

---

# NOSQL DATABASES AND CASSANDRA

---

Semester Project: Advanced Databases



DECEMBER 14, 2015  
WANG CAN, EVABRIGHT BERTHA  
Université Libre de Bruxelles

## Preface

The goal of this report is to introduce the new evolving technology of database management systems (DMSs). When people are introducing to the notion of databases, the most famous and established relational database management system (RDBMS) model is firstly presented to them. Therefore, one begins to think that such relational technology can easily handle voluminous, and variety nature of data as described in three Vs of Big Data known as velocity, variety, and volume.. For sure that is not true, there are other models that have been proposed so far to provide service to the databases community.

The new technologies introduced in the literature include NoSQL databases. We are going to study this class of database technologies known as the column Family database (column-store) and analyze the tool known as Cassandra, which handle such column-store databases, analyze its properties and point out its limitations it.

We begin by introducing the Cassandra, its features and properties, and give a brief introduction of NoSQL. We then describe the difference between relational databases and NoSQL, and test the different properties of Cassandra based on studied work. We would also comment on the testbed of Cassandra on local machine environment at smaller scale, hence can only comment on its properties and features.

## Table of Contents

INTRODUCTION.....	3
Tools and Technologies.....	4
A. NoSQL :.....	4
Schema:.....	4
Data placement.....	4
Extensibility:.....	5
Partition:.....	5
Asynchronous replication:.....	5
Architecture:.....	5
B. <b>Categories</b> .....	6
Apache Cassandra:.....	6
MongoDB:.....	7
Hypergraph:.....	7
C. <b>CHARACTERISTICS</b> of NoSQL Storage Systems.....	7
a) Key-Value Store:.....	7
b) Column store databases.....	7
c) Graph databases.....	8
d) Document oriented databases.....	8
D. <b>Relational DATABASE (RDBMS) V.S NoSQL</b> .....	8
E. <b>Example Case Study: CASSANDRA</b> .....	9
Cassandra:.....	10
CASSANDRA ARCHITECTURE.....	11
Properties of Evaluation.....	12
Cassandra Query Language: CQL.....	14
CASSANDRA DATA MODEL.....	14
Data Models of Cassandra and RDBMS.....	17
CASSANDRA KEYSPACE OPERATION.....	17
Case Study:.....	18
Conclusion.....	21
References.....	22

## INTRODUCTION

In the last decade, there have been significant increase in data generation from variety of sources including monitoring sensors, medical devices, networking and social media sources and websites, and real-time applications. These sources generate data in different formats from structured (relational data), unstructured (text, files, etc.), and semi-structured data (XML, Tab-separated, key-val etc.). All these variant data structures cannot be handled by the traditional relational database management systems (RDBMS) because they only handle relational structured data. Similarly, the huge volume of data can also not be efficiently handled due to dynamic scale up and down to cope with changing demands of online data generation tools and sources. These recent advancements pose serious challenges to handle such data in Tera bytes, and Exa bytes per second in a fault tolerant flexible system. However, when large scale data and high need of concurrent processing appear, plenty of serious problems arise, relational databases are not suitable to tackle such situations especially in case of handling variety and voluminous data that is dynamic.

SQL is one of the earliest RDBMS management engine and has lots of advantages. It is a rich language and many tools exist to support it. All the classical relation databases follow ACID rules which provide availability, consistency, integrity to the data and user accessing it. ACID stands for A-Atomic-Consistency, I-isolation, D-durability. In modern internet, classical RDBMS face large scale data size, high input/output rate, frequent changes, and their requirement are not the same as ACID for RDBMS. RDBMS is best suited for fixed schema and statically defined queries to be executed on the data to process transactions from users.

With the passage of time and the advancement of sensor and internet of things, new technologies emerged as data management including column-store, row-store, key-value store etc. A NoSQL database is designed in order to solve the problems of large scale data. NoSQL stands for Not Only SQL. All the databases store, process, and retrieve data, but NoSQL is special in a sense that It handles large volume of data in a column-storage model and was introduced in 2009. In the coming sections we introduce some of the technologies and tools that have been introduced in the literature as data management technologies.

# Tools and Technologies

## A. NoSQL :

NoSQL is a distributed system, it consists of multiple computers through a network, it is because of the characteristic of software, a distributed system is cohesion and transparency. Brewer's CAP Theorem states that we cannot guarantee more than two of three basic requirements for a distributed system. They are coherence, availability and partition tolerance.



Figure 1 Brewer's CAP theorem

If you want availability and consistency, means all the nodes are in connect, but when partition appears, system will block. If you want consistency and partition tolerance, it means maybe some data will not in contact, but rest nodes still work, usually performance is not high. If you want availability and partition tolerance, it means system is always available under partitioning, but less accuracy.

There are lots of implementations we can use now. The most popular NOSQL databases are MongoDB, Apache Cassandra, Redis, Solr, Elasticsearch, HBase, Splunk, Memcached, and Neo4j.

In general, NoSQL systems have different features as compared to typical relational systems. All NoSQL systems have some common features described as follows:

### Schema:

These NoSQL systems are normally less schema oriented at design time. They do not need to define the data model, the structure of the predefined table. Each record in the data may have different properties and formats and attributes. When you insert data, you do not need to define their patterns.

### Data placement:

In NoSQL stores, the data after insertion is stored on the local server. Because the performance of the data from the local disk is better than the performance of the data transmission through the network, the performance of the system is improved. The transmission only takes place in case there is need to move.

## Extensibility:

These systems run with the dynamic increasing or deleting nodes which means dynamically scaling up and down in response to the needs of data sources and needs of computation. Without stopping, data migrate automatically. Various tools have been described to support dynamic data migration without system downtime such as Squall.

## Partition:

NoSQL database disperse the data, the record is scattered on multiple nodes. So it can improve the performance of parallel, and guarantee that no single point of failure.

## Asynchronous replication:

In this way, the data can be written as soon as possible to a node, and will not be caused by the network transmission delay.

## Architecture:

In this section, we are going to explain how is the architecture of a NOSQL database different to RDBMS especially in data read and writes, which characteristics in NOSQL database differ with RDBMS, and show some examples of the NOSQL works applications.

In figure 2 below, the architecture of some of the NoSQL stores is mentioned. These store manage data in different formats including column-store, row-store, document-store, and key-value store.

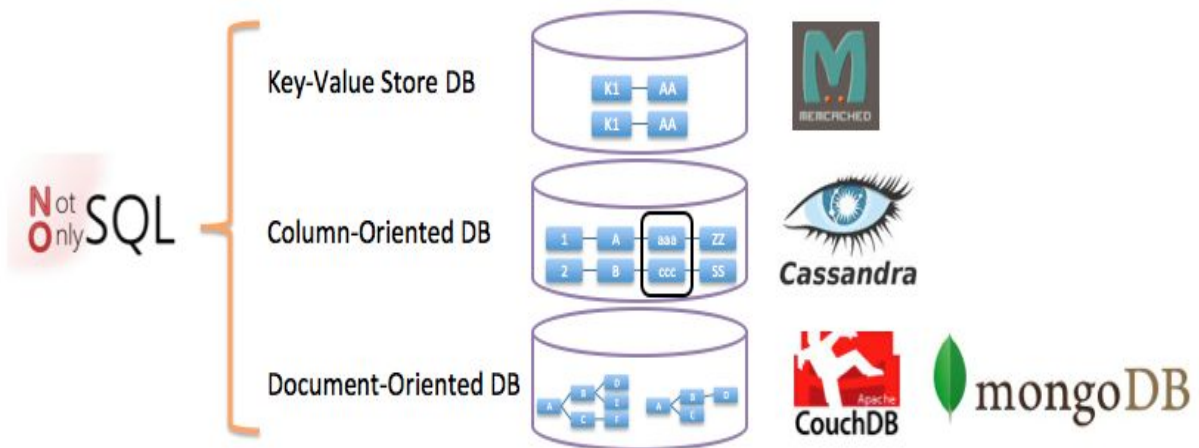


Figure 2 NoSQL storage systems

Similarly, compared to RDBMS, inspite of the change in its data model, there is also significant change in its distributed query read and writes. In the following figure 3, a NoSQL system can write as concurrent operation to different nodes while an RDBMS has to write to a single node. This key feature makes NoSQL fast, efficient.

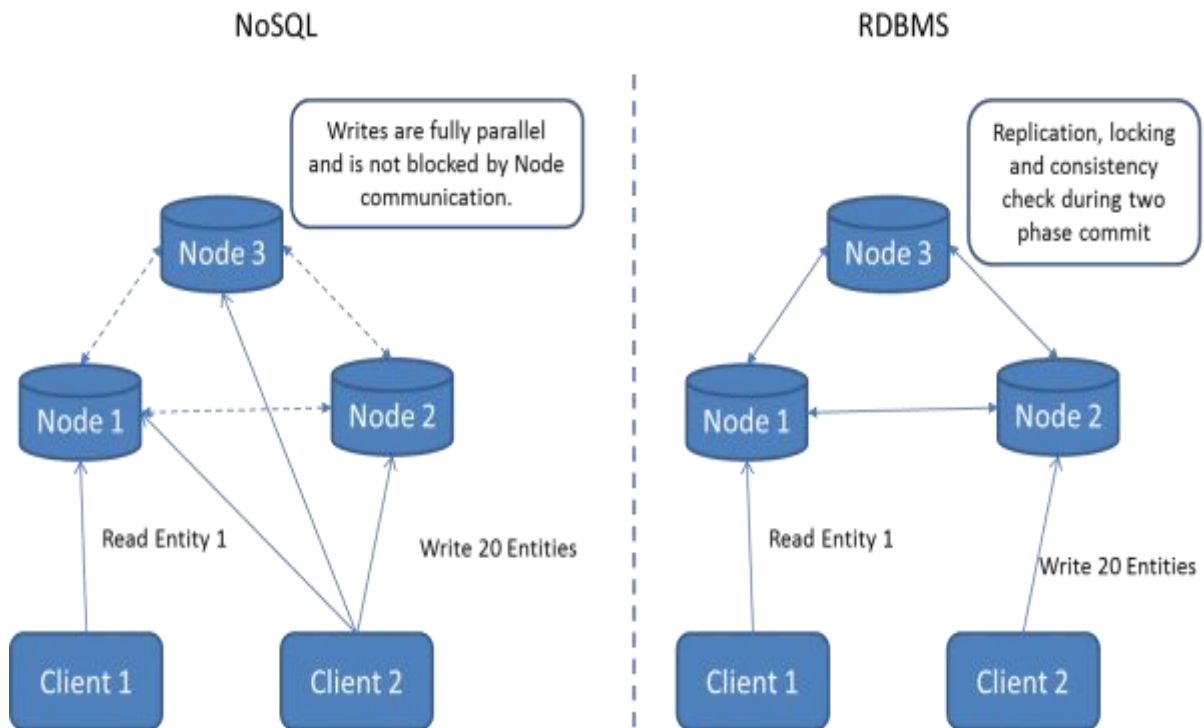


Figure 3 Comparison of Processing NoSQL and RDBMS

## B. Categories

There are four kinds of NoSQL databases.

1. Key-value stores (KV Durable, KV Volatile)
2. Column store
3. Graph
4. Document store

Different tools have different kinds of NoSQL databases, for example

- Memcached and Redis for KV Volatile.
- Infobright and Monet DB for Column Store
- Riak, Voldemort and Dynamo for KV Durable
- HyperGraph and Neo4j for Graph.
- eXist, MongoDB and CouchDB for Document Store

In the following sections, we briefly describe some of these technologies.

### Apache Cassandra:

It is a distributed, column oriented database, extremely scalable and fault tolerant. It provides no single point of failure, great performance, open source, decentralized, elastically scalable with high availability and made to handle huge amount of data structured.

## MongoDB:

MongoDB fall into the class of document-oriented database system, it is an open source product, highly scalable with high performance, developed and supported by 10gens.

## Hypergraph:

Hypergraph is a graph storage tool that is mostly used to store graph data in the form of vertices, and edges or in the form of triples as in RDF graphs. Neo4J is a its graph processing framework to process the data in a distributed manner and efficiently.

## C. CHARACTERISTICS of NoSQL Storage Systems

### a) Key-Value Store:

Key-value stores are the most basic types of NoSQL databases. In the key-value storage, Key-values have no type constraint and each key is unique .Key-Value stores follow the 'Availability' and 'Partition'. Key-value stores usually use B-trees or extensible hash tables.

For example, any record insert would be: `put (key, value)`

And data retrieval is like the function to retrieve data based on keys: `get (key)`

The returned value can be assigned to another variable: `value = get (key)`

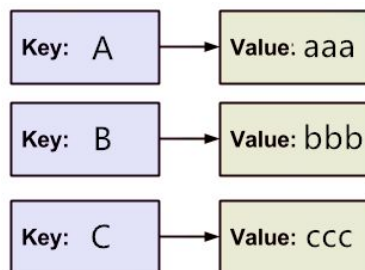


Figure 4: Key-value Store

### b) Column store databases

Column store databases store data in column, a single column are stored contiguously, and every column is treated individually. Query works by columns too, so the performance of queries is better than other ways, because in this way, queries can access specific column data.

Because all the data in the same column are same type, so column store databases is easy to compress.



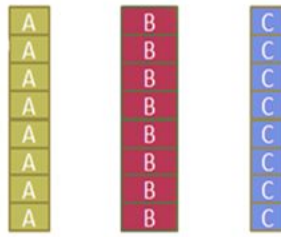


Figure 5: Column Store DB example

### c) Graph databases

Graph data structure consists of edges and nodes. Graph database stores data in graphs. It clearly shows any kind of data in an easy way. Each node stands for an entity and each edge stands for a connection between two nodes. Every node and edge is unique and knows the relationship between other nodes around it. If the amount of nodes becomes large, the cost also increases.

### d) Document oriented databases

Document oriented databases store data in documents. A document is a key value collection where the key allows access to its value. Documents are not typically forced to have a schema and therefore are flexible and easy to change. Documents are stored into collections in order to group different kinds of data. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents. Document stores are just like normal filesystems and the most common include Hadoop distributed file system (HDFS), Google file system (GFS) etc.

## D. Relational DATABASE (RDBMS) V.S NoSQL

The difference between relational database management system such as SQL server, ORACLE, MySQL, PostgreSQL and NoSQL Cassandra, Hbase, MongoDB can be described in detail in regards to each system differently. However, in this section, we briefly describe their key differences.

Before describing the details of differences, we describe some basic notions.

**Structured data:** It is the data in the form of tuples consisting of attributes such as in relational databases.

**Unstructured data:** In this type of data, it includes all the data that does not conform to a formal structural semantics including HTML, text, and other types of data such as sensory information.

**Semi-structured data:** It is mostly referred to as XML, CSV, and XHTML data where some form hierarchy and fields are maintained to define linkage in the data.

Hence, based on these definitions, we describe the differences as follows:

- RDBMS are made to handle structured data whereas NoSQL handle semi-structure and unstructured data

- RDBMS support complex query languages whereas NoSQL support simplified query language
- RDBMS are design to handle ACID(atomicity, consistency, isolation and durability) whereas NoSQL are design on the premise of eventual consistency
- RDBMS support transaction whereas NoSQL does not support transaction
- NoSQL systems are envisioned to perform wide area of big data analytics while RDBMS only support traditional BI analytics.
- NoSQL systems are adaptive and extensible while RDBMS are rigid.
- NoSQL systems are memory efficient and many in-memory and realtime systems have been developed while RDBMSs are mainly fixed architecture and does not support real-time streaming.

## E. Example Case Study: CASSANDRA

Before introducing what Cassandra is and how it works actually, let us discover the different challenges which relational database management system find it difficult to deal with.

The interest of most people is to obtained high availability as fast as possible when looking for a data or an information

When there is huge amount of data, it need to be distributed and partition, if the data are not partition or distributed, or if the data is centralized searching or updating the data will be time consuming and many people get annoyed with such situation, to avoid this situation distribute and partition your data and this data need to be available in real time and one of the difficulty we may face here is ensuring consistency.

*What are the common Key features of Cassandra?*

1. Huge data Scheme: Cassandra is made to handle high volume of data
2. Fast random Access
3. The data processing and the data access need to act fast
4. Variable schema
5. We equally need to have high flexibility: This enable flexible storage, flexible structure or schema and avoid time consuming if we need to be following a structure when modifying or adding something.
6. Need of compression: since Cassandra handle huge data, we need technique to be able to compress the data and are one of the key requirements.
7. High availability : Availability guarantee that every request receives a response about whether it succeeded or failed
8. Need for consistency : all nodes see the same data at the same time
9. Need of distribution or partition tolerance: the system continues to operate despite arbitrary partitioning due to network failures

Recalling the CAP theorem which states that: *“having a computer system, it is impossible to guarantee availability, consistency or distribution equally in your system.”*

Therefore, if the key thing in my system is to guarantee consistency, you may need to sacrifice either availability or partition tolerance.

RDBMS guarantee consistency; ensure availability and partition tolerance is compromise whereas Cassandra key aspect is to ensure availability and partition tolerance at the cost of consistency. Other system such as MongoDB, HBase, and Redis offers Consistency and partition tolerance.

Therefore, based on your requirement you can go in for Cassandra, RDBMS or others base on what you want to ensure.

## Cassandra:

Apache Cassandra is defined based on its features: It is a distributed, column oriented database, extremely scalable, fault tolerant meaning with no single point of failure, great performance, open source, decentralized, elastically scalable with high availability and made to handle huge amount of data structured.

Cassandra is used in the most famous social platforms such as Face book, Twitter and in much other application Google, Netflix, Cisco and others.

### *Brief recall of Cassandra Origin*

In terms of Cassandra story line;

- it is basically a combination of Google big table and Amazon Dynamo which face book build for its search inbox,
- in March 2009 apache foundation made Cassandra an open source so that everyone can benefit from Cassandra
- Equally, since February 2010, apache made of Cassandra one of its main projects.

### *The main structure of Cassandra*

The main items of Cassandra structure are:

- **Node:** The first basic structure of Cassandra is Node. A node is responsible of storing data in Cassandra
- **Data center:** A data center is a group of node which are related to each other. A data center can be physical or virtual. It is advisable to used different datacenter for each workload
- **Cluster:** A cluster is a collection of one or more data center
- **Commit log:** The commit log is an important component in Cassandra and responsible for crash-recovery mechanism. When data are written, it is firstly written in the commit log and later be transfer to the SSTable.
- **Table:** A is a collection of column in which data can be retrieve from it by means of row and each row have a key.
- **Mem-table:** It is a memory resident for data. Data are immediately transfer to the mem-table after the commit log
- **SSTable:** Store data on a disk in a file, this data are retrieving from the mem-table.
- **Bloom filter:** These are algorithm used to verify whether a certain element belong to a particular set.

## CASSANDRA ARCHITECTURE

In terms of Cassandra architecture, Cassandra was really build on the ground with the understanding that system/hardware failures can and do occur, so because of this, it was design as Peer-to-peer distributed system database management system, meaning all nodes are the same in Cassandra meaning, there is no concept of the master nodes or the slave node, or other thing alike.

Cassandra automatically partitioned the data that you write in the database across all the nodes in the cluster and you have the possibility to control the data replication with Cassandra to ensure fault tolerance meaning, you determine the number of copies you want duplicated on node database which participate on a particular cluster

Moreover, because of the Peer-to-peer design, Cassandra is then a Read and write anywhere design, this means you can read and write on any node in Cassandra

The nodes in the same cluster have the same function and are independent of each other. Cassandra uses the Gossip protocol to enable nodes of the same cluster to communicate with each other, which exchanges information across the cluster every second

A earlier seen above, when you write data in Cassandra, it is first written in a commit log which ensure data durability and then the data is also written to an in-memory structure called the mem-Table and when the mem- Table becomes full is the push to disk to another structure called an SStable meaning storage screen table.

The scheme used in Cassandra is mirrored after Google big table. It is a row -oriented, column structure. Cassandra have what is called keyspace it is akin to database in the RDBMS world. Cassandra used the notion of column family and is similar to an RDBMS table but it is more flexible and dynamic, used to manage data. In Cassandra, row in a column family can be indexed by its key. Other column may be indexed as well.

In the figure 6, we have used the actual Cassandra architecture from Cassandra. In this architectural diagram, there are two Cassandra clusters each consisting of various number of nodes and a web client accesses both of these clusters and user queries are processed for processing and analytics. The middle tier architectural part of the architecture works as configuration for the cluster where each cluster configuration files reside and the expired sessions from clients are managed. Cassandra is a distributed system which can be scaled-up/down based on the needs. It also supports data replication for efficiency and fault tolerance, whenever a node fails or is straggling executing tasks, then its load can be scheduled on another node or another node can be started to run its data or data from one of its replication.

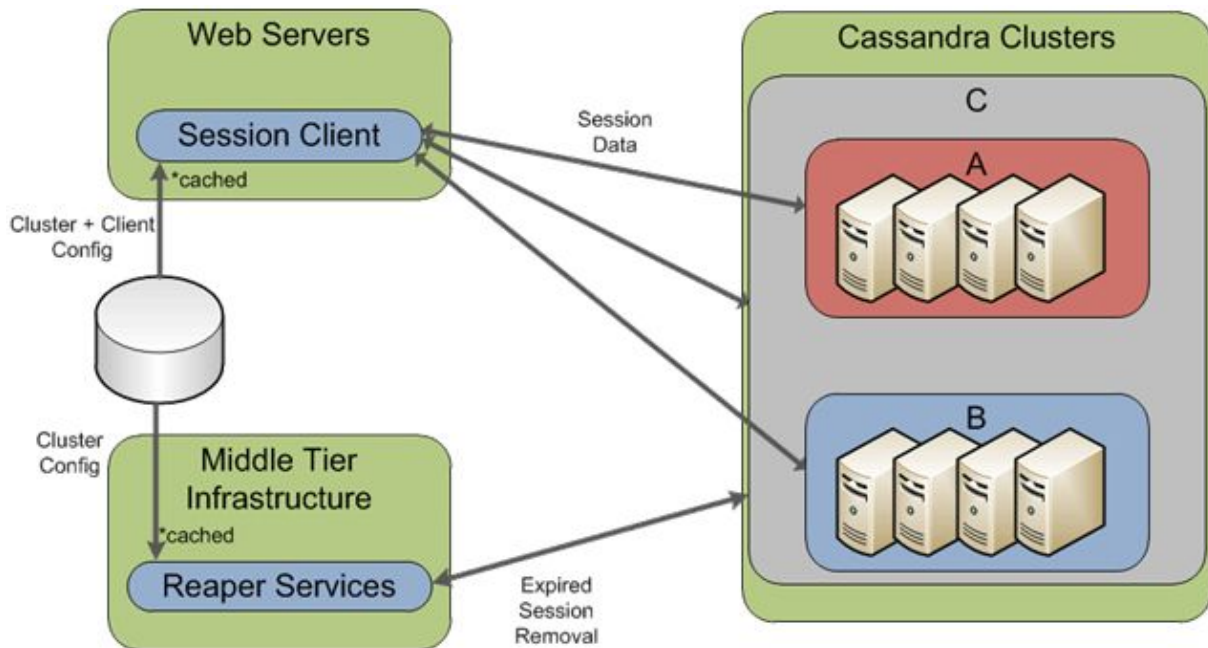


Figure 6: Cassandra Architecture

## Properties of Evaluation

### Data Replication in Cassandra

In Cassandra, one or more of the nodes in a cluster act as replicas for a given piece of data. If it is detected that some of the nodes responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a **read repair** in the background to update the stale values.

The following figure shows a schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure.

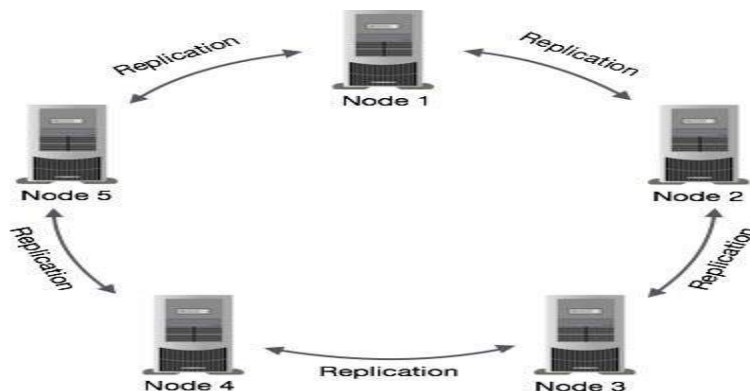


Figure 7 Cassandra Replication

Note – Cassandra uses **the Gossip Protocol** in the background to allow the nodes to communicate with each other and detect any faulty nodes in the cluster.

Cassandra has a variety of different features and benefit as earlier cited above, let us go through some of them.

### *SCALABILITY:*

Cassandra is a well-known as big data scalable meaning that is scalable style design is really capable of moving mega bytes to Terabytes very easily and you can add new nodes to the cluster online and this actually increases the linear performance, e.g. you can move from 4 nodes to the cluster to 8 nodes, thus doubling by throughput capability.

### *No Single point of failure:*

As mention earlier, Cassandra also offer no single point of failure because of its peer-peer architecture and all nodes are the same, and the Customized replication affords tunable data redundancy, so you have to possibility to decide how many tune of data you want propagated to the various node in the system. There is also the possibility to Read and write from any node.

Cassandra is also capable of replicating data between different physical data center racks. Therefore, there is a copy of data on another rack such that, if a first rack failed actually, the data is save on the second rack.

### *Easy Replication/ Data Distribution:*

Cassandra also offers very easy replication capable. Everything is transparently handled by Cassandra. This replication can be done with either one datacenter or with Multi-data center capable. Because of this, Cassandra exploits all the benefits of cloud computing. It also enables a high level implementation where you may have data in the cloud and other data on premise.

### *No Need for caching Software:*

Because, the Peer-peer architecture removes need for special caching layer and programming that goes with it, Cassandra have no need of caching software, though some employee may add some catching software but they don't really need to do that with Cassandra. The database cluster uses the memory from all participating nodes to cache the data assigned to each node and there is no irregularities between a memory cache and database are encountered.

### *Tunable Data Consistency:*

Cassandra also offer what is called tunable data consistency, Choose between strong and eventual consistency (all to any node responding) depending on the need. It can be done on a per- operation basis, and for both reads, writes and equally it Handles Multi-data center operation.

### *Flexible Scheme:*

Cassandra uses a column Family to store the data inside Cassandra keyspace, the schema with the column family is much more dynamic schema design allows for much more flexible data storage than rigid RDBMS and allow to handles structured, semi-structured and unstructured data. If you have to make a particular change to the column family, counters also support no offline and downtime for schema changes. Cassandra equally support primary and secondary indexes.

### *Data Compression:*

Cassandra uses Google's snappy data compression algorithm and compresses data on a per column family level. This compression has no performance penalty and some increases the overall performance due to less physical I/O.

### Cassandra Query Language: CQL

Cassandra uses the CQL language which stands for Cassandra Query Language, which mapped to RDBMS SQL language syntax. When you are creating column family

### CASSANDRA DATA MODEL

The data model of Cassandra is significantly different from what we normally see in an RDBMS. This chapter provides an overview of how Cassandra stores its data

#### *Cluster*

A cluster is basically a container of different key spaces. Basically in cluster we have different nodes and Key spaces (which are nothing but databases). In Cassandra we have different nodes connected to each other to form a cluster and Cassandra reorder those nodes in the form of a ring. Cassandra is design in the way that is distributed on a large number of machine operating

#### *Keyspace*

A Keyspace in cassandra is analogues to database in RDMS, as we said cluster is a container for key spaces and key space are used to logically group column families together. There existe some attribute which we can set up at the keyspace level which described the keyspace –wide behavior and each key space have a name. Some attributes which we can defined at the level of key space are

- **Replication factor:** A replication factor is nothing but the number of application you want to maintain in that keyspace
- **Replication placement strategy:** The replication placement strategy defined how data is place on the ring. Cassandra have basically two difficult strategies which is using. Firstly, the **simple strategy** is network unaware, it does not know whether there are differents datacenters or different racks, the simple strategy assume your network to have a single rack, and some time called the rack-aware strategy. Secondly, the **network topology strategy** in contrast is ware and understands that the network may have different datacenter and at the same time that each dta center may have multiple rack, and this strategy is usually known as datacenter-shared strategy.
- **Column families:** The Keyspace can have one or multiple column families. The column family is similar to tables in RDBMS. Cassandra defines a column family to be a logical division that associates similar data. Column families on the otherside have multiple rows, where each row contains organized column. Therefore looking at the column family, you obtained the structure of your data.

There exist have different logical types of column family

- a. **The static column family:** The static column family is a column family where the column name is defined at the time of entries. So when you defined a column family, you have the option to define it at the time of creation or not. Thus, it uses a relatively static set of column names and is more similar to a relational database table
- b. **The dynamic Column Family:** In the dynamic column family we don't defined the column family name at the time of entries, it is allow to takes advantages of Cassandra ability to use arbitrary application supplied column names to store data. A dynamic column family allows to pre compute result set and stores them in a single row for efficient data retrieval.

The syntax of creating a Keyspace is as follows –

```
CREATE KEYSPACE Keyspace name
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
```

The following illustration shows a schematic view of a Keyspace[2].

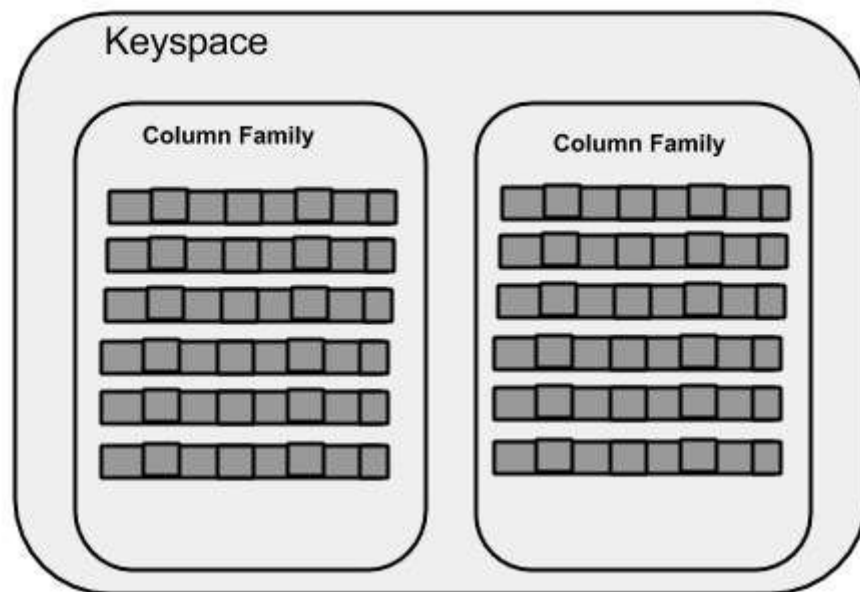


Figure 8: KeySpace Example

The difference between Cassandra column family and relational table

1. Cassandra column family is schema free meaning you can add a number of column family at any time and that is one of the best features of Cassandra, making your application more scalable whereas the relational table does not have a flexible schema, the schema here is fixed.
2. Cassandra column family has two principle attributes, Name and Comparator. So, when you defined Cassandra column family, you have to give a given name which is



mandatory and the comparator is basically the datatype for the column name whereas in relational table, there is no comparator for the table

3. Cassandra has a concept known as column families which allows for nesting that is it allows grouping of columns whereas relational database does not allow the grouping of column

Cassandra column family has a number of attributes:

**Keys\_cached:** The keys cached gives the number of places to store cache per SStable

**Rows\_cached:** The rows cached gives the number of rows which content can be store in memory

**Preload\_row\_cache :** The preload row cache is used when you want to pre-populate the row cache.

The following figure shows an example of a Cassandra column family.

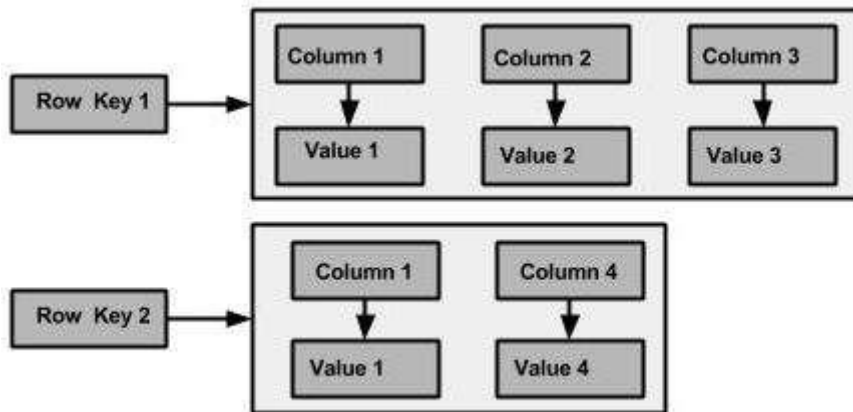


Figure 9: Key Structure of Cassandra

**Column:** A column is the smallest increment of data structure in Cassandra. Cassandra column contain the following values; A column name, value, a time stamp, expiring column and counter

**Super Columns:** A super column is nothing but a grouping of some column to provide a certain needs.

Column which you can be brought to query together, bring them together and form a super column family

Super Column	
name : byte[]	cols : map<byte[], column>

Figure 10: Super Column Cassandra

## Data Models of Cassandra and RDBMS

Therefore the difference between Cassandra data Model and RDMS data model are described as follow:

1. RDBMS deals with structured data whereas Cassandra deals with unstructured data. It has a fixed schema whereas Cassandra has a flexible schema.
2. In RDBMS, a table is an array of arrays. (ROW x COLUMN) whereas In Cassandra, a table is a list of "nested key-value pairs". (ROW x COLUMN key x COLUMN value)
3. Database is the outermost container that contains data corresponding to an application whereas Keyspace is the outermost container that contains data corresponding to an application.
4. Tables are the entities of a database whereas Tables or column families are the entity of a keyspace.
5. Row is an individual record in RDBMS whereas Row is a unit of replication in Cassandra.
6. Column represents the attributes of a relation whereas Column is a unit of storage in Cassandra.
7. RDBMS supports the concepts of foreign keys, joins. Whereas Relationships are represented using collections.

## CASSANDRA KEYSPACE OPERATION

### *Creating a Keyspace using Cqlsh*

A keyspace in Cassandra is a namespace that defines data replication on nodes. A cluster contains one keyspace per node. Given below is the syntax for creating a keyspace using the statement **CREATE KEYSPACE**.

### *Cassandra Experiment and Testing*

#### **Cassandra cluster setup:**

During this literature study and experimentation, we performed Cassandra experiment on our local machines. The setup include the following:

Virtual box based virtual machines, machine has 2GB RAM and we assigned 50 GB HDD. In the first step, we setup Cassandra on Linux operating system as follows:

- Preliminaries installation
  - Java version 7.
  - SSH setup and password less access to local host
- Cassandra Setup:
  - Following the online tutorials, we first setup the configurations of Cassandra and directory structures.
  - We setup Cassandra functional and performed necessary experimentation of using it for practical application as described below.

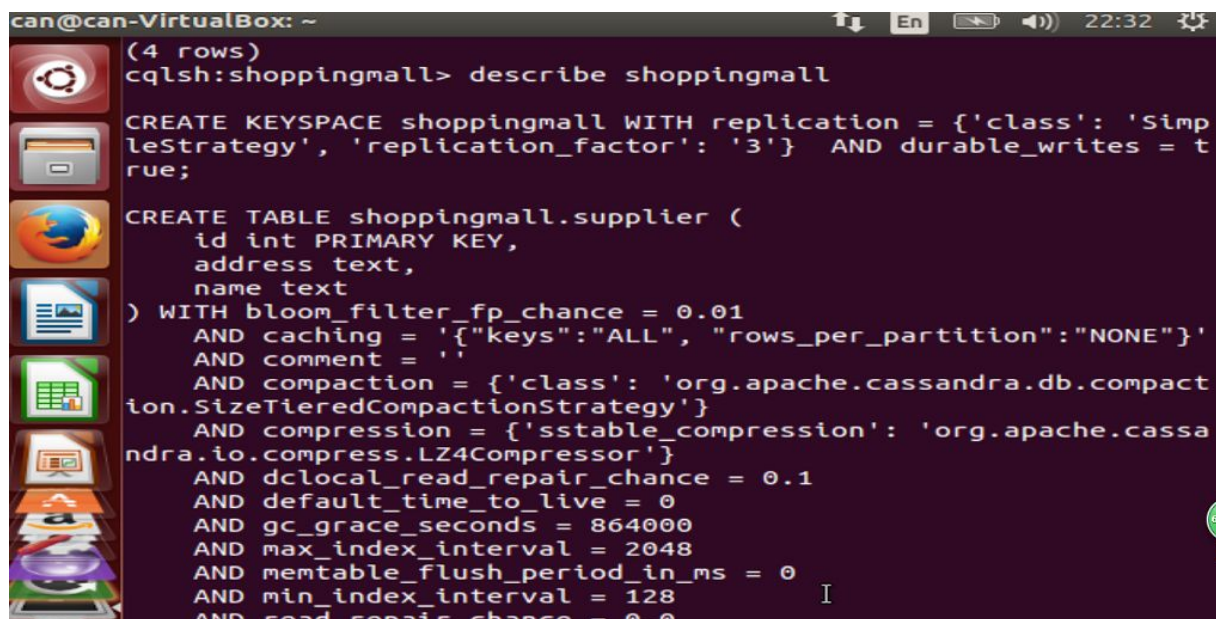
## Case Study:

We adopted the ShoppingMall case study, where we have many customers ordering items from the mall and we have suppliers who deliver them their ordered items. This is a small case study just to understand the working and understanding of Cassandra and how to develop applications in it using either CQL or using Eclips in Java.

We created a keyspace with 2 replication factor as we are only running on local virtual box machines. And the keyspace is named as “ShoppingMall” with four preliminary tables as below:

- Customers
- Suppliers
- Orders
- Items

Inserted some data into tables. These tables creation, query testing, and data manipulation were performed using CQL to test command line testing of Cassandra. The following figure shows snapshot of these commands and the Keyspace description.



```
can@can-VirtualBox: ~
(4 rows)
cqlsh:shoppingmall> describe shoppingmall

CREATE KEYSPACE shoppingmall WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'} AND durable_writes = true;

CREATE TABLE shoppingmall.supplier (
  id int PRIMARY KEY,
  address text,
  name text
) WITH bloom_filter_fp_chance = 0.01
AND caching = '{"keys": "ALL", "rows_per_partition": "NONE"}'
AND comment = ''
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'}
AND compression = {'sstable_compression': 'org.apache.cassandra.io.compress.LZ4Compressor'}
AND dclocal_read_repair_chance = 0.1
AND default_time_to_live = 0
AND gc_grace_seconds = 864000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair_chance = 0.0
```

Similarly, we executed CQL commands to create, alter, delete, drop, and insert data to manipulate tables in Cassandra.

### *Case Study implementation in Eclipse Java:*

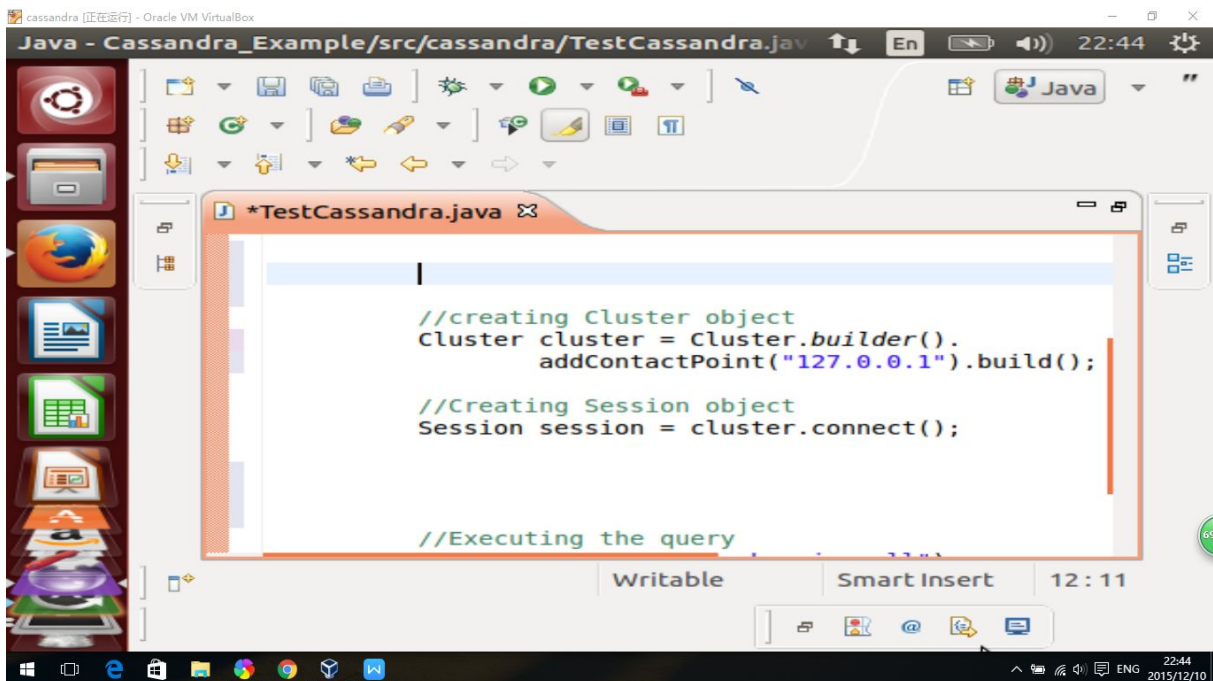
Apart from the command line testing of Cassandra, we used Java eclipse to interact with it. In Java eclipse setup, the following steps are necessary prior to implementation:

- Eclipse installation
- Jar files necessary to import form Cassandra, they include:
  - Cassandra-driver-core-2.0.2.jar
  - Guava-16.0.1.jar

- Metrics-core-3.0.2.jar
- Netty-3.9.0-Final.jar
- Slf4j-api-1.7.5.jar

After all these setup and configuration, we use these jars to communicate with Cassandra server in java programming in eclipse.

First of all, using the builder method of the referenced API, we try to connect the contactpoint at the localhost and create a connection. The snapshot of the program is shown in the following figure.



```
Java - Cassandra_Example/src/cassandra/TestCassandra.java
//creating Cluster object
Cluster cluster = Cluster.builder()
    .addContactPoint("127.0.0.1").build();

//Creating Session object
Session session = cluster.connect();

//Executing the query
```

*Connect to LocalCassandra Server*

Once the server is connected, then using the keyspace through the 'Executequery' method, we can select the keyspace, and perform various insertion, deletion, and other query manipulations. A simple example of data insert ion is shown in the following snapshot.

```
Java - Cassandra_Example/src/cassandra/TestCassandra.java
class TestCassandra {
    public static void main(String args[]){

        //Query
        String query = "insert into orders(id, ";
        query=query+"customerid, quantity, supplierid) values
        //creating Cluster object
        Cluster cluster = Cluster.builder().addContactPoint("1
        //Creating Session object
        Session session = cluster.connect();
        //Executing the query
        session.execute("use shoppingmall");
        session.execute(query);
        System.out.println("Data inserted");
    }
}
```

*:Cassandra Insert data using Eclipse Java*

Apart from these simple manipulations, we were able to import data from files into the database just like in a simple SQL based data.

From our experience with Cassandra, we are able to use Cassandra for application building and using it for our analytical purposes in future and its storage system is very useful and easy. For example, unlike traditional RDBMS, we were able to create column families, and then insert into tables with some columns droppings and this experiment clearly tell us incorporation of any kind semi-structured or unstructured data use in Cassandra. However, in RDBMS, we can not use this kind of data.

In conclusion, Cassandra is an easy to use NoSQL database framework with a lot of easy to use, experience, and good things in hand. It is mostly suitable for large scale data manipulation and processing.

## Conclusion

In this report, we have described the properties, features, data model, architecture, and language specifications for NoSQL data management systems especially Cassandra in detail. We have compared various properties of the traditional relational database management systems (RDBMS) and NoSQL systems. No doubt, RDBMS systems are the most widely used systems in commercial industry applications from last few decades. However, with the advancement in new hardware technologies and in the era of Internet of Things (IoT), it is un-deniable to have such high level data models which can accommodate heterogeneous data sources and also provide scalable data management mechanisms. There are initiatives which also incorporate real-time data management along with traditional historical analytics data. These new advancement pose challenges for technology experts to design and model flexible technology that can cope with demand of the advanced and new technology trend. Bog data has recently compelled the community to process large amount of data on highly scalable streaming, and file systems with heterogeneous data such as HDFS, MapReduce, Hive, HBase, and many more.

## References

- <http://inside.godaddy.com/wp-content/uploads/2012/04/ArchitectureOverview.gif>
- [https://www.google.be/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjTpN\\_kztbJAhVEWRoKHWW\\_D4IQjxwIAw&url=http%3A%2F%2Fwww.tutorialspoint.com%2Fcassandra%2Fcassandra\\_data\\_model.htm&psig=AFQjCNFPHTCvdKye98u\\_qHiiiPh3LPqVQw&ust=1450019777467270](https://www.google.be/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjTpN_kztbJAhVEWRoKHWW_D4IQjxwIAw&url=http%3A%2F%2Fwww.tutorialspoint.com%2Fcassandra%2Fcassandra_data_model.htm&psig=AFQjCNFPHTCvdKye98u_qHiiiPh3LPqVQw&ust=1450019777467270)
-