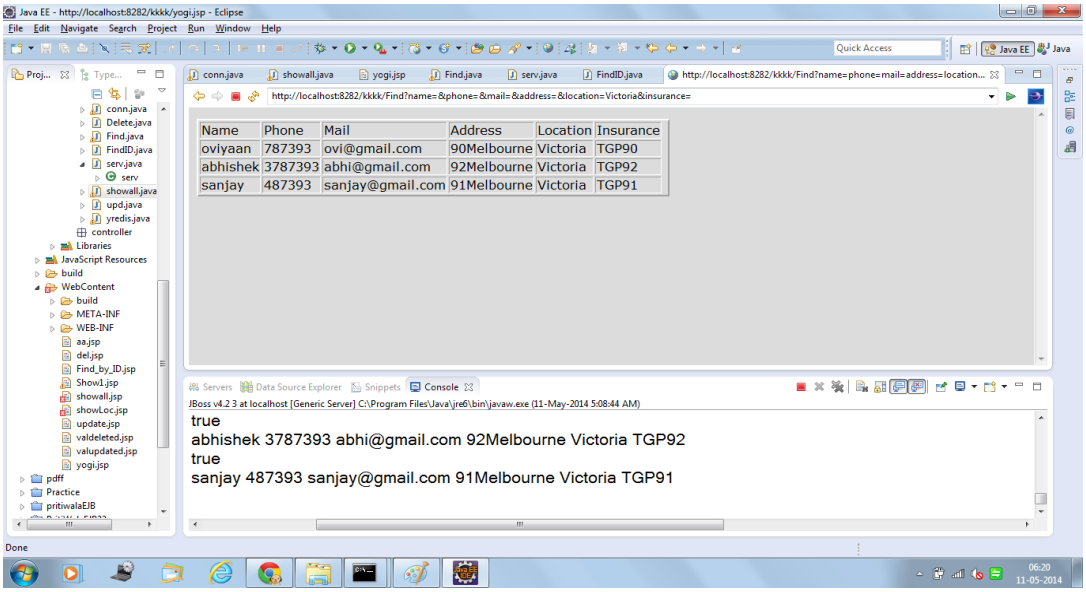

Fig 5.1.10 Values retrieved by Location

Comparing MongoDB and Redis in finding by location type :

5.1.11 Retrieving values according to Location through MongoDB:

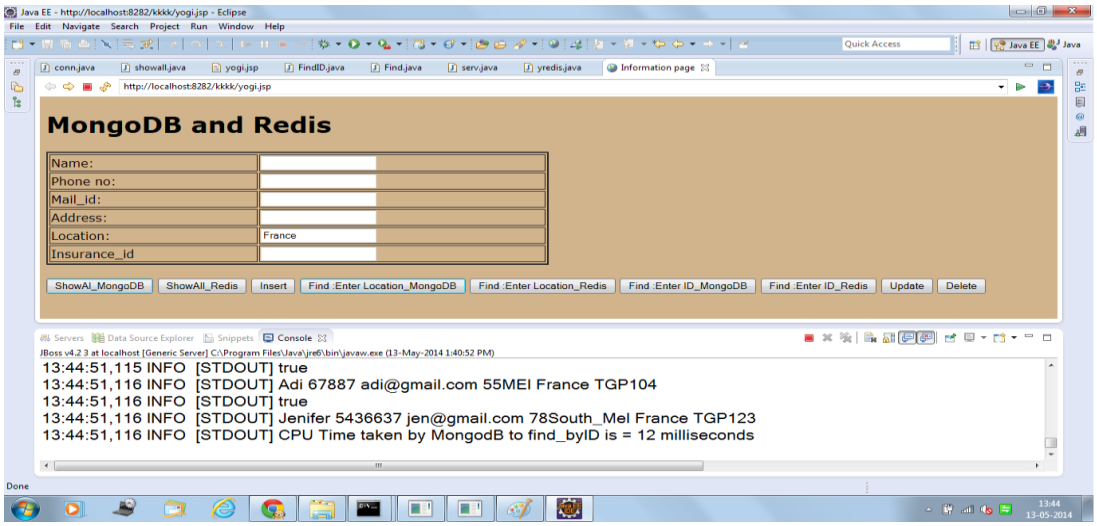

Fig 5.1.11 Values retrieved by Location through MongodB (12 millisec**)**

5.1.12 Retrieving values according to Location through Redis :

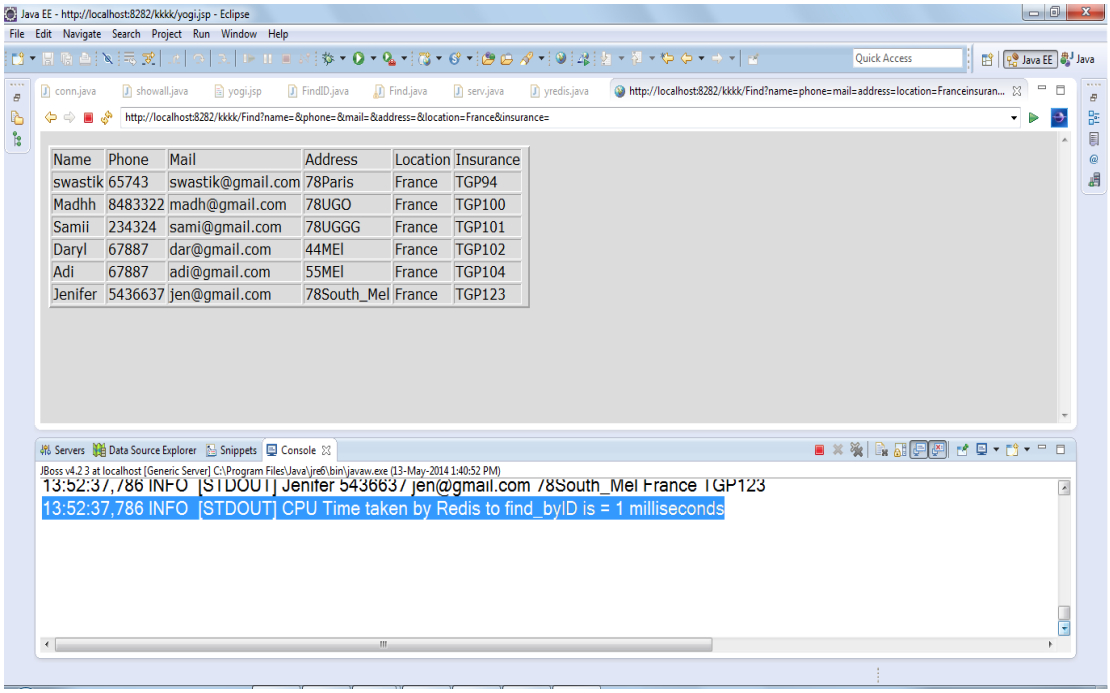

Fig 5.1.12 Values retrieved by_Location through Redis(1 millisec)

5.1.13 Retrieving values according to Unique Insurance_id:

Enter the unique id here(Insurance_id like TGP100) and all the clients are retrieved from the Redis server at a much faster rate than mongodb. The Insurance_id is entered in the following fig 5.1.13. Find_By_ID button leads to the action of searching the information of that particular client. The searched data is shown in the next figure (Fig 5.1.14 and Fig 5.1.15).

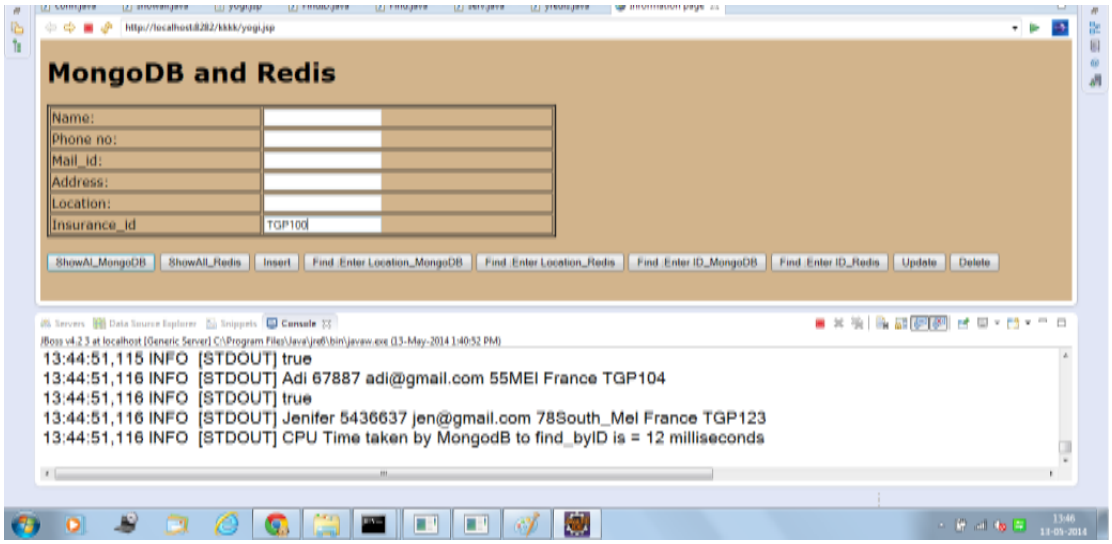

Fig 5.1.13 Search by unique id in MongodB and Redis

Comparing Mongodb and Redis in finding through Insurance_id:

5.1.14 Retrieving values according to unique id through Mongodb:

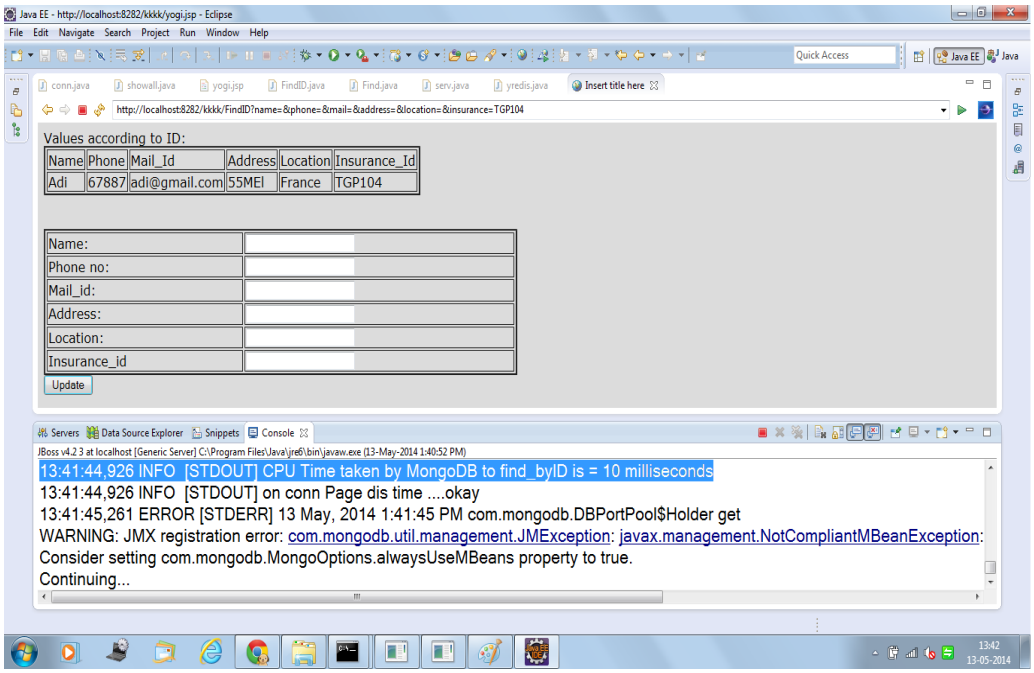

Fig 5.1.14 Values retrieved by ID through MongodB(10 millisec)

5.1.15 Retrieving values according to unique id through Redis:

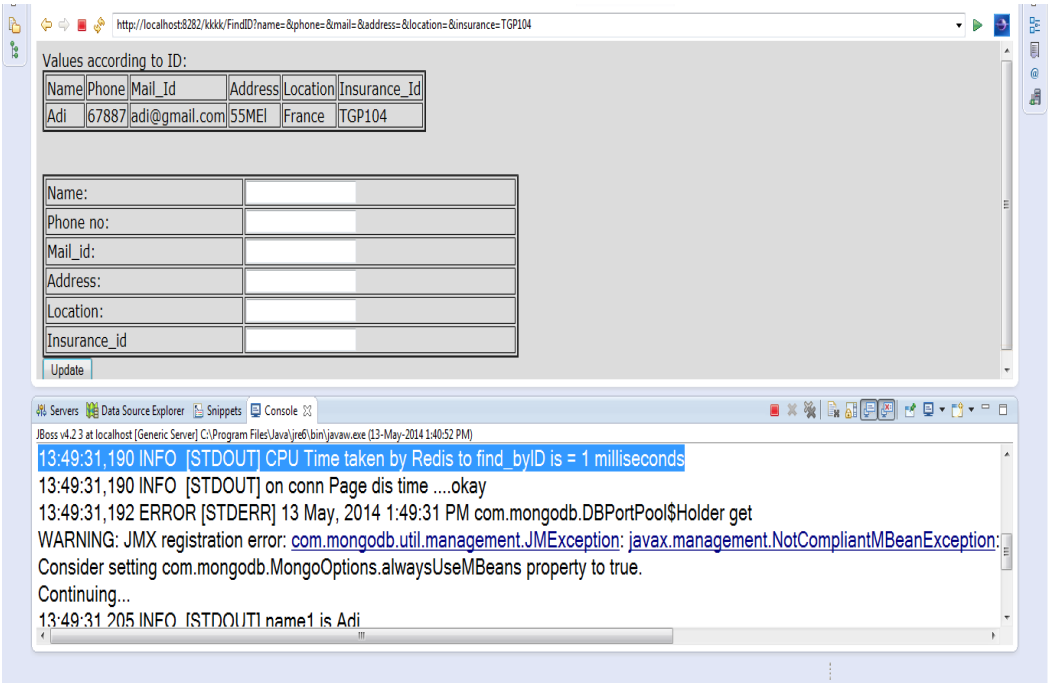

Fig 5.1.15 Values retrieved by ID through Redis(1 millisec)

5.1.16 Updating values through ID:

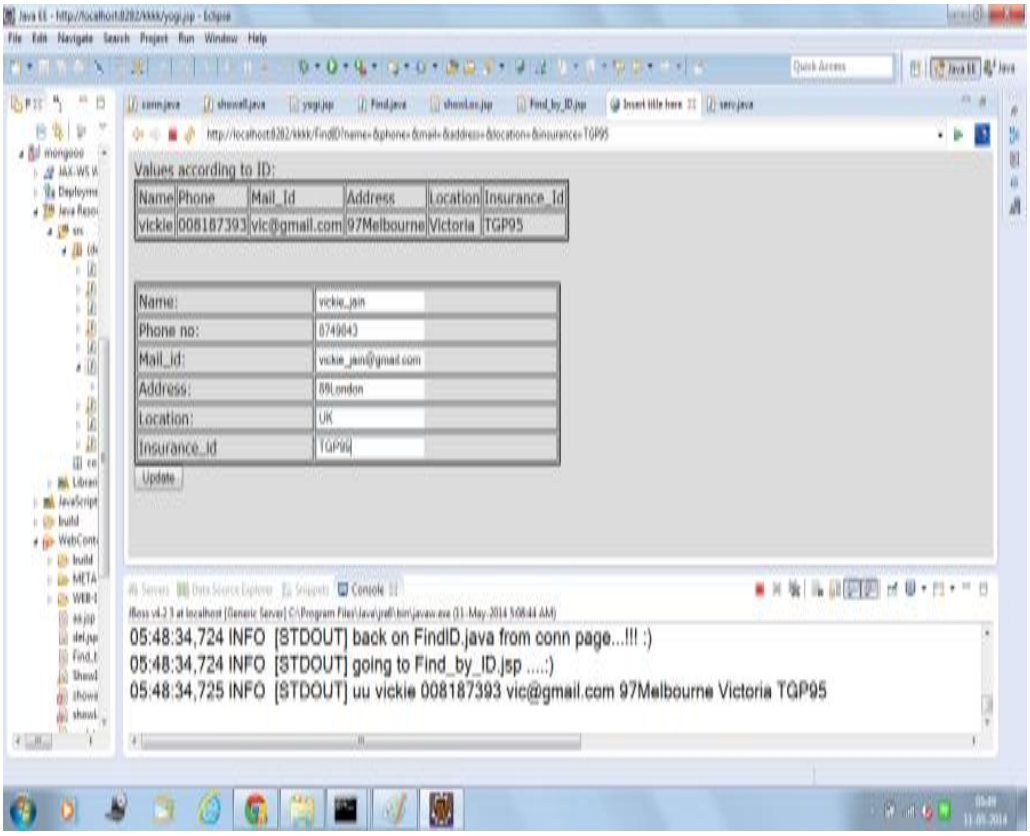

Fig 5.1.16 Updating values through ID:

5.1.17 Values updated in Mongodb :

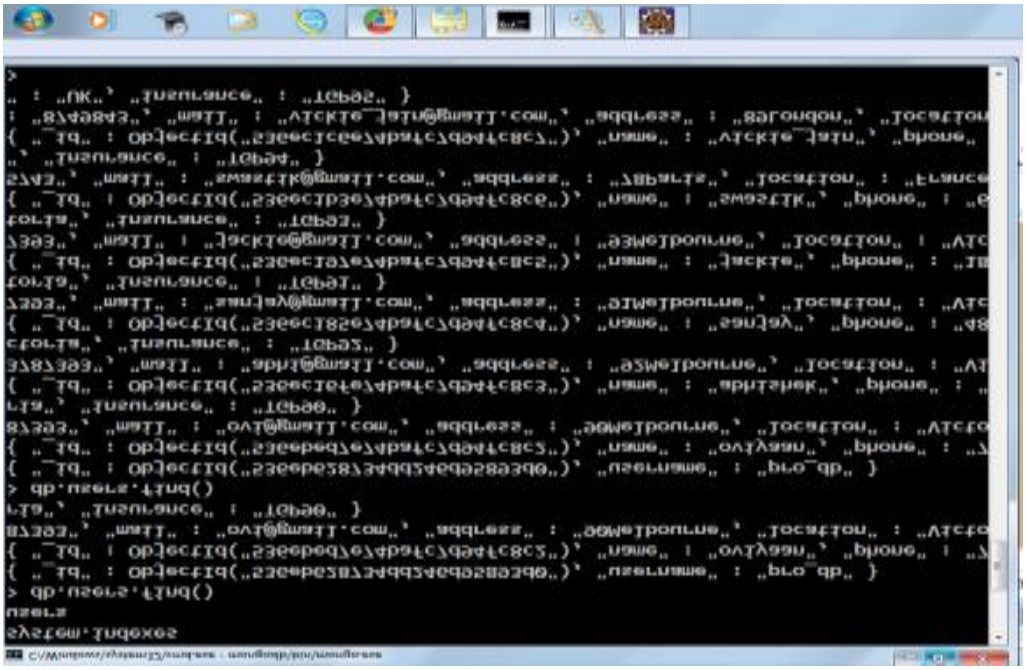

Fig 5.1.17 Values updated in Mongodb

5.1.18 All documents shown in Mongodb (Show_ALL) :

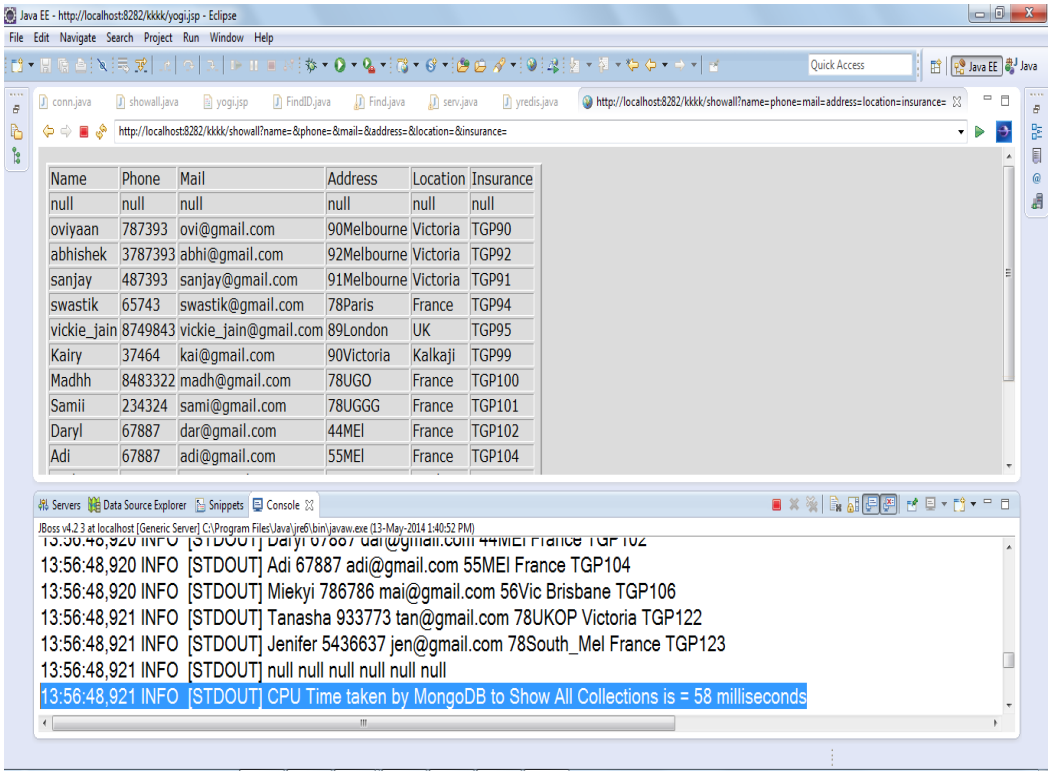

Fig 5.1.18 Documents shown in Mongodb(58 millisec)

5.1.19 All documents shown in Redis(Show_ALL) :

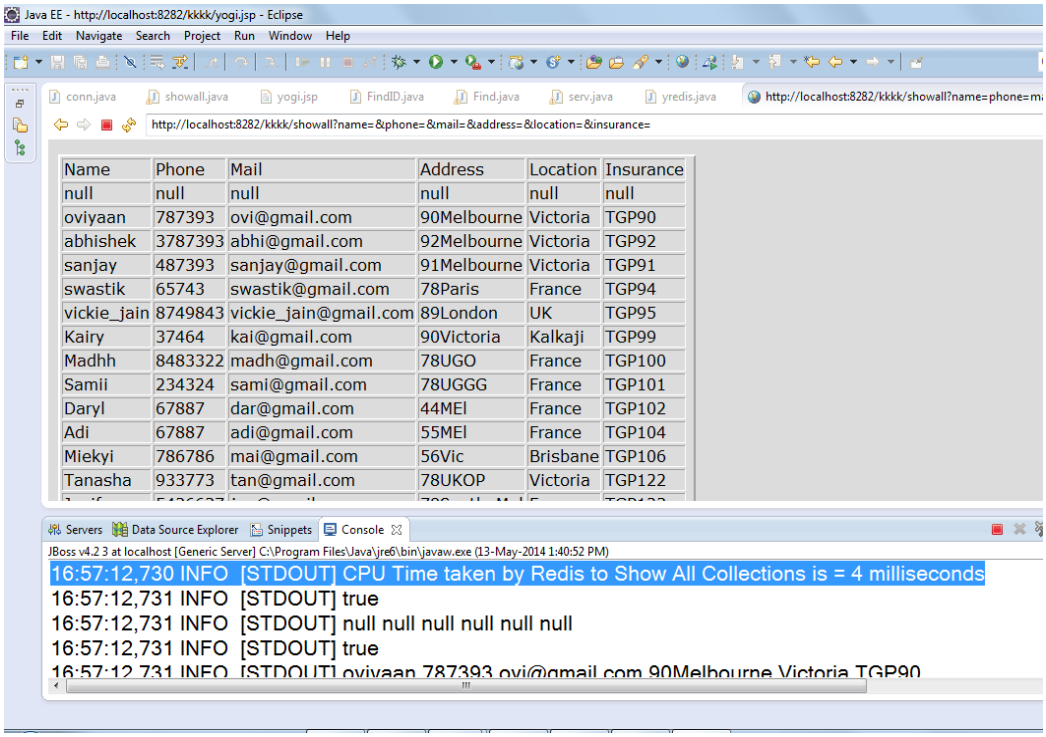

Fig 5.1.19 Documents shown in Mongodb(4 millisec)

5.1.20 Documents deleted from Mongodb (Delete) :

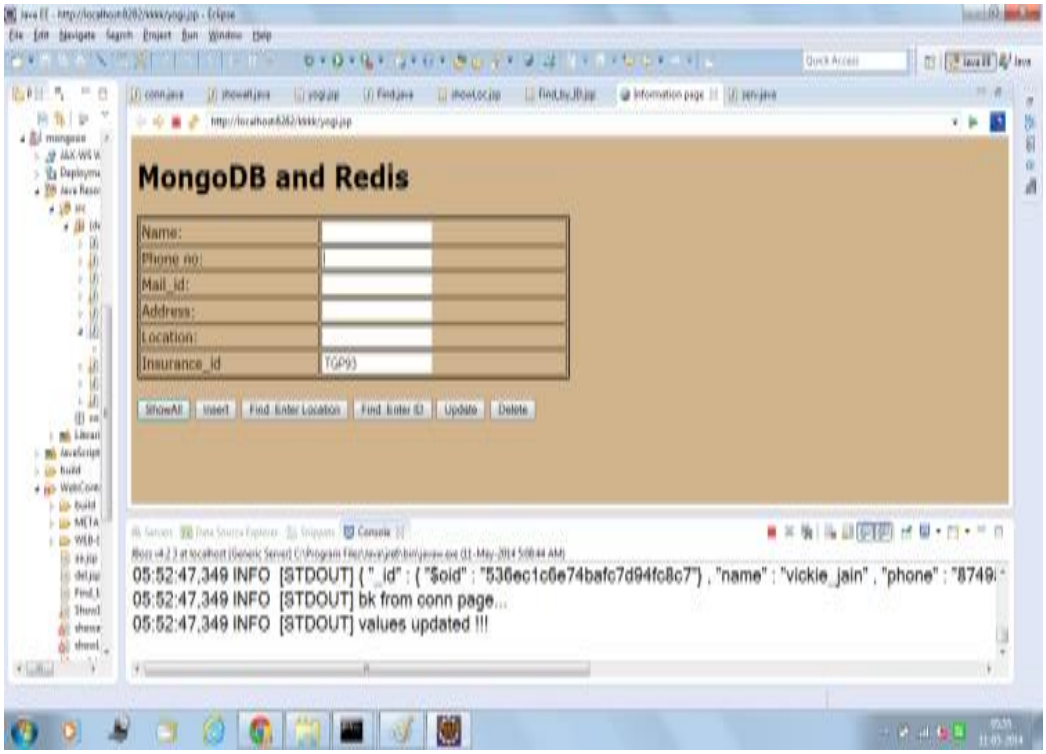

Fig 5.1.20 Documents deleted in Mongodb through ID

5.1.21 Documents deleted shown from Mongodb :

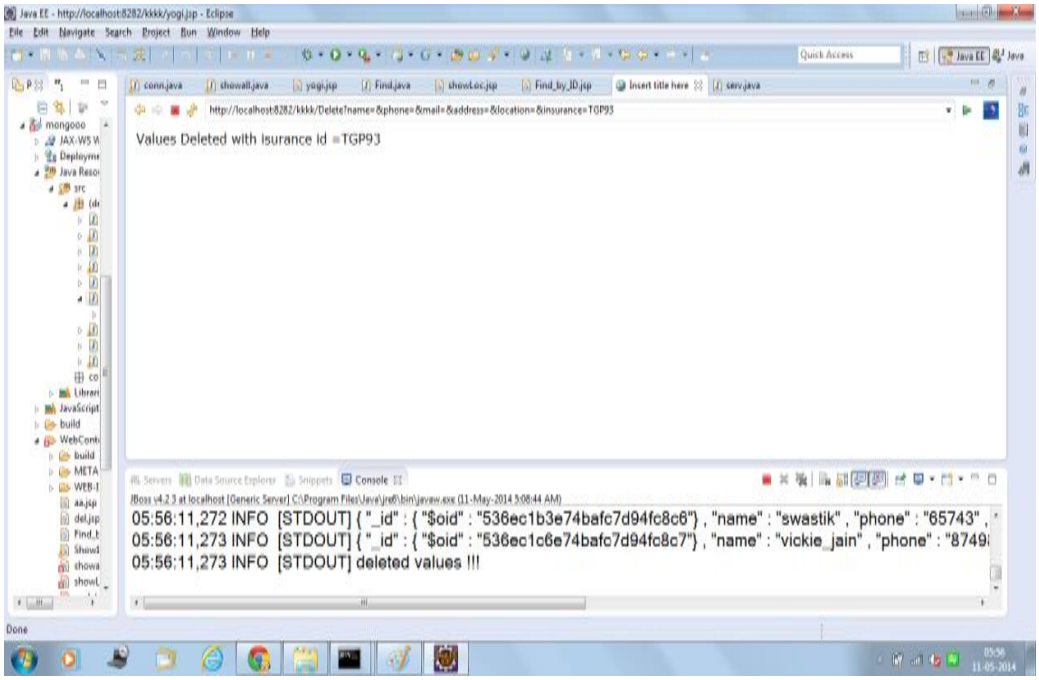

Fig 5.1.21 Documents deleted shown in Mongodb

5.2 Implemeting Information Management System Using Converted Database (from SQL-to-NOSQL)

Based on the database that have been converted from SQL to NoSQL , i.e from MySql to MongoDB, information Management System have been implemented that stores the input from user and also from the old system and can perform operation on the database.The system stores the basic information about the user like his name, date of birth, father's name, blood group, address, phone number, email_id, location and photograph. The system generates the html file with the filled details and a Uniqueid is generated with the file that can be use for further purposes. There are two basic operation available for the user : update, user can update his phone number, emalid and location. And can find_id to get information of someone with his location and his Uniqueid.

5.2.1 Inserting values in System

User needs to insert values in the system, the data from old legacy system is entered through conversion, and this database act as backend for Information Management system. As show in fig 4.3.1 the data is inserted by user, the code for inserting the values in MongoDB is as following

```
mongo = new Mongo("localhost", 27017);
db = mongo.getDB("ims");
table = db.getCollection("information");
document = new BasicDBObject();
document.put("Name", name);
document.put("Father's Name", fn);
document.put("Sex", sex);
document.put("BloodGroup", bg);
document.put("DOB", dob);
document.put("Street 1", s1);
document.put("Street 2", s2);
document.put("city", city);
```

```
document.put("state",state);

document.put("zipcode", zc);

document.put("Email", email);

document.put("Phone", phone);

document.put("Location", location);

document.put("photo",imagefile);

document.put("uid",uid);

table.insert(document);
```

first we need to make connection with mongo client at localhost on port number 27012 then getting database and then creating collection and finally inserting values in the document.

After inserting the values a new page is generated with uniqueid printed on the top of it and a copy is stored in the current file system and same copy is mailed to the user on the inserted email_id as can be seen in fig 4.3.2. , this is the file generated after the user presss insert button and the same copy is mailed to user through mail.

5.2.2. Updating Values

There is hyper link for the update operation which direct to another page , where only phone number , email_id and location can be updated ,so to update information user must enter the values and press update ,updated file will be sent to the email_id filled

```
BasicDBObject updateDocument = new BasicDBObject();

updateDocument.append("$set", new BasicDBObject().append("Email",
email)) BasicDBObject searchQuery2 = new
BasicDBObject().append("uid",uid);

table.update(searchQuery2, updateDocument);

searchQuery2= null;

 updateDocument.append("$set", new BasicDBObject().append("Phone",
phone));

searchQuery2 = new BasicDBObject().append("uid",uid);

table.update(searchQuery2, updateDocument);
```

```
searchQuery2= null;

updateDocument.append("$set", new BasicDBObject().append("location",
location));

searchQuery2 = new BasicDBObject().append("uid",uid);

table.update(searchQuery2, updateDocument);
```

5.2.3 Find ID

User can find its user by entring uid given to him with current location and the data will be printed according to the query made. The code used for the following are as following

```
BasicDBObject whereQuery= new BasicDBObject();

whereQuery.put("uid", uid);

DBCursor cursor = table.find(whereQuery);
```

The whereQuery put the uid in search query and cursor find the result , the information will be printed , as the user enter it .

To compare performance of MongoDB with Oracle following implementation of Oracle is needed. Same data has been saved, updated, deleted and retrieved by firing the queries in the next sub-section.

a) Oracle data model

The performance testing is done in comparison to Oracle which is a structured relational database. It uses common SQL language. SELECT, UPDATE, DELETE are the commonly used data manipulation statements. Modern relational databases are established based on the relational model of data proposed by E.F. Codd [6]. The data is stored in tables. Afterwards to split the data joins are applied which take time to retrieve the data, whereas in MongoDB it is a matter of only a small fraction of time as compared to time taken by oracle. As the number of records are added on a larger level this difference in time taken is a great slot. In MongoDB, for the operations like adding new records, updating and deleting the existing records, functions are used.

b) Traditional system implementation

In Oracle, the creation of table is done using CREATE statement:

```
CREATE TABLE info_insurance(
name varchar 2(50),
phone number(10),
```

```
mail varchar2(20),
address varchar2(50),
location varchar2(20),
insurance_id varchar2(10) not null
);
```

For operation like deletion of data, following query is fired when no other table is related to this one.

```
DROP TABLE info_insurance;
```

To insert data into Oracle databases, a new record is saved into database by using INSERT command. As for instance, following code is written:

```
INSERT INTO info_insurance (name, phone, mail, address, location, insurance_id)
VALUES ('Ram',9183929295,'54/4 GandhiMarg', 'delhi', 'I904" );
```

To update into Oracle databases either the entire table or part of an entry following statement is used:

```
UPDATE info_insurance
Set address='23 Mall Road'
WHERE location='Noida';
```

c) Comparing with MongoDB in insertion of data

A bunch of data is inserted both in MongoDB and in Oracle as well in order to check the amount of data taken in case of insertion in milliseconds. Table 5.1 shows the number of milliseconds for each bunch of inserts.

Table 5.1 Inserting time (in milliseconds)

No. of Records	**Oracle DB**	**MongoDB**
10	31	800
100	46	4
1000	1346	40
10000	8766	678
100000	84678	4321
1000000	884567	56789

Figure 5.2.1 explains the how the amount of time spent while insert operation in Oracle database is comparatively high. It is put in the form of graphs as shown below:

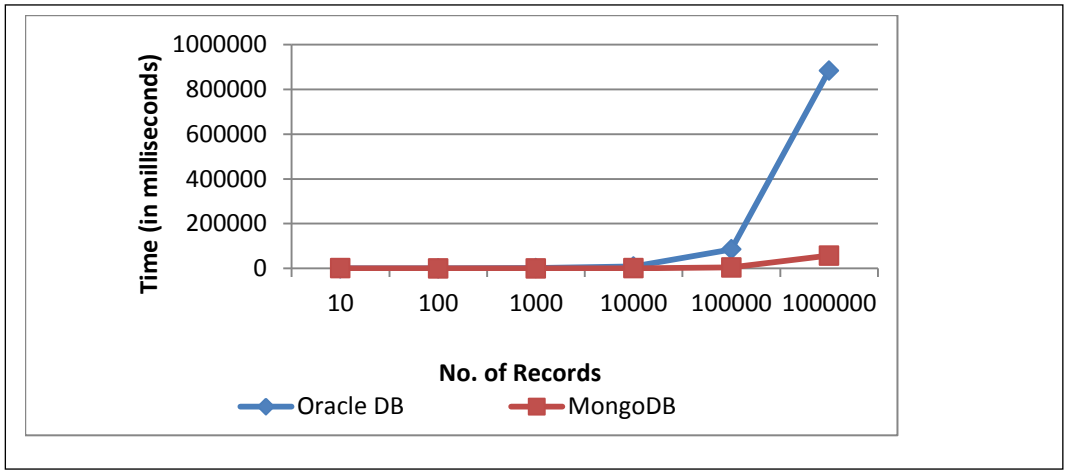

Figure 5.2.1 Inserting data

MongoDB is efficient for large insertion of large amount of data. For a small bunch of data it took a little too long to insert records. Oracle database on the other hand deals very nice for small amount of data but as the number of records is increasing, the time spent is bigger.

MongoDB and Redis are combined that is data is further segregated and stored on Redis database for fast retrieval. It is mandatory to show how the performance would have been affected if Redis has not been used and data would have been retrieved from MongoDB only.

Table 5. 2-MongoDB vd Redis

No. of Elements	**Mongo Read**	**Redis Read**	**Mongo Write**	**Redis write**
10	5	2	8	5
100	11	13	34	8
1000	93	38	153	31
10000	980	238	1394	220
50000	5218	958	8713	979

The following graph depicts the comparison between read an write operations in MongoDB and Redis:

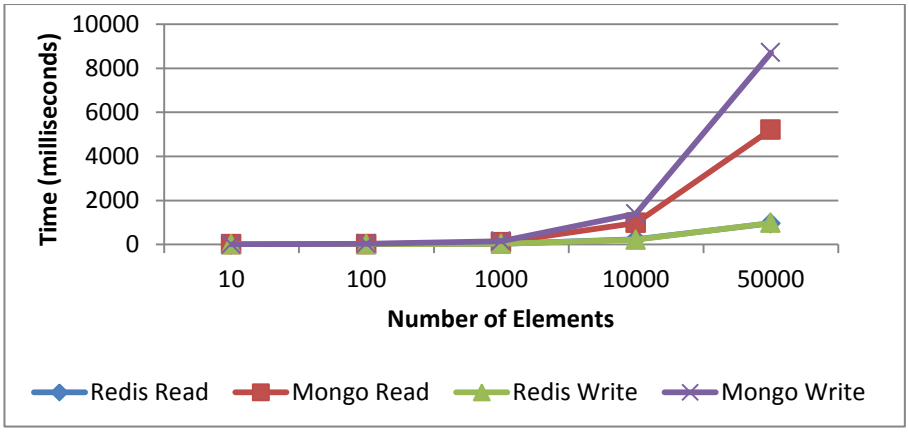

Figure 5.2.2 MongoDB vs Redis

Chapter 6

CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusions

Database from MySql server to MongoDB has been ported. All tables are stored in MongoDB as collections, all rows are stored as documents containing field and its associated value same operation can be performed on these rows as MongoDb is rich query syntax, Information Management System has been implemented on this transformed database and big data has been implemented using combination of MongoDB and Redis at the backend. This is done in order to do fast transactions of data that is fast insertion and retrieval of data. Both MongoDB and Redis are scalable and Redis is used for its speed as it used KVS whereas MongoDB is used for its flexibility, scalabilty, and efficient performance. With the help of MongoDB complex data into one field. An array or an object or a reference could be stored in a field in MongoDB. Also the deletion time in MongoDB was constant where as it generally increases in case of Oracle[6].

MongoDB is able to store large amount of data in schema-less structure and it is scalable. It is quite flexible in nature. It is easy to deploy and copy database from one server to other by using import-export tools. Complex data may be stored into a single field like an array or a reference.

Oracle is a bit complex database, it has relations between tables and tables have a fix structure. These relations may be one to one or many to many. These relations may be helpful to join tables and create complex queries.

The problem with Oracle is replication, if the database is to be copied, it is quite difficult. Even if the tools are considered for this purpose, they are not fast enough. It is very much slower than MongoDB.

6.2 Future Scope

The present implementation can be used to design more complex applications using

some map-reducing functions like to aggregate result functions. Also some plugins could be developed as it is an open source community. Big data can be completely implemented using a combination of various NoSQL database.

Even though MongoDB is also rich in queries but still it can’t perform all the queries performed by SQL databases, so some mechanism is needed so that MongoDB can replace SQL completely.

REFERENCES

[1] Zhu Wei-ping, Li Ming-xin and Chen Huan, "Using MongoDB to implement textbook management system instead of MySQL", Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, pp.303-305, 27-29 May 2011

[2] Changqing Ji; Yu Li; Wenming Qiu; Awada, U.; Keqiu Li, "Big Data Processing in Cloud Computing Environments", Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on, pp.17-23, 13-15 Dec. 2012

[3] Leavitt, N., "Will NoSQL Databases Live Up to Their Promise?", Computer , vol.43, no.2, pp.12-14, Feb. 2010

[4] Persisting Objects in Redis Key-Value Database Matti Paksula University of Helsinki, Department of Computer Science Helsinki, Finland

[5] Michael Armbrust, Nick Lanham, Stephen Tu, Armando Fox, Michael J. Franklin and David A. Patterson, "The case for PIQL: a performance insightful query language", June 10-11, 2010, Indianapolis, Indiana, USA

[6] Alexandru Boicea, Florin Radulescu, Laura Ioana Agapin,"MongoDB vs Oracle database comparison", Emerging Intelligent Data and Web Technologies (EIDWT) IEEE, 2012 Third International Conference, pp.330 – 335, 19-21 Sept. 2012

[7]Karamjit Kaur and Rinkle Rani, "Modeling and Querying Data in NoSQL Databases",IEEE international Conference on Big Data, 2013, pp.1 – 7, 6-9 Oct. 2013, Silicon Valley, CA

[8]Rupali Arora and Rinkle Rani, "Modeling and Querying Data in MongoDB", International Journal of Scientific and Engineering Research (IJSER) -Volume 4, Issue 7, pp.141-144 July 2013

[9]Priyanka Raichand and Rinkle Rani, "A SHORT SURVEY OF DATA COMPRESSION TECHNIQUES FOR COLUMN ORIENTED DATABASES",

Journal of Global Research in Computer Science(GJRCS), Volume 4, No. 7, pp.43-46, July 2013

[10]Priyanka Raichand ” Query Execution and Effect of Compression on NoSQL Column Oriented Data Store using Hadoop and HBase”, International Journal of Scientific and Engineering Research (IJSER), Volume 4, Issue 9 , Sept. 2013

[11] Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang,” Modeling MongoDB with Relational Model”, Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference IEEE, pp. 115-121, 9-11 Sept. 2013,

[12] Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J., "Security Issues in NoSQL Databases," Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference, pp.541-547, 16-18 Nov. 2011

[13] Yimeng Liu; Yizhi Wang; Yi Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," Computer Science & Education (ICCSE), 2012 7th International Conference, pp.851-854, 14-17 July 2012

[14] Alexandru Boicea et al. “MongoDB vs Oracle - database comparison”, Emerging Intelligent Data and Web Technologies (EIDWT) IEEE, 2012 Third International Conference, pp.330-335, 19-21 Sept. 2012

[15] Nyati, S.S.; Pawar, S.; Ingle, R., "Performance evaluation of unstructured NoSQL data over distributed framework," Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference, pp.1623-1627, 22-25 Aug. 2013

[16] Yimeng Liu, Yizhi Wang and Yi Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," Computer Science & Education (ICCSE), 2012 7th International Conference, pp.851-854, 14-17 July 2012

[17] Scott Poe, Susan V. Vrbsky and Zachary Parker, “Comparing NoSQL MongoDB to an SQL DB”, ACMSE'13, April 4-6, 2013, Savannah, GA, USA. ACM

[18] Veronika Abramova and Jorge Bernardino ”NoSQL Databases: MongoDB vs Cassandra”, C3S2E '13 Proceedings of the International C* Conference on Computer Science and Software Engineering, pp. 14-22, July 10–12, 2013, Porto, Portugal,

[19] Abdul Sattar, Torben Lorenzen and Keerthi Nallamaddi, “Incorporating NoSQL into a Database Course”, pp. 50-53, June 2013, ACM New York, NY, USA,

[20]Sumita Barahmand, Shahram Ghandeharizadeh and Jason Yap,” A Comparison of Two Physical Data Designs for Interactive Social Networking Actions”, CIKM '13 Proceedings of the 22nd ACM international conference on Conference on information & knowledge managementSan Francisco, CA, USA., pp. 949-958, Oct. 27–Nov. 1, 2013.

[21] Wenbin Jiang, Lei Zhang, Xiaofei Liao, Hai Jin and Yaqiong Peng, ”A novel clustered MongoDB-based storage system for unstructured data with high availability”, pp 455-478, 2013.

[22] Javier Espinazo Pagán, Jesús Sánchez Cuadrado and Jesús García Molina, ” A repository for scalable model management”, Software & Systems Modeling Springer-Verlag, 17 March 2013.

[23] PaoloAtzeni, Francesca Bugiotti and Luca Rossi. ”Uniform access to NoSQL systems”, pp. 117-133, June 2013.

[24] C. Mohan , ”History Repeats Itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla”, EDBT '13 Proceedings of the 16th International Conference on Extending Database Technology, pp.11-16, ACM New York, USA

[25] Yishan Li; Manoharan, S., "A performance comparison of SQL and NoSQL databases," Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on , vol., no., pp.15-19, 27-29 Aug. 2013

[26] Tudorica, B.G.; Bucur, C., "A comparison between several NoSQL databases with comments and notes," Roedunet International Conference (RoEduNet), pp.1-5, 23-25 June 2011

[27] Jing Han, Haihong, E., Guan Le and Jian Du, "Survey on NoSQL database," Pervasive Computing and Applications (ICPCA), 2011 6th International Conference, pp.363-366, 26-28 Oct. 2011

[28] D. Bartholomew, “SQL vs. NoSQL,” Linux Journal, no. 195, July 2010

[29] A. Boicea, F. Radulescu, and L. I. Agapin, “MongoDB vs Oracle –

database comparison,” in Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference, pp. 330-335., sept. 2012

[30] Bhat U. and Jadhav S., “Moving Towards Non-Relational Databases”. International Journal of ComputerApplications, 13 (February 2010)

[31] Kristina Chodorow, Michael Dirolf. “MongoDB: The Definitive Guide”.

O. Reilly Media,September 2010. p135

PUBLICATIONS

[1] Yogesh Punia,Rinkle Aggarwal, “Performance comparison of System using MongoDB-Redis vs System using Relational Database”, Second International Conference on Emerging Research in Computing, Information, Communication and Applications- (ERCICA-14), Elsevier, Banglore, to be held on Aug 1-2, 2014.

Status-Accepted.

[2] Yogesh Punia, Rinkle Aggarwal ,” Implementing Information System using MongoDB and Redis”, International Conference on Advances in Computer Science and Engineering (ICACE 2014), Mysore, pp.16-21, March 10, 2014.1

Status-Published