KADIR HAS UNIVESITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

THE PERFORMANCE –WISE COMPARISON OF THE MOST WIDELY USED NOSQL DATABASES

Ahmed Tallal Aladily

June, 2015

AHMED TALLAL ALADILY

M.Sc. Thesis

2015

THE PERFORMANCE –WISE COMPARISON OF THEMOST WIDELY USED NOSQL DATABASES

Ahmed Tallal Aladily

Submitted to the Graduate School of Science and Engineering

In partial fulfillment of the requirements for the degree of

Master of Science In

Information Technologies

KADIR HAS UNIVERSITY

June, 2015

KADIR HAS UNIVERSITY GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

THE PERFORMANCE –WISE COMPARISON OF TheMOST WIDELY USED NOSQL DATABASES

Ahmed Tallal Aladily

APPROVED BY:

Prof. Dr. Hasan Dağ (Advisor), Kadir Has University ____________________ [illegible]

Asst. Prof. Selçuk A. Öğrenci, Kadir Has University ____________________ [illegible]

Assoc. Prof. Dr. Mehmet .N. Aydın, Kadir Has University ____________________ [illegible]

Acknowledgments

I would like to acknowledge all the guidance and constant support of my thesis advisor Prof. Dr. Hasan Dağ and also I would like to acknowledge my friend and classmate Tayfun Yalçınkaya for his assistant in my research and all the professors who taught me in my study life.

Table of Contents

Abstract

This work deals with the comparison of the most widely used noSQLdatabases. Chapter one deals in great details with the SQL databases and the noSQL databases including characteristics and the four types of noSQL databases, the second Chapter deals with the characteristics of the SQL and noSQL databases and the main differences between SQL databases and the noSQL databases,The third chapter deals with the architecture of the Couchdb, Mongodb, Cassandra, and Hbase.Chapter four deals with installation of the Couchdb, Mongodb, Cassandra and Hbase and Chapter five deals with analysis of the four noSQL databases and it also include the performance wise comparison.

Chapter 1

1. Introduction

Data is growing at a very rapid pace these days. There are many of new business applications that are centered around processing large amount of data, The nature of data is also quite diverse, for example data can now be anything from blogs, photos, tweets, messages and videos etc. so there is a massive need for unstructured database like noSQLdatabases, also in term of big data analytics noSQL is one of the solution that were adopted by companies in IT community that are became more interested in data driven to make their decision depending on data analytics.

1.1 Background

Of many different types of data models the relational model has been choose since the 80s, MySQL, oracle and SQL servers known as (RDBMS) Relational Database Management System, later on however the number of the cases the RDBMS use show problems and deficits in the data modeling, big amounts of data cannot be handled by RDBMS in term of scalability.
There are two main trends that IT and software community should give attention to:(Hossain, 2013).

1. The volume the data is growing exponentially generated by the users, social media, networking, big distributed systems, like Google, LinkedIn, Amazon and other services in the cloud.

2. The variety of data and complexity beside interdependency on the internet, open sources from very large number of different systems.

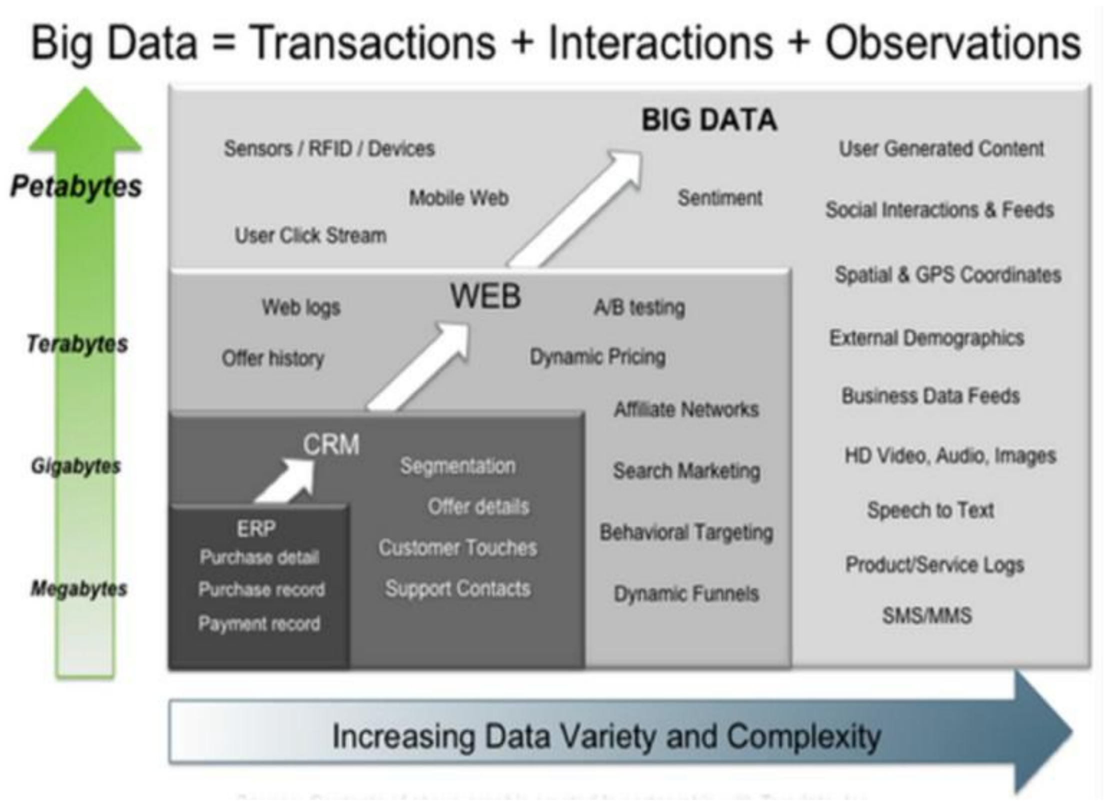


Figure 2: Big Data Concepts(Connolly, 2012).

1.2 noSQLBased Databases

"noSQL, for not Only SQL, refers to an eclectic and increasingly familiar group of non-relationaldata management systems; where databases are not built primarily on tables, and generally do not useSQL for data manipulation. noSQL database management systems are useful when working with a hugequantity of data when the data's nature does not require a relational model. noSQL systems aredistributed, non-relational databases designed for large-scale data storage and for massively-parallel dataprocessing across a large number of commodity servers. They also use non-SQL languages andmechanisms to interact with data (though some new feature APIs that convert SQL queries to thesystems native query language or tool). noSQL database systems arose alongside major Internetcompanies, such as Google, Amazon, and Facebook; which had challenges in dealing with hugequantities of data where conventional RDBMS solutions could not cope.
They can support multiple activities, including exploratory and predictive analytics, ETL-style data transformation, and non-mission-critical OLTP (for example, managing long-duration or inter-organization transactions). Originally motivated by Web 2.0 applications, these systems are designed to scale to thousands or millions of users doing updates as well as reads, in contrast to traditional DBMSs and data warehouses"(Hossain, 2013).

noSQLDatabases and how it started

"Google started off with BigTablewhich is a massive distributed scalable system that effectively operated on Big Data. Looking at this, other companies followed suit by investing heavily in research to create their own custom scalable solutions for their needs"(Narde, 2013).

"The conventional ACID (Atomicity, Consistency, Isolation and Durability) properties were deemed as too regressive and inflexible towards scaling out, so another model namely BASE (Basically available and eventually consistent) came into practice, variousnoSQLsystems were derived from Google's BigTable or Amazons Dynamo"(Narde, 2013), figure(1) below shows the benefits of noSQL databases.

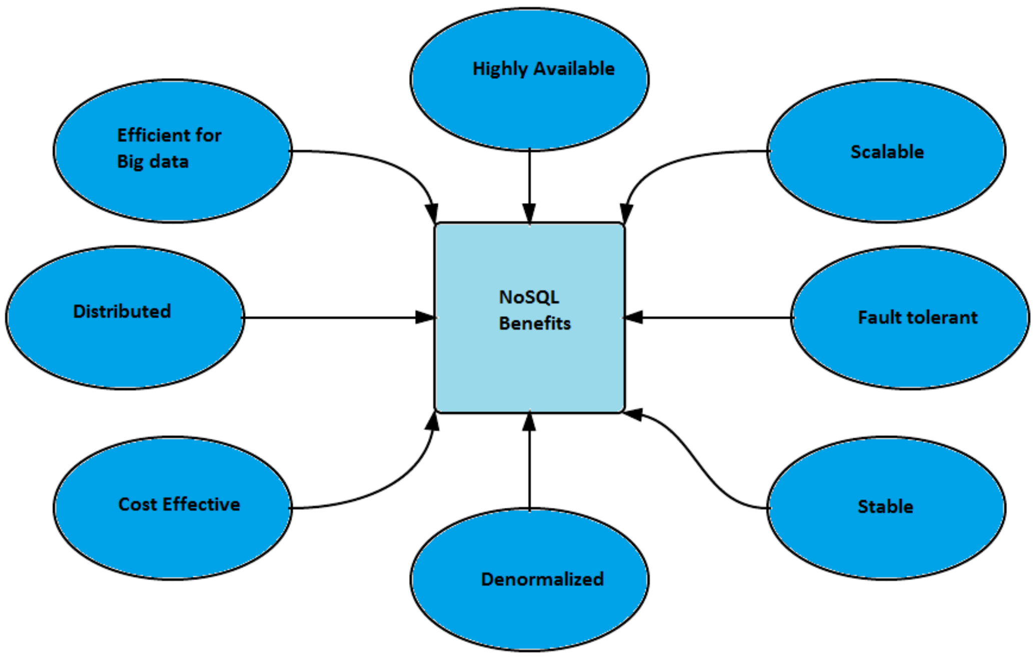


Figure 1: Benefits and advantages of noSQLdatabases(N.Rohan, 2013).

noSQL database are horizontally scalable, transportable, unstructured, low inmaintenance and much faster compare to their relational counter parts, since noSQL databases don’t assign or employ between data relational constraints it employs other different method to store the data.

SQL Characteristics

This section describes the characteristics ofSQL databases:

a) Data is stored in columns and tables.
b) Relationships are represented by objects.
c) There exists a Data Manipulation Language.
d) There exists a Data Definition Language.
e) One is able to conduct Transactions.
f) There is an abstraction from physical layer(Hare, 2011).

SQL Physical Layer Abstraction

This section describes Physical Layer Abstraction:
Applications specify what, not how.
There is aquery optimization engine.
Physical layer can change without modifying applications.
One can easily create indexes to support queries.

Data Manipulation Language (DML)

This section describes Data Manipulation Language:

Manipulation of data with select, update, inserts, and deletes statements.
```
    Select T1.Col1, T2.Col2
     From Tbl1, Tbl2
Where T1.Col1 = T2.Col1 …
```
Datais being aggregated.
procedures and functions using
using compound statements
using transaction control.

Data Definition Language

This section describes Data Definition Language:

The schema is defined at the beginning as shown below:
Create table (col1.datatype1, col2.datatype..)
To define the relationships there should constraints to manage the relationships.
Primary key "first"
Foreign key.
Triggers should be existing to manipulate statements of insert, update and delete.
Models have to be stored.
Alter statements...
Drop statements...
Access control and security.

Transactions – ACID Properties

This section describes ACID Properties briefly:

a) Atomicity:means that all of the tasks in a transaction completesor none of the task completes.
b) Consistency:means that a transaction transforming the databasefrom one specific consistent state to the other consistentstate.
c) Isolation:The results of any changes which ismade during atransaction are not visible until the transaction becamecommitted.
d) Durability:The results of any committed transaction survive failures.

Below are examples of commercial SQL Database:
IBM DB2,MySQL, Microsoft SQL Server, Sybase SQL, Oracle RDMS.
Abig portion of the world's community use SQL relational database management systems, RDBMS.

noSQLDefinition

noSQL databases knows as one of the tools that has been created to manage and analyze unstructured database for example the data of the web2 on the internet, it is unstructured, horizontally scalable, transportable, low in maintenance, non-schema and if we compare it with RDBMS its more faster, since noSQL do not employ between data relationalconstraints, they employ other methods to store the data.

noSQLDatabases Characteristics

The CAP theorem means three basic characters Consistency means that all the clients see current data regardless of updates and deletes, Availability means the system continues to operate as expected even with node failures and Partition Tolerance means that the system continues to operate as expected despite network or message failures. The CAP theorem shows that only two of the three aspects of scaling can be achieved at the same time.

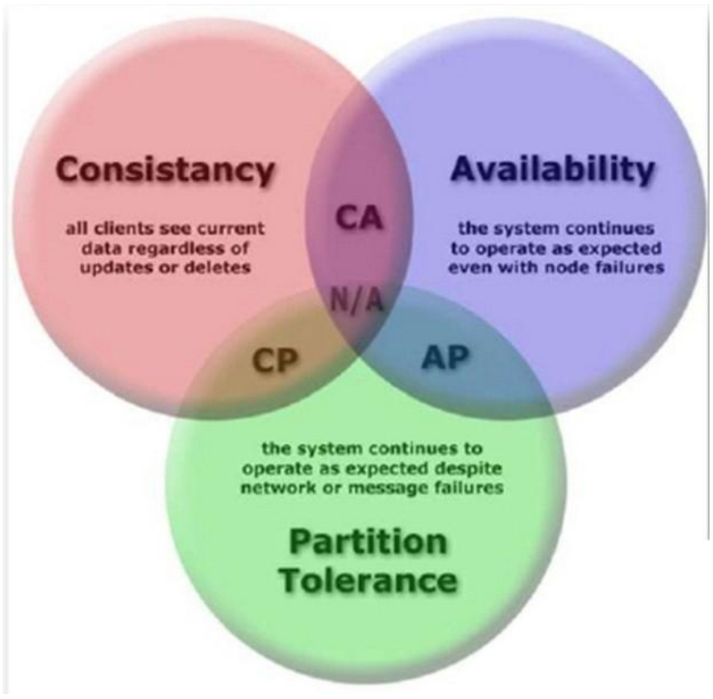

Consistancy
all clients see current data regardless of updates or deletes
Availability
the system continues to operate as expected even with node failures
CA
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance

Figure 3: noSQLDatabase Characteristics(Hossain, 2013).

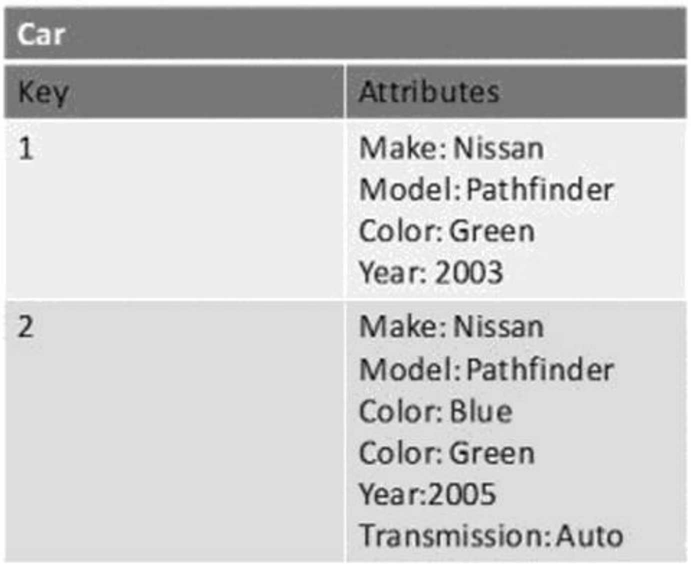
1.3 noSQL Databases Classification:

According to Leavitt N. classification in "2010" the noSQL databases are classified into three types, key value store example on this type SimpleDB (Amazon SimpleDB, 2015),column oriented database example on this type Cassandra (CassandraPro., 2015), Hbase(Apache HBase, 2015), BigTable(Chang F., 2008), and document store example on this type CouchDB(Couchdoc., 2015),MongoDB(MongoDB, 2015). We classify noSQL databases in this research into four basic types:

1) Key-Value stores.
2) Document databases.
3) Wide-Columnor ColumnFamily.
4) Graph databases.

1.3.1 Key Value stores

"Key-Value stores typically, these DMS store items as alpha-numeric identifiers (keys) and associated values in simple, standalone tables (referred to as hash tables). The values may be simple text strings or more complex lists and sets. Data searches can usually only be performed against keys, not values, and are limited to exact matches"(Manoj, 2014).



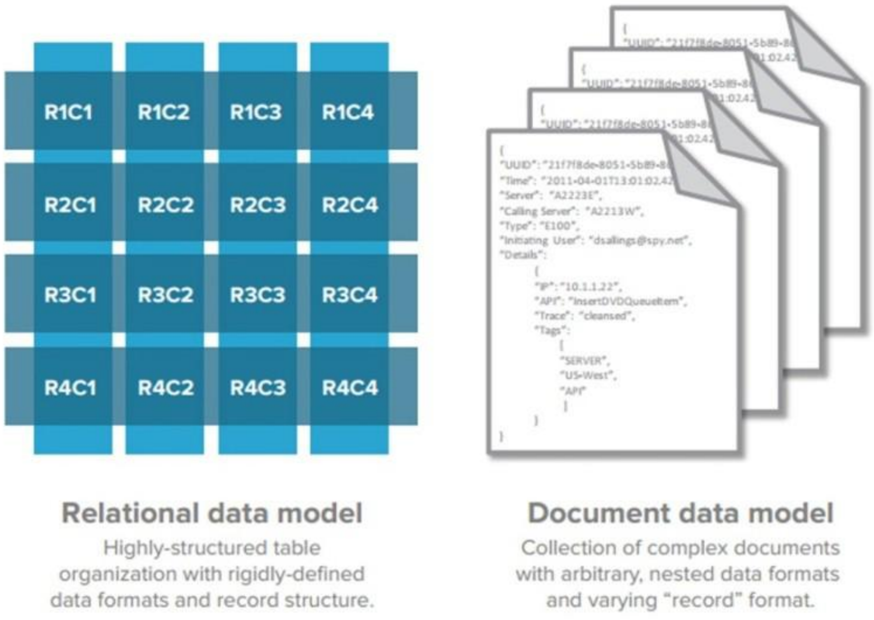
Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Figure4: Key/Value Store(Hossain, 2013)**.**

Because of the simplicity of this type ,its became suited and fast, highly scalable retrieving of the values which is needed for application like manage the user profiles or retrieving names, this is one of the reasons that make Amazon do extra use of its own key value system(G. DeCandia, 2007). Examples of KeyValue StoresDynamo (Amazon), Voldemort (LinkedIn), Redis(Redis, 2015), BerkeleyDB, Riak(Riak, 2015).

1.3.2 Document databases

This type of noSQL databases are designed to manage and store different format of data files like XML and JSON files or BSON files, this type differ from key value store type that column value inside the databases are semi structured and much more complicated than key value store type, single column can have hundreds of same attributes and the number and the type of the attributes can be different in each row inside the database. Examples on this type are CouchDB and MongoDB.


R1C1
R1C2
R1C3
R1C4
R2C1
R2C2
R2C3
R2C4
R3C1
R3C2
R3C3
R3C4
R4C1
R4C2
R4C3
R4C4
Relational data model
Highly-structured table organization with rigidly-defined data formats and record structure.
Document data model
Collection of complex documents with arbitrary, nested data formats and varying "record" format.

```
{
"UUID":"21f7f8de-8051-5b89-8
"Time": "2011-04-01T13:01:02.42
"Server": "A2223E",
"Calling Server": "A2213W",
"Type": "E100",
"Initiating User": "dsallings@spy.net",
"Details":
        {
        "IP":"10.1.1.22",
        "API": "InsertDVDQueueItem",
        "Trace": "cleansed",
        "Tags":
            [
            "SERVER",
            "US-West",
            "API"
            ]
        }
}
```

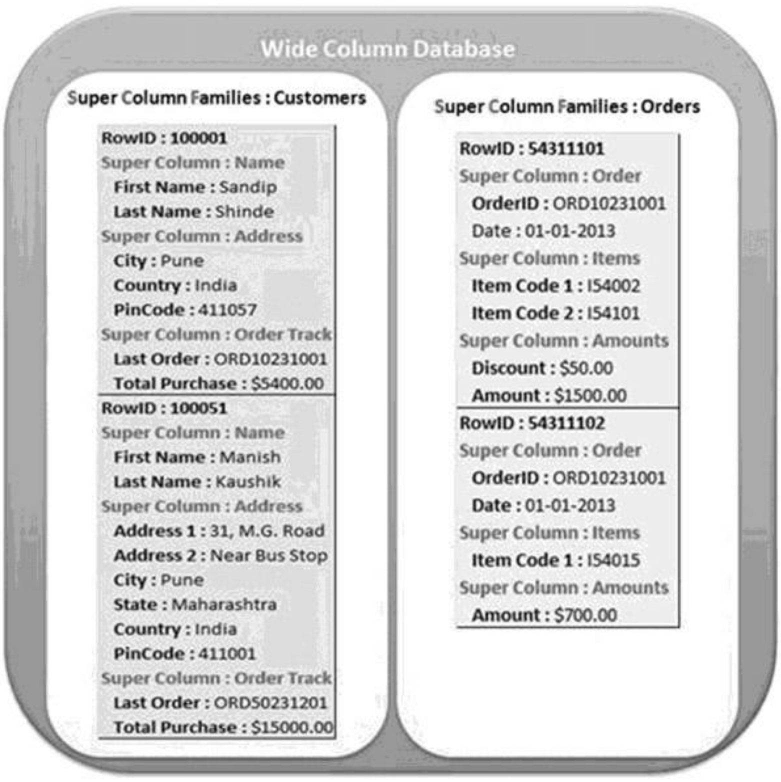
(Harris, 2011)**Figure 5: Document Store noSQL Database**

Document store is good to store big amount of data collections like link in files, email messages , text document,XML documents in addition to conceptual documents, they are also good for storing sparse data generally which is irregular semi structured (Orend, 2010).

1.3.3 Wide-Column or Column FamilyStores

This type of noSQL databases is very similar structure to the classic relational database management system, since all the data are stored in databases in order of sets of columns and rows, the data stored related that is retrieved together may be gathered as a group.

This type is designed to manage huge amount of data complexity, for example Cassandra is on the column family type which is designed to provide salability and high availability, two important examples of this type is Cassandra and Hbase which is open source runs on the top of Hadoop (HDFS).


Wide Column Database

Super Column Families : Customers

RowID : 100001
Super Column : Name
First Name : Sandip
Last Name : Shinde
Super Column : Address
City : Pune
Country : India
PinCode : 411057
Super Column : Order Track
Last Order : ORD10231001
Total Purchase : $5400.00
RowID : 100051
Super Column : Name
First Name : Manish
Last Name : Kaushik
Super Column : Address
Address 1 : 31, M.G. Road
Address 2 : Near Bus Stop
City : Pune
State : Maharashtra
Country : India
PinCode : 411001
Super Column : Order Track
Last Order : ORD50231201
Total Purchase : $15000.00

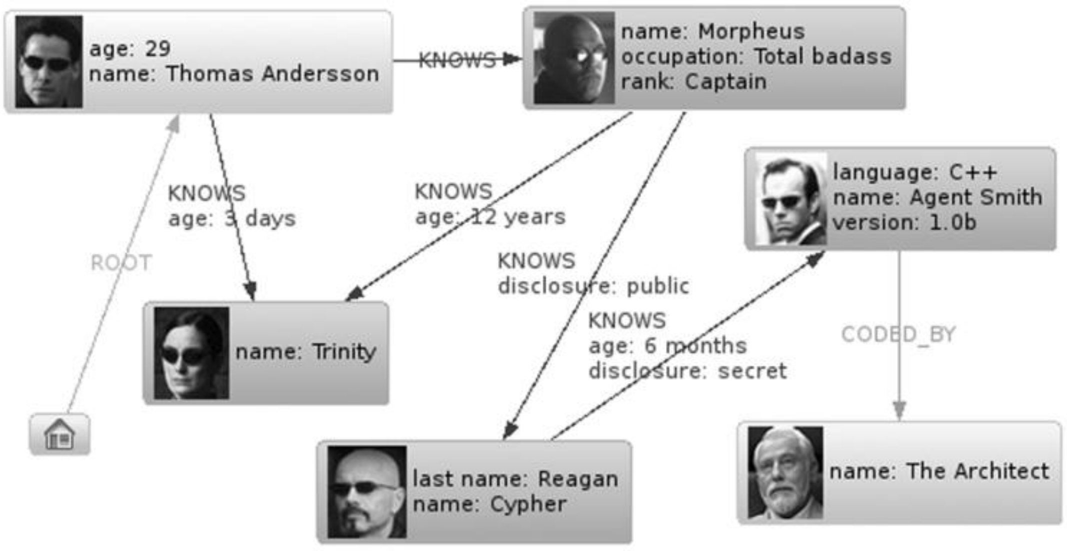
Super Column Families : Orders

RowID : 54311101
Super Column : Order
OrderID : ORD10231001
Date : 01-01-2013
Super Column : Items
Item Code 1 : I54002
Item Code 2 : I54101
Super Column : Amounts
Discount : $50.00
Amount : $1500.00
RowID : 54311102
Super Column : Order
OrderID : ORD10231001
Date : 01-01-2013
Super Column : Items
Item Code 1 : I54015
Super Column : Amounts
Amount : $700.00

(Shinde, 2015)**Figure 6: WideColumn Store**

1.3.4 Graph Databases

In this type of noSQL database system relational tables are represented as structure graphs, and it is similar to the object oriented databases or object oriented network, it focuses on the visual represent of information and make it more friendly to the other noSQL databases.



(Staff, 2010)**Figure 7: Graph Database**

Generally graph databases became very useful when the interest is in the relationship between the data more than the data itself, as example social networks, Neo4j and InfoGrid are two examples of graph databases.

Chapter 2

2. Characteristics of both SQL and noSQL databases

2.1 Differences between RDBMS and noSQL

The main difference of noSQL against SQL databases is noSQL is unstructured databases horizontally scalable, highly available,denormalized as we know when designing the structure of the SQL database at the end stage we normalize it, exactly the reverse content of the noSQL which is denormalized, According to big data prospective noSQL is much more efficient and productive than SQL, and SQL is much suitable for the variety types of data, since every noSQL database is designed or created for a specific task,noSQL databases is much more cost effective and efficient than SQL, noSQL databases is distributed and highly available not as SQL.
The amount of data and the distribution databases system is very important issue for choosing noSQL databases instead of other relational databases (Kopp, 2011).

2.2 When to choose RDBMS

Using noSQL databases for such conditions like small or not huge amount of data is not a good way because when the amount of data are small like bank databases or workshop databases which is properly structured, SQL databases is quite enough to handle the data, Actually there is some properties or features that is better to choose SQL databases instead of noSQL databases: ACID properties, joins and queries, integrity, table based, relations between tables.

2.3 When and why we choose noSQL

According to the issues below we can notice when to choose noSQL databases as a solution:

1) Matter of size: when we have very large amount of data sets, consistency and scalability is much easier to achieve.
2) Speed issue: usually noSQL databases work faster sometimes extremely faster read and writes operation very fast depending on the data being queried.
3) Schema design: noSQL database as a solution shows very big flexibly.
4) Easy replication: noSQL databases are growing very rapidly, and can easily being scaled and work with clusters.
5) Multiple options (choices): the variety of noSQL data models depending on the type of data.

noSQLscale andRDBMS does not scale.

Usually the system scale in two ways one of them is the nodes is upgraded, the other way is adding more nodes, the system should be capable to handle these two conditions, the system performance should increase in linear way, according to RDBMS prospective the limitation in the scalability ofRDBMS, replication and data sharding are not the suitable choice to achieve scalability in RDBMS, relational databases cannot achieve the general data sharding because it requires distinct entities of data, in relational database it is very difficult to achieve the reason why because its table based nature.

Table based property of relational database make them the proper choice to ensure ACID(Atomicity, Consistency, Integrity, Durability) features or properties, comparing to noSQL database can easily distribute the rows over the nodes because of the table less schema and each entity (record) is not distributed over multiple nodes rather the whole entity (record) is stored in a single node(Kopp, 2011).

Why choosing noSQL ?

For the last few decades the big companies like Amazon, Facebook and link-in who are depending on the concept of "Data Driven" which means that decisions they make depend on the customer data analytics, in addition to that the huge growth of the customer data and the complexity of the data (text, audio, video, login files, maps, doc,etc.) shows there is a massive need to a new tools or systems that deals with this huge amount of unstructured data.

For example Cassandra noSQL database was adopted by Facebook company, and Voldemort noSQL database was adopted by link in company, However when creating and designing the structure of these systems, being open sourced and assigned to a specific task for example CouchDB which is apache noSQL database is designed to analyses web application data using Jason file format, Hbase for example designed to run the much complexity data on the internet like data of web2.0, in addition to what we describe above the noSQL database system can be applied using the cloud(CouchBase, 2015).

Chapter 3

3. Architectures of noSQL databases

3.1 MongoDB

MongoDB is non-schema noSQL database open source document store or database, which provides high availability, high performance and scalability.

Document format

Each record in MongoDB is represented as document consist of field and value pairs, the document of MongoDB are known as objects, fields values may include any other document, arrays, arrays of documents.

```
{
name: "sue"                        ← filed:value
age   : 26                           ← filed:value
status: "A"              ← filed:value
groups: [ "news", " sports"] ← filed:value

}
```

The benefits of using the documents are:
• In programming languages documents are correspond to native type.
• Reducing the need expensive joins because of having embedded documents and arrays.
• Fluent polymorphismare supported by dynamic schema.

Key Features

High Performance

MongoDBprovidedata persistence and high performance as follows,
• reducing the input/output activity on the database system by supporting for embedded data models.
• in embedded documents and arrays indexes always support faster queries and can include keys.

High Availability

To supply or achieve high availability there is for MongoDB facility of replication, which is called replica sets. Also provide:
• Failoverautomatically.
• Redundancy of data.
Replica set in MongoDB is a group of servers that manipulate or maintain the same data set by increasing data availability and providing redundancy.

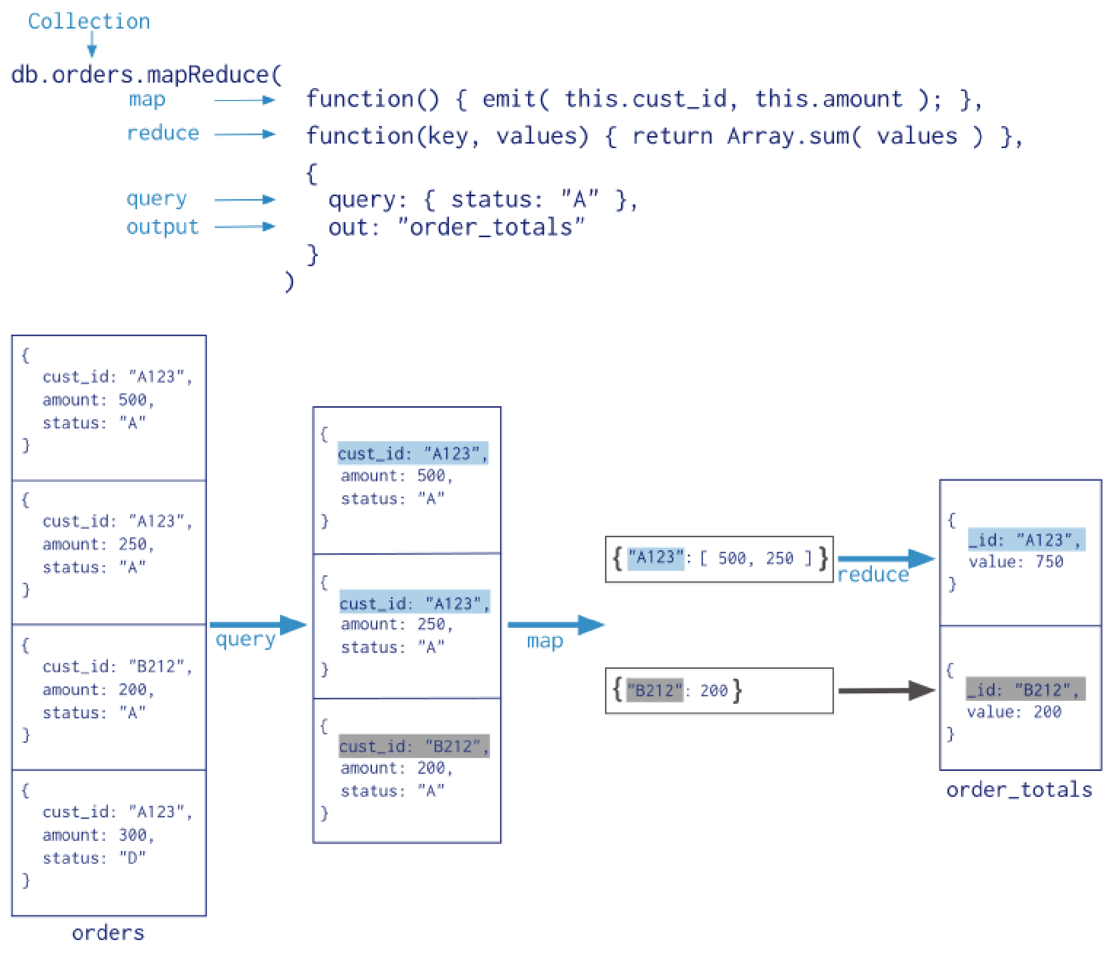
Automatically Scaling

MongoDBalso provides horizontal scalability as one of its basic functionality.
• Sharding is distributing the data across the cluster of machines automatically.
• Eventually consistent reads and low latency throughput high deployments can be provided by replica sets.

Map Reduce

Map reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. For map-reduce operations, MongoDBprovides the Map Reduce database command.

The figure below shows the map-reduce operation:



```
db.orders.mapReduce(
                       function() { emit( this.cust_id, this.amount ); },
                       function(key, values) { return Array.sum( values ) },
                       {
                         query: { status: "A" },
                         out: "order_totals"
                       }
                     )
```

Collection
map
reduce
query
output
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
query
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
map
{ "A123": [500, 250] }
{ "B212": 200 }
reduce
{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
order_totals

Figure(8) Map Reduce operation (MongoDB, 2015).

"In this map-reduce operation, MongoDBapplies the map phase to each input document (i.e. the documents in the collection that match the query condition). The map function emits key-value pairs. For those keys that have multiple values, MongoDBapplies the reduce phase, which collects and condenses the aggregated data. MongoDBthen stores the results in a collection. Optionally, the output of the reduce function may pass through a finalize function to further condense or process the results of the aggregation.

The entire map-reduce functions in MongoDBare javascripts and run within the MongoDBprocess.

Map-reduce operations take the documents of a single collection asthe input and can perform any arbitrary sorting and limitingbefore beginning the map stage. MapReduce can return the results of a map-reduce operation as a document, or may write the results to collections. The input and the output collections may besharded"(MongoDBInc, 2015).

Map-Reduce JavaScript Functions

"In MongoDB, map-reduce operations use custom JavaScript functions to map, or associate, values to a key. If a key has multiple values mapped to it, the operationreduces the values for the key to a single object.

The use of custom JavaScript functions provides flexibility to map-reduce operations. For instance, when processing a document, the map function can create more than one key and value mapping or no mapping. Map-reduce operations can also use a custom JavaScript function to make final modifications to the results at the end of the map and reduce operation, such as perform additional calculations"(MongoDB, 2015).

3.2 CouchDB

CouchDB database is no schema, and semi structured data model based on JSON, materialized views structure produced from document collections, the views with the map reduce method allowing bothincremental maintenance and parallelcomputation of their content, data management distributed techniques:horizontal scalability,consistent hashing,parallelcomputing,data replication and reconciliation etc.(Abiteboul, 2011).

Data model

JSON file is a text file format, which is designed for serializing java scripts objects,
Basic use: in a web environment data exchange.
Extend use: storage of data and serialization XML document is very easy to integrate to any programming language with minimum parsing effort, in addition to that no schema, no query language at the moment.

Core: keyvalue construct

Example:
"title": "The Social network"
Atomic data types: character strings, integers, floating-point number and Booleans (true or false). Non-string values need not be surrounded by ’"’.
"year": 2010

Complex values: objects

An unordered set of name /value pairs called as objects, the types of objects can be distinct , also the key appearance only once, each object can be used as value component of a key value construct.

```
"director": {
"last_name": "Fincher",
"first_name": "David",
"birth_date": 1962
}
```

Complex values: arrays
An ordered collection of values that contain different data types are known as arrays in these values.
"actors": ["Eisenberg", "Mara", "Garfield", "Timberlake"]

A document here is the object which can be represented with a nesting of array and object constraints.

```
{
"title": "The Social network",
"year": "2010",
"director": {"last_name": "Fincher",
"first_name": "David"},
"actors": [
{"first_name": "Jesse", "last_name": "Eisenberg"},
{"first_name": "Rooney", "last_name": "Mara"}
}
```

3.3 Cassandra

"Apache Cassandra in a nutshell is an open source, peer to peer distributed database architecture, decentralized, easily scalable, fault tolerant, highly available, eventually consistent, schema free, column oriented database. Generally in a master/slave setup, the master node can have far reaching effects if it goes offline. By contrast, Cassandra has a peer-to-peer distribution model, such that any given node is structurally identical to any other node—that is, there is no "master" node that acts differently than a "slave" node. The aim of Cassandra's design is overall system availability and ease of scaling. Cassandra data model comprises of Key space (something like a database in relational databases) and column families (tables). Cassandra defines a column family to be a logical division that associates similar data. Basic Cassandra data structures: The column, which is a name/value pair and a client-supplied timestamp of when it was last, updated, and a column family, which is a container for rows that have similar, but not identical, column sets"(Aircc.Corporation, 2015).
"There is no need to store a value for every column every time a new entity is stored. For example, column family data model,A cluster is a container for key spaces typicallya single key space. A key space is the outermost container for data in Cassandra, but it's perfectly fine to create as many keyspaces as the application needs. A column family is a container for an ordered collection of rows, each of which is itself an ordered collection of columns"(Manoj, 2014).

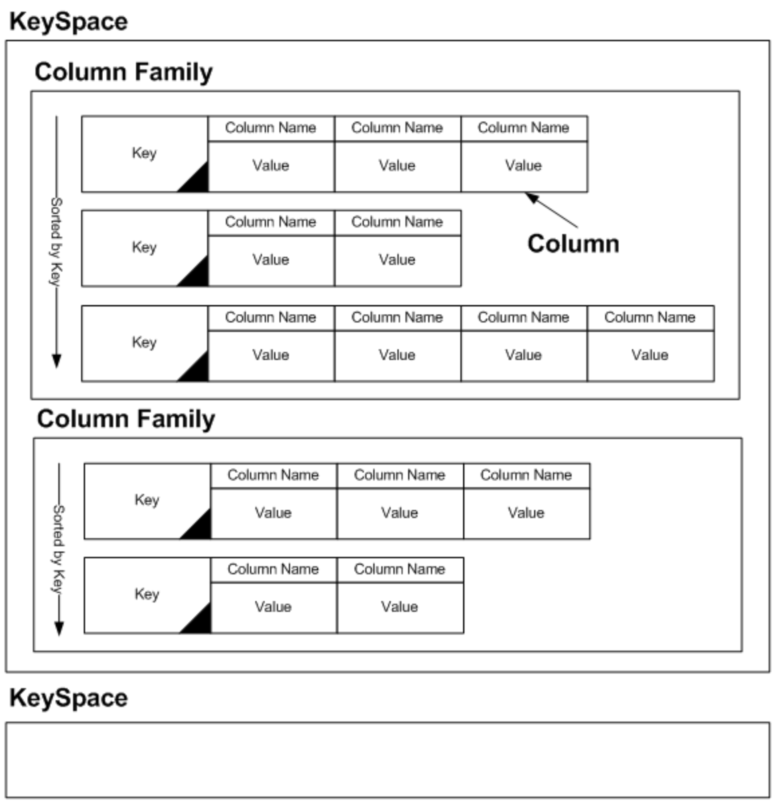


Figure: 9Cassandra data Model(Hewitt, 2010).

3.4 Hbase

"Apache HBase is an open source, non-relational, persistent, strictly consistent fault tolerantdistributed database runs on top of HDFS (Hadoop Distributed File System) modeled afterGoogle's Big table providing the capabilities on Hadoop. Hbase is a Master/Slave approachcomprising of one master server and many region servers where the master node is responsiblefor assigning or load balancing across region servers. Region servers are slaves like responsiblefor all read and write requests for all regions they serve, and also split regions that have exceededthe configured region size thresholds. The store files are typically saved in the HadoopDistributed File System (HDFS), which provides a scalable, persistent, replicated storage layer
for HBase. It guarantees that data is never lost by writing the changes across a configurablenumber of physical servers" (Manoj, 2014).
"Hbase follows Big table data model, a sparse, distributed, persistent multidimensional sortedmap. The map is indexed by a row key, column key, and a timestamp; each value in the map is anuninterrupted array of bytes. Hbase data model is column oriented storage structure typicallygrouped into one or more tables. Row keys in table are arbitrary strings; Bitable maintains datain lexicographic order by row key. The row range for a table is dynamically partitioned. Columnkeys are grouped into sets called column families, which form the basic unit of access control.
Alldata stored in a column family is usually of the same type. A column family must be createdbefore data can be stored under any column key in that family each cell in a BigTable cancontain multiple versions of the same data; these versions are indexed by 64-bit integerstimestamp. Although conceptually a table is a collection of rows with columns in HBase,physically they are stored in separate partitions called regions. Every region is served by exactlyone region server, which in turn serves the stored values directly to clients"(Manoj, 2014).

Chapter 4

4. Implementation

4.1 Installation CouchDB on windows:

Usually there are methodsfor installingCouchDB on windows system.

Installation from binaries

Get the latest Windows binaries from CouchDB website.
This is the simplest way to goto the old release and it's available at archive.
"We have to follow the installation steps:
a)Next on “Welcome” screen.
b) Accept the License agreement.
c)Select the installation directory.
d)Specify “Start Menu” group name."(CouchDocument, 2015)
After that approve like installing CouchDB as service and let it started automatically after installation.
(Probably, you should see the statement)
."After that verify installation settings
.After thatInstall CouchDB
Open Futon (if you didn't selectauto start CouchDB after installation you have to start it as manual" (CouchDocument, 2015).
"**Note:**in some cases you might be asked to reboot Windows to complete installation process, because of using on different Microsoft Visual C++ runtimes by CouchDB" (CouchDocument, 2015).

4.1.1 Installation CouchDB on Unix:

Dependencies

"You should have the following installed:
Erlang OTP (>=R14B01, =<R17)
ICU
OpenSSL
Mozilla SpiderMonkey (1.8.5)
GNU Make
GNU Compiler Collection
libcurl
help2man
Python (>=2.7) for docs
Python Sphinx (>=1.1.3)"(CouchDBLearn, CouchDB Learing, 2014).
After satisfying the dependences you have to run the configuration for it, the default CouchDB be installed into usr/local the script above will set it up, for customizing the installation write pass-help to this script,
If everything was done successfully you should see on the screen the following message:
We have configured Apache CouchDB.

To install CouchDB you should run:

"Make && sudo make install we only need to use sudo if we’re installing into a system directory.
Try gmake if make is giving you any problems.
If everything was successful we should see the following message.
we have installed Apache CouchDB.

First Run

You can start the CouchDB server by running:
sudo -i -u CouchDBCouchDB
This uses the sudo command to run the Couchdb command as the Couchdb user.
When CouchDB starts it should eventually display the following message:
Apache CouchDB has started.
To check that everything has worked, point your web browser to :

http://127.0.0.1:5984/_utils/index.html

From here you should verify your installation by pointing your web browser to:

http://localhost:5984/_utils/verify_install.html"(CouchDBLearn, 2014).

4.2 Install MongoDBon Linux:

"Restart MongoDByou can restart the mongod process by issuing the following command (as root or with sudo):
service mongod restart"(MongoDBDoc., 2015)

"Follow the state of this process by watching the output in the /var/log/mongo/mongod.log file to watch forerrors or important messages from the server" (MongoDBDoc., 2015)

"Install MongoDBon Ubuntu This tutorial outlines the steps to install MongoDBon Ubuntu Linux systems. The tutorial uses deb packages to install. Although Ubuntu include its own MongoDBpackages, the official MongoDBpackages are generally more up to date" (MongoDBDoc., 2015).

"Note: If you use an older Ubuntu that does not use Upstart, (i.e. any version before 9.10 “Karmic”) please follow thePackage Options TheMongoDBdownloads repository provides the mongodb-10gen package, which contains the latest stable release. Additionally you can install previous releases of MongoDB" (MongoDBDoc., 2015).

"You cannot install this package concurrently with the MongoDB, MongoDB-server, or MongoDB-clients packages provided by Ubuntu"(MongoInstall, 2015).

Install MongoDB

"Configure Package Management System (APT) The Ubuntu package management tool (i.e. dpkg and apt).
ensure package consistency and authenticity by requiring that distributors sign packages with GPG keys.Issue the following command to import the MongoDBpublic GPG Key1" (MongoInstall, 2015)

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 recv 7F0CEB10
```

"Create a /etc/apt/sources.list.d/mongodb.list file using the following command,

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list" (MongoInstall, 2015)
```

Now write the following command to reload your repository:

```
sudo apt-get update
```

"Install Packages Issue the following command to install the latest stable version of MongoDB:

```
sudo apt-get install mongodb-10gen
```

When this command completes, you have successfully installed MongoDBContinue(MongoInstall, 2015)for configuration and start-upSuggestions"

"Manage Installed Versions You can use the mongodb-10gen package to installprevious versions of MongoDB.
To install a specific release, append the version number to the package name, as in the following example:(MongoInstall, 2015)"

```
apt-get install mongodb-10gen=2.2.3
```

"This will install the 2.2.3 release of MongoDB. You can specify any available version of MongoDB; however:

apt-get will upgrade the mongodb-10gen package when a newer version becomes available. Use the following pinning procedure to prevent unintended upgrade, to pin a package, issue the following command at the system prompt to pin the version of MongoDBat the currently installed version" (MongoInstall, 2015).

"echo "mongodb-10gen hold" | sudo dpkg --set-selections" (MongoInstall, 2015).

4.3 Installing Cassandra on Linux:

"Check that Java is installed by running the following command in a terminal window:
java -version
Note:Any version of Java is fine for installing an evaluation instance. For production deployments, see InstallingOracle JRE.
Download the Cassandra package (required), and the optional OpsCenter.

To get the latest versions of DataStax and OpsCenter Community Editions and the Portfolio Demo"(DataStax, 2014):

"$ curl -OL http://downloads.datastax.com/community/dsc-cassandra-1.0.9-bin.tar.gz
$ curl -OL http://downloads.datastax.com/community/opscenter.tar.gz
Unpack the distributions
$ tar -xzvf dsc-cassandra-1.0.9-bin.tar.gz
$ tar -xzvf opscenter.tar.gz
(DataStaxTut., 2015)$ rm *.tar.gz "

4.4 Installing Hbase:

"This chapter explains how HBase is installed and initially configured. Java and Hadoop are required to proceed with HBase, so you have to download and install java and Hadoop in your system" (Hbasetutorials, 2015).

"Pre-Installation Setup

Before installing Hadoop into Linux environment, we need to set up Linux using ssh (Secure Shell). Follow the steps given below for setting up the Linux environment" (Hbasetutorials, 2015).

"Creating a User

First of all, it is recommended to create a separate user for Hadoop to isolate the Hadoop file system from the Unix file system. Follow the steps given below to create a user"(Hbasetutorials, 2015).

- "Open the root using the command su"(Hbasetutorials, 2015).
- "Create a user from the root account using the command user add username " (Hbasetutorials, 2015).
- "Now you can open an existing user account using the command suusername." (Hbasetutorials, 2015)

"Open the Linux terminal and type the following commands to create a user." (Hbasetutorials, 2015)

```
 "$ su
password:
# useraddHadoop
# passwdHadoop
New passwd:
Retype new passwd" (Hbasetutorials, 2015)
```

"Setup of SSH and Key Generation

SSH setup is required to perform different operations on the cluster such as start, stop, and distributed daemon shell operations. To authenticate different users of Hadoop, it is required to provide public/private key pair for a Hadoop user and share it with different users" (Hbasetutorials, 2015).

"The following commands are used to generate a key value pair using SSH. Copy the public keys form id_rsa.pub to authorized_keys, and provide owner, read and write permissions to authorized_keys file respectively"(Hbasetutorials, 2015).

```
"$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys " (Hbasetutorials, 2015)
```

"Verify ssh

```
sshlocalhost" (Hbasetutorials, 2015)
```

"Installing Java

Java is the main prerequisite for Hadoop and HBase. First of all, you should verify the existence of java in your system using java -version. The syntax of java version command is given below" (Hbasetutorials, 2015).

```
"$ java –version " (Hbasetutorials, 2015)
```

"If everything works fine, it will give you the following output" (Hbasetutorials, 2015).

```
"java version 1.7.0_71
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)" (Hbasetutorials, 2015)
```

"If java is not installed in your system, then follow the steps given below for installing java" (Hbasetutorials, 2015).

"Step 1

Download java (JDK <latest version> - X64.tar.gz) by visiting the following link Oracle Java.

Then jdk-7u71-linux-x64.tar.gz will be downloaded into your system" (Hbasetutorials, 2015).

"Step 2

Generally you will find the downloaded java file in Downloads folder. Verify it and extract the jdk-7u71-linux-x64.gz file using the following commands"(Hbasetutorials, 2015).

```
"$ cd Downloads/
$ ls
jdk-7u71-linux-x64.gz
$ tar zxf jdk-7u71-linux-x64.gz
$ ls
jdk1.7.0_71 jdk-7u71-linux-x64.gz" (Hbasetutorials, 2015)
```

"Step 3

To make java available to all the users, you have to move it to the location /usr/local/. Open root and type the following commands"(Hbasetutorials, 2015).

```
"$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit" (Hbasetutorials, 2015)
```

"Step 4

For setting up PATH and JAVA_HOME variables, add the following commands to ~/.bashrc file" (Hbasetutorials, 2015).

```
"export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH= $PATH:$JAVA_HOME/bin" (Hbasetutorials, 2015)
```

"Now apply all the changes into the current running system"(Hbasetutorials, 2015).

```
"$ source ~/.bashrc " (Hbasetutorials, 2015)
```

"Step 5

Use the following commands to configure java alternatives"(Hbasetutorials, 2015):

```
"# alternatives --install /usr/bin/java java usr/local/java/bin/java 2

# alternatives --install /usr/bin/javac javac usr/local/java/bin/javac 2

# alternatives --install /usr/bin/jar jar usr/local/java/bin/jar 2

# alternatives --set java usr/local/java/bin/java

# alternatives --set javac usr/local/java/bin/javac

# alternatives --set jar usr/local/java/bin/jar" (Hbasetutorials, 2015)
```

"Now verify the java -version command from the terminal as explained above"(Hbasetutorials, 2015).

"Downloading Hadoop

After installing java, you have to install Hadoop. First of all, verify the existence ofHadoop using Hadoopversion command as shown below"(Hbasetutorials, 2015).

```
"Hadoopversion "(Hbasetutorials, 2015)
```

"If everything works fine, it will give you the following output" (Hbasetutorials, 2015).

```
"Hadoop 2.6.0
Compiled by jenkins on 2014-11-13T21:10Z
Compiled with protoc 2.5.0
From source with checksum 18e43357c8f927c0695f1e9522859d6a
This command was run using
/home/Hadoop/Hadoop/share/Hadoop/common/Hadoop-common-2.6.0.jar" (Hbasetutorials,
2015)
```

"If your system is unable to locate Hadoop, then download Hadoop in your system.Follow the commands given below to do so.

Download and extract Hadoop-2.6.0 from Apache Software Foundation using thefollowing commands"(Hbasetutorials, 2015).

```
"$ su
password:
# cd /usr/local
# wget http://mirrors.advancedhosters.com/apache/Hadoop/common/Hadoop-
2.6.0/Hadoop-2.6.0-src.tar.gz
# tarxzfHadoop-2.6.0-src.tar.gz
# mvHadoop-2.6.0/* Hadoop/
# exit" (Hbasetutorials, 2015)
```

Chapter 5

5. Analysis

In this chapter we show the Comparison in Data model, query model, replication model between the noSQL databases and also the performance of the noSQLdatabases:

5.1 MongoDB

"MongoDBis an open-source document-oriented database written in C++ and is completely schema-free and manages JSON-style documents. It focuses on high-performance providing the developer with a set of features to easily model and query data"(AbertoAcademy., 2008).

a) Data model

"MongoDBstores data as BSON objects, which is a binary-encoded serialization of JSON-like documents. It supportsall the data types that are part of JSON but also defines new data types, i.e. the Date data type and the Big Data type"(AbertoAcademy., 2008)"and also support all the data type that is the part of Jason. The key advantage of using BSON is efficiency as it is abinary format"(ResGate, 2015). "Documents are contained in collections, they can be seen as an equivalent to relational databasetables. Collections can contain any kind of document, no relationship is enforced, and still documents within acollection usually have the same structure as it provides a logical way to organize data. As data within collections isusually contiguous on disk, if collections are smaller better performance is achieved. Each document is identifiedby a unique ID (−id field), which can be given by the user upon document creating orautomatically generated bythe database"(ResGate, 2015). "An index is automatically created on the ID field although other indexes can be manually createdin order to speed up common queries. Relationships can be modeled in two different ways embedding documents orreferencing documents. Embedding documents means that a document might contain other data fields related to thedocument, i.e. a document modeling a blog post would also contain the post's comments. This option might lead tode-normalization of the database, as the same data might be embedded in different documents. Referencing documentscan be seen as the relational database equivalent of using a foreign-key. Instead of embedding the whole data, thedocument might instead store the ID of the foreign document so that it can fetch. It is important to note thatMongoDBdoes not provide the ability to join documents, therefore when referencing documents, any necessary join has tobe done on the client-side"(AbertoAcademy., 2008).

b) Query model

"Many traditional SQL queries have a similar counterpart on MongoDB's query Language. Queries are expressedas JSON objects and are sent to MongoDBby the database driver (typically using the find method)"(ResGate, 2015). "Morecomplex queries can be expressed using a Map Reduce operation, and it may be useful for batch processing of dataand aggregation operations. The user specifies the map and reduces functions in JavaScript and they are executed onthe server side. The results of theoperation are stored in a temporary collection, which is automatically removedafter the client gets the results. It is also possible for the results to be stored in a permanent collection, so that they arealways available" (ResGate, 2015).

c) Replication model

"MongoDBprovides Master-Slave replication and Replica sets, where data is asynchronously replicated betweenservers. In either case only one server is used for write operations at a given time, while read operations can be redirectedto slave servers. Replica sets are an extension of the popular Master-Slave replication scheme in order toprovide automatic failover and automatic recovery of member nodes (MongoRep, 2011). A replicate set is a group of servers where atany point in time there is only one master server, but the set is able to elect a new master if the current one goes down.

Data is replicated among all the servers from the set. Figure (10) illustrates the two different replication models. SinceMongoDBhas only one single active master at any point in time, strong consistency can be achieved if allthe read operations are done on the master " (AbertoAcademy., 2008).

"MongoDBdoes not provide version concurrency control; reads from the slave servers employ eventual consistencysemantics. The ability to read from slave servers is usually desired to achieve load balance, therefore the client is alsoable to enforce

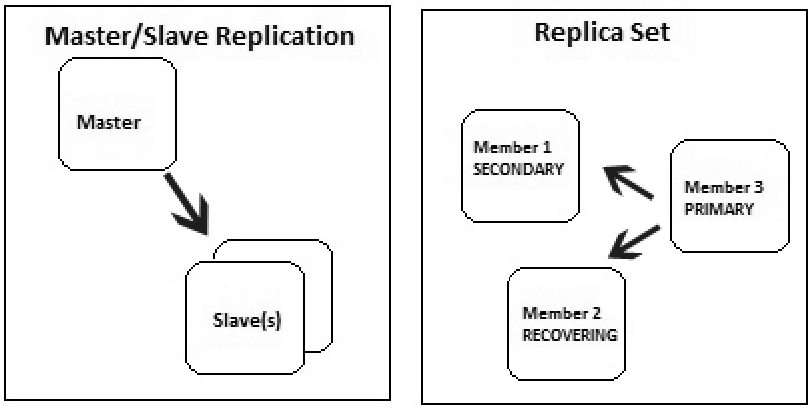


Figure:(10) Illustration of master-slave replication(AbertoAcademy., 2008).

that a certain write has replicated to a certain number of slave servers. This feature helps dealing withimportant writes, where eventual consistency semantics might not be suitable, while at the same time providing theflexibility to read from slave servers" (AbertoAcademy., 2008).

5.2 CouchDB

Opensource document database and which written in Erlang. It concern with the ACID properties, providing much more serializability.

a) Data model

"Data is stored as semi-structured documents in CouchDB. A document is a JSON file which is a collection of namedkey-value pairs. Values can be numbers, string, Booleans, lists or dictionaries. Documents are not bound to follow anystructure and can be schema-free. Each document CouchDB database is identified by a unique ID (the –id field).
CouchDB is a simple container of a collection of documents and it does not establish any mandatory relationshipbetween them" (AbertoAcademy., 2008).

b) Query model

"CouchDB exposes a Restful HTTP API to perform basic CRUD operations on all stored items and it uses the HTTPmethods POST, GET, PUT and DELETE to do so. More complex queries can be implemented in the form of views(as was seen before) and the result of these views can also be read using the REST API"(AbertoAcademy., 2008).

c) Replication model

"CouchDB is a peer-based distributed database system (CouchIntro, 2011) it allows peers to update and change data and then directionallysynchronizes the changes. Therefore, we can model either master-slave setups (were synchronizations areunidirectional) or master-master setups were changes can happen in either of the nodes and they must be synchronizedin a bidirectional way.
Each document is assigned a revision id and every time a document is updated, the old version is kept and theupdated version is given a different revision id. Whenever a conflict is detected, the winning version is saved as themost recent version and the losing version is also saved in the document’s history. This is done consistently throughoutall the nodes so that the exact same choices are made. The application can then choose to handle the conflict by itself(ignoring one version or merging the changes)" (AbertoAcademy., 2008).

d) Consistency model

"CouchDB provides eventual consistency. As multiple masters are allowed, changes need to be propagated to theremaining nodes, and the database does not lock on writes. Therefore, until the changes are propagated from node tonode the database remains in an inconsistent state. Still, single master setups (with multiple slaves) are also supported,and in this case strong consistency can be achieved"(AbertoAcademy., 2008).

5.3 Cassandra

Apache Cassandra is a free, open-source distributed, structured key-value store with eventual consistency. It is developed by Facebook. While providing high availability and scalability Cassandra was designed tohandle very huge amounts of data,(CassandraPro., 2015).

a) Data model

"A typical instance of Cassandra consists of only one table with a distributed multidimensional map indexed by a keyand tables. Rows are generally identified by a string-key and row operations are atomic even if there are manyColumns being read or written. As in the big table, column-families have to be defined in advance, that is before acluster of servers comprising a Cassandra instance is launched. The number of column-families per table is infinitehowever it is expected to specify only few of them. A column family consists of columns and super columns, whichcan be added dynamically (i.e. at runtime) to column-families and are not restricted in number.Columns have aname and store a number of values per row, which are identified by a timestamp (as in Big table). Every row in a tablehas different number of columns. Client specific applications may use ordering of columns within a column family inwhich name and timestamp can be considered as super-column. Super-columns have a name and number of columnsassociated with them randomly" (AbertoAcademy., 2008).

b) Query model

"Since Cassandra is in the essence of a key-value store, the querying model provides only three simple operations.

A key can access all the values and return the column value associated with that key. Additional methods for gettingmultiple values or getting multiple columns are also available. But they all rely on the key as input. Additionally it is also possible to run Map Reduce jobs on the database, in order to process complex queries. This can be done byusing Apache Hadoop, which is a framework for distributed data processing. To simplify this task, Cassandra alsoprovides Pig (Hadoop, 2011) support that enables the user to write complex Map Reduce jobs in a high-level language similar to that of SQL. These queries are generally used as an offline method for the analysis of data" (AbertoAcademy., 2008).

c) Replication model

"Since Cassandra is distributed in nature, data is partitioned over a set of nodes and each data item is assigned to aspecific node. Data items are assigned to a position by hashing the data item's key, using consistent hashing.

Nodes are also assigned a random value within the hashing's space, which represents its position on the ring. To findwhich node is the coordinator for a given data item, it is necessary to go through the ring in a clockwise manner, andfind the first node with a position larger than the data item's position. The client will be able to specify the numberof replicas that each data item should have. The data item's coordinator node is then responsible for the replication. Several replication policies are available, were the system might be aware of a node's geographic position (i.e. replicateto other datacenters or to other racks)"(AbertoAcademy., 2008).

5.4 HBase

Hbase is a distributed, open-sourceand columndatabasewritten in Java runs as pat the top of Hadoop which is run on the top of HDFS (Hadoop Distributed File System) which provide capabilities for Hadoop and give a good way to store huge amount of data.

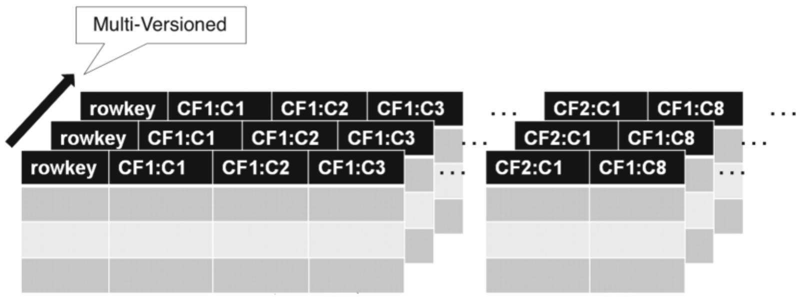


Figure 11.Representation of HBase's data model.

a) Data model

"HBase's data model is very similar to that of Cassandra, as they are both based on Google's BigTable. It follows a key-value model where the value is a tuple consisting of a row key, a technologies.
The Column key and a timestamp (i.e. a cell)(Cass.Benchmark, 2013) made of rows and columns, as seen in figure (11). A row can be a row key with number of columns. The Rows are lexicographically sorted with the lowest order appearing first in the table. The Columns contain a version history of their contents, which is ordered by a timestamp. Columns are grouped into Column Families and they have a common prefix in the form of family: qualifier manner. At column family level both tunings and storage specifications are done, it is advisable that all column family members have the same general access pattern and size characteristics" (AbertoAcademy., 2008).

b) Query model

"HBase provides a Java API which can be used to make queries. The obvious map operations such as put, delete andupdate are provided by this API An abstraction layer Filter, gives the applications the ability to define severalfilters for a row key, column families, column key and timestamps and the results can be iterated using the Scannerabstraction. For more complex queries it is possible to write MapReduce jobs which run on the underlying Hadoopinfrastructure. Using Pig it is very much possible to query data using a high-level, SQLlike, and language. It isalso possible to use Cascadingto simplify the process of writing MapReduce jobs. In addition to the Java API, itis also possible to access HBase through REST, Avro or Thrift gateway APIs. A shell for direct user interaction withHBase also exists here"(AbertoAcademy., 2008).

c) Replication model

"HBase uses a simple master-slave replication model. The replication is done asynchronously, that is, the clusters can be geographically distant and the links connecting them can be offline for some time, and rows inserted on the master server may not be available at the same time on the slave servers, hence providing only eventual consistency.
Replication is performed by replicating wholewaledits in order to maintain atomicity.WALEdits are used in HBase's transaction log (WAL) to represent the collection of edits corresponding to a single transaction " (AbertoAcademy., 2008).

d) Consistency model

"HBase is strictly consistent. Every value appears in one region only which is assigned to one region server at a time, within the appropriate boundary for its row. If thereplication is enabled and one reads it from the slave servers, only eventual consistency can be guaranteed, as the replication is done asynchronously. Rowoperations are atomic. It is possible to make transactions inside a row. Transactions over multiple rows are unsupported at the moment of this writing"(AbertoAcademy., 2008).

Ease of Use

According to the installation part and setting MongoDB shows that the most simple and Straightforward way comparing with others noSQL databases Cassandra, Hbase, and CouchDB.
The MongoDB server have been set up on the default part, looks like downloading the MongoDB and setting the configuration as simple as getting tar zips from 10g website after unzipping and running the MongoDB daemon thread.
And Cassandra shows the most complexity of the installed systems because it is required a lot of parameters and editing of the Cassandra.yaml file to run it on micro instance.
The need of restating the Cassandra server with the memory allocated for JVM and the frequent memory issues, because the previous allocated memory could not keep the collection of the garbage efficiently due to JVM heap lack size specified to the process of Cassandra(Patil, 2011).

This section deals with the comparison of Query form of the noSQL:
Couchdb query,MongoDBquery,SQL, Cassandra Query Language (CQL).
In this section we will compare the four types query standard statements, first we write the SQL standard query statements as most developers are familiar with.

SQL
```
update tablename set colField1 = colFieldValue
where primary key = primary keyValue
```

CouchDB
```
function(doc)
if document(doc.last_name){
emit(doc.last_name,doc)
}
}
```

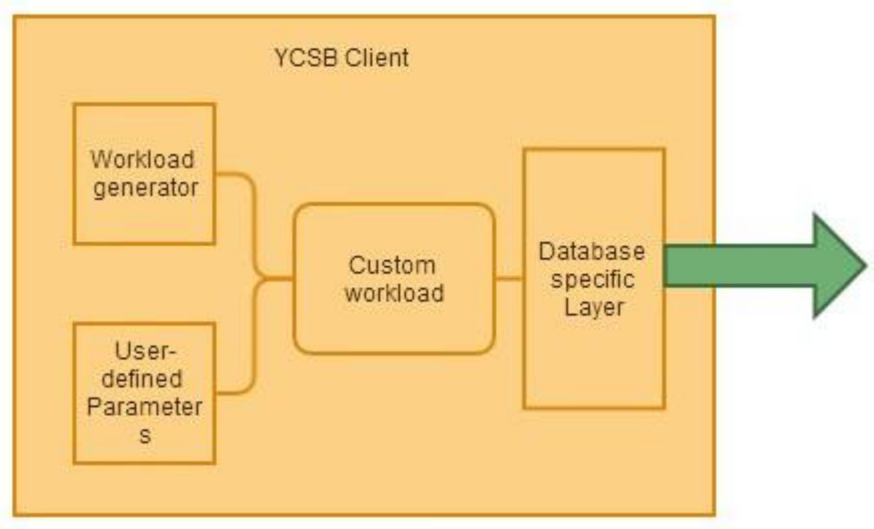
MongoDB
```
db.tableName.update({ fcolField1: f $eq: primaryKeyValue}},
   {f $set: fcolField1 = coalfield Value gg}})
```

CQL(Cassandra)
```
Update tableName SET colField0=value0, colField1=value1
WHERE primary Key=primaryKeyValue
```

So we can see that there is a small different between CQL Cassandra Query Language Because of its column data structure, MongoDB queries are little different using JSON type structure document and it's not too complex to adjust either.

5.5 Yahoo Cloud Serving Benchmark

"The Yahoo Cloud Computing benchmark or YCSB is a Java based tool that is used tobenchmark various databases and noSQLsystems. It is an open source project and caneasily be extended for new and upcoming noSQLor SQL database systems. It is a Javabased program which can be installed on the system by downloading via github or compiling source code directly. The main feature of YCSB is extensibility"(N.Rohan, 2013). Followingis the architecture of YCSB.



Figure(12) : YCSB Architecture(N.Rohan, 2013)

Workload generator

Workload generator splits the workload work into two basic phases, a) first phase called load phase means the environment which has to be set up to the database which we are going to test it by creating tables and populating the tables. b) The second phase called run phase means running different operations/queries against the database to show the output results to the user(N.Rohan, 2013).

The Following workloads are the workloads that we haveused for evaluation:

Workload A: can be named as standard workload benchmark and it contains 50% read of the operations and 50% update.

Workload B: This workload contains 100% read operations only inside thedatabase.

There are more than this two workload adopted by YCSB and we also can vary the percentage of the parameters to execute more workloads, However the basic workloads for the YCSB is workload A and workload B.

The variety of the parameters can be shown in the table below.

Table 1: workload properties

Parameter	Description
record count	number of insertion record
read %	read operation percentage
write%	write operation percentage
record size	size in bytes
fields	number of columns

Database layer
The main part of the YCSB is these layers which pair up different noSQL databases can be as an option, it converts the calls which is made by user workload into read and write calls to the database which is under the test specifically.

There are six steps for running a workload:
1. Set up the database system to test.
2. Choose the appropriate database interface layer.
3. Choose the appropriate workload.
4. Choose the appropriate runtime parameters (number of nodes, target throughput,e.t.c)
5. Load the data.
6. Execute the workload.

Nodejs creating and push record into Cassandra

Firstwe create my source code to generate my data with node.js for more speed performance not like php apache running code it's a synchronously "working fast ".

```
<?php

$dbh=mysql_connect('37.230.106.13','onrdgcom','ujSh9l7W11');

if (!$dbh) {

            die('Baglanamadi: ' . mysql_error());

}

mysql_select_db('onrdgcom_twissandra');

for ($i=0;$i<10000;$i++){

          $username= tkg.$i;

           $follower= ktr.$i;

$since="1970-01-01 00:00:01";

         $result = mysql_query("INSERT INTO `followers` (follower, since, username) VALUES
('$follower', '$since', '$username')");

}

?>
```

After that using Cassandra query language CQL to create table "twissandra"and the fields "follower, since, username" :

```
CQL> CREATE TABLE twissandra.followers (username text, follower text, since
timestamp, primary key( username, follower));
```

After that generating the records (600000) :

```
CQL> select count (*) from twissandra .followers limit 600000;

Output:

Count
    600000
<1 row>
```

First showing the transfer code in php:

```
<?Php
//'require('autoload.php);
usephpcassa\ColumnFamily;
usephpcassa\ColumnSlice;
usephpcassa\Connection\ConnectionPool;
usephpcassa\SystemManager;
usephpcassa\Schema\StrategyClass;
'dbh=mysql_connect('37.230.106.13','onrdgcom','ujSh9l7W11$);
if (!$dbh ) {
die('Baglanamadi: ' . mysql_error (());
}
mysql_select_db('onrdgcom_twissandra');
$result1 = mysql_query("SELECT * FROM followers");
$row1 = mysql_fetch_array($result1,MYSQL_ASSOC);
//Create a new keyspace and column family
 $'sys = new SystemManager('127.0.0.1);
  $sys->create_keyspace(twissandra, array)
"strategy_class" =>StrategyClass::SIMPLE_STRATEGY,
"strategy_options" =>array('replication_factor' => '1")));

'sys->create_column_family(twissandra, 'followers$);

//Start a connection pool, create our ColumnFamily instance
pool = new ConnectionPool('twissandra', array('127.0.0.1',’cassandra'));$
//array parameters must be name of cassandra nodes
$ ');followers = new ColumnFamily($pool, 'followers
while ($row1 = mysql_fetch_array($result1,MYSQL_ASSOC )){
'user=$row1{'username$};
followers->insert($user, array("follower" => $row1{'follower'}, "since" => $
'$row1{'since}));
}
"echo "end of file";
? >
```

Transfer Data from Node.js to Cassandra

For data transfer with Node.js to Cassandra we have to write a small Node.js program which shown below.

```
varhelenus = require('helenus'),
pool = new helenus.ConnectionPool({
hosts : ['localhost:9160'],
keyspace : 'twissandra',
user:    '',
password:     '',
timeout :  800000  ,
});
pool.on('error',
function(err){
console.error(err.name, err.message);
            });
varsinc="1970-01-01 00:00:01 ";
var i=0 ;
pool.connect(function(err, keyspace )}
if(err){
throw(err);
} else {
while (i<1000000){
                              i++;
varstr="tkg" + i;
var pass="ktr" + i ;
pool.cql("insert into followers (username,follower,since) VALUES (?,?,?)",[str, pass,
sinc],function(err, results));

                                                        }
}
});
console.log("end of program");
```

The workload of the YCSB executed as shown below:

Workload A:50% Read, 50% update , we can call it [update heavy workload].

Workload B: 95% Read, 5% update, we can call it [Read Mostly workload]

Workload C:100% Read only, we can call it [Read only workload]

Workload F: 50% Read, 50% modify-write, we can call it [Read-modify-write workload],In this workload a recordsare read, updated and all the changes are saved.

Workload G: 5% Read, 95% update, we can call it [update mostly workload]

Workload H: 100% update, we can call it [update only workload]

Read and write (insert) testing noSQL databases Using Yahoo Cloud Serving Benchmark(YCSB):

The Data:Loading

The figure below showing us the results of data loading with (600,000)record into the computer.

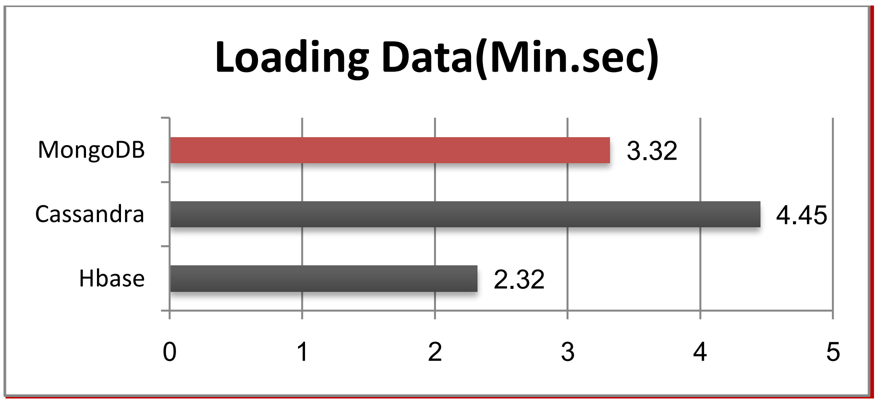


Figure:13Time of loading

This figure shows the loading time of two different types of noSQL databases bottom thebest loading time was presented by Hbase(column family) with a loading time of only (2:32) two minutes and thirty two seconds which is muchfaster than Cassandra (5.45) and MongoDB execution time was (3.32).

Workload A

The figure 14explain the results obtained whenwe execute workload A which is contain 50% of reads and 50% of updates on (600,000)record.

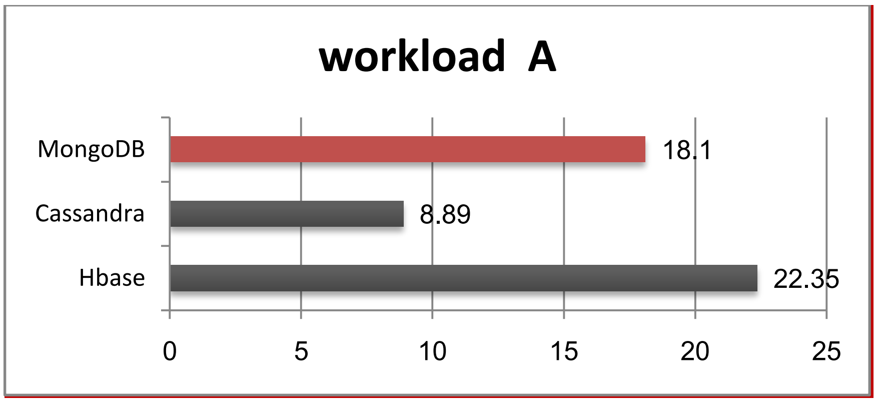


Figure:14Time of executing workload A

As we can see a good performance presented by the column family Cassandra (8.89) seven minutes and eighty nine seconds faster than MongoDB(18:10), while Hbase shows low performance in this workload .

Workload B

The figure 15explain the results obtained when we execute workload B that is consist of 95% reads and 5% updates on(600,000) records.

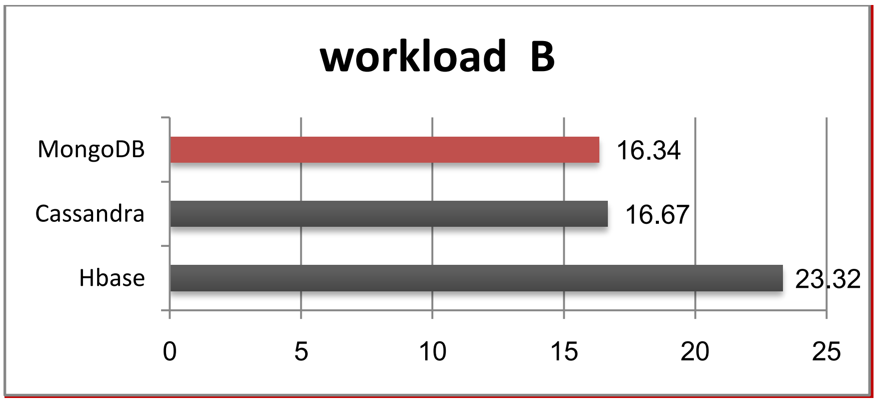


Figure:15Time of executing workload B

During execution of workload B, Column Family databases and Document Store databases had similar resultsCassandra time is (16.67) while MongoDBtime is (16.34) which is very close to Cassandra execution time performance, while Hbase shows much time longer than the others(23.32).

Workload C

The figure 16explain the results obtained while executing workload C that consists of execution of 100% reads onlyon(600,000)record.

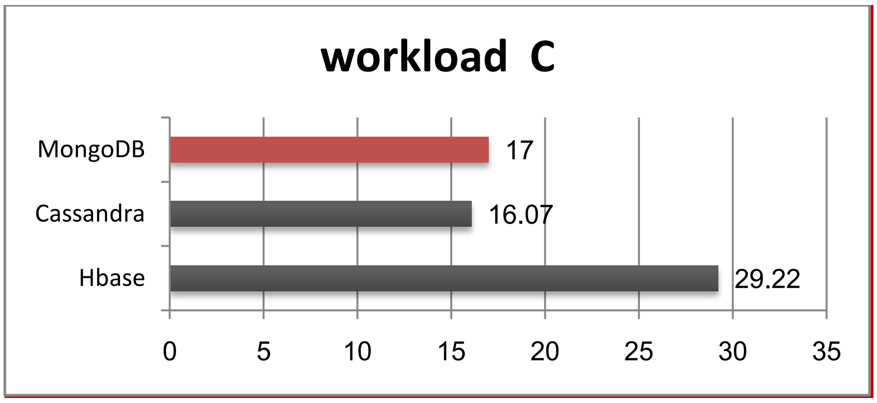


Figure:16Time of executing workload C

Here we can notice that Cassandra (16,7) gives better performance than MongoDB(17min),and better than Hbase (29,22).

Workload F

The figure 17explain the results when we execute workload F that is (50% read, 50%modify-write)on(600,000) records stored.

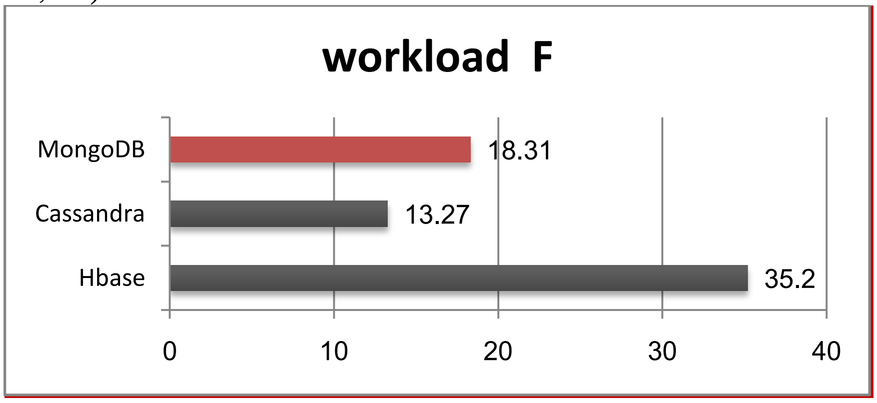


Figure:17Time of executing workload F

As we can see MongoDB give more efficient performance with execution time (18.31) And Cassandra execution time is (13.27) while Hbase is (35.2).

Workload G

Figure 18explain the results when we execute workload G with (600,000) records, performing (5% reads and 95% updates).

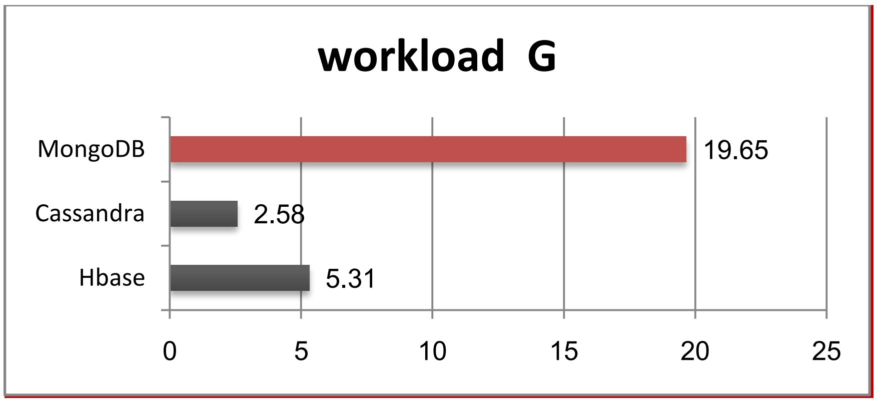


Figure:18Time of executing workload G

We can notice that Cassandra execution time (2:58) two minute and fifty eight seconds is eleven times faster than the execution time ofMongoDB (19.65).

Workload H

Figure 19explain the results of the execution of workload H with 100% updates into a database with (600,000) records.

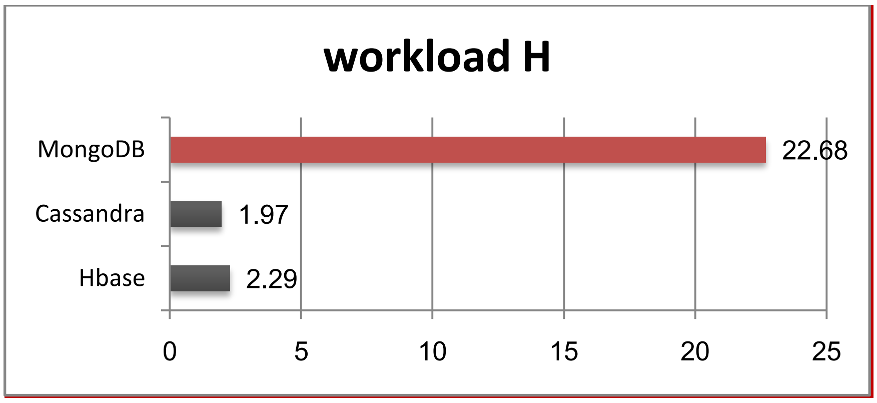


Figure:19. Time of executing workload H

As we can see that the execution time for Cassandra (1.97) nighty seven seconds only which is twenty one times faster than the execution time of MongoDB (22.68), while Hbase shows (2.29).

Note: all of the above workloads are done on a single node computer and its specification is:

CPU Intel(R) core (TM) i7-4200 CPU 2.29 GH
RAM 4,00 GB
Hard disk   100 GB

Overall Evaluation

During the previous tests we described the results classified by the type of operation assigned, andshow Cassandra is the most effective and faster than the other (MongoDBand Hbase).

CPU Utilization

In addition to various benchmark workload we also record the CPU utilization on the noSQL databases Cassandra and MongoDB while CouchDB was not available yet on the web interface of EC2 cloud watch metrics, we used this metrics to record the number of the CPU utilized while the workload A is running we insert (test) the records as following (1000rec, 10000rec, 50000rec).

We can notice as shown in the table below that Cassandra is much lower than MongoDB for the three test phases, means that Cassandra schema better performance in term of CPU utilization

The table below shows how much of CPU resources has been used in percentage for both MongoDB and Cassandra.

Table 2: Comparison of CPU utilization server side

no _records	MongoDB	Cassandra
1000	26	13
10000	45	29
50000	83	69

Sorting:

"Providing a sorted view of a given data set is a typical task for a database.

All examined databases can handle the ordering of objects by a single attribute, but only the both document oriented databases MongoDBand CouchDB and the App Engine data store allow the ordering by multiple attributes. All three of them require specifying an index to be able to perform the ordering efficiently.

A sometimes-important feature of databases is the ordering of a set by the result of an aggregation, like a count operation or summation. This is for example required for Top N lists. If an application requires such a feature, one possible solution would be to let the client handle the creation of top n list in a periodically batch process, it could even use the database to store and order the result"(k.Orend, 2010). Table3 shows the sorting of the different databases.

Table 3: Comparison of the sorting capabilities

	Singleattribute ordering	multiple attributes ordering	aggregation result ordering
MongoDB	Yes	Yes	No
CouchDB	Yes	Yes	No
Cassandra	Yes	Yes	No

Range Queries

CouchDB allows range queries at the same time on different attributes;MongoDB also allows range queries on multiple attributes that connected to a logical operator, CouchDB uses complex keys in the task which means the developers have to specify an index with a key contain the attributes on that rang query being performed(k.Orend, 2010).

Table 4 shows the different in the expression power for range queries of the different databases.

Table 4: The range querying Comparison

noSQL DBs	Onsingle attributes	on multipleattributes	on aggregation results
MongoDB	yes	yes	no
CouchDB	yes	yes	no
Cassandra	yes	yes	no

Aggregations

Aggregation means the task like counting ,summations and Average calculations and other statistics calculations for example SimpleDB noSQL database support only counting and don't support other aggregation functions, CouchDB and big table aggregation can be implemented with map reduce functions usage, but they do it differently, big table uses Goggles map reduce frame on it, which can be as data input source, in contrast CouchDB map reduce system stores the output as index which is updated incrementally when the document is written, the advantage of CouchDB database is that map reduce result always reflect the documents current state in the database(k.Orend, 2010).

MongoDB also provides some extra functions for aggregation like summation and average, grouped by statements, the functions in MongoDB can be used efficiently with the index that can be specified, also MongoDB support the use of map reduce function, in MongoDB map reduce shows better performance since it can instrument previously specifying indexes, especially map reduce task strict to a query, which is used to reduce the large amount number of documents that map reduce task has to monitor.Table 5describe the difference of the aggregation functionalities

Table 5: Comparison of the aggregation functionalities.

noSQL DBs	Counting	Aggregation primitives like Summations and Averages	Complex aggregations with MapReduce
MongoDB	yes	yes	yes
CouchDB	Map reduce	Map reduce	yes
Cassandra	yes	yes	no

Durability

"Durability means that all write operations that are committed to a database can never be lost. This requires that new data is written in some way onto the hard disk before a database can return the acknowledgement of a successful commit operation, so that committed data cannot be lost even in the event of a power failure. InRDBMS, durability is usually ensured through the usage of a transaction log file.

In this file, all transactions are written, before they are committed. If a failurelike a power outage or some other kind of software or hardware failure happens, the database can use this log to recover committed but not completely applied transactions.

For distributed databases the term of durability can even be more extended, because some databases can not only make sure that committed data is always persistent on a single machine, but they can also make sure that the data is replicated to at least a specific number of machines or even to different data centers distributed around the world to be safe against geological disasters and permanent failures of single machines"(k.Orend, 2010).

Table6: Comparison of the durability.

noSQL DBs	Written on at least one hard diskafter commit	Replicated to different machines after commit	On disk integrity
MongoDB	no	only late replication	Recovery may cause data loss and requires consistency checks of the database files
CouchDB	not butcan be enabled.	onlylate replication.	TheDatabase files always be in validity
Cassandra	nodewrites data into the local commit log file in a sequential manner.	memtablegets updatedin asynchronous mode	Cassandra tool "nodetool" provides an option to explicitly flush the data in commit log or memtable into SSTables,

5.6 Conclusion

In this study we have compared most-widely used noSQL database systems. Namely, CouchDB, MongoDB, and Cassandra. Although it is not exactly in the same group of noSQL database systems we also added Hbase for the comparison when conducting the tests to see its performance on a single node. All of our test results are average of 6 times. All the test workloads are done on single computer node.

The questions, for which we seek an answer, is what would be the best noSQL database system to use for analyzing my data?, Our findings on this research for the most widely used noSQL databases, namely MongoDB, Cassandra, Hbase as follow:

- MongoDB shows better performance than Cassandra when loading (reading in) the data,
- when we have (50%) read and (50%) update operations Cassandra’s performance is better than the others,
- when we have read heavy operationCassandra shows better performance than the others,
- when we have execution of 100% read only operations Cassandra shows the best performance,
- when we perform with (5%) reads and (95%) updates means update heavy Cassandra shows the best performance.
- when we have only " updates " means100% updates only into a database Cassandra shows better performance than the others.

5.7 Suggestions

Using the appropriate noSQL database for the task is very important issue for the IT community, since there is no absolute evaluate for the question " which tool is the write tool for the task?" ,**the performance–wise (Read, Write, and Update) were be compared in this researchon most widely used noSQL databases.**

First, to avoid unnecessary complexity in transformation or mapping tasks developers should evaluate the big amount of data to identify a proper data model and putting in consider the queries supported by the database.

The requirements mentioned above are influencing the design of the data model in massive way, developers have to deal with and manage the high performance of partitioning and balancing the servers, high availability should be supported by rigid consistency and asynchronous replication.

Bibliography

The MongoDBnoSQL Database Blog. (2009). Retrieved 2015, from BSON: http://blog.mongodb.org/post/114440717/bson

Binary JSONx . (2011). Retrieved 2011, from BSON. BSON : http://bsonspec.org/

SQL to Mongo Mapping Chart –. (2011). Retrieved 2015, from Mongo DB: http: www.mongodb.org/display/DOCS/SQL+to+Mongo+ Mapping+Chart

Futon: Web GUI Administration Panel. (2015). Retrieved 2015, from CouchDB: http://docs.couchdb.org/en/latest/intro/futon.html

Amazon SimpleDB. (2015). Retrieved 2015, from Amazon: http://aws.amazon.com/simpledb/

Apache HBase. (2015). Retrieved 2015, from Hbase: http://hbase.apache.org/

MongoDB. (2015). Retrieved 2015, from MongoDB: https://www.mongodb.org/

Redis. (2015). Retrieved 2015, from Redis: http://redis.io/

AbertoAcademy. (2008). *Uporto*. Retrieved 2015, from aberto: http://repositorio-aberto.up.pt/

Abiteboul, S. (2011, Sep 23). *An introduction to CouchDB, a "NoSQL" document database*. Retrieved 2015, from http://webdam.inria.fr/Jorge/files/slcouchdb.pdf

Aircc.Corporation. (2015). *AIRCC Puplication*. Retrieved 2015, from AIRCC Puplication: http://www.airccse.org/

Cass.Benchmark. (2013). *Accelerating Cassandra Performance and Reducing Read Latency .* California: Benchmarking Cassandra on Violin.

CassandraPro. (2015). *Apache Cassandra*. Retrieved 2015, from Cassandra: http://cassandra.apache.org/

Chang F., D. J. (2008). Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems*.

Connolly, S. (2012, May). *7 Key Drivers for the Big Data Market*. Retrieved May 2015, from Hortonworks: http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market/

CouchBase. (2015). *Why Nosql*. Retrieved 2015, from CouchDb: http://www.couchbase.com/nosql-resources/what-is-no-sql

CouchDBLearn. (2014). Retrieved 2015, from http://training.couchbase.com/

CouchDBLearn. (2014). *CouchDB Learing*. Retrieved 2015, from CouchDB: http://webdam.inria.fr/Jorge/files/slcouchdb.pdf

Couchdoc. (2015). *Apache CouchDB*. Retrieved 2015, from CouchDB: http://couchdb.apache.org/

CouchDocument. (2015). *couchDocument*. Retrieved 2015, from CouchDB: http://couchdb.apache.org/

CouchIntro. (2011). *Apache CouchDB: Introduction*. Retrieved 2015, from The Apache Software Foundation,: http://couchdb.apache.org/docs/intro.html

DataStax. (2014). *Apache Cassandr1.0 Documentation.* DataStaxCommunity.

DataStaxTut. (2015). *DataStax*. Retrieved 2015, from DataStax: http://www.datastax.com/

Futon. (2015). *Futon: Web GUI Administration Panel*. Retrieved 2015, from CouchDB: http://docs.couchdb.org/en/latest/intro/futon.html

G. DeCandia, D. H. (2007, Oct 14). Dynamo: Amazon's Highly Available Key-value Store.

Hadoop. (2011). *The apache software foundation*. Retrieved 2015, from Welcome to Apache Pig: http://pig.apache.org/

Hare, K. W. (2011). A Comparison of SQL And Comparison of SQL. *JCC Consulting, Inc.Convenor, ISO/IEC JTC1 SC32 WG3.* ISO/IEC JTC1/SC32/WG2 N1537.

Harris, D. (2011, july 29). *GigaOM Research*. Retrieved 2015, from GIGAOM: https://gigaom.com/2011/07/29/couchbase-2-0-unql-sql-noSQL/

Hbasetutorials. (2015). *tutorialspoint*. Retrieved May 2015, from tutorialspoint: http://www.tutorialspoint.com//hbase/hbase_installation.htm

Hewitt, E. (2010). *Cassnadra The Defnitive Guide.* O Really Media.

Hossain, A. B. (2013). NoSQL Database New Era of Databases for Big data Analytics Classification, Characteristics and Comparison. *International Journal of Database Theory and Application, 6*, 1.

k.Orend. (2010). *Analysis and Classification of NoSQL Databases and Evaluation of their Ability to Replace an Object-relational.* Munchen: Technical University Munchen.

Kopp, M. (2011). *NoSQL or RDBMS?* Retrieved 2015, from About Performance: http://apmblog.dynatrace.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/

Manoj, V. (2014, August). Comparative Study Of Nosql Documen Column Store Database And Evaluation Of Cassandra. *International Journal of Database Management Systems (IJDMS), 6*.

MongoDatabase. (2015). *MongoDB - Create Database*. Retrieved 2015, from Tutorialspoint: http://www.tutorialspoint.com/mongodb/mongodb_create_database.htm

MongoDB. (2015). *map reduce*. Retrieved 2015, from mongoDB: http://docs.mongodb.org/manual/core/map-reduce/

MongoDBDoc. (2015). *MongoDBmanual*. Retrieved 2015, from MongoDB: http://docs.mongodb.org/

MongoDBInc. (2015). *mapreduce*. Retrieved 2015, from MongoDB: http://docs.mongodb.org/manual/core/map-reduce/

MongoInstall. (2015). *MongoDB*. Retrieved 2015, from MongoDB: http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/

MongoRep. (2011). *Replica Sets*. Retrieved 2015, from MongoDB: http://www.mongodb.org/

N.Rohan. (2013, December). *A Comparison of Nosql Systems.* New york: Rochester Institute of Technology.

Narde, R. (2013). *Comparison of NoSQL databases on the Cloud based on data model.* Rochester Institute, Computer Science. New York: Rochester Institute of Technology.

Orend, k. (2010). *Analysis and compilation of nosql databases.* Munchen: faculty of inforamtion systems.

Patil, N. A. (2011). *Sql versus noSQL data management in the cloud.* New york: Rochester Institute of Technology.

ResGate. (2015). *Research Gate*. Retrieved 2015, from Research Gate: http://www.researchgate.net/

Riak. (2015). *Riak Kv*. Retrieved 2015, from Riak: http://basho.com/products/riak-kv/

Shinde, S. (2015). *What is Wide Column Stores*. Retrieved 2015, from Bigdata: http://bi-bigdata.com/2013/01/13/what-is-wide-column-stores/

Staff, N. (2010, Feb). *The top 10 ways to get to know Neo4j*. Retrieved 2015, from Neo4j: http://neo4j.com/blog/the-top-10-ways-to-get-to-know-neo4j/

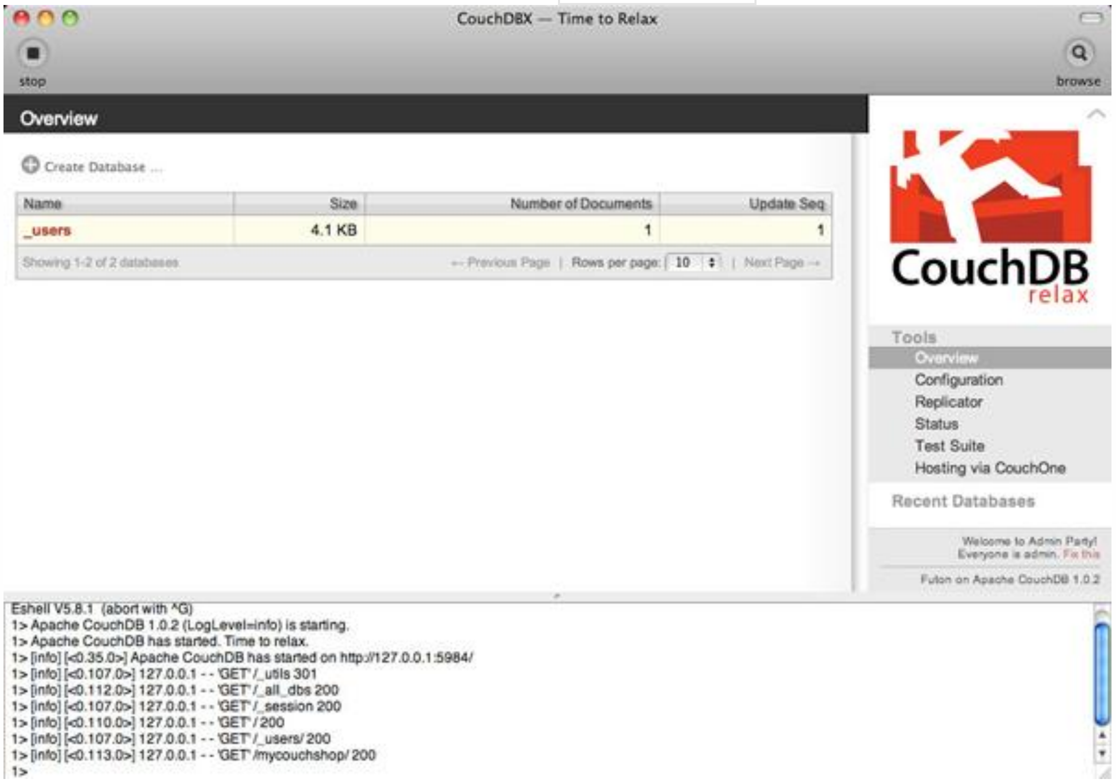
Appendix : Creating databases in noSQL

Creating database in CouchDB "Futon"

Step1:

After CouchDB has started, you should see the Futon control panel in the CouchDBX application. In case you can't, you can access Futon via your browser. Looking at the log, CouchDBX tells us CouchDB was started at `http://127.0.0.1:5984/` (may be different on your system).

Open a browser and go to `http://127.0.0.1:5984/_utils/` and you should see Throughout the rest of this tutorial I will be using Futon in Firefox. I'll also have Firebug and the console view open to see all the HTTP requests Futon is sending behind the scenes. This is useful as your application can do everything Futon is doing. Let's go ahead and create a database called `mycouchshop`.



Name	Size	Number of Documents	Update Seq
_users	4.1 KB	1	1

Showing 1-2 of 2 databases ← Previous Page | Rows per page: 10 | Next Page →

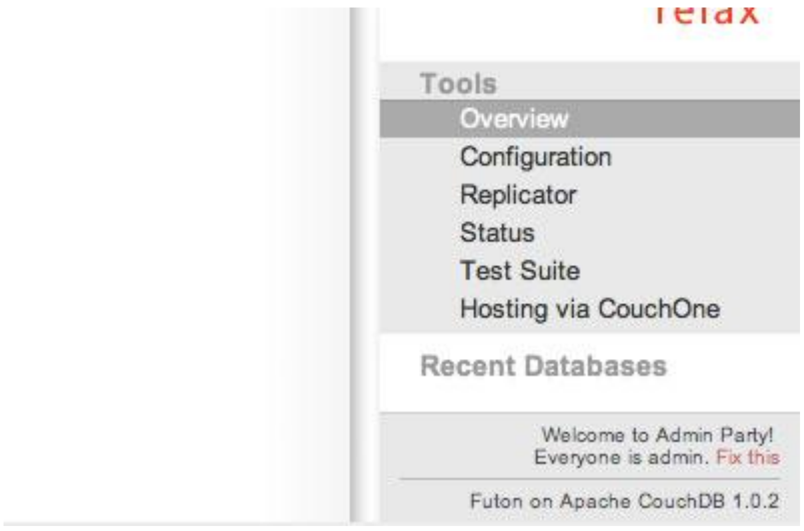
```
Eshell V5.8.1  (abort with ^G)
1> Apache CouchDB 1.0.2 (LogLevel=info) is starting.
1> Apache CouchDB has started. Time to relax.
1> [info] [<0.35.0>] Apache CouchDB has started on http://127.0.0.1:5984/
1> [info] [<0.107.0>] 127.0.0.1 - - 'GET' /_utils 301
1> [info] [<0.112.0>] 127.0.0.1 - - 'GET' /_all_dbs 200
1> [info] [<0.107.0>] 127.0.0.1 - - 'GET' /_session 200
1> [info] [<0.110.0>] 127.0.0.1 - - 'GET' / 200
1> [info] [<0.107.0>] 127.0.0.1 - - 'GET' /_users/ 200
1> [info] [<0.113.0>] 127.0.0.1 - - 'GET' /mycouchshop/ 200
1>
```

CouchDB jQuery PluginFuton is actually using a jQuery plugin to interact with CouchDB. You can view that plugin at http://127.0.0.1:5984/_utils/script/jquery.couch.js (bear in mind your port may be different). This gives you a great example of interacting with CouchDB.

Step 2 - Users in CouchDB

CouchDB, by default, is completely open, giving every user admin rights to the instance and all its databases. This is great for development but obviously bad for

production. Let's go ahead and setup and admin. In the bottom right, you will see "Welcome to Admin Party! Everyone is admin! Fix this".
Go ahead and click fix this and give yourself a username and password. This creates an admin account and gives anonymous users access to read and write operations on all the databases, but no configuration privileges(Futon, 2015).



More on Users

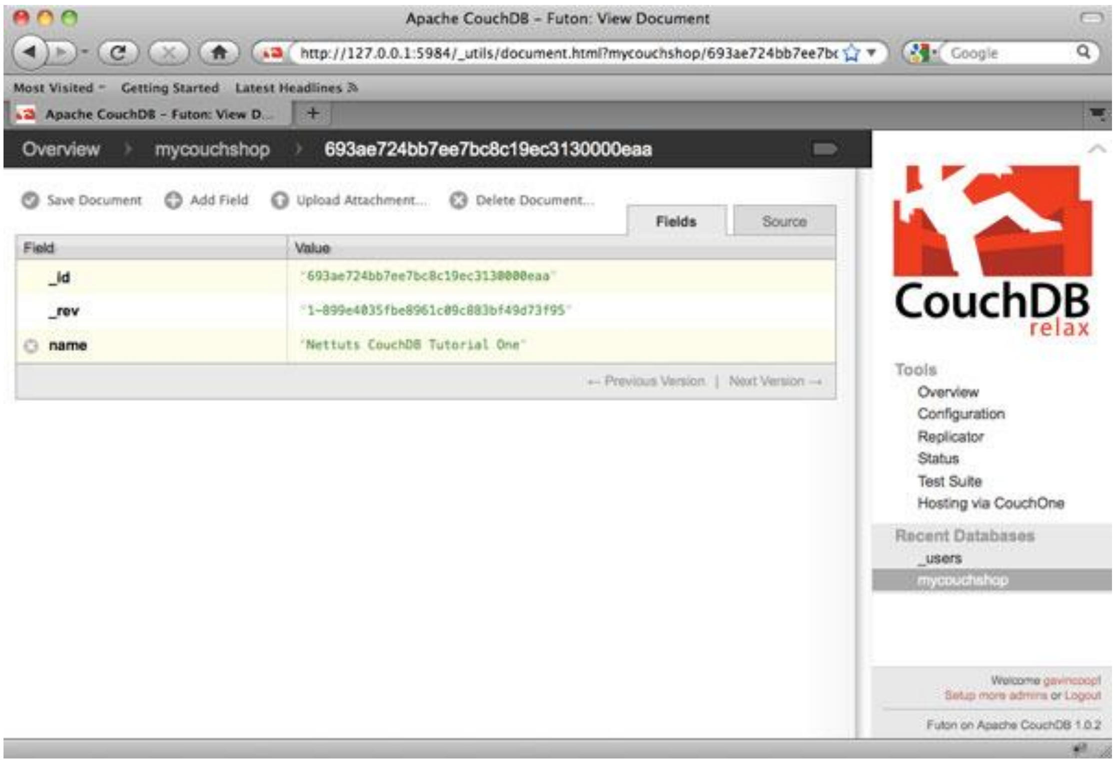
In CouchDB it would be unwise to create a single super user and have that user do all the read/write.
Users in CouchDB can be a little confusing to grasp initially, especiallyif you're used to creating a single user for your entire application and then managing users yourself within a user's table (not the MySQL users table). In CouchDB, it would be unwise to create a single super user and have that user do all the read/write, because if your app is client-side then this super user's credentials will be in plain sight in your JavaScript source code.
CouchDB has user creation and authentication baked in. You can create users with the jQuery plugin using $.couch.signup(). These essentially become the users of your system. Users are just JSON documents like everything else so you can store any additional attributes you wish like email for example.You can then use groups within CouchDB to control what documents each user has write access to. For example, you can create a database for that user to which they can write to and then add them to a group with read access to the other databases as required.

Step 3 - Creating a Product Document

Now let's create our first document using Futon through the following steps:

1. Open the mycouchshop database.
2. Click "New Document".
3. Click "Add Field" to begin adding data to the JSON document. Notice how an ID is pre-filled out for you, I would highly advise not changing this. Add key "name" with the value of "Nettuts CouchDB Tutorial One".
4. Make sure you click the tick next to each attribute to save it.
5. Click "Save Document".

Go up a level, back to the database and you should see one document listed with the previous ID as the key and a value beginning with{rev: . This is the JSON document you just created.



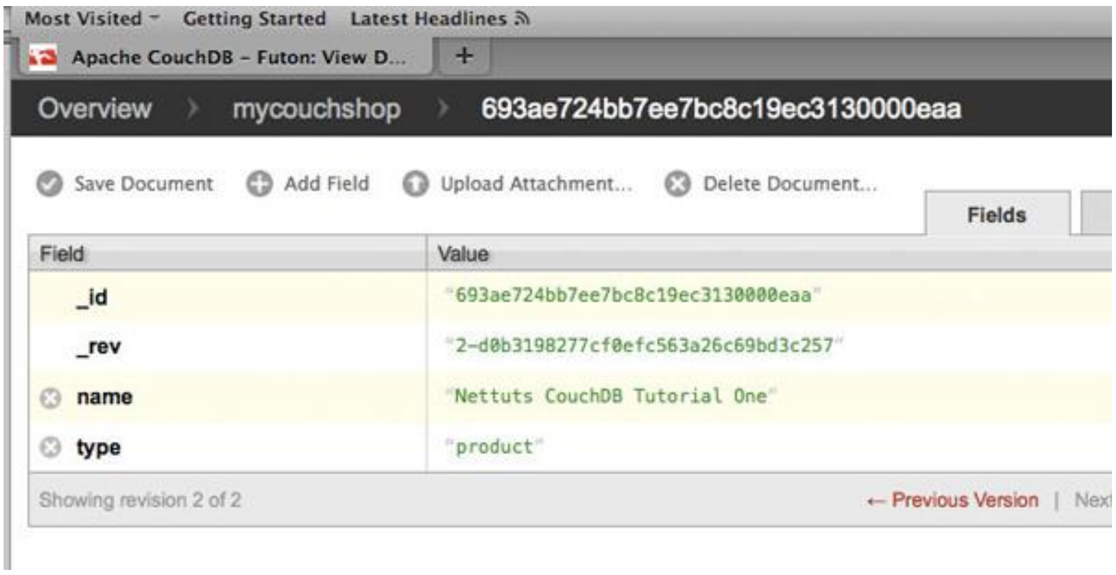
Field	Value
_id	"693ae724bb7ee7bc8c19ec3130000eaa"
_rev	"1-899e4035fbe8961c09c883bf49d73f95"
name	"Nettuts CouchDB Tutorial One"

Step 4 - Updating a Document

CouchDB is an append only database -- new updates are appended to the database and do not overwrite the old version. Each new update to a JSON document with a pre-existing ID will add a new revision. This is what the automatically inserted revision key signifies. Follow the steps below to see this in action:

- Viewing the contents of the mycouchshop database, click the only record visible.
- Add another attribute with the key "type" and the value "product".
- Hit "Save Document".

After hitting save, a new revision key should be visible starting with the number 2. Going back a level to the mycouchshop database view, you will still see just one document, this is the latest revision of our product document(Futon, 2015).



Field	Value
_id	"693ae724bb7ee7bc8c19ec3130000eaa"
_rev	"2-d0b3198277cf0efc563a26c69bd3c257"
name	"Nettuts CouchDB Tutorial One"
type	"product"

Showing revision 2 of 2

Revisions

While CouchDB uses revisions internally, try to not lean on it too much. The revisions can be cleaned through Futon quite easily and it is not designed to be used as a revision control system. CouchDB uses the revisions as part of its replication functionality.

Step 5 - Creating a Document Using cURL

I've already mentioned that CouchDB uses a RESTful interface and the eagle eyed reader would have noticed Futon using this via the console in Firebug. In case you didn't, let's prove this by inserting a document using CURL via the Terminal.

First, let's create a JSON document with the below contents and save it to the desktop calling the file person.json.

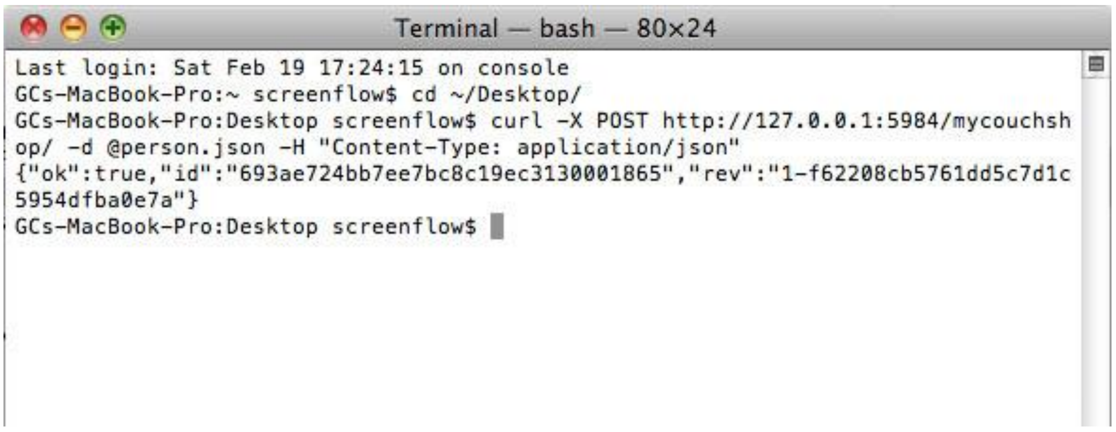
```
{
    "forename": "Gavin",
    "surname":  "Cooper",
    "type":     "person"
}
```

Next, open the terminal and execute cd ~/Desktop/ putting you in the correct directory and then perform the insert with curl -X POST http://127.0.0.1:5984/mycouchshop/ -d @person.json -H "Content-Type: application/json". CouchDB should have returned a JSON document similar to the one below.

```
{"ok":true,"id":"c6e2f3d7f8d0c91ce7938e9c0800131c","rev":"1-abadd48a09c270047658dbc38dc8a892"}
```

This is the ID and revision number of the inserted document. CouchDB follows the restful convention and thus:(Futon, 2015)

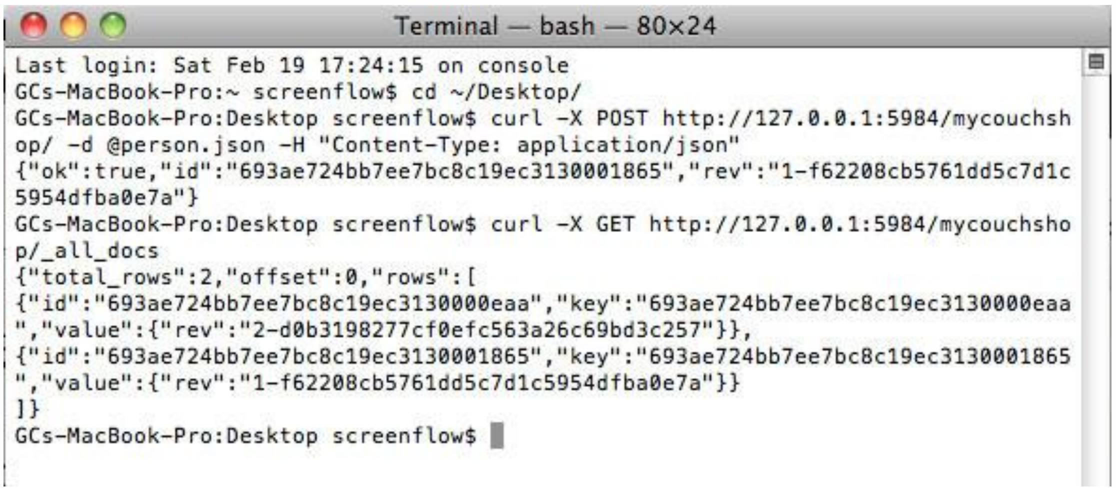
- POST - creates a new record
- GET - reads records
- PUT - updates a record
- DELETE - deletes a record



```
Last login: Sat Feb 19 17:24:15 on console
GCs-MacBook-Pro:~ screenflow$ cd ~/Desktop/
GCs-MacBook-Pro:Desktop screenflow$ curl -X POST http://127.0.0.1:5984/mycouchsh
op/ -d @person.json -H "Content-Type: application/json"
{"ok":true,"id":"693ae724bb7ee7bc8c19ec3130001865","rev":"1-f62208cb5761dd5c7d1c
5954dfba0e7a"}
GCs-MacBook-Pro:Desktop screenflow$ 
```

Step 6 - Viewing All Documents

We can further verify our insert by viewing all the documents in our mycouchshopdatabase by executing curl -X GET http://127.0.0.1:5984/mycouchshop/_all_docs.



```
Last login: Sat Feb 19 17:24:15 on console
GCs-MacBook-Pro:~ screenflow$ cd ~/Desktop/
GCs-MacBook-Pro:Desktop screenflow$ curl -X POST http://127.0.0.1:5984/mycouchsh
op/ -d @person.json -H "Content-Type: application/json"
{"ok":true,"id":"693ae724bb7ee7bc8c19ec3130001865","rev":"1-f62208cb5761dd5c7d1c
5954dfba0e7a"}
GCs-MacBook-Pro:Desktop screenflow$ curl -X GET http://127.0.0.1:5984/mycouchsho
p/_all_docs
{"total_rows":2,"offset":0,"rows":[
{"id":"693ae724bb7ee7bc8c19ec3130000eaa","key":"693ae724bb7ee7bc8c19ec3130000eaa
","value":{"rev":"2-d0b3198277cf0efc563a26c69bd3c257"}},
{"id":"693ae724bb7ee7bc8c19ec3130001865","key":"693ae724bb7ee7bc8c19ec3130001865
","value":{"rev":"1-f62208cb5761dd5c7d1c5954dfba0e7a"}}
]}
GCs-MacBook-Pro:Desktop screenflow$ 
```

Step 7 - Creating a Simple Map Function

Viewing all documents is fairly useless in practical terms. What would be more ideal is to view all product documents. Follow the steps below to achieve this:

- Within Futon, click on the view drop down and select "Temporary View".
- This is the map reduce editor within Futon. Copy the code below into the map function.

```
function(doc) {
    if(doc.type === "product"&& doc.name) {
        emit(doc.name, doc);
    }
}
```

Creating database in MongoDB:

MongoDB use DATABASE_NAME is used to create database. The command will create a new database, if it doesn't exist otherwise it will return the existing database.

Syntax:

Basic syntax of use DATABASE statement is as follows:

```
use DATABASE_NAME
```

Example:

If you want to create a database with name <mydb>, then use

DATABASE statement would be as follows:

```
>use mydb
switched to db mydb
```

To check your currently selected database use the command db

```
>db
mydb
```

If you want to check your databases list, then use the command show dbs.

```
>show dbs
local     0.78125GB
test      0.23012GB
```

Your created database (mydb) is not present in list. To display database you need to insert at least one document into it.

```
>db.movie.insert({"name":"tutorials point"})
>show dbs
local0.78125GB
mydb0.23012GB
test0.23012GB
```

In MongoDB default database is test.If youdidn't create any database thencollections will be stored in test database.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases(MongoDatabase, 2015).

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do notenforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Below given table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Sample document

Below given example shows the document structure of a blog site which is simply a comma separated key value pair.

```
{
   _id:ObjectId(7df78ad8902c)
title:'MongoDB Overview',
description:'MongoDB is no sql database',
by:'tutorials point',
url:'http://www.tutorialspoint.com',
tags:['mongodb','database','NoSQL'],
likes:100,
comments:[
{
user:'user1',
message:'My first comment',
dateCreated:newDate(2011,1,20,2,15),
like:0
},
{
user:'user2',
message:'My second comments',
dateCreated:newDate(2011,1,25,7,45),
like:5
}
]
}
```

Creating Database in Cassandra:

Creating a Table using Cqlsh

You can create a table using the command CREATE TABLE. Given below is the syntax for creating a table.

Syntax

```
CREATE (TABLE | COLUMNFAMILY) <tablename>
('<column-definition>' , '<column-definition>')
(WITH <option> AND <option>)
```

Defining a Column

You can define a column as shown below.

```
column name1 data type,
column name2 data type,
example:
ageint,
name text
```

Primary Key

The primary key is a column that is used to uniquely identify a row. Therefore,defining a primary key is mandatory while creating a table. A primary key is made of one or more columns of a table. You can define a primary key of a table as shown below(MongoDatabase, 2015).

```
CREATE TABLE tablename(
column1 name datatype PRIMARYKEY,
column2 name data type,
column3 name data type.
   )
```

or

```
CREATE TABLE tablename(
column1 name datatype PRIMARYKEY,
column2 name data type,
column3 name data type,
PRIMARY KEY (column1)
   )
```

Example

Given below is an example to create a table in Cassandra using cqlsh. Here we are:
Using the keyspace tutorialspoint
Creating a table named emp
It will have details such as employee name, id, city, salary, and phone number. Employee id is the primary key.

```
cqlsh> USE tutorialspoint;
cqlsh:tutorialspoint>; CREATE TABLE emp(
emp_idint PRIMARY KEY,
emp_name text,
emp_city text,
emp_salvarint,
emp_phonevarint
   );
```

Verification

The select statement will give you the schema. Verify the table using the select statement as shown below.

```
cqlsh:tutorialspoint> select * from emp;


emp_id | emp_city | emp_name | emp_phone | emp_sal
--------+----------+----------+-----------+---------


(0 rows)
```

Here you can observe the table created with the given columns. Since we have deleted the keyspace tutorials point, you will not find it in the keyspaces list(MongoDatabase, 2015).

Cassandra Data Model

The Cassandra data model is a dynamic schema, column-oriented data model. This means that, unlike a relational database, you do not need to model all of the columns required by your application up front, as each row is not required to have the same set of columns.
Columns and their metadata can be added by your application as they are needed without incurring downtime to your application.
There is an SQL like query language to query Cassandra Key Spaces. The name of the Cassandra query language is CQL. CQL has some versions but the last version of CQL is 3, so right now CQL3 is used to query Cassandra. CQL 3 tables, rows, and columns can be viewed much the same way as SQL, which is different from

theinternal implementation in Cassandra. In SQL you define tables, which have definedcolumns. The table defines the column names and their data types, and the client application then supplies rows conforming to that schema. In Cassandra, you also define tables and metadata about the columns, but the actual columns that make up a row are determined by the client application.
CQL 3 has terms which are different from relational database terms. These terms are given below.
Cluster: You can create different Cassandra datacenters means that different Cassandra topologies can be created as datacenters. If you have more than 1 datacenter or topology, you can establish a communication among these datacenters. Cluster names are given to different datacenters(MongoDatabase, 2015).

KeySpace: The name of Cassandra's database which has same meaning with database of RDBMS.

Column Family: The name of Cassandra's tables which has same meaning with tables or relations of RDBMS.

RowKey: In Cassandra each record is represented a row key.

An Example of Cassandra Data Model (Music Service)
This example contrasts two approaches to applying Cassandra's storage model:
• Using legacy sparse objects. Objects and fields correspond to rows and columns, respectively, but we do not know what the fields are in advance.
• Using CQL 3 to transpose data partitions (wide rows) into familiar row-based result sets, dramatically simplifying data modeling.

This example of a social music service requires a songs table having a title, album, and artist column, plus a column called data for the actual audio file itself. The table uses a UUID as a primary key.

```
CREATE TABLE songs (
        id uuid PRIMARY KEY,
        title text,
        album text,
        artist text,
        data blob
        );
```

In a relational database, you would create a playlists table with a foreign key to the songs, but in Cassandra, you denormalize the data. Cassandra packs each song in a playlist into one column, so fetching a playlist is just a single primary key lookup. The column value holds the song id. To represent the playlist data in CQL3, you can create a table like this:

```
CREATE TABLE playlists (
        id uuid,
        song_id uuid,
        title text,
        album text,
        artist text,
        PRIMARY KEY (id, song_id)
        );
```

The combination of the id and song_id in the playlists table uniquely identifies a row in the playlists table. You can have more than one row with the same id as long as the rows contain different song_ids. Using UUIDs as surrogate keys instead of sequential integers is a Cassandra best practice; using natural keys is also a good option. UUIDs are handy for sequencing the data or automatically incrementing synchronization across multiple machines(MongoDatabase, 2015).

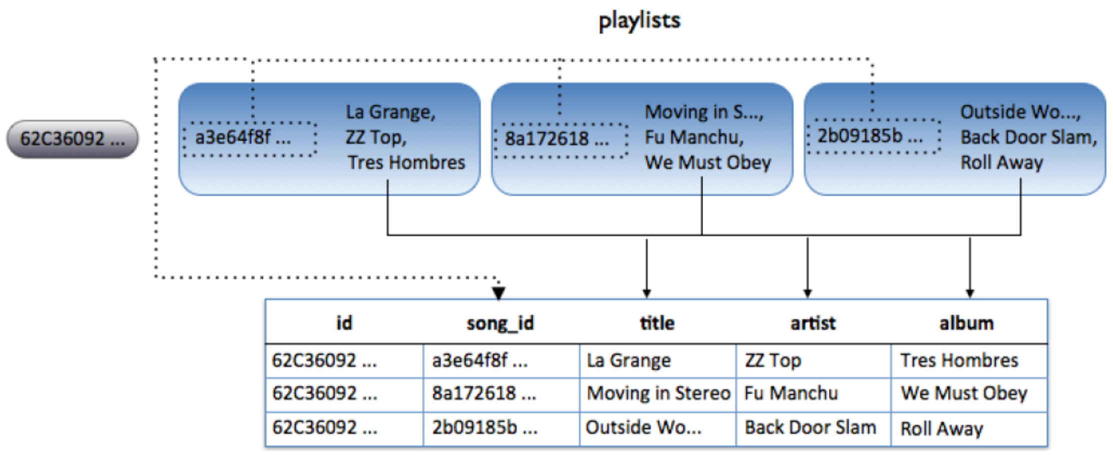
We can insert some records which are shown below to “playlist”.
INSERT INTO playlists (id, song_id, title, artist, album)
VALUES (62c36092-82a1-3a00-93d1-46196ee77204,
'a3e64f8f-bd44-4f28-b8d9-6938726e34d4', 'La Grange', 'ZZ Top', 'Tres Hombres');
INSERT INTO playlists (id, song_id, title, artist, album)
VALUES (62c36092-82a1-3a00-93d1-46196ee77204,
'8a172618-b121-4136-bb10-f665cfc469eb', 'Moving in Stereo', 'Fu Manchu', 'We Must Obey');
INSERT INTO playlists (id, song_id, title, artist, album)
VALUES (62c36092-82a1-3a00-93d1-46196ee77204,
'2b09185b-fb5a-4734-9b56-49077de9edbf', 'Outside

INSERT INTO playlists (id, song_id, title, artist, album)
VALUES (62c36092-82a1-3a00-93d1-46196ee77204,
'7db1a490-5878-11e2-bcfd-0800200c9a66',
'No One Rides for Free', 'Fu Manchu', 'OjoRojo');
After inserting the example data into playlists, the output of selecting all the data looks like this:

```
SELECT * FROM playlists;

 id            | song_id     | album         | artist          | title
-------------+-------------+-------------+---------------+--------------------
 62c36092... | 2b09185b... |     Roll Away | Back Door Slam | Outside Woman Blues
 62c36092... | 8a172618... | We Must Obey |       Fu Manchu |    Moving in Stereo
 62c36092... | a3e64f8f... | Tres Hombres |          ZZ Top |           La Grange
```

The CQL 3 data maps to the storage engine's representation of the data as follows:


playlists
62C36092 ...
a3e64f8f ...
La Grange, ZZ Top, Tres Hombres
8a172618 ...
Moving in S..., Fu Manchu, We Must Obey
2b09185b ...
Outside Wo..., Back Door Slam, Roll Away

id	song_id	title	artist	album
62C36092 ...	a3e64f8f ...	La Grange	ZZ Top	Tres Hombres
62C36092 ...	8a172618 ...	Moving in Stereo	Fu Manchu	We Must Obey
62C36092 ...	2b09185b ...	Outside Wo...	Back Door Slam	Roll Away

The solid lines show how the cell values get unpacked into three CQL 3 columns. The dotted lines show how the clustering key portion of the compound primary key becomes the song_id column. Presenting a storage engine row as a partition of two or more object rows is a more natural row-column representation than earlier CQL versions and the Thrift API(MongoDatabase, 2015).

Creating a Table using HBase Shell

You can create a table using the create command, here you must specify the table name and the Column Family name. The syntax to create a table in HBase shell is shown below.

```
create ‘<table name>’,’<column family>’
```

Example

Given below is a sample schema of a table named emp. It has two column families: “personal data” and “professional data”.

Row key	personal data	professional data

You can create this table in HBase shell as shown below.

```
hbase(main):002:0> create 'emp', 'personal data', ’professional data’
```

And it will give you the following output.

```
0 row(s) in 1.1300 seconds
=> Hbase::Table - emp
```

Verification

You can verify whether the table is created using the list command as shown below. Here you can observe the created emp table.

```
hbase(main):002:0> list
TABLE
emp
2 row(s) in 0.0340 seconds
```

Creating a Table Using java API

You can create a table in HBase using the createTable() method of HBaseAdmin class.

This class belongs to the org.apache.Hadoop.hbase.client package. Given below are the steps to create a table in HBase using java API.

Step1: Instantiate HBaseAdmin

This class requires the Configuration object as a parameter, therefore initially instantiate the Configuration class and pass this instance to HBaseAdmin.

```
Configuration conf = HBaseConfiguration.create();
```

```
HBaseAdmin admin = new HBaseAdmin(conf);
```

Step2: Create Table Descriptor

HTableDescriptor is a class that belongs to the org.apache.Hadoop.hbaseclass. This class is like a container of table names and column families.

```
//creating table descriptor
HTableDescriptor table = new HTableDescriptor(toBytes("Table name"));

//creating column family descriptor
HColumnDescriptor family = new HColumnDescriptor(toBytes("column family"));

//adding coloumn family to HTable
table.addFamily(family);
```

Step 3: Execute through Admin

Using the CreateTable() method of HBaseAdmin class, you can execute the created table in Admin mode.

```
admin.createTable(table);
```

Given below is the complete program to create a table via admin.

```
importjava.io.IOException;
importorg.apache.Hadoop.hbase.HBaseConfiguration;
importorg.apache.Hadoop.hbase.HColumnDescriptor;
importorg.apache.Hadoop.hbase.HTableDescriptor;
importorg.apache.Hadoop.hbase.client.HBaseAdmin;
importorg.apache.Hadoop.hbase.TableName;
importorg.apache.Hadoop.conf.Configuration;
publicclassCreateTable{
publicstaticvoid main(String[]args)throwsIOException{
// Instantiating configuration class
Configuration con =HBaseConfiguration.create();
// Instantiating HbaseAdmin class
HBaseAdmin admin =newHBaseAdmin(con);
// Instantiating table descriptor class
HTableDescriptortableDescriptor=new
TableDescriptor(TableName.valueOf("emp"));
// Adding column families to table descriptor
tableDescriptor.addFamily(newHColumnDescriptor("personal"));

tableDescriptor.addFamily(newHColumnDescriptor("professional"));

// Execute the table through admin

admin.createTable(tableDescriptor);

System.out.println(" Table created ");

}

}
```

Compile and execute the above program as shown below(MongoDatabase, 2015).

```
$javac CreateTable.java
$java CreateTable
```

The following should be the output:

```
Table created
```