

414466: Software Laboratory-VI

BEIT (2012 Course) Semester - II

Teaching Scheme	Examination Scheme	
Practical : 04 Hrs. / Week	Practical	50 Marks
	Oral	50 Marks



LABORATORY MANUAL V 1.0

DEPARTMENT OF INFORMATION TECHNOLOGY
SINHGAD COLLEGE OF ENGINEERING,PUNE-41
2015-16

PROGRAM EDUCATIONAL OBJECTIVES

The students of Information Technology course after passing out will

- I. Graduates of the program will possess strong fundamental concepts in mathematics, science, engineering and Technology to address technological challenges.
- II. Possess knowledge and skills in the field of Computer Science & Engineering and Information Technology for analyzing, designing and implementing complex engineering problems of any domain with innovative approaches.
- III. Possess an attitude and aptitude for research, entrepreneurship and higher studies in the field of Computer Science & Engineering and Information Technology.
- IV. Have commitment to ethical practices, societal contributions through communities and life-long learning.
- V. Possess better communication, presentation, time management and team work skills leading to responsible & competent professionals and will be able to address challenges in the field of IT at global level.

PROGRAM OUTCOMES

The students in the Information Technology course will attain:

- i. an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering and technology;
- ii. an ability to define a problem and provide a systematic solution with the help of conducting experiments, as well as analyzing and interpreting the data;
- iii. an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints;
- iv. an ability to identify, formulate, and provide systematic solutions to complex engineering problems;
- v. an ability to use the techniques, skills, and modern engineering technologies tools, standard processes necessary for practice as a IT professional;
- vi. an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems with necessary constraints and assumptions;
- vii. an ability to analyze the local and global impact of computing on individuals, organizations and society;
- viii. an ability to understand professional, ethical, legal, security and social issues and responsibilities;
- ix. an ability to function effectively as an individual or as a team member to accomplish a desired goal(s);
- x. an ability to engage in life-long learning and continuing professional development to cope up with fast changes in the technologies/tools with the help of electives, professional organizations and extra-curricular activities;
- xi. an ability to communicate effectively in engineering community at large by means of effective presentations, report writing, paper publications, demonstrations;
- xii. an ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice;
- xiii. an ability to apply design and development principles in the construction of software systems of varying complexity.

Compliance Document Control

Reference Code	<<College Abbreviation >>/<<Course Title>> Lab.Manual
Version No	1.0
Compliance Status	Complete
Revision Date	Month and Year
Security Classification	Department Specific
Document Status	Definitive
Review Period	Academic Year

Author(s)		Authorizer
Prof. Avinash N Bhute	Mr.K.B.Sadafale	Dr. G.V. Garje
SCOE, Pune-41	SCOE, Pune-41	BOS – Information Technology

Document History

Revision No.	Revision Date	Reason For Change
1.0		Modification to existing manual for unique format and elaboration of assignment.

Summary of Changes to Computer Practice Laboratory Manual

Sr. No	Changes	Change type
01	Case Study	
02	FAQs	
03	Detailed Conceptual elaboration of each assignment	

Software Laboratory-VI

COURSE OBJECTIVES

1. To learn and understand Database Modeling, Architectures.
2. To learn and understand Advanced Database Programming Frameworks.
3. To learn and understand web database language, XML, JDOQL.
4. To learn NoSQL Databases (Open source) such as Hive/ Hbase/ Cassandra/DynamoDB.

COURSE OUTCOME

1. Understanding of Advanced Database Programming Languages.
2. Master the basics of web and object oriented database languages and construct queries using XML and JDOQL.
3. Master the basic concepts of NoSQL Databases.
4. Understand how analytics and big data affect various functions now and in the future.
5. Appreciate the impact of analytics and big data on the information industry and the external ecosystem for analytical and data services

INDEX

Sr. No.	Title	Page No.
1	Study and Configure Hadoop for Big Data	
2	Study of NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	
3	Design Data Model using NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	
4	Implement any one Partitioning technique in Parallel Databases	
5	Implement Two Phase commit protocol in Distributed Databases	
6	Design Persistent Objects using JDO and implement min 10 queries on objects using JDOQL in ObjectDB NOSQL DATABASE	
7	Create XML, XML schemas , DTD for any database application and implement min 10 queries using XQuery FLOWR expression and XPath	
8	Design database schemas and implement min 10 queries using Hive/ Hbase/ Cassandra column based databases	
9	Design database schemas and implement min 10 queries using DynamoDBkeyValue based databases	
10	Implement Web Page ranking algorithm	
11	Implement any one machine learning algorithm for classification / clustering task in BIG data Analytics	
12	Design and Implement social web mining application using NoSQL databases, machine learning algorithm, Hadoop and Java/.Net	

SCHEDULE

Assign No.	Assignment Title	No. of Hrs.	Week No.
1	Study and Configure Hadoop for Big Data	2	Week 1
2	Study of NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	2	Week 1
3	Design Data Model using NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	2	Week 2
4	Implement any one Partitioning technique in Parallel Databases	2	Week 2
5	Implement Two Phase commit protocol in Distributed Databases	2	Week 3
6	Design Persistent Objects using JDO and implement min 10 queries on objects using JDOQL in ObjectDB NOSQL DATABASE	4	Week 3 Week 4
7	Create XML, XML schemas , DTD for any database application and implement min 10 queries using XQuery FLOWR expression and XPath	2	Week 4
8	Design database schemas and implement min 10 queries using Hive/ Hbase/ Cassandra column based databases	4	Week 5
9	Design database schemas and implement min 10 queries using DynamoDBkeyValue based databases	2	Week 6
10	Implement Web Page ranking algorithm	4	Week 6 Week 7
11	Implement any one machine learning algorithm for classification / clustering task in BIG data Analytics	2	Week 7
12	Design and Implement social web mining application using NoSQL databases, machine learning algorithm, Hadoop and Java/.Net	2	Week 8

NOTE: The schedule should be inclusive of mock practical, partial and final submission.

Assignment No 1

Study and Configure Hadoop
For Big Data

Assignment No 1

Study and Configure Hadoop for Big Data

Aim
To Study and Configure Hadoop for Big Data

Objective(s)	
1	To Learn and understand the concepts of Hadoop
2	To learn and understand the Hadoop framework for Big Data.
3	To understand and practice installation and configuration of Hadoop.

Theory
<p>Introduction</p> <p>Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.</p> <p>Due to the advent of new technologies, devices, and communication like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected</p>
<p>Big Data</p> <p>Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks. Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.</p> <p>Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.</p> <ul style="list-style-type: none">• Black Box Data: It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.• Social Media Data: Social media such as Facebook and Twitter hold information and

the views posted by millions of people across the globe.

- **Stock Exchange Data** : The stock exchange data holds information about the ‘buy’ and ‘sell’ decisions made on a share of different companies made by the customers.
- **Power Grid Data** : The power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data** : Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data** : Search engines retrieve lots of data from different databases.

Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

- **Structured data** : Relational data.
- **Semi Structured data** : XML data.
- **Unstructured data** : Word, PDF, Text, Media Logs.

Benefits of Big Data

Big data is really critical to our life and its emerging as one of the most important technologies in modern world. Follow are just few benefits which are very much known to all of us:

- Using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions, and other advertising mediums.
- Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.
- Using the data regarding the previous medical history of patients, hospitals are providing better and quick service.

Big Data Technologies

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business.

To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in realtime and can protect data privacy and security.

There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that handle big data, we examine the following two classes of technology:

Operational Big Data

This include systems like MongoDB that provide operational capabilities for real-time, interactive workloads where data is primarily captured and stored.

NoSQL Big Data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow massive computations to be run inexpensively and efficiently. This makes operational big data workloads much easier to manage, cheaper, and faster to implement.

Some NoSQL systems can provide insights into patterns and trends based on real-time data with minimal coding and without the need for data scientists and additional infrastructure.

Analytical Big Data

This includes systems like Massively Parallel Processing (MPP) database systems and MapReduce that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.

MapReduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines.

These two classes of technology are complementary and frequently deployed together.

Big Data Challenges

The major challenges associated with big data are as follows:

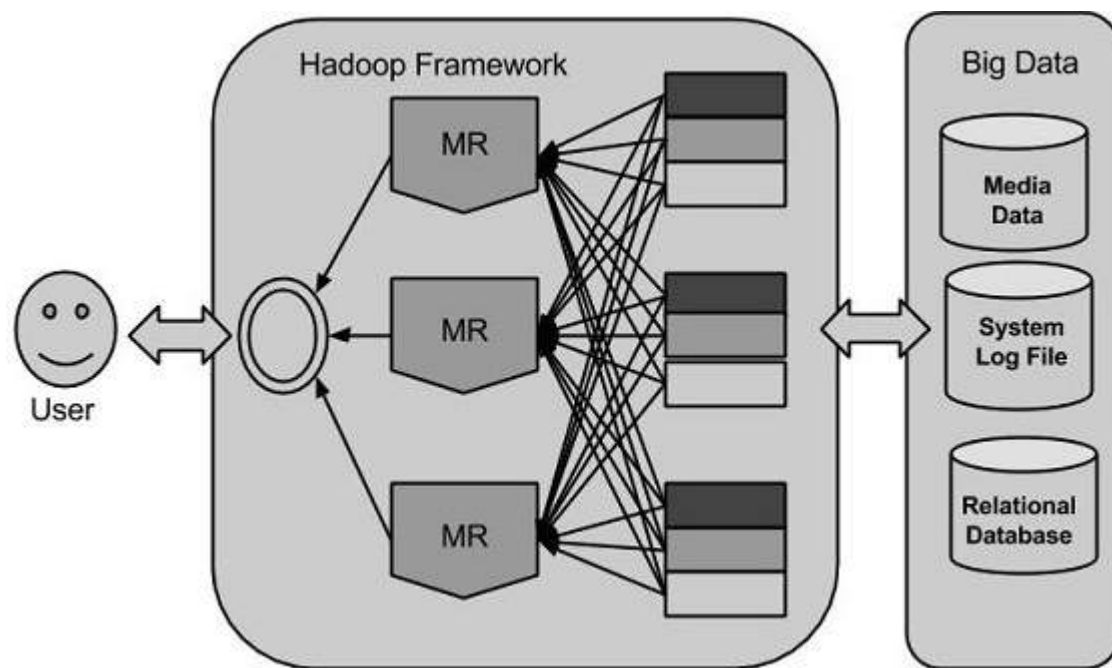
- Capturing data
- Curation
- Storage
- Searching
- Sharing
- Transfer
- Analysis
- Presentation

To fulfill the above challenges, organizations normally take the help of enterprise servers.

Hadoop

Doug Cutting, Mike Cafarella and team took the solution provided by Google and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. In short, Hadoop framework is capable enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for a huge amounts of data.



Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

Hadoop framework includes following four modules:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides

high-throughput access to application data.

- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

We can use following diagram to depict these four components available in Hadoop framework.

Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above, but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark etc.

MapReduce

Hadoop **MapReduce** is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

The term MapReduce actually refers to the following two different tasks that Hadoop programs perform:

- **The Map Task:** This is the first task, which takes input data and converts it into a set of data, where individual elements are broken down into tuples (key/value pairs).
- **The Reduce Task:** This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

Typically, both the input and the output are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master **JobTracker** and one slave **TaskTracker** per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves TaskTracker execute the tasks as directed by the master and provide task-status information to the master periodically.

The JobTracker is a single point of failure for the Hadoop MapReduce service which means if JobTracker goes down, all running jobs are halted.

Hadoop Distributed File System

Hadoop can work directly with any mountable distributed file system such as Local FS, HFTP FS, S3 FS, and others, but the most common file system used by Hadoop is the Hadoop Distributed File System (HDFS).

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on large clusters (thousands of computers) of small computer machines in a reliable, fault-tolerant manner.

HDFS uses a master/slave architecture where master consists of a single **NameNode** that manages the file system metadata and one or more slave **DataNodes** that store the actual data.

A file in an HDFS namespace is split into several blocks and those blocks are stored in a set of DataNodes. The NameNode determines the mapping of blocks to the DataNodes. The DataNodes takes care of read and write operation with the file system. They also take care of block creation, deletion and replication based on instruction given by NameNode.

HDFS provides a shell like any other file system and a list of commands are available to interact with the file system. These shell commands will be covered in a separate chapter along with appropriate examples.

Working of a Hadoop.

Stage 1

A user/application can submit a job to the Hadoop (a hadoop job client) for required process by specifying the following items:

1. The location of the input and output files in the distributed file system.
2. The java classes in the form of jar file containing the implementation of map and reduce functions.
3. The job configuration by setting different parameters specific to the job.

Stage 2

The Hadoop job client then submits the job (jar/executable etc) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

Stage 3

The TaskTrackers on different nodes execute the task as per MapReduce implementation and output of the reduce function is stored into the output files on the file system.

Pre-installation Setup

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

Creating a User

At the beginning, it is recommended to create a separate user for Hadoop to isolate Hadoop file

system from Unix file system. Follow the steps given below to create a user:

- Open the root using the command “su”.
- Create a user from the root account using the command “useradd username”.
- Now you can open an existing user account using the command “su username”.

Open the Linux terminal and type the following commands to create a user.

```
$ su
password:
# useradd hadoop
# passwd hadoop
New passwd:
Retype new passwd
```

SSH Setup and Key Generation

SSH setup is required to do different operations on a cluster such as starting, stopping, 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.

The following commands are used for generating a key value pair using SSH. Copy the public keys from `id_rsa.pub` to `authorized_keys`, and provide the owner with read and write permissions to `authorized_keys` file respectively.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Installing Java

Java is the main prerequisite for Hadoop. First of all, you should verify the existence of java in your system using the command “java -version”. The syntax of java version command is given below.

```
$ java -version
```

If everything is in order, it will give you the following output.

```
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)

Downloading Hadoop

Download and extract Hadoop 2.4.1 from Apache software foundation using the following commands.

```
$ su
```

```
password:
```

```
# cd /usr/local
```

```
# wget http://apache.claz.org/hadoop/common/hadoop-2.4.1/
```

```
hadoop-2.4.1.tar.gz
```

```
# tar xzf hadoop-2.4.1.tar.gz
```

```
# mv hadoop-2.4.1/* to hadoop/
```

```
# exit
```

Hadoop Operation Modes

Once you have downloaded Hadoop, you can operate your Hadoop cluster in one of the three supported modes:

- **Local/Standalone Mode** : After downloading Hadoop in your system, by default, it is configured in a standalone mode and can be run as a single java process.
- **Pseudo Distributed Mode** : It is a distributed simulation on single machine. Each Hadoop daemon such as hdfs, yarn, MapReduce etc., will run as a separate java process. This mode is useful for development.
- **Fully Distributed Mode** : This mode is fully distributed with minimum two or more machines as a cluster. We will come across this mode in detail in the coming chapters.

Installing Hadoop in Standalone Mode

Here we will discuss the installation of **Hadoop 2.4.1** in standalone mode.

There are no daemons running and everything runs in a single JVM. Standalone mode is suitable for running MapReduce programs during development, since it is easy to test and debug them.

Setting Up Hadoop

You can set Hadoop environment variables by appending the following commands

to ~/.bashrc file.

```
export HADOOP_HOME=/usr/local/hadoop
```

Before proceeding further, you need to make sure that Hadoop is working fine. Just issue the following command:

```
$ hadoop version
```

If everything is fine with your setup, then you should see the following result:

Hadoop 2.4.1

Subversion <https://svn.apache.org/repos/asf/hadoop/common> -r 1529768

Compiled by hortonmu on 2013-10-07T06:28Z

Compiled with protoc 2.5.0

From source with checksum 79e53ce7994d1628b240f09af91e1af4

It means your Hadoop's standalone mode setup is working fine. By default, Hadoop is configured to run in a non-distributed mode on a single machine.

Example

Let's check a simple example of Hadoop. Hadoop installation delivers the following example MapReduce jar file, which provides basic functionality of MapReduce and can be used for calculating, like Pi value, word counts in a given list of files, etc.

```
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar
```

Let's have an input directory where we will push a few files and our requirement is to count the total number of words in those files. To calculate the total number of words, we do not need to write our MapReduce, provided the .jar file contains the implementation for word count. You can try other examples using the same .jar file; just issue the following commands to check supported MapReduce functional programs by hadoop-mapreduce-examples-2.2.0.jar file.

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduceexamples-2.2.0.jar
```

Step 1

Create temporary content files in the input directory. You can create this input directory anywhere you would like to work.

```
$ mkdir input
```

```
$ cp $HADOOP_HOME/*.txt input
```

```
$ ls -l input
```

It will give the following files in your input directory:

```
total 24
```

```
-rw-r--r-- 1 root root 15164 Feb 21 10:14 LICENSE.txt
```

```
-rw-r--r-- 1 root root 101 Feb 21 10:14 NOTICE.txt
```

```
-rw-r--r-- 1 root root 1366 Feb 21 10:14 README.txt
```

These files have been copied from the Hadoop installation home directory. For your experiment, you can have different and large sets of files.

Step 2

Let's start the Hadoop process to count the total number of words in all the files available in the input directory, as follows:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduceexamples-2.2.0.jar wordcount input output
```

Step 3

Step-2 will do the required processing and save the output in output/part-r00000 file, which you can check by using:

```
$cat output/*
```

It will list down all the words along with their total counts available in all the files available in the input directory.

Installing Hadoop in Pseudo Distributed Mode

Follow the steps given below to install Hadoop 2.4.1 in pseudo distributed mode.

Step 1: Setting Up Hadoop

You can set Hadoop environment variables by appending the following commands to ~/.bashrc file.

```
export HADOOP_HOME=/usr/local/hadoop
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

Step 2: Hadoop Configuration

You can find all the Hadoop configuration files in the location “\$HADOOP_HOME/etc/hadoop”. It is required to make changes in those configuration files according to your Hadoop infrastructure.

```
$ cd $HADOOP_HOME/etc/hadoop
```

In order to develop Hadoop programs in java, you have to reset the java environment variables in **hadoop-env.sh** file by replacing **JAVA_HOME** value with the location of java in your system.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
```

The following are the list of files that you have to edit to configure Hadoop.

core-site.xml

The **core-site.xml** file contains information such as the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, and size of Read/Write buffers.

Open the core-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
<property>
<name>fs.default.name </name>
```

```
<value> hdfs://localhost:9000 </value>
```

```
</property>
```

```
</configuration>
```

hdfs-site.xml

The **hdfs-site.xml** file contains information such as the value of replication data, namenode path, and datanode paths of your local file systems. It means the place where you want to store the Hadoop infrastructure.

Let us assume the following data.

dfs.replication (data replication value) = 1

(In the below given path /hadoop/ is the user name.

hadoopinfra/hdfs/namenode is the directory created by hdfs file system.)

namenode path = //home/hadoop/hadoopinfra/hdfs/namenode

(hadoopinfra/hdfs/datanode is the directory created by hdfs file system.)

datanode path = //home/hadoop/hadoopinfra/hdfs/datanode

Open this file and add the following properties in between the <configuration></configuration> tags in this file.

```
<configuration>
```

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>1</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.name.dir</name>
```

```
<value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.data.dir</name>
```

```
<value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
</property>
</configuration>
```

In the above file, all the property values are user-defined and you can make changes according to your Hadoop infrastructure.

yarn-site.xml

This file is used to configure yarn into Hadoop. Open the yarn-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
</configuration>
```

mapred-site.xml

This file is used to specify which MapReduce framework we are using. By default, Hadoop contains a template of yarn-site.xml. First of all, it is required to copy the file from **mapred-site.xml.template** to **mapred-site.xml** file using the following command.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Open mapred-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Verifying Hadoop Installation

The following steps are used to verify the Hadoop installation.

Step 1: Name Node Setup

Set up the namenode using the command “hdfs namenode -format” as follows.

```
$ cd ~
```

```
$ hdfs namenode -format
```

The expected result is as follows.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
```

```
/*****
```

```
STARTUP_MSG: Starting NameNode
```

```
STARTUP_MSG: host = localhost/192.168.1.11
```

```
STARTUP_MSG: args = [-format]
```

```
STARTUP_MSG: version = 2.4.1
```

```
10/24/14 21:30:56 INFO common.Storage: Storage directory
```

```
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.
```

```
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to  
retain 1 images with txid >= 0
```

```
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
```

```
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
```

```
/*****
```

```
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
```

```
*****/
```

Step 2: Verifying Hadoop dfs

The following command is used to start dfs. Executing this command will start your Hadoop file system.

```
$ start-dfs.sh
```

The expected output is as follows:

10/24/14 21:37:56

Starting namenodes on [localhost]

localhost: starting namenode, logging to /home/hadoop/hadoop

2.4.1/logs/hadoop-hadoop-namenode-localhost.out

localhost: starting datanode, logging to /home/hadoop/hadoop

2.4.1/logs/hadoop-hadoop-datanode-localhost.out

Starting secondary namenodes [0.0.0.0]

Step 3: Verifying Yarn Script

The following command is used to start the yarn script. Executing this command will start your yarn daemons.

```
$ start-yarn.sh
```

The expected output as follows:

starting yarn daemons

starting resourcemanager, logging to /home/hadoop/hadoop

2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out

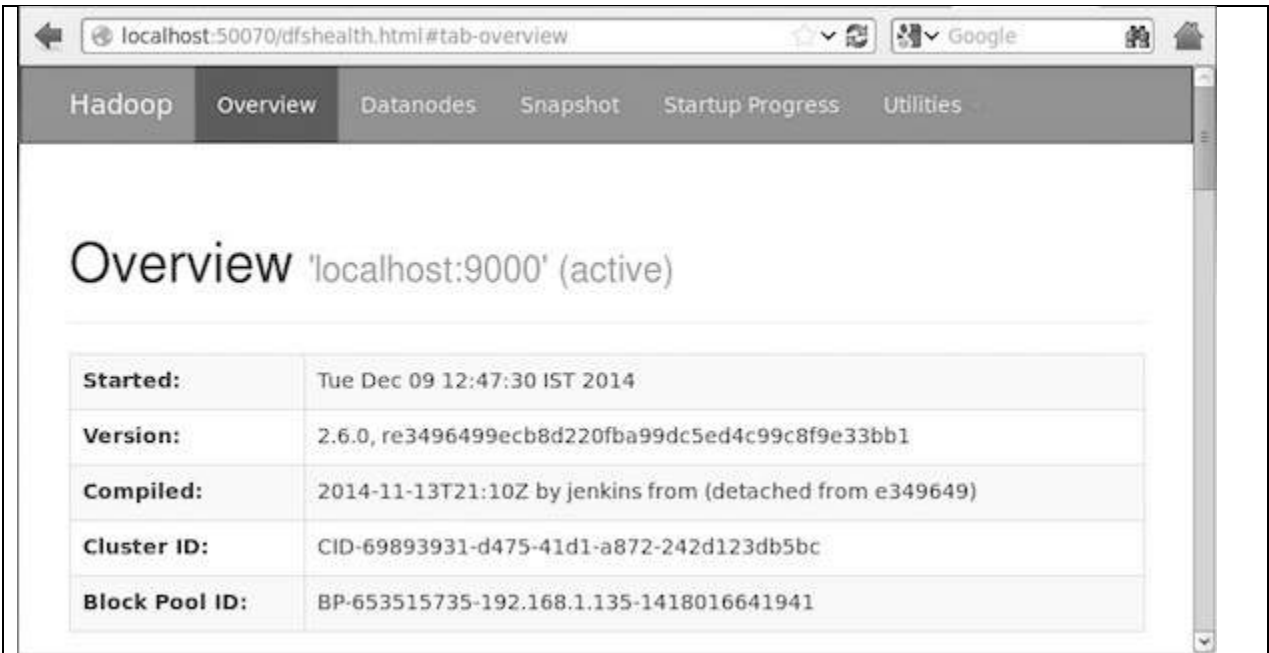
localhost: starting nodemanager, logging to /home/hadoop/hadoop

2.4.1/logs/yarn-hadoop-nodemanager-localhost.out

Step 4: Accessing Hadoop on Browser

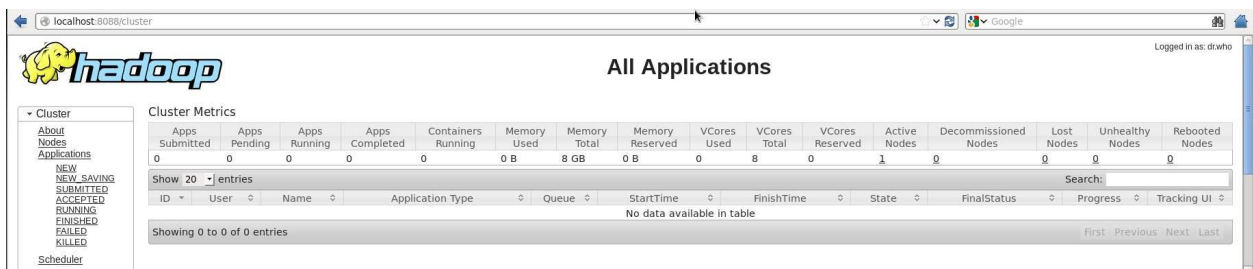
The default port number to access Hadoop is 50070. Use the following url to get Hadoop services on browser.

<http://localhost:50070/>



Step 5: Verify All Applications for Cluster

The default port number to access all applications of cluster is 8088. Use the following url to visit this service.



Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly faulttolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

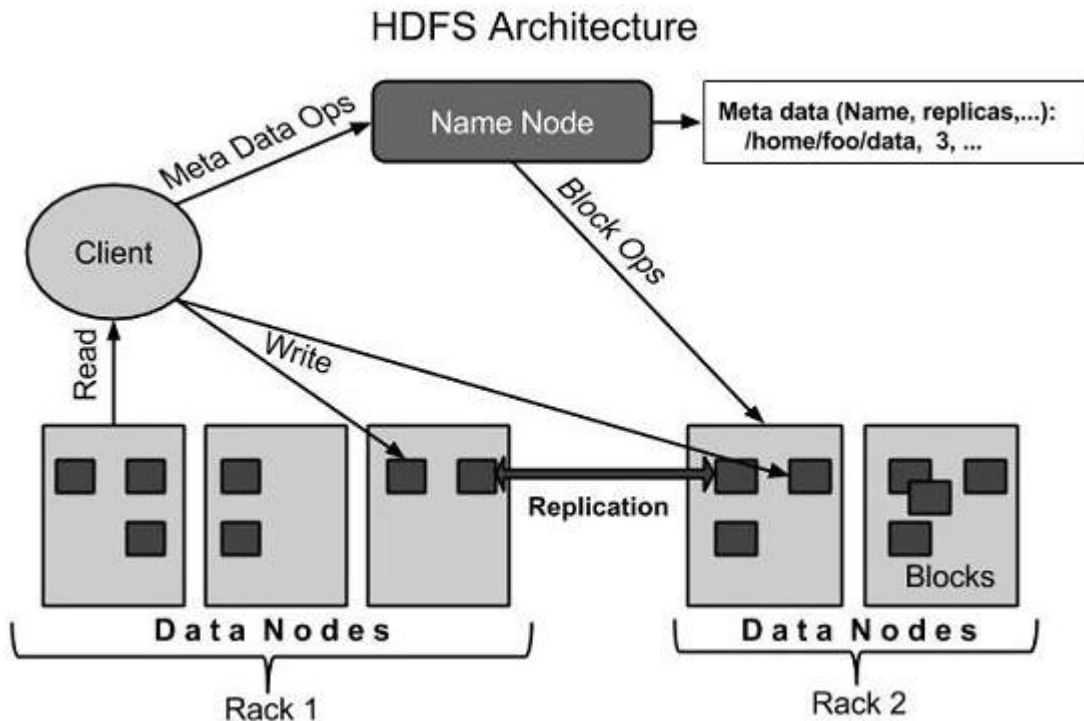
Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.

- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according

to the instructions of the namenode.

Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

Goals of HDFS

- **Fault detection and recovery** : Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.
- **Huge datasets** : HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.
- **Hardware at data** : A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

Starting HDFS

Initially you have to format the configured HDFS file system, open namenode (HDFS server), and execute the following command.

```
$ hadoop namenode -format
```

After formatting the HDFS, start the distributed file system. The following command will start the namenode as well as the data nodes as cluster.

```
$ start-dfs.sh
```

Listing Files in HDFS

After loading the information in the server, we can find the list of files in a directory, status of a file, using 'ls'. Given below is the syntax of ls that you can pass to a directory or a filename as an argument.

```
$ $HADOOP_HOME/bin/hadoop fs -ls <args>
```

Inserting Data into HDFS

Assume we have data in the file called file.txt in the local system which is ought to be saved in the hdfs file system. Follow the steps given below to insert the required file in the Hadoop file

system.

Step 1

You have to create an input directory.

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
```

Step 2

Transfer and store a data file from local systems to the Hadoop file system using the put command.

```
$ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input
```

Step 3

You can verify the file using ls command.

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/input
```

Retrieving Data from HDFS

Assume we have a file in HDFS called outfile. Given below is a simple demonstration for retrieving the required file from the Hadoop file system.

Step 1

Initially, view the data from HDFS using cat command.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile
```

Step 2

Get the file from HDFS to the local file system using get command.

```
$ $HADOOP_HOME/bin/hadoop fs -get/user/output/ /home/hadoop_tp/
```

Shutting Down the HDFS

You can shut down the HDFS by using the following command.

There are many more commands in "\$HADOOP_HOME/bin/hadoop fs" than are demonstrated here, although these basic operations will get you started. Running ./bin/hadoop dfs with no additional arguments will list all the commands that can be run with the FsShell system. Furthermore, \$HADOOP_HOME/bin/hadoop fs -help commandName will display a short usage summary for the operation in question, if you are stuck.

A table of all the operations is shown below. The following conventions are used for

parameters:

"<path>" means any file or directory name.

"<path>..." means one or more file or directory names.

"<file>" means any filename.

"<src>" and "<dest>" are path names in a directed operation.

"<localSrc>" and "<localDest>" are paths as above, but on the local file system.

Conclusion

Study and Configure Hadoop for Big Data

Assignment No. 2

**Study of NoSQL Databases such as
Hive/Hbase/Cassandra/DynamoDB**

Assignment No 2

Study of NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB

Aim	
Study of NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	

Objective(s)	
1	To understand the concepts of NoSQL databases
2	To understand the purpose of NoSQL databases
3	To Learn the CRUD Operation in NoSQL Databases

Theory
<p>Introduction</p> <p>Efficient Storage and retrieval of data with availability and scalability is the main purpose of NoSQL databases. NoSQL does not stand for no to SQL; it means “NOT ONLY SQL”. NoSQL database is just an alternative to traditional relational database. The industry of database has seen an introduction of many non relational databases such as MongoDB, Hbase , Neo4j in last few years. Depending upon the business requirement and strategy a cloud vendor can go with any of the database type.</p> <p>Importance Of NoSQL</p> <p>For last couple of years, SQL vs. NoSQL has been emerged as a heated argument over the Internet. The argument “SQL vs NoSQL,” actually talks about relational versus nonrelational databases. Because of normalized data model and enforcement of strict ACID properties, traditional relational database is considered to be a schema based transaction oriented database. It requires a strict predefined schema prior to storing data into it. Redefining a schema in case of a future change, once after data got inserted into the database is disruptive. Whereas in the era of Big Data, there is a constant need for adding new types of data to enrich the applications. Again the storage solution of relational database can make a big impact on speed and scalability. Web services like Amazon and Google have terabytes and petabytes of</p>

data stored in their big data centers and have to respond to massive read-write requests without a noticeable latency. To scale a relational database, data needs to get distributed on multiple servers. Before providing to the application the desired information has to be collected from many tables and combined. Similarly, while writing data also; it has to be performed on many tables in a coordinated manner. For any application, it could be a bottleneck to handle tables across multiple servers. In relational databases “join” operation slow-down the system to a crawl, especially when millions of users are doing lookups against tables with millions of rows of data. Large scale web services such as Google, Amazon, Yahoo, Facebook found these to be the cases to develop their own non-relational database in order to meet the scalability and performance needs.

Features of NoSQL

NoSQL databases may not require a predefined table schema, typically scale horizontally and usually avoid join operations. Because of schema less nature and involvement of smaller subset analysis of NoSQL system, this database can be better described as structured data stores. Three important basic features of NoSQL databases are scale-out, flexible data structure and replication, which are explained as follows.

- **Scale-out:** Scaling out refers to achieve high performance in a distributed environment by using many general-purpose machines. NoSQL databases allow the distribution of the data over a large number of machines with a distributed processing load. Many NoSQL databases allow automatic distribution of data to new machines when they are added to the cluster. Scale-out is evaluated in terms of scalability and elasticity.
- **Flexibility:** Flexibility in terms of data structure says that there is no need to define a schema for databases. NoSQL databases do not require a predefined schema. This allows the users to store data of various structures in the same database table. However, support for high-level query languages such as SQL is not supported by most of the NoSQL databases.
- **Data Replication:** One of the features of NoSQL databases is data replication. In this process a copy of the data is distributed to different systems in order to achieve redundancy and load distribution. However there is a chance of losing data consistency among the replicas. But it is believed that sometimes this consistency may be achieved eventually. Consistence and availability are the factors for evaluating replication.

NoSQL Data Models

These are some categories of NoSQL database models discussed as follows.

1. Key-Value Data Stores

In order to handle highly concurrent access to database, the category of NoSQL designed is key-value stores. It is the simplest, still the most powerful data store. In a key-value store each data consists of a pair of a unique key and value. In order to save data a key gets generated by the application and value gets associated with the key. And this key-value pair gets submitted to the data store. The data values stored in key value stores can have dynamic sets of attributes attached to it and is opaque to the database management system. Hence key is the only means to access the data values. The type of binding from the key to value depends on the programming language used in the application. An application needs to provide a key to the data stores in order to retrieve data. Many key-value data stores use a hash function. The application hashes the key and find out the location of the data in the database. The key-value data stores are row focused. Which means it enables the application to retrieve data for complete entities. Fig.1 describes retrieval of data from a key-value database.

The application has specified a key 'Emp102' to the data store in order to retrieve data. Using the hash function the application hashes the key in order to trace the location of data in the data store. The design of the key should support the most frequent queries fired on the data store. Efficiency of the hash function, design of the key and size of the values being stored are the factors which affect the performance of a keyvalue data store. The operations performed on such data stores are mostly limited to read and write operations. Because of the simplicity of the key-value data store, it provides users with fastest means of storing and fetching data. All other categories of NoSQL are built upon the simplicity, scalability and performance of key-value data stores. Redis, Voldemort and Membase database systems are examples of prominent keyvalue data stores.

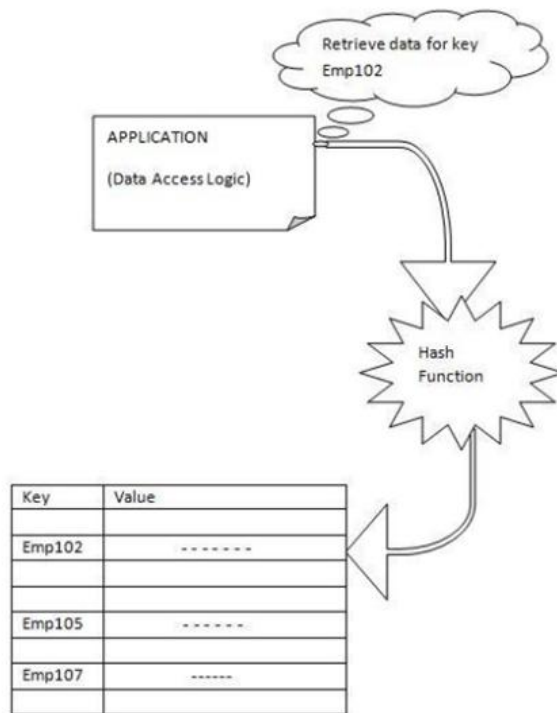


Fig1. An Example of Key value data store

2. Document Oriented Data Stores

At an abstract level document oriented database is similar to key-value data store. It also holds value, which an application can read or fetch by using a key. Several document databases automatically generate the unique key while creating a new document. A document in a document database is an entity, which is a collection of named fields. The feature which distinguishes the document oriented database from a key-value data store is transparency of the data held by the database. Hence the query possibility is not restricted with the key only. In order to support scenarios where the application requires querying the database not only based on its key but also with attribute values, can switch for document databases.

A document needs to be self-describing in a document oriented database. Information is stored in a portable and well understood format such as XML, BSON or JSON. As shown in Fig. 2, the document database stores data in form of key-value pairs. But the data stored in the database is transparent to the system unlike key-value databases. The application can query the database not only with the key i.e. 'Employee ID' but also with the defined fields in the document i.e. FirstNm, LastNm, age etc. Document data stores are an efficient approach to model data based on common software problems. But it comes at the cost of slightly lower

performance and scalability in comparison to key-value data stores. Few of the most prominent document stores are Riak, MongoDB, CouchDB.

(Key) Employee ID	Documents
EMP101	FirstName: Harshit LastName:B Age:22
EMP102	FirstName: Sada LastName:Fale Age:32
.	.
.	.
EMP705	FirstName: Prashant LastName: Kulkarni Age:32

Fig 2. Example of Document Data Store

3. Column Family Data Stores

Sometimes an application may want to read or fetch a subset of fields, similar to the SQL's projection operation. Column family data store enables storing data in column centric approach. The column family data store partitions the key space. In NoSQL a key space is considered to be an object which holds all column families of a design together. It is the outer most grouping of the data in the data store. Each partition of the key space is known to be a Table. Column families are declared by these tables. Each column family consists of number of columns. A row in a column family is structured as collections of arbitrary number of columns. Each column is a map of a key-value pair. In this map, keys are the names of columns and columns themselves are the values.

Each of these mappings is called a cell. Each row in a column family database is identified by a unique row key, defined by the application. Use of these row keys makes the data retrieval quicker. In order to avoid overwriting of the cell values few of the popular column-family databases add timestamp information automatically to individual columns. Every time there is

an update, it creates a new version of the cells which have been affected by the update operation. Always the reader reads the value which is last written or committed. A row key, column family, column and timestamp constitute a key. Hence the exact mapping can be represented as (row key, column family, column, timestamp) --> value.

A generalized structure of a column family database has been shown in Fig. 3 as follows.

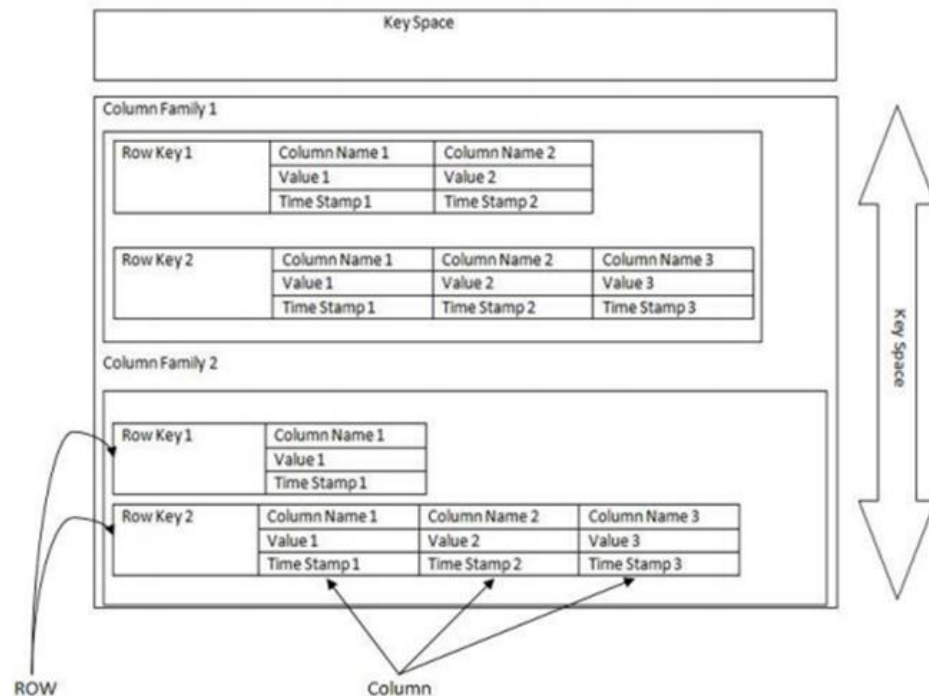


Fig 3. Column-Family Data Store

This data model has been popularly accepted as "sparse, distributed, consistent multidimensional sorted map". An advantage of using a column family data store over a traditional database is in handling NULL values. In a relational database, when a value for an attribute is not applicable for a particular row, NULL gets stored. While in a column family database the column can be simply removed for corresponding row in case the data is not available. That's why Google calls it a sparse database. One of the key features of this database is that it can be distributed in billion of cells over thousands of machines. The cells are sorted on basis of row keys. Sorting of keys allows searching data for a range of keys. Since the data in such kind of model get organized as a set of rows and columns, representation wise this database is most similar to the relational database. But like a relational database it does not need any predefined schema. At runtime, rows and columns

can be added flexibly but oftentimes the column families have to be redefined, which leads the data store to be less flexible than key-value or document data stores. Developers should understand the data captured by the application and the query possibilities before deciding the column families. A well-designed column-family database enables an application to satisfy majority of its queries by visiting less number of column families as possible. Compared to a relational database holding equivalent amount of data, a column family data store is more scalable and faster. But the performance comes at the price of the database being less generalized than a relational database as it is designed in support for a specific set of queries. Hbase and Hypertable database systems are based on the data model described above. Whereas another database system Cassandra differs from the data model, as it is having a new dimension added called super column. As shown in Fig. 4 a super column consists of multiple columns. A collection of super columns along with a row key constitute a row of a super column family. As in columns, the super column names and the sub column names are sorted. Super column is also a name-value entity but with no timestamps.

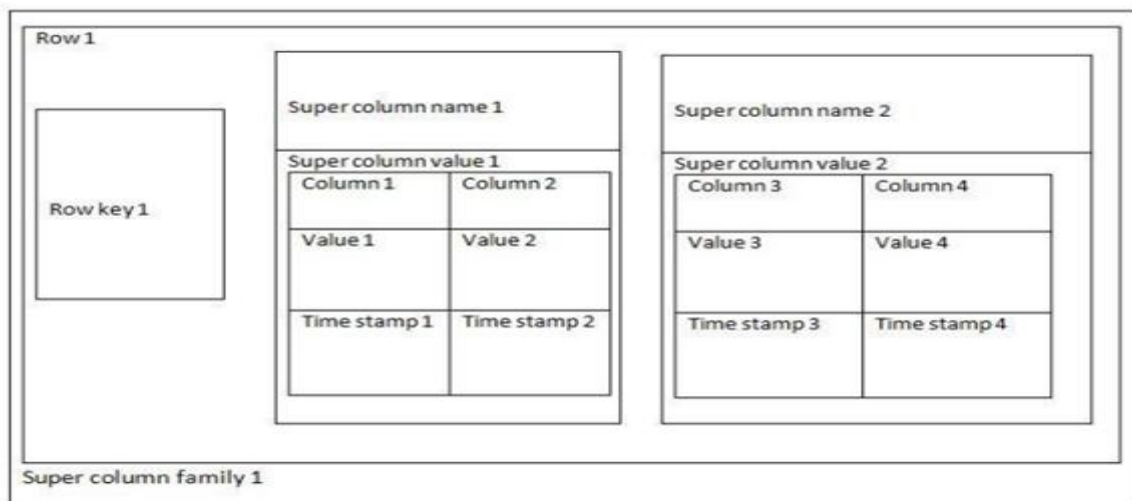


Fig 4. Column Family datastore (Cassandra)

4. Graph Database

Graph databases are considered to be the specialists of highly linked data. Therefore, it handles data involving a huge number of relationships. There are basically three core abstractions of graph database. These are nodes, edges which connect two different nodes, and properties. Each node holds information about an entity. The edges represent the existence of relationship between the entities. Each relationship is having a relationship type

and is directional with a start point (node) and an end point. The end point can be some other node than that of the start node or possibly the same node. Key-value properties are associated not only with the nodes but also with the relationships. The properties of the relationships provide additional information about the relationships. The direction of the relationship determines the traversal path from one node to the other in a graph database.

Fig. 5 represents a part of the 'Employee' database structured as graph database. Each node in this graph database represents an employee entity. These entities are related with each other through a relationship of relationship type “knows”. The property associated with the relationship is “Duration”. The key difference between a graph and relational database is data querying. Instead of using cost intensive process like recursive join as in relational database, graph databases use traversal method. While querying through graph database, a start node has to be specified by the application. Traversal starts from the start node and progresses via relationships to nodes connected to the start node, based upon some rule defined by the application logic. The traversal method involves only nodes which are relevant to the application not the entire data set. Hence, a huge increase in number of nodes does not affect the traversal rate much.

Social networking, data mining, managing networks, and calculating routes are few of the fields where graph database has been used extensively. Neo4j, GraphDB are popular graph databases in use today.

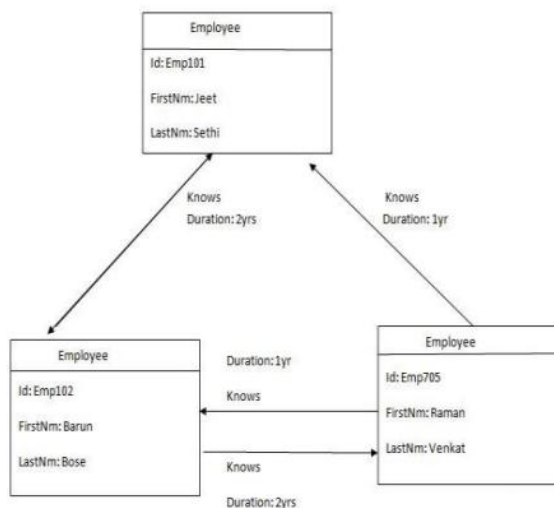


Fig 5. Example of Graph database

Transaction In Nosql Databases

When we talk about SQL vs. NoSQL, the competition is actually not between the databases. The comparison is between the transaction models of both the databases. Transaction is defined to be the logical unit of a database processing formed by an executing program. The transaction of SQL database is based upon strict ACID properties. Where ACID is the abbreviation for Atomicity, Consistence, Isolation and Durability. But designers of the NoSQL database came up with a decision, that ACID property is too restrictive to achieve the demands of big data. Hence Professor Eric Brewer in the year of 2000 came up with a new theorem known as CAP theorem. CAP is the abbreviation for Consistency, Availability and Partition tolerance. The theorem says that the designers can achieve any two of these properties at a time in a distributed environment. The designers can ensure Consistency and Availability at the cost of Partition tolerance, i.e. CA based database. If the designer goes for availability and partition tolerance at the cost of Consistency, then it is an AP based database. And if ensure Consistency and Partition tolerance at the cost of availability then the database is CP based. The transaction of NoSQL can be classified as follows.

- **Concerned about consistency and availability (CA):**

This kind of database system ensures its priority more towards data availability and consistency by using replication approach. Part of database doesn't bother about partition tolerance. In case of occurrence of a partition between nodes, the data will go out of sync. The relational database, Vertica, and Greenplum database systems fall under such category of databases.

- **Concerned about consistency and partition tolerance (CP):**

The priority of such database system is to ensure data consistency. But it does not support for good availability. Data gets stored in distributed nodes. When a node goes down, data becomes unavailable to maintain consistency between the nodes. It maintains partition tolerance by preventing resynchronization of data. Hypertable, BigTable, HBase are few database systems which are concerned about CP.

- **Concerned about availability and partition tolerance (AP):**

The priority of such database system is to ensure data availability and partition

tolerance primarily. Even if there is a communication failure between the nodes, nodes remain online. Once after the partition gets resolved, resynchronization of data takes place, but without the guarantee of consistency. Riak, CouchDB, KAI are few databases which follow this principle.

Afterwards CAP theorem gets expanded into PACELC. PACELC is an abbreviation for partition, availability, consistency, else, latency, consistency. According to this model the tradeoff between availability and consistency is not only based upon partition tolerance, but it is also dependent on the existence of network partition. It suggests latency to be one of the important factors, since most of the distributed database systems use replication technology for ensuring availability. Later eBay introduced a new theorem known as BASE theorem. BASE aims to achieve availability instead of consistency of databases. BASE is the abbreviation for basically available, soft state and eventually consistent.

- **Basically Available:** Basically available says that even if a part of the database becomes unavailable, other parts of the database continue to function as expected. In case of a node failure, the operation continues on the replica of the data stored in some other node.
- **Soft State:** Soft state says that on the basis of user interaction a data may be dependent on time. These data may also have possible expiration after a certain period of time. Hence to keep the data relevant in a system it has to be updated or accessed.
- **Eventually Consistent:** Eventual consistency says after any data update, data may not become consistent across the entire system but it will become consistent with time eventually. Therefore, the data is said to be consistent in the future.

There is not any hard and fast rule to decide which NoSQL database is best for an enterprise. Business Model, strategy, cost and transaction model demand are few of the important factors that an enterprise should consider while choosing a database. Following are few of the facts which may help in choosing a database for an enterprise.

- If the applications simply store and retrieve data items which are opaque to the database management system and blobs by using a key as identifier, then a key-value store is the best choice. But if the application likes to query the database with some

attribute value other than the key, it fails. Also while updating or reading an individual field in a record key-value store is a failure.

- When applications are more selective and need to filter records based on non-key fields, or retrieve or update individual fields in a record as it, then document database is an efficient solution. Document data stores offer better query possibility than key-value data stores.
- When the applications need to store records with hundreds or thousands of fields, but retrieves a subset of those fields in most of the queries that it performs, in that case column-family data store is an efficient choice. Such data stores are suitable for large datasets that scale high.

If the applications need to store and process information on heavily linked data with highly complex relationship between the entities, graph database is the best choice. In a graph database, entities and relationship between the entities are treated with equal importance.

Table 1 represents a list of databases, their corresponding data models, along with transaction model and query language used by these databases. Cassandra for facebook, HBase for Google, DynamoDB for Amazon is few of the databases which were developed by different companies in order to meet their demand for high data storage requirement. On the other hand database systems such as Neo4j, Riak, and MongoDB were developed in order to serve other organizations. In terms of transaction model, most of the databases such as DynamoDB, Riak, Cassandra and Voldemort give more preference to availability over consistency. Whereas Tokyo Cabinet, Hbase prefer consistency over availability. NoSQL database was designed in order to handle large volume data processing, excluding some of the support system of RDBMS like ad-hoc query.

Though many of NoSQL databases mentioned in Table. 1 support ad-hoc queries but the level of programming expertise in writing queries needs to be much higher than that of a relational database.

TABLE I. A COMPARISON OF DIFFERENT NOSQL DATABASES

Database Tool	Data Model	Transaction Model (CAP)	Ad-HOC query
DynamoDB	Key-value	AP	Built in API
Riak	Key-value	AP	CorrugatedIron
Voldemort	Key-value	AP	No
Tokyo Cabinet	Key-value	PC	No
CouchDB	Document	AP	Cloudant, Lucene
MongoDB	Document	AP	BSON based format
RavenDB	Document	ACID	Built in, Limited
Cassandra	Column-family	AP	HIVE, PIG
Hbase	Column-family	PC	HIVE, PIG
Neo4j	Graph	CA	Chyper

Conclusion

Studied NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB and there comprasion.

Assignment No. 3

**Design Data Model using NoSQL
Databases such as
Hive/Hbase/Cassandra/DynamoDB**

Assignment No 3

Design Data Model using NoSQL Databases such as
Hive/Hbase/**Cassandra**/DynamoDB

Aim	
To design Data Model using NoSQL Databases such as Hive/Hbase/Cassandra/DynamoDB	

Objective(s)	
1	To Learn and Understand the design principal of NoSQL.
2	To Design the model for Cassandra.
3	Performed the curd Operarions in Cassandra.

Theory
<p>Cassandra:</p> <ul style="list-style-type: none">• Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.• Features :<ol style="list-style-type: none">1) Scalability:<ul style="list-style-type: none">○ Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.2) Fault-tolerant:<ul style="list-style-type: none">○ Data is automatically replicated to multiple nodes for fault-tolerance.○ Replication across multiple data centers is supported.○ Failed nodes can be replaced with no downtime.3) MapReduce support:<ul style="list-style-type: none">○ Cassandra has Hadoop integration, with MapReduce support.4) Query language:<ul style="list-style-type: none">○ Cassandra introduces CQL (Cassandra Query Language), a SQL-like alternative to the traditional RPC interface.• Keyspace:<p>Keyspace is the outermost container for data. It is similar to the schema in a relational database. Basic attributes of Keyspace are:</p> <p>Replication Factor: It is the number of machines in the cluster that will receive copies of the same data</p> <p>Replica Placement Strategy: It is a strategy to place replicas in the ring</p> <ul style="list-style-type: none">○ Simple Strategy

- Old Network Topology Strategy
- Network Topology Strategy

Column Families

Column family is a NoSQL object that contains columns of related data.

It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value that is a set of columns.

It is similar to a table in a relational database and each key-value pair being a row.

Each column is a tuple (triplet) consisting of

- Column name
- Value
- Timestamp

- Verifying Installation :

DataStax Community Edition must be installed on system before installing Cassandra. Verify the Cassandra installation using the following command:

```
$ Cassandra version
```

If Cassandra is already installed on system, then you will get the following response:

```
Connected to Test Cluster at 127.0.0.1:9042.
```

```
[cqlsh 5.0.1 | Cassandra 2.2.4 | CQL spec 3.3.1 | Native protocol v4]
```

```
Use HELP for help.
```

```
WARNING: pyreadline dependency missing. Install to enable tab completion. From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

- Creating KEYSPACE :

Cassandra is technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it.

Create **KEYSPACE** Statement

Create **KEYSPACE** is a statement used to create a **KEYSPACE** in Cassandra. A **KEYSPACE** in Cassandra is a **KEYSPACE** or a collection of tables. The **syntax** for this statement is as follows:

```
cqlsh> CREATE KEYSPACE ABC userdb replication = { 'class': 'SimpleStrategy', 'replication_factor': '1' };
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

The following query is executed to create a database named **userdb**:

```
cqlsh> userdb;
```

or

```
cqlsh> CREATE SCHEMA userdb;
```

The following query is used to verify a databases list:

```
cqlsh>:userdb> show tables;
```

Improper show command.

default

userdb

For creating Table:

Create Table

```
CREATE TABLE test_table (  
    id int,  
    address text,  
    name text,  
    PRIMARY KEY ((id))  
);
```

CURD using cql Updating Table:

Update Table

```
insert into test_table (id, name, address) values  
(4, 'somnath', 'Sus');
```

CURD using cql Delete Table

Deleting rows from Table

delete from test_table where id =1;

The following queries are used to drop a database. Let us assume that the database name is **userdb**.

```
cqlsh:userdb> delete from Tablename where condition;
```

For describing tables

```
cqlsh:userdb> describe tables;
```

show all table names

```
cqlsh:userdb>
```

For help of any topic:

For Help of any Topic

Cqshl> Help;

Display topics

Cqshl> Help topic name;

Help open in Browser.

HBASE

HBase is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (Hadoop Distributed Filesystem), providing BigTable-like capabilities for Hadoop.

Features:

- HBase is not an “eventually consistent” DataStore. This makes it very suitable for tasks such as high-speed counter aggregation(Strongly consistent reads/writes).
- HBase tables are distributed on the cluster via regions, and regions are automatically split and re-distributed as your data grows(Automatic sharding)

- AutomaticRegion Server failover
- HBase supports **HDFS** out of the box as its distributed file system(Hadoop/HDFS Integration)
- HBase supports massively parallelized processing via MapReduce for using HBase as both source and sink.(MapReduce)
- HBase supports an easy to use Java API for programmatic access(Java Client API)
- HBase also supports Thrift and REST for non-Java front-ends(Thrift/REST API)
- HBase supports a Block Cache and Bloom Filters for high volume query optimization(Block Cache and Bloom Filters)
- HBase provides build-in web-pages for operational insight as well as JMX metrics(Operational Management)

Applications store data into an HBase table. Tables are made of rows and columns. All columns in HBase belong to a particular column family. Table cells -- the intersection of row and column coordinates -- are versioned. A cell's content is an uninterpreted array of bytes.

Table row keys are also byte arrays so almost anything can serve as a row key from strings to binary representations of longs or even serialized data structures. Rows in HBase tables are sorted by row key. The sort is byte-ordered. All table accesses are via the table row key -- its primary key.

Data Model Operations

The four primary data model operations are Get, Put, Scan, and Delete. Operations are applied via HTable instances.

- Get

Get returns attributes for a specified row. Gets are executed via HTable.get.

- Put

Put either adds new rows to a table (if the key is new) or can update existing rows (if the key already exists). Puts are executed via HTable.put (writeBuffer) or HTable.batch (non-writeBuffer).

- Scans

Scan allow iteration over multiple rows for specified attributes.

The following is an example of a on an HTable table instance. Assume that a table is

populated with rows with keys "row1", "row2", "row3", and then another set of rows with the keys "abc1", "abc2", and "abc3". The following example shows how startRow and stopRow can be applied to a Scan instance to return the rows beginning with "row".

```
HTable htable = ...    // instantiate HTable

Scan scan = new Scan();
scan.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("attr"));
scan.setStartRow( Bytes.toBytes("row"));           // start key is inclusive
scan.setStopRow( Bytes.toBytes("row" + (char)0)); // stop key is exclusive
ResultScanner rs = htable.getScanner(scan);
try {
    for (Result r = rs.next(); r != null; r = rs.next()) {
        // process result...
    } finally {
        rs.close(); // always close the ResultScanner!
    }
}
```

- Delete

Delete removes a row from a table. Deletes are executed via HTable.delete.

HBase does not modify data in place, and so deletes are handled by creating new markers called tombstones. These tombstones, along with the dead values, are cleaned up on major compactions.

ACID in HBase

HBase supports ACID in limited ways, namely Puts to the same row provide all ACID guarantees. HBase employs a kind of MVCC. And HBase has no mixed read/write transactions.

The highlevel flow of a write transaction in HBase looks like this:

1. lock the row(s), to guard against concurrent writes to the same row(s)
2. retrieve the current writenumber
3. apply changes to the WAL (Write Ahead Log)
4. apply the changes to the Memstore (using the acquired writenumber to tag the KeyValues)
5. commit the transaction, i.e. attempt to roll the Readpoint forward to the acquired Writenumber.
6. unlock the row(s)

The highlevel flow of a read transaction looks like this:

1. open the scanner
2. get the current readpoint
3. filter all scanned KeyValues with memstore timestamp > the readpoint
4. close the scanner (this is initiated by the client)

Hive:

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. While initially developed by Facebook, Apache Hive is now used and developed by other companies such as Netflix. Amazon maintains a software fork of Apache Hive that is included in Amazon Elastic MapReduce on Amazon Web Services.

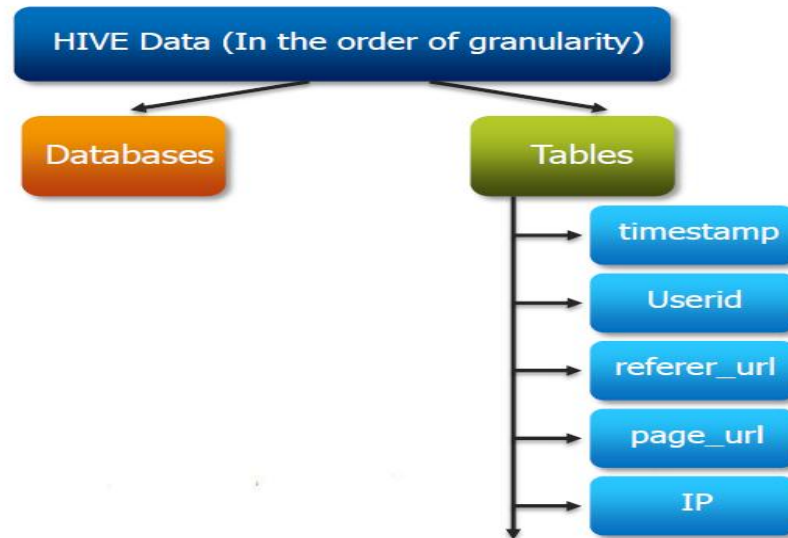
Features:

- Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem.
- Apache Hive provides an SQL-like language called HiveQL with schema on read and transparently converts queries to map/reduce, Apache Tez and Spark jobs.
- By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.
- Indexing to provide acceleration, index type including compaction and Bitmap index as of 0.10, more index types are planned.
- Different storage types such as plain text, RCFile, HBase, ORC, and others.
- Metadata storage in an RDBMS, significantly reducing the time to perform semantic checks during query execution.
- Operating on compressed data stored into the Hadoop ecosystem using algorithms including DEFLATE, BWT, snappy, etc.
- Built-in user defined functions (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.
- SQL-like queries (HiveQL), which are implicitly converted into MapReduce or Tez, or Spark jobs.

Hive Data Models:

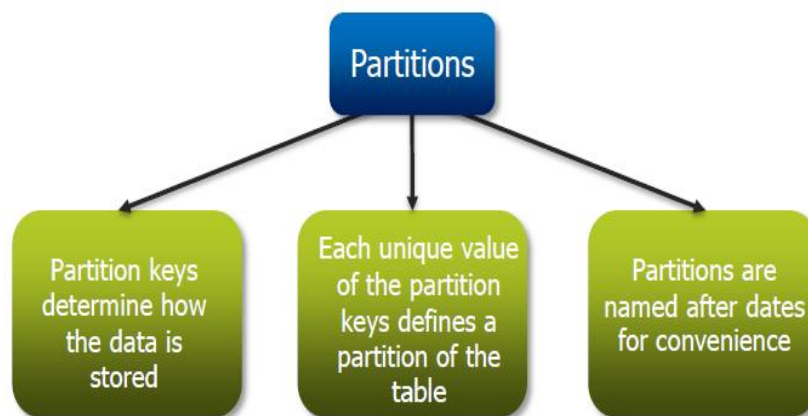
The Hive data models contain the following components:

- Databases
- Tables
- Partitions
- Buckets or clusters



Partitions:

Partition means dividing a table into a coarse grained parts based on the value of a partition column such as 'data'. This makes it faster to do queries on slices of data

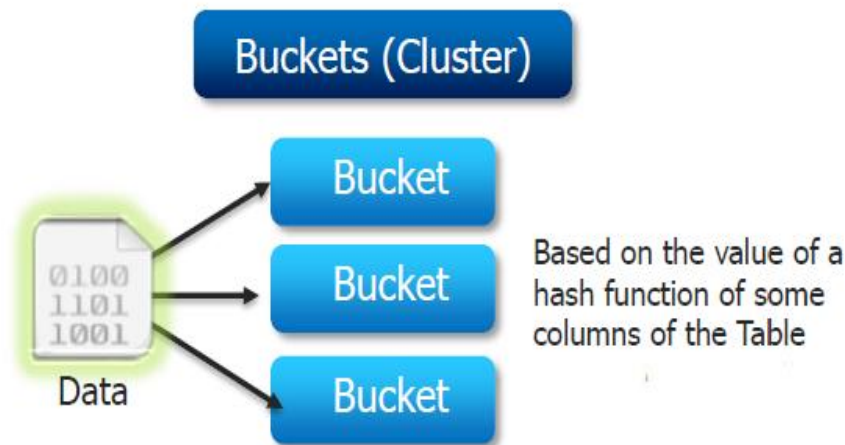


The Partition keys determine how data is stored. Here, each unique value of the Partition key

defines a Partition of the table. The Partitions are named after dates for convenience. It is similar to 'Block Splitting' in HDFS.

Buckets:

Buckets give extra structure to the data that may be used for efficient queries. A join of two tables that are bucketed on the same columns, including the join column can be implemented as a Map-Side Join. Bucketing by used ID means we can quickly evaluate a user-based query by running it on a randomized sample of the total set of users.



DynamoDB:

Amazon DynamoDB is a fully managed proprietary NoSQL database service that is offered by Amazon.com as part of the Amazon Web Services portfolio. DynamoDB exposes a similar data model and derives its name from Dynamo, but has a different underlying implementation. Dynamo had a multi-master design requiring the client to resolve version conflicts and DynamoDB uses synchronous replication across multiple datacenters for high durability and availability. DynamoDB was announced by Amazon CTO Werner Vogels on January 18, 2012.

Features:

- Document Data Model Support

DynamoDB supports storing, querying, and updating documents. Using the AWS SDK you can write applications that store JSON documents directly into Amazon DynamoDB tables. This capability reduces the amount of new code to be written to insert, update, and retrieve

JSON documents and perform powerful database operations like nested JSON queries using just a few lines of code.

- Key-value Data Model Support

Amazon DynamoDB supports key-value data structures. Each item (row) is a key-value pair where the primary key is the only required attribute for items in a table and uniquely identifies each item. DynamoDB is schema-less. Each item can have any number of attributes (columns). In addition to querying the primary key, you can query non-primary key attributes using Global Secondary Indexes and Local Secondary Indexes.

- Seamless Scaling

Amazon DynamoDB delivers seamless throughput and storage scaling via API and the AWS Management Console. There is virtually no limit on how much throughput or storage that you can dial up at a time.

- High Availability

Amazon DynamoDB is highly available, with automatic and synchronous data replication across three facilities in a Region. This helps protect your data against individual machine, or even facility level failures.

- Develop Locally, Scale Globally

The downloadable version of DynamoDB makes it easy to develop and test applications on your laptop or in an EC2 instance. Once your solution is ready, easily scale your application on the cloud with DynamoDB.

- Secondary Indexes

Amazon DynamoDB gives you the flexibility to efficiently query on any attribute (column) using secondary indexes. You can create and delete secondary indexes for your table at any time.

- Streams (New)

Amazon DynamoDB Streams is a time ordered sequence of item level changes in any DynamoDB table. Using DynamoDB Streams, you can keep track of the latest item level change or get all item level updates in the last 24 hours, and use the data to build creative applications for replication, materialized views, backups and integration with other services.

- Cross-region Replication (New)

Amazon DynamoDB supports cross-region replication that automatically replicates DynamoDB tables across AWS regions. You can use cross-region replication to build

globally distributed applications with lower-latency data access, better traffic management, and easier disaster recovery and data migration.

- Triggers (New)

Amazon DynamoDB integrates with AWS Lambda to provide Triggers. Using Triggers, you will be able to automatically execute a custom function when item level changes in a DynamoDB table are detected.

- Free-text Search (New)

DynamoDB is integrated with Elasticsearch using the Amazon DynamoDB Logstash Plugin. With this integration, you can easily search DynamoDB content such as messages, locations, tags, and keywords.

- DynamoDB Titan Graph Database Integration (New)

DynamoDB is integrated with Titan, enabling you to efficiently store and traverse both small and large graphs up to hundreds of billions of vertices and edges. Graph databases are optimized for fast traversal of all kinds of relationships, such as social networks, customer relationship management (CRM), inventory, logistics management, pattern matching, and recommendation engines and more.

- Schema-less

Amazon DynamoDB has a flexible database schema. The data items in a table need not have the same attributes or even the same number of attributes. Multiple data types (strings, numbers, binary data, and sets) add richness to the data model.

- Strong Consistency, Atomic Counters

Unlike many non-relational databases, Amazon DynamoDB makes development easier by allowing you to use strong consistency on reads so that you are reading the latest values. The service also natively supports Atomic Counters, allowing you to atomically increment or decrement numerical attributes with a single API call.

- Integrated Monitoring

Amazon DynamoDB displays key operational metrics for your table in the AWS Management Console. The service also integrates with Amazon CloudWatch so you can see your request throughput and latency for each Amazon DynamoDB table, and easily track your resource consumption.

- Secure

Amazon DynamoDB uses proven cryptographic methods to authenticate users and prevent unauthorized data access. It also integrates with AWS Identity and Access Management (IAM) for fine-grained access control for users within your organization.

- Elastic MapReduce Integration

Amazon Elastic MapReduce (Amazon EMR) allows businesses to perform complex analytics of their large datasets using a hosted Hadoop framework on AWS. It is easy for customers to use Amazon EMR to analyze datasets stored in DynamoDB and archive the results in Amazon Simple Storage Service (Amazon S3), while keeping the original dataset in DynamoDB intact.

- Redshift Integration

Amazon Redshift complements Amazon DynamoDB with advanced business intelligence capabilities and a powerful SQL-based interface. When you copy data from a DynamoDB table into Amazon Redshift, you can perform complex data analysis queries on that data, including joins with other tables in your Amazon Redshift cluster.

- Data Pipeline Integration

You can use AWS Data Pipeline to automate data movement and transformation into and out of Amazon DynamoDB. The built-in scheduling capabilities of AWS Data Pipeline let you schedule and execute recurring jobs, without having to write your own complex data transfer or transformation logic.

- Management Console and APIs

The AWS Management Console for DynamoDB allows you to create, update, delete and query tables, adjust throughput, and set alarms with just a few clicks. In addition, DynamoDB supports rich functionality with a small number of APIs. Fewer APIs to learn means you can focus more on developing your application.

Conclusion

Studied different data models of Hive/Hbase/Cassandra/DynamoDB, and design data model for Cassandra.

Assignment No. 4

**Implement any one Partitioning
technique in Parallel Databases**

Assignment No 4

Implement any one Partitioning technique in Parallel Databases

Aim
To implement Partitioning technique in Parallel Databases

Objective(s)	
1	Distributes data over a number of processing elements
2	Different parallel operations use different types of parallelism.
3	These operations include parallel creation of partitioned indexes, and parallel creation of partitioned tables.
4	Partitioning can improve performance in data warehouses.

Theory
<p>Introduction</p> <p>A variety of hardware architectures allow multiple computers to share access to data, software, or peripheral devices. A parallel database is designed to take advantage of such architectures by running multiple instances which "share" a single physical database. In appropriate applications, a parallel server can allow access to a single database by users on multiple machines, with increased performance.</p> <p>A parallel server processes transactions in parallel by servicing a stream of transactions using multiple CPUs on different nodes, where each CPU processes an entire transaction. Using parallel data manipulation language you can have one transaction being performed by multiple nodes. This is an efficient approach because many applications consist of online insert and update transactions which tend to have short data access requirements. In addition to balancing the workload among CPUs, the parallel database provides for concurrent access to data and protects data integrity.</p> <p>Elements of Parallel processing</p> <ul style="list-style-type: none">• Speedup and Scaleup: Measure the performance goals of parallel processing• Synchronization: A Critical Success Factor• Locking• Messaging

Parallel Execution

Parallel execution dramatically reduces response time for data-intensive operations on large databases typically associated with decision support systems (DSS) and data warehouses. You can also implement parallel execution on certain types of online transaction processing (OLTP) and hybrid systems. Parallel execution is sometimes called parallelism. Simply expressed, parallelism is the idea of breaking down a task so that, instead of one process doing all of the work in a query, many processes do part of the work at the same time. An example of this is when four processes handle four different quarters in a year instead of one process handling all four quarters by itself. The improvement in performance can be quite high. In this case, each quarter will be a partition, a smaller and more manageable unit of an index or table.

Partitioning

The SQL standard does not provide much in the way of guidance regarding the physical aspects of data storage. The SQL language itself is intended to work independently of any data structures or media underlying the schemas, tables, rows, or columns with which it works. Nonetheless, most advanced database management systems have evolved some means of determining the physical location to be used for storing specific pieces of data in terms of the file system, hardware or even both. In MySQL, the InnoDB storage engine has long supported the notion of a tablespace, and the MySQL Server, even prior to the introduction of partitioning, could be configured to employ different physical directories for storing different databases.

Partitioning takes this notion a step further, by enabling you to distribute portions of individual tables across a file system according to rules which you can set largely as needed. In effect, different portions of a table are stored as separate tables in different locations. The user-selected rule by which the division of data is accomplished is known as a partitioning function, which in MySQL can be the modulus, simple matching against a set of ranges or value lists, an internal hashing function, or a linear hashing function. The function is selected according to the partitioning type specified by the user, and takes as its parameter the value of a user-supplied expression. This expression can be a column value, a function acting on one or more column values, or a set of one or more column values, depending on the type of partitioning that is used.

Partitioning Techniques

The partitioning techniques are as follows

- 1) RANGE Partitioning
- 2) LIST Partitioning
- 3) HASH Partitioning
- 4) KEY Partitioning

- 1) **RANGE partitioning.** This type of partitioning assigns rows to partitions based on column values falling within a given range.

A table that is partitioned by range is partitioned in such a way that each partition contains rows for which the partitioning expression value lies within a given range. Ranges should be contiguous but not overlapping, and are defined using the VALUES LESS THAN operator. For the next few examples, suppose that you are creating a table such as the following to hold personnel records for a chain of 20 video stores, numbered 1 through 20:

```
CREATE TABLE employees (  
  id INT NOT NULL,  
  fname VARCHAR(30),  
  lname VARCHAR(30),  
  hired DATE NOT NULL DEFAULT '1970-01-01',  
  separated DATE NOT NULL DEFAULT '9999-12-31',  
  job_code INT NOT NULL,  
  store_id INT NOT NULL  
);
```

This table can be partitioned by range in a number of ways, depending on your needs. One way would be to use the store_id column. For instance, you might decide to partition the table 4 ways by adding a PARTITION BY RANGE clause as shown here:

```
CREATE TABLE employees (  
  id INT NOT NULL,  
  fname VARCHAR(30),  
  lname VARCHAR(30),  
  hired DATE NOT NULL DEFAULT '1970-01-01',  
  separated DATE NOT NULL DEFAULT '9999-12-31',  
  job_code INT NOT NULL,  
  store_id INT NOT NULL  
)
```

```

PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN (21)
);

```

In this partitioning scheme, all rows corresponding to employees working at stores 1 through 5 are stored in partition p0, to those employed at stores 6 through 10 are stored in partition p1, and so on. Note that each partition is defined in order, from lowest to highest. This is a requirement of the PARTITION BY RANGE syntax; you can think of it as being analogous to a series of if ... elseif ... statements in C or Java in this regard.

It is easy to determine that a new row containing the data (72, 'Michael', 'Widenius', '1998-06-25', NULL, 13) is inserted into partition p2, but what happens when your chain adds a 21st store? Under this scheme, there is no rule that covers a row whose store_id is greater than 20, so an error results because the server does not know where to place it. You can keep this from occurring by using a “catchall” VALUES LESS THAN clause in the CREATE TABLE statement that provides for all values greater than the highest value explicitly named:

```

CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)
PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);

```

- 2) **LIST partitioning.** Similar to partitioning by RANGE, except that the partition is selected based on columns matching one of a set of discrete values.

List partitioning in MySQL is similar to range partitioning in many ways. As in partitioning by RANGE, each partition must be explicitly defined. The chief difference between the two types of partitioning is that, in list partitioning, each partition is defined and selected based on the membership of a column value in one of a set of value lists, rather than in one of a set of contiguous ranges of values. This is done by using PARTITION BY LIST(*expr*) where *expr* is a column value or an

expression based on a column value and returning an integer value, and then defining each partition by means of a VALUES IN (*value_list*), where *value_list* is a comma-separated list of integers.

To partition this table in such a way that rows for stores belonging to the same region are stored in the same partition, you could use the CREATE TABLE statement shown here:

```
CREATE TABLE employees (  
  id INT NOT NULL,  
  fname VARCHAR(30),  
  lname VARCHAR(30),  
  hired DATE NOT NULL DEFAULT '1970-01-01',  
  separated DATE NOT NULL DEFAULT '9999-12-31',  
  job_code INT,  
  store_id INT  
)  
PARTITION BY LIST(store_id) (  
  PARTITION pNorth VALUES IN (3,5,6,9,17),  
  PARTITION pEast VALUES IN (1,2,10,11,19,20),  
  PARTITION pWest VALUES IN (4,12,13,14,18),  
  PARTITION pCentral VALUES IN (7,8,15,16)  
);
```

- 3) **HASH partitioning.** With this type of partitioning, a partition is selected based on the value returned by a user-defined expression that operates on column values in rows to be inserted into the table. The function may consist of any expression valid in MySQL that yields a nonnegative integer value.

Partitioning by HASH is used primarily to ensure an even distribution of data among a predetermined number of partitions. With range or list partitioning, you must specify explicitly into which partition a given column value or set of column values is to be stored; with hash partitioning, MySQL takes care of this for you, and you need only specify a column value or expression based on a column value to be hashed and the number of partitions into which the partitioned table is to be divided.

To partition a table using HASH partitioning, it is necessary to append to the CREATE TABLE statement a PARTITION BY HASH (*expr*) clause, where *expr* is an expression that returns an integer. This can simply be the name of a column whose type is one of MySQL's integer types. In addition, you most likely want to follow this with PARTITIONS *num*, where *num* is a positive integer representing the number of partitions into which the table is to be divided.

```
CREATE TABLE employees (
  id INT NOT NULL,
  fname VARCHAR(30),
  lname VARCHAR(30),
  hired DATE NOT NULL DEFAULT '1970-01-01',
  separated DATE NOT NULL DEFAULT '9999-12-31',
  job_code INT,
  store_id INT
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
```

- 4) **KEY partitioning.** This type of partitioning is similar to partitioning by HASH, except that only one or more columns to be evaluated are supplied, and the MySQL server provides its own hashing function. These columns can contain other than integer values, since the hashing function supplied by MySQL guarantees an integer result regardless of the column data type. An extension to this type, LINEAR KEY, is also available. See Section 19.2.5, “KEY Partitioning”.

Partitioning by key is similar to partitioning by hash, except that where hash partitioning employs a user-defined expression, the hashing function for key partitioning is supplied by the MySQL server. MySQL Cluster uses MD5() for this purpose; for tables using other storage engines, the server employs its own internal hashing function which is based on the same algorithm as PASSWORD().

The syntax rules for CREATE TABLE ... PARTITION BY KEY are similar to those for creating a table that is partitioned by hash. The major differences are listed here:

- KEY is used rather than HASH.
- KEY takes only a list of zero or more column names. Any columns used as the partitioning key must comprise part or all of the table's primary key, if the table has one. Where no column name is specified as the partitioning key, the table's primary key is used, if there is one.

```
CREATE TABLE k1 (
  id INT NOT NULL PRIMARY KEY,
  name VARCHAR(20)
)
PARTITION BY KEY()
PARTITIONS 2;
```

```
CREATE TABLE k1 (
  id INT NOT NULL,
```

```
name VARCHAR(20),  
  UNIQUE KEY (id)  
)  
PARTITION BY KEY()  
PARTITIONS 2;
```

Conclusion

Studied different partitioning technique and implement Range Partitioning technique in Parallel Databases.

Assignment No. 5

**Implement Two Phase commit
protocol in Distributed Databases**

Assignment No 5

Implement Two Phase commit protocol in Distributed Databases

Aim	
Implement Two Phase commit protocol in Distributed Databases	

Objective(s)	
1	It is better utilizes the underlying communication infrastructure.
2	It is the strategy to designed to ensure that all the databases are updated.

Theory	
Introduction <p>The two phase commit protocol is a distributed algorithm which lets all sites in a distributed system agree to commit a transaction. The protocol results in either all nodes committing the transaction or aborting, even in the case of site failures and message losses. However, due to the work by Skeen and Stonebraker, the protocol will not handle more than one random site failure at a time. The two phases of the algorithm are broken into the COMMIT-REQUEST phase, where the COORDINATOR attempts to prepare all the COHORTS, and the COMMIT phase, where the COORDINATOR completes the transactions at all COHORTS.</p>	
Assumptions <p>The protocol works in the following manner: One node is designated the coordinator, which is the master site, and the rest of the nodes in the network are called cohorts. Other assumptions of the protocol include stable storage at each site and use of a write ahead log by each node. Also, the protocol assumes that no node crashes forever, and eventually any two nodes can communicate with each other. The latter is not a big deal since network communication can typically be rerouted. The former is a much stronger assumption; suppose the machine blows up!</p>	
Basic Algorithm <p>During phase 1, initially the coordinator sends a query to commit message to all cohorts. Then it waits for all cohorts to report back with the agreement message. The cohorts, if the transaction was successful, write an entry to the undo log and an entry to the redo log. Then the cohorts reply with an agree message, or an abort if the transaction failed at a cohort node. During phase 2, if the coordinator receives an agree message from all cohorts, then it writes a commit record into its log and sends a commit message to all the cohorts. If all agreement messages do not come back the coordinator sends an abort message. Next the coordinator waits for the acknowledgement from the cohorts. When acks are received from all cohorts the coordinator writes a complete record to its log. Note the coordinator will wait forever for all the acknowledgements to come back. If the cohort receives a commit message, it releases all the</p>	

locks and resources held during the transaction and sends an acknowledgement to the coordinator. If the message is abort, then the cohort undoes the transaction with the undo log and releases the resources and locks held during the transaction. Then it sends an acknowledgement.

Disadvantages

The greatest disadvantage of the two phase commit protocol is the fact that it is a blocking protocol. A node will block while it is waiting for a message. This means that other processes competing for resource locks held by the blocked processes will have to wait for the locks to be released. A single node will continue to wait even if all other sites have failed. If the coordinator fails permanently, some cohorts will never resolve their transactions. This has the effect that resources are tied up forever.

Another disadvantage is the protocol is conservative. It is biased to the abort case rather than the complete case.

The Detailed Commit Protocol

At the COORDINATOR:

The COORDINATOR sends the message to each COHORT. The COORDINATOR is now in the preparing transaction state.

Now the COORDINATOR waits for responses from each of the COHORTS. If any COHORT responds ABORT then the transaction must be aborted, proceed to step 5. If all COHORTS respond AGREED then the transaction may be committed, and proceed to step 3. If after some time period all COHORTS do not respond the COORDINATOR can either transmit ABORT messages to all COHORTS or transmit COMMIT-REQUEST messages to the COHORTS that have not responded. In either case the COORDINATOR will eventually go to state 3 or state 5.

Record in the logs a COMPLETE to indication the transaction is now completing. Send COMMIT message to each of the COHORTS.

Wait for each COHORT to respond. They must reply COMMIT. If after some time period some COHORT has not responded retransmit the COMMIT message. Once all COHORTS have replied erase all associated information from permanent memory (COHORT list, etc.). DONE.

Send the ABORT message to each COHORT.

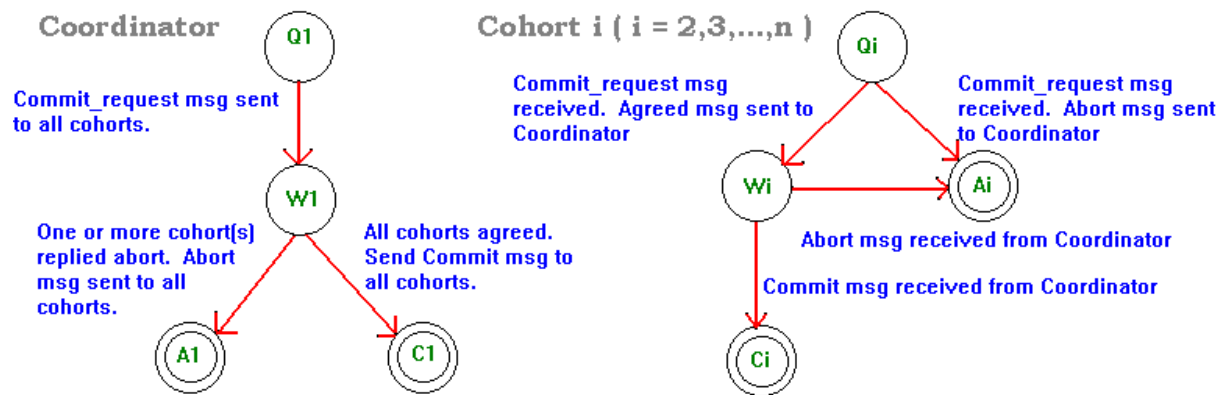
At COHORTS:

If a COMMIT-REQUEST message is received for some transaction t which is unknown at the COHORT (never ran, wiped out by crash, etc), reply ABORT. Otherwise write the new state of the transaction to the UNDO and REDO log in permanent memory. This allows for the old state to be recovered (in event of later abort) or committed on demand regardless of crashes. The read locks of a transaction may be released at this time; however, the write locks are still

maintained. Now send AGREED to the COORDINATOR.

If an ABORT message is received then kill the transaction, which involves deleting the new state if the transaction from the REDO and UNDO log the new state of the transaction and restoring any state before the transaction occurred.

If a COMMIT message is received then the transaction is either prepared for commital or already committed. If it is prepared, perform all the operations necessary to update the database and release the remaining locks the transaction possesses. If it is already committed, no further action is required. Respond COMMITTED to the COORDINATOR.



Finite State Diagram of Commit Protocol for Coordinator and Cohort

Legend

- A? - accept state
- C? - commit state
- Q? - query state
- W? - wait state

We assert the claim that if one COHORT completes the transaction all COHORTS complete the transaction eventually. The proof for correctness proceeds somewhat informally as follows: If a COHORT is completing a transaction, it is so only because the COORDINATOR sent it a COMMIT message. This message is only sent when the COORDINATOR is in the commit phase, in which case all COHORTS have responded to the COORDINATOR AGREED. This means all COHORTS have prepared the transaction, which implies any crash at this point will not harm the transaction data because it is in permanent memory. Once the COORDINATOR is completing, it is insured every COHORT completes before the COORDINATOR's data is erased. Thus crashes of the COORDINATOR do not interfere with the completion. Therefore if any COHORT completes, then they all do. The abort sequence can be argued in a similar manner. Hence the atomicity of the transaction is guaranteed to fail or complete globally.

Conclusion

Studied and implement two phase commit (2PC) protocol.

Assignment No. 6

**Design Persistent Objects using JDO and
implement min 10 queries on objects using
JDOQL in ObjectDB NOSQL DATABASE**

Assignment No 6

Design Persistent Objects using JDO and implement min 10 queries on objects using JDOQL in ObjectDB NOSQL DATABASE

Aim	
Design Persistent Objects using JDO and implement min 10 queries on objects using JDOQL in ObjectDB NOSQL DATABASE	

Objective(s)	
1	Study of JDO
2	Study of Persistent Object
3	Implement min 10 queries using JDOQL

Theory
<p>Introduction:</p> <p>Java Data Objects (JDO)</p> <p>The Java Data Objects (JDO) specification is part of the Sun Java Community Process. JDO is a Java application program interface (API) for transparent persistence. It works with object and relational databases as well as other types of systems. It is designed to also work well with EJB and J2EE.</p> <p>The JDO object model is determined by a set of Java classes and XML metadata file. The metadata file contains modeling directives that either override the semantics specified in Java or provides semantics that cannot be expressed in Java. An enhancer is provided that enhances the Java classes based on these modeling directives.</p> <p>The JDO specification provides a standard interface for accessing, storing, and processing persistent objects. The primary aspects in the JDO model are:</p> <p>Persistence Manager: Persistence Managers negotiate accesses, transactions, and queries between applications and the underlying data store. More on the JDO Persistence Manager.</p> <p>Transaction: Transactions provide for atomic, consistent, isolated, and durable management of data (ACID properties).</p> <p>Query: The JDO Query Language (JDOQL) allows users to search for persistent objects matching specific criteria. JDOQL is meant to be query language neutral so that the underlying query language could be SQL, an object database query language such as OQL, or a specialize API to a hierarchical database or EIS system.</p>

Persistence Capable classes: The actual entities those are stored and fetched. There are three identity models to allow for different underlying database management systems.

The JDO specification uses the Java language as much as possible, which allows the transparent integration of Java.

Benefits of Using JDO for Application Programming

Ease of use: Application programmers can focus on their domain object model and leave the details of persistence (field-by-field storage of objects) to the JDO implementation.

Portability: Applications written with the JDO API can be run on multiple implementations without recompiling or changing source code. Metadata, which describes persistence behavior external to the Java source code including most commonly used features of O/R mapping, is highly portable.

Database independence: Applications written with the JDO API are independent of the underlying database. JDO implementations support many different kinds of transactional data stores, including relational and object databases, XML, flat files, and others.

High performance: Application programmers delegate the details of persistence to the JDO implementation, which can optimize data access patterns for optimal performance.

Integration with EJB: Applications can take advantage of EJB features such as remote message processing, automatic distributed transaction coordination, and security, using the same domain objects models throughout the enterprise.

Interface PersistenceManagerFactory

The PersistenceManagerFactory is the interface to use to obtain PersistenceManager instances.

All PersistenceManager instances obtained from the same PersistenceManagerFactory will have the same default properties.

PersistenceManagerFactory instances may be configured and serialized for later use. They may be stored via JNDI and looked up and used later. Any properties configured will be saved and restored. Once the first PersistenceManager is obtained from

the `PersistenceManagerFactory`, the factory can no longer be configured.

If the `ConnectionFactory` property is set (non-null) then all other `Connection` properties including `ConnectionFactoryName` are ignored; otherwise, if `ConnectionFactoryName` is set (non-null) then all other `Connection` properties are ignored. Similarly, if the `ConnectionFactory2` property is set (non-null) then `ConnectionFactory2Name` is ignored.

Operational state (`PersistenceManager` pooling, connection pooling, operational parameters) must not be serialized.

Interface `PersistenceManager`

`PersistenceManager` is the primary interface for JDO-aware application components. It is the factory for `Query` and `Transaction` instances, and contains methods to manage the life cycle of `PersistenceCapable` instances.

A `PersistenceManager` is obtained from the [PersistenceManagerFactory](#) (recommended) or by construction.

Interface `Transaction`

The JDO `Transaction` interface provides for initiation and completion of transactions under user control. It is a sub-interface of the [PersistenceManager](#) that deals with options and transaction demarcation.

`Transaction` options include whether optimistic concurrency control should be used for the current transaction, whether instances may hold values in the cache outside transactions, and whether values should be retained in the cache after transaction completion. These options are valid for both managed and non-managed transactions.

`Transaction` initiation and completion methods have similar semantics to `javax.transaction.UserTransaction` when used outside a managed environment. When used in a managed environment, transaction initiation and completion methods may only be used with bean-managed transaction semantics.

Interface `Query`

The `Query` interface allows applications to obtain persistent instances, values, and aggregate data from the data store. The [PersistenceManager](#) is the factory for `Query` instances. There

may be many Query instances associated with a PersistenceManager. Multiple queries might be executed simultaneously by different threads, but the implementation might choose to execute them serially. In either case, the implementation must be thread safe.

There are three required elements in a Query: the class of the results, the candidate collection of instances, and the filter.

There are optional elements: parameter declarations, variable declarations, import statements, ordering and grouping specifications, result and result class, the range of results, and flags indicating whether the query result is unique and whether the query can be modified.

The query namespace is modeled after methods in Java:

- `setClass` corresponds to the class definition
- `declareParameters` corresponds to formal parameters of a method
- `declareVariables` corresponds to local variables of a method
- `setFilter` and `setOrdering` correspond to the method body

There are two namespaces in queries. Type names have their own namespace that is separate from the namespace for fields, variables and parameters.

The method `setClass` introduces the name of the candidate class in the type namespace. The method `declareImports` introduces the names of the imported class or interface types in the type namespace. Imported type names must be unique. When used (e.g. in a parameter declaration, cast expression, etc.) a type name must be the name of the candidate class, the name of a class or interface imported by method `declareImports`, or denote a class or interface from the same package as the candidate class.

The method `setClass` introduces the names of the candidate class fields.

The method `declareParameters` introduces the names of the parameters. A name introduced by `declareParameters` hides the name of a candidate class field of the same name. Parameter names must be unique.

The method `declareVariables` introduces the names of the variables. A name introduced by `declareVariables` hides the name of a candidate class field if equal. Variable names must

be unique and must not conflict with parameter names.

The result of the query by default is a list of result class instances, but might be specified via setResult. The class of the result by default is the candidate class, but might be specified via setResultClass.

A hidden field may be accessed using the 'this' qualifier: this.fieldName.

The Query interface provides methods which execute the query based on the parameters given. They return a single instance or a List of result class instances which the user can iterate to get results. The signature of the execute methods specifies that they return an Object which must be cast to the appropriate result by the user.

Any parameters passed to the execute methods are used only for this execution, and are not remembered for future execution.

Interface Extent<E>

Instances of the Extent class represent the entire collection of instances in the data store of the candidate class or interface possibly including its subclasses or sub interfaces.

.

The Extent instance has two possible uses:

1. to iterate all instances of a particular class or interface
2. to execute a Query in the data store over all instances of a particular class or interface

Conclusion

Studied Persistent Objects using JDO and implement min 10 queries on objects using JDOQL in ObjectDB NOSQL DATABASE.

Assignment No. 7

**Create XML, XML schemas, DTD
for any database application and
implement min 10 queries using
XQuery FLOWR expression and
XPath**

Assignment No 7

Create XML, XML schemas, DTD for any database application and implement min 10 queries using XQuery FLOWR expression and XPath

Aim
To Create XML schemas , DTD for a database application and implement queries using XQuery FLOWR expression and XPath

Objective(s)	
1	XML DTD and XML Schema define rules to describe data.
2	XPath is a syntax for defining parts of an XML document.
3	XPath uses path expressions to navigate in XML documents.

Theory
<p>Introduction:</p> <p>XML</p> <p>Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications all of which are free open standards. The design goals of XML emphasize simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services. Several schema systems exist to aid in the definition of XML-based languages, while many application programming interfaces (APIs) have been developed to aid the processing of XML data.</p> <p>XML Schema</p> <p>An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints. There are languages developed specifically to express XML schemas. The Document Type Definition (DTD) language, which is native to the XML specification, is a schema language that is of relatively limited capability, but that also has other uses in XML aside from the expression of schemas. Two more expressive XML schema languages in widespread use are XML Schema (with a capital S) and RELAX NG. The mechanism for associating an XML document with a schema varies according to the schema</p>

language. The association may be achieved via markup within the XML document itself, or via some external means.

DTD

A document type definition (DTD) is a set of markup declarations that define a document type for an SGML-family markup language (SGML, XML, HTML). A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference. XML uses a subset of SGML DTD. As of 2009, newer XML namespace-aware schema languages (such as W3C XML Schema and ISO RELAX NG) have largely superseded DTDs. A namespace-aware version of DTDs is being developed as Part 9 of ISO DSDL. DTDs persist in applications that need special publishing characters, such as the XML and HTML Character Entity References, which derive from larger sets defined as part of the ISO SGML standard effort.

Xml DTD

```
<?xml version="1.0"?><!DOCTYPE PARTS [  
<!ELEMENT PARTS (TITLE,(PART)+)>  
<!ATTLIST PARTS xmlns CDATA #FIXED "">  
<!ELEMENT TITLE (#PCDATA)>  
<!ATTLIST TITLE xmlns CDATA #FIXED "">  
<!ELEMENT PART (ITEM,MANUFACTURER,MODEL,COST)>  
<!ATTLIST PART xmlns CDATA #FIXED "">  
<!ELEMENT ITEM (#PCDATA)>  
<!ATTLIST ITEM xmlns CDATA #FIXED "">  
<!ELEMENT MANUFACTURER (#PCDATA)>  
<!ATTLIST MANUFACTURER xmlns CDATA #FIXED "">  
<!ELEMENT MODEL (#PCDATA)>  
<!ATTLIST MODEL xmlns CDATA #FIXED "">  
<!ELEMENT COST (#PCDATA)>  
<!ATTLIST COST xmlns CDATA #FIXED "">  
>]
```

XML Document

```
<PARTS>  
<TITLE>Computer Parts</TITLE>  
<PART>  
<ITEM>Motherboard</ITEM>  
<MANUFACTURER>ASUS</MANUFACTURER>  
<MODEL>P3B-F</MODEL>  
<COST> 123.00</COST>  
</PART>  
<PART>  
<ITEM>Video Card</ITEM>  
<MANUFACTURER>ATI</MANUFACTURER>  
<MODEL>All-in-Wonder Pro</MODEL>
```

```

<COST> 160.00</COST>
</PART>
<PART>
<ITEM>Sound Card</ITEM>
<MANUFACTURER>Creative Labs</MANUFACTURER>
<MODEL>Sound Blaster Live</MODEL>
<COST> 80.00</COST>
</PART>
<PART>
<ITEM> inch Monitor</ITEM>
<MANUFACTURER>LG Electronics</MANUFACTURER>
<MODEL> 995E</MODEL>
<COST> 290.00</COST>
</PART>
</PARTS>

```

XPATH

- XPath (the XML Path language) is a language for finding information in an XML document.
- XPath is a syntax for defining parts of an XML document.
- XPath uses path expressions to navigate in XML documents.
- XPath is also used in Xquery.

FLWOR EXPRESSIONS

FLWOR is an acronym for "For, Let, Where, Order by, Return".

The **for** clause selects all book elements under the bookstore element into a variable called \$x.

The **where** clause selects only book elements with a price element with a value greater than 30.

The **order by** clause defines the sort-order. Will be sort by the title element.

The **return** clause specifies what should be returned. Here it returns the title elements.

How to Select Nodes From "books.xml" With FLWOR

Look at the following path expression:

doc("books.xml")/bookstore/book[price>30]/title

The expression above will select all the title elements under the book elements that are under the bookstore element that have a price element with a value that is higher than 30.

The following FLWOR expression will select exactly the same as the path expression above:

*for \$x in doc("books.xml")/bookstore/book
 where \$x/price>30
 return \$x/title*

Conclusion
Studied XML, XML scemas , DTD for database application and execute queries using XQuery, Flower expression and XPath.

Assignment No. 8

**Design database schemas and
implement min 10 queries using
Hive/ Hbase/ Cassandra column
based databases**

Assignment No 8

Design database schemas and implement min 10 queries using Hive/ Hbase/ Cassandra column based databases

Aim	
To Design database schemas and implement queries using Cassandra databases	

Objective(s)	
1	Study of NOSQL Cassandra.
2	Study the procedure to execute a query using Apache Cassandra.
3	Execute min 10 queries using Cassandra column based database.

Theory	
<p>Cassandra:</p> <ul style="list-style-type: none">• Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.• Features :<ol style="list-style-type: none">1) Scalability:<ul style="list-style-type: none">○ Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.2) Fault-tolerant:<ul style="list-style-type: none">○ Data is automatically replicated to multiple nodes for fault-tolerance.○ Replication across multiple data centers is supported.○ Failed nodes can be replaced with no downtime.3) MapReduce support:<ul style="list-style-type: none">○ Cassandra has Hadoop integration, with MapReduce support.4) Query language:<ul style="list-style-type: none">○ Cassandra introduces CQL (Cassandra Query Language), a SQL-like alternative to the traditional RPC interface.<ul style="list-style-type: none">• Keyspace:<p>Keyspace is the outermost container for data It is similar to the schema in a relational database Basic attributes of Keyspace are:</p><p>Replication Factor: It is the number of machines in the cluster that will receive copies of the same data</p><p>Replica Placement Strategy: It is a strategy to place replicas in the ring</p><ul style="list-style-type: none">○ Simple Strategy○ Old Network Topology Strategy○ Network Topology Strategy<p style="text-align: center;">Column Families</p>	

Column family is a NoSQL object that contains columns of related data. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to a value that is a set of columns.

It is similar to a table in a relational database and each key-value pair being a row.

Each column is a tuple (triplet) consisting of

- Column name
- Value
- Timestamp

- **Verifying Installation :**

DataStax Community Edition must be installed on system before installing Cassandra. Verify the Cassandra installation using the following command:

```
$ Cassandra version
```

If Cassandra is already installed on system, then you will get the following response:

```
Connected to Test Cluster at 127.0.0.1:9042.
```

```
[cqlsh 5.0.1 | Cassandra 2.2.4 | CQL spec 3.3.1 | Native protocol v4]
```

```
Use HELP for help.
```

```
WARNING: pyreadline dependency missing. Install to enable tab completion. From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

- **Creating KEYSPACE :**

Cassandra is technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it.

Create **KEYSPACE** Statement

Create **KEYSPACE** is a statement used to create a **KEYSPACE** in Cassandra. A **KEYSPACE** in Cassandra is a **KEYSPACE** or a collection of tables. The **syntax** for this statement is as follows:

```
cqlsh> CREATE KEYSPACE ABC userdb replication = { 'class': 'SimpleStrategy', 'replication_factor': '1' };
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command. The following query is executed to create a database named **userdb**:


```
cqlsh> userdb;
```

or

```
cqlsh> CREATE SCHEMA userdb;
```

The following query is used to verify a databases list:

```
cqlsh>:userdb> show tables;
```

Improper show command.

default

userdb

For creating Table:

Create Table

```
CREATE TABLE test_table (  
    id int,  
    address text,  
    name text,  
    PRIMARY KEY ((id))  
);
```

CURD using cql Updating Table:

Update Table

```
insert into test_table (id, name, address) values  
(4, 'somnath', 'Sus');
```

CURD using cql Delete Table

Deleting rows from Table

```
delete from test_table where id =1;
```

The following queries are used to drop a database. Let us assume that the database name

is **userdb**.

```
cqlsh:userdb> delete from Tablename where condition;
```

For describing tables

```
cqlsh:userdb> describe tables;
```

show all table names

```
cqlsh:userdb>
```

For help of any topic:

For Help of any Topic

```
Cqshl> Help;
```

Display topics

```
Cqshl> Help topic name;
```

Help open in Browser.

Conclusion

Studied Cassandra database and implement queries using Cassandra databases.

Assignment No. 9

**Design database schemas and
implement min 10 queries using
DynamoDB key Value based
databases**

Assignment No 9

Design database schemas and implement min 10 queries using DynamoDB key Value based databases

Aim
To design database schemas and implement queries using DynamoDB key Value based databases

Objective(s)	
1	Study DynamoDB the latest NoSql-database-as-a-service offering by Amazon.
2	Study procedure to execute a query using DynamoDB .
3	Study different concept of DynamoDB's data model like tables, items and attributes.

Theory
<p>Introduction:</p> <p>Database Schema:- A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects. You can assign a user login permissions to a single schema so that the user can only access the objects they are authorized to access.</p> <p>DynamoDB Data Model:- Tables, Items, and Attributes</p> <p>In DynamoDB, a <i>table</i> is a collection of <i>items</i> and each item is a collection of <i>attributes</i>.</p> <p>In a relational database, a table has a predefined schema such as the table name, primary key, list of its column names and their data types. All records stored in the table must have the same set of columns. In contrast, DynamoDB only requires that a table has a primary key, but does not require you to define all of the attribute names and data types in advance. Individual items in a DynamoDB table can have any number of attributes, although there is a limit of 400 KB on the item size. An item size is the sum of lengths of its attribute names and values (binary and UTF-8 lengths).</p> <p>Each attribute in an item is a name-value pair. An attribute can be a scalar (single-valued), a JSON document, or a set. For example, consider storing a catalog of products in DynamoDB. You can create a table, Product Catalog, with the <i>Id</i> attribute as its primary key. The primary key uniquely identifies each item, so that no two products in the table can have the same <i>Id</i>.</p> <p>Primary Key</p> <p>When you create a table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key.</p> <p>DynamoDB supports two different kinds of primary keys:</p> <p>1]Partition Key – A simple primary key, composed of one attribute known as the partitionkey. DynamoDB uses the partition key's value as input to an internal hash function; the output from</p>

the hash function determines the partition where the item will be stored. No two items in a table can have the same partition key value.

Partition Key and Sort Key – A composite primary key, composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*. DynamoDB uses the partition key value as input to an internal hash function; the output from the hash function determines the partition where the item will be stored. All items with the same partition key are stored together, in sorted order by sort key value. It is possible for two items to have the same partition key value, but those two items must have different sort key values.

Table Name	Primary Key Type	Partition Key Name	Sort Key Name
Forum (<u>Name</u> , ...)	Simple	Name	-
Thread (<u>ForumName</u> , <u>Subject</u> , ...)	Composite	ForumName	Subject
Reply (<u>Id</u> , <u>ReplyDateTime</u> , ...)	Composite	Id	ReplyDateTim

Secondary Indexes

DynamoDB supports two kinds of secondary indexes:

- **Global secondary index** – an index with a partition key and sort key that can be different from those on the table.
- **Local secondary index** – an index that has the same partition key as the table, but a different sort key.

DynamoDB Data Types

DynamoDB supports the following data types:

- **Scalar types** – Number, String, Binary, Boolean, and Null.
- **Document types** – List and Map.
- **Set types** – String Set, Number Set, and Binary Set.

For example, in the Product Catalog table, the Id is a Number type attribute and Authors is a String Set type attribute. Note that primary key attributes must be of type String, Number, or Binary.

Item Distribution

DynamoDB stores data in partitions. A *partition* is an allocation of storage for a table, backed by solid state drives (SSDs) and automatically replicated across three facilities within an AWS region. Partition management is handled entirely by DynamoDB—customers never need to manage partitions themselves. If your storage requirements exceed a partition's capacity, DynamoDB allocates additional partitions automatically.

When you create a table, the initial status of the table is CREATING. During this phase, DynamoDB allocates one partition for the table. You can begin writing and reading table data after the table status changes to ACTIVE.

As the amount of data in the table approaches the partition's maximum capacity, DynamoDB

allocates another partition to the table, and then distributes the data items among the old partition and the new one. This activity occurs in the background, and is transparent to your applications. The more data you add to the table, the more partitions that DynamoDB will allocate—as many as necessary to store your table's data.

DynamoDB does not deallocate or coalesce partitions. If a table spans multiple partitions, and you delete most of the data (or all of it), the partitions will still be allocated to the table

Query

A *Query* operation uses the primary key of a table or a secondary index to directly access items from that table or index.

Use the *KeyConditionExpression* parameter to provide a specific value for the partition key. The *Query* operation will return all of the items from the table or index with that partition key value. You can optionally narrow the scope of the *Query* operation by specifying a sort key value and a comparison operator in *KeyConditionExpression*. You can use the *ScanIndexForward* parameter to get results in forward or reverse order, by sort key.

Queries that do not return results consume the minimum number of read capacity units for that type of read operation.

If the total number of items meeting the query criteria exceeds the result set size limit of 1 MB, the query stops and results are returned to the user with the *LastEvaluatedKey* element to continue the query in a subsequent operation. Unlike a *Scan* operation, a *Query* operation never returns both an empty result set and a *LastEvaluatedKey* value. *LastEvaluatedKey* is only provided if the results exceed 1 MB, or if you have used the *Limit* parameter.

You can query a table, a local secondary index, or a global secondary index. For a query on a table or on a local secondary index.

Steps to install Dynamo DB.

- 1) Copy and extract the archive [[\\10.10.223.32\Installables\DynamoDB\dynamodb_local_2015-07-16_1.0.zip](#)] in local m/c
e.g. d:\dynamodb\
- 2) Change directory and execute below command
D:\>cd dynamodb

D:\dynamodb>java -Djava.library.path=./DynamoDBLocal_lib -jar
DynamoDBLocal.jar -sharedDb

```
ca. Administrator: Command Prompt - java -Djava.library.path=../DynamoDBLocal_lib -jar DynamoDBLocal...
Connection-specific DNS Suffix . : ATT.TechMahindra.com
Tunnel adapter Teredo Tunneling Pseudo-Interface:
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . :
C:\Users\pk00339344>d:
D:\>cd dynamodb
D:\dynamodb>
D:\dynamodb>
D:\dynamodb>java -Djava.library.path=../DynamoDBLocal_lib -jar DynamoDBLocal.jar
-sharedDb
Initializing DynamoDB Local with the following configuration:
Port: 8000
InMemory: false
DbPath: null
SharedDb: true
ShouldDelayTransientStatuses: false
CorsParams: *
```

Dynamo DB will start and will use 8000 as a default port.

- 3) Open a web browser on your computer and go to the following URL:<http://localhost:8000/shell>

Conclusion

We have design database schemas and implement 10 queries using DynamoDB keyValue based database.

Assignment No. 10

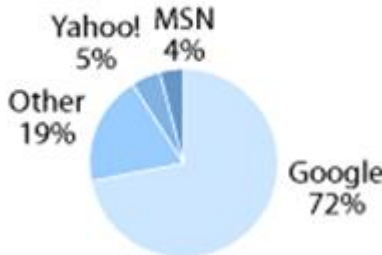
Implement Web Page ranking algorithm

Assignment No 10

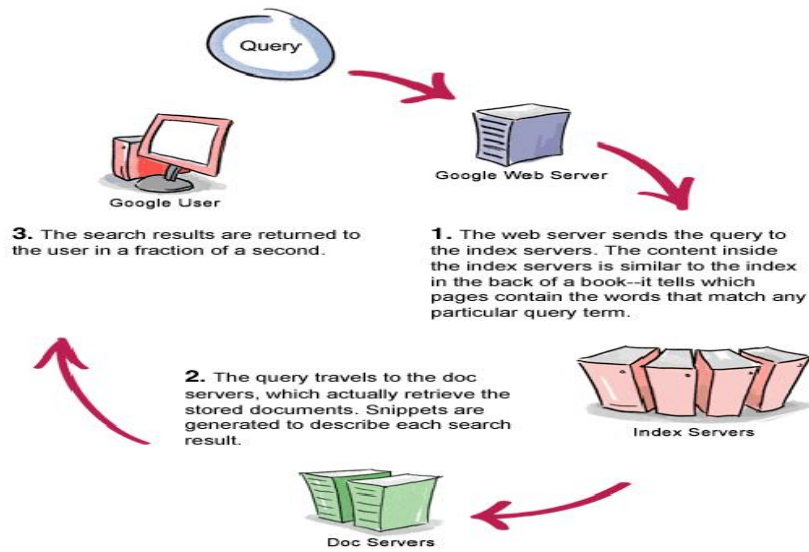
Implement Web Page ranking algorithm

Aim	
Implement Web Page ranking algorithm	

Objective(s)	
1	Study of web ranking algorithm
2	Study of web ranking algorithm for Google

Theory										
<p>Introduction:</p> <ul style="list-style-type: none">• The Web can be seen as a very large, unstructured but ubiquitous database.• So we need for efficient tools to manage, retrieve and filter the information.• There are 3 different forms of searching the Web:<ol style="list-style-type: none">1. Search Engines, which index a portion of Web pages as a full-text database.2. Web Directories, which classify selected Web documents by subject.3. Searching by hyperlinks structure. <p>About Google</p> <ul style="list-style-type: none">• The name "Google" is a play on the word "googol", which refers to the number represented by 1 followed by one hundred zeros.• Google receives over 200 million queries each day through its various services.• As of January 2006, Google has indexed 9.7 billion web pages, 1.3 billion images, and over one billion Usenet messages — in total, approximately 12 billion items. It also caches much of the content that it indexes. <p>"Which search engines do you use most frequently to search the Internet"</p>  <p>A pie chart illustrating the frequency of search engine usage. The chart is divided into four segments: a large light blue segment for Google (72%), a medium blue segment for Other (19%), a small dark blue segment for Yahoo! (5%), and a very small dark blue segment for MSN (4%).</p> <table><tr><th>Search Engine</th><th>Percentage</th></tr><tr><td>Google</td><td>72%</td></tr><tr><td>Other</td><td>19%</td></tr><tr><td>Yahoo!</td><td>5%</td></tr><tr><td>MSN</td><td>4%</td></tr></table>	Search Engine	Percentage	Google	72%	Other	19%	Yahoo!	5%	MSN	4%
Search Engine	Percentage									
Google	72%									
Other	19%									
Yahoo!	5%									
MSN	4%									

How Google works



- The idea is that the documents on the web have different degrees of "importance".
- Google will show the most important pages first.
- The idea is that more important pages are likely to be more relevant to any query than non-important pages.
- Google considers over 100 factors, including:

PageRank algorithm

Popularity of page.

Position and size of the search terms within page.

Unique Content.

Terms order.

Page size and load time.

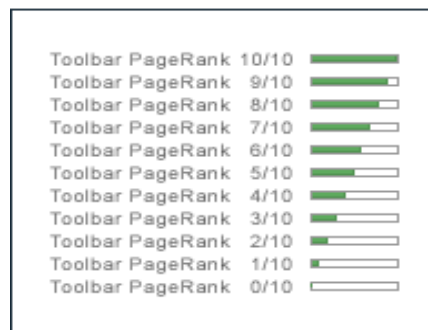
Error free websites.

Important incoming links.

Website Optimization.

Google PageRank

- Page Rank is a Numeric value to measure how important a page is.
- PageRank (PR) is the actual ranking of a page, as determined by Google.
- A probability is expressed as a numeric value between 0 and 10



Toolbar PageRank	Real PageRank		
0/10	0.15	-	0.9
1/10	0.9	-	5.4
2/10	5.4	-	32.4
3/10	32.4	-	194.4
4/10	194.4	-	1,166.4
5/10	1,166.4	-	6,998.4
6/10	6,998.4	-	41,990.4
7/10	41,990.4	-	251,942.4
8/10	251,942.4	-	1,511,654.4
9/10	1,511,654.4	-	9,069,926.4
10/10	9,069,926.4	-	$0.85 \times N + 0.15$

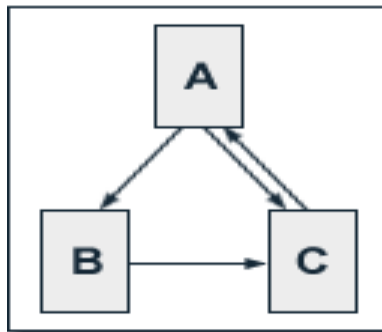
- A probability is expressed as a numeric value between 0 and 10.
 - Purpose: to increase the quality of search results and has left all of its competitors for dead.
 - The Google Page Rank is based on how many links you have to and from your pages, and their respective page rank.
 - Google update its index every four weeks.
 - Each time they update their database of web pages, index invariably shifts:
 - They find new sites, lose some sites, and sites ranking may change.
 - The Google PageRank (PR) is calculated for every webpage that exists in Google's database.
 - It's real value varies from 0 to infinite, but for representation purposes it is converted to a value between 0 and 10 (from low to high).
 - The calculation of the PR for a page is based on the quantity and quality of webpages that contain links to that
-
- www.prchecker.info/to chk pgrank of given webpage)
 - If you've installed the Google Toolbar, you MAY have the PageRank meter installed.



Google System Features

- PageRank – Bring order to the web
 - $PR(A) = (1-d) + d (PR(T1)/C(T1) + + PR (Tn)/C(Tn))$
 - ✓ $PR(A)$ is the PageRank of page A.
 - ✓ $PR(T1)$ is the PageRank of the page that links to our (A) page.
 - ✓ $C(T1)$ is the number of links going out of page T1.
 - ✓ d is a damping factor, usually set to 0.5.

Example



- PageRank calculation:
- $PR(A) = 0.5 + 0.5 PR(C)$
- $PR(B) = 0.5 + 0.5 (PR(A) / 2)$
- $PR(C) = 0.5 + 0.5 (PR(A) / 2 + PR(B))$

These equations can easily be solved. We get the following PageRank values for the single pages:

- $PR(A) = 14/13 = 1.07692308$
- $PR(B) = 10/13 = 0.76923077$
- $PR(C) = 15/13 = 1.15384615$

$$(PR(A) = (1-d) + d (PR(T1)/C(T1) + + PR (Tn)/C(Tn)))$$

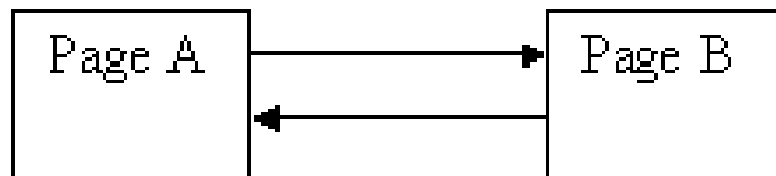
- How is PageRank Calculated?**
- This is where it gets tricky. The PR of each page depends on the PR of the pages pointing to it.
- But we won't know what PR those pages have until the pages pointing to **them** have

their PR calculated and so on...

- And when you consider that page links can form circles it seems impossible to do this calculation!
- **PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.**
- What that means to us is that we can just go ahead and calculate a page's PR **without knowing the final value of the PR of the other pages.**

That seems strange but, basically, each time we run the calculation we're getting a closer estimate of the final value. So all we need to do is remember the each value we calculate and repeat the calculations lots of times until the numbers stop changing much

- Lets take the simplest example network: two pages, each pointing to the other:
- Each page has one outgoing link (the outgoing count is 1, i.e. $C(A) = 1$ and $C(B) = 1$).

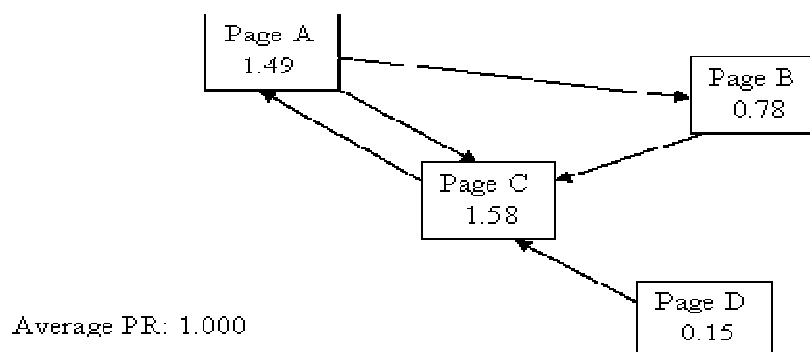
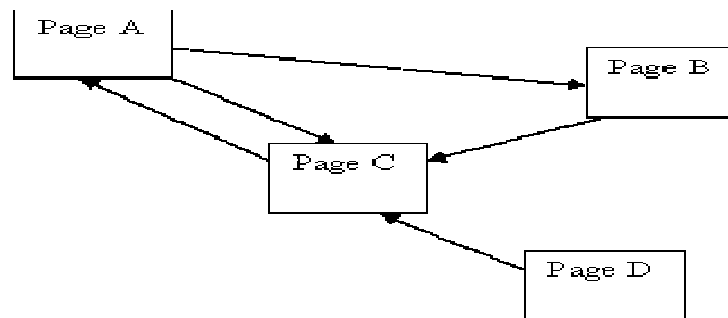


- **Guess 1**
- We don't know what their PR should be to begin with, so let's take a guess at 1.0 and do some calculations:
- $d = 0.85$
- $PR(A) = (1 - d) + d(PR(B)/1)$
- $PR(B) = (1 - d) + d(PR(A)/1)$
- i.e.
- $PR(A) = 0.15 + 0.85 * 1 = 1$
- $PR(B) = 0.15 + 0.85 * 1 = 1$
- Hmm, the numbers aren't changing at all! So it looks like we started out with a lucky guess!!!
- **Guess 2**
- No, that's too easy, maybe I got it wrong (and it wouldn't be the first time). Ok, let's start the guess at 0 instead and re-calculate:
- $PR(A) = 0.15 + 0.85 * 0 = 0.15$
- $PR(B) = 0.15 + 0.85 * 0.15 = 0.2775$
- NB. we've already calculated a "next best guess" at $PR(A)$ so we use it here
- And again:
- $PR(A) = 0.15 + 0.85 * 0.2775 = 0.385875$
- $PR(B) = 0.15 + 0.85 * 0.385875 = 0.47799375$
- And again
- $PR(A) = 0.15 + 0.85 * 0.47799375 = 0.5562946875$
- $PR(B) = 0.15 + 0.85 * 0.5562946875 = 0.622850484375$

And so on. The numbers just keep going up.

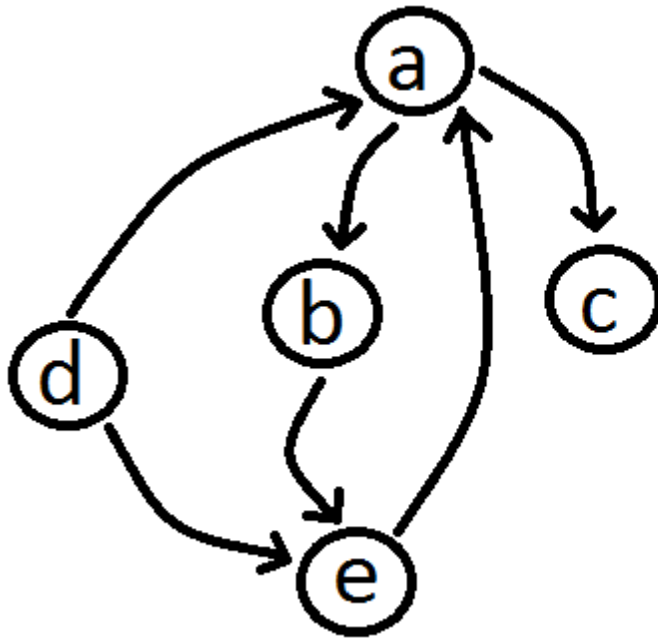
- **Guess 3**
- Well let's see. Let's start the guess at 40 each and do a few cycles:
- $PR(A) = 40$
- $PR(B) = 40$
- First calculation

- $PR(A) = 0.15 + 0.85 * 40 = 34.25$
- $PR(B) = 0.15 + 0.85 * 0.385875 = 29.1775$
- And again
- $PR(A) = 0.15 + 0.85 * 29.1775 = 24.950875$
- $PR(B) = 0.15 + 0.85 * 24.950875 = 21.35824375$
- Yup, those numbers are heading down alright! It sure looks the numbers will get to 1.0 and stop
- Here's the code used to calculate this example starting the guess at 0
- **Principle:** it doesn't matter where you start your guess, once the PageRank calculations have settled down, the "*normalized probability distribution*" (the average PageRank for all pages) will be 1.0



When you go and type some keywords in Google Search Engine a list of Web Pages will be displayed, but how does the search engine know which page to be shown first to the user? To solve this problem a algorithm called PageRank was developed at Stanford university by Larry Page and Sergey Brin in 1996. The PageRank Algorithm uses probabilistic distribution to calculate rank of a Web page and using this rank display the search results to the user. The Pagerank is recalculated every time the search engine crawls the web.

To calculate PageRank for the n Webpages ,First we initialise all Webpages with equal page rank of $1/n$ each. Then Step by Step we calculate Page Rank for each Webpage one after the other.



There are 5 Web pages represented by Nodes a , b ,c , d .The hyperlink from each webpage to the other is represented by the arrow head.

steps	A	B	C	D	E
0	0.2	0.2	0.2	0.2	0.2
1	0.3	0.1	0.1	0.1	0.2
2	0.25	0.15	0.15	0.05	0.1

At 0th Step we have all Webpages PageRank values 0.2 that is $1/5$ ($1/n$) . To get PageRank of Webpage A, consider all the incoming links to A .So we have half the Page Rank of D is pointed to A and Full Rank of E is pointed to A . So it will be $(1/5)*(1/2) + (1/5)*(1/1)$ which is $(3/10)$ or 0.3 the Page Rank of A. Similarly the Page Rank of B will be $(1/5)*(1/2)$ which is $(1/10)$ or 0.1 because A's PageRank value is $1/5$ or 0.2 from Step 0 . Even though we got 0.3 of A's PageRank in Step 1 we are considering 0.3 when we are Calculating Page Rank of B in Step 2.

The general rule is --> we consider (N-1)th step values when we are calculating the Page Rank values for Nth Step .

In Similar way we calculate all the Page Rank Values and Sort them to Get the Most important Webpage to be displayed in the Search Results .

Conclusion

Studied Implement Web Page ranking algorithm

Assignment No. 11

**Implement any one machine learning
algorithm for classification / clustering task
in BIG data Analytics**

Assignment No 11

Implement any one machine learning algorithm for classification / clustering task in BIG data Analytics

Aim	
Implement any one machine learning algorithm for classification / clustering task in BIG data Analytics	

Objective(s)	
1	Discuss What is machine learning algorithm
2	Implement K means algorithm for clustering

Theory:

Introduction:

k -means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, k -means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes. The algorithm has a loose relationship to the k -nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k -means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k -means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm. Given a set of observations (x_1, x_2, \dots, x_n) , where each observation is a d -dimensional real vector, k -means clustering aims to partition the n observations into k ($\leq n$) sets $S = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the K center). In other words, its objective is to find:

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2$$

1. Initialize the center of the clusters	$\mu_i = \text{some value}, i=1, \dots, k$
--	--

2. Attribute the closest cluster to each data point	$c_i = \{j: d(x_j, \mu_i) \leq d(x_j, \mu_l), l \neq i, j=1, \dots, n\}$
3. Set the position of each cluster to the mean of all data points belonging to that cluster	$\mu_i = \frac{1}{ c_i } \sum_{j \in c_i} x_j, \forall i$
4. Repeat steps 2-3 until convergence	
Notation	$ c $ = number of elements in c

The Lloyd's algorithm, mostly known as k-means algorithm, is used to solve the k-means clustering problem and works as follows. First, decide the number of clusters k. Then further steps in above table. The algorithm eventually converges to a point, although it is not necessarily the minimum of the sum of squares. That is because the problem is non-convex and the algorithm is just a heuristic, converging to a local minimum. The algorithm stops when the assignments do not change from one iteration to the next.

Deciding the number of clusters

The number of clusters should match the data. An incorrect choice of the number of clusters will invalidate the whole process. An empirical way to find the best number of clusters is to try K-means clustering with different number of clusters and measure the resulting sum of squares. The most curious can look at this paper for a benchmarking of 30 procedures for estimating the number of clusters.

Initializing the position of the clusters

It is really up to you! Here are some common methods:

- Forgy: set the positions of the k clusters to k observations chosen randomly from the dataset.
- Random partition: assign a cluster randomly to each observation and compute means as in step 3.

Since the algorithm stops in a local minimum, the initial position of the clusters is very important.

Advantages

- 1) Fast, robust and easier to understand.
- 2) Relatively efficient: $O(tknd)$, where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d \ll n.
- 3) Gives best result when data set are distinct or well separated from each other.

Disadvantages

- 1) The learning algorithm requires apriori specification of the number of cluster centers.
- 2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.
- 3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).
- 4) Euclidean distance measures can unequally weight underlying factors.
- 5) The learning algorithm provides the local optima of the squared error function.

- 6) Randomly choosing of the cluster center cannot lead us to the fruitful result. Pl. refer Fig.
- 7) Applicable only when mean is defined i.e. fails for categorical data.
- 8) Unable to handle noisy data and outliers.
- 9) Algorithm fails for non-linear data set.

Conclusion
Studied and Implement K means machine learning algorithm for classification / clustering task in BIG data Analytics.

Assignment No. 12

**Design and Implement social web mining
application using NoSQL databases,
machine learning algorithm, Hadoop and
Java/.Net**

Assignment No 12

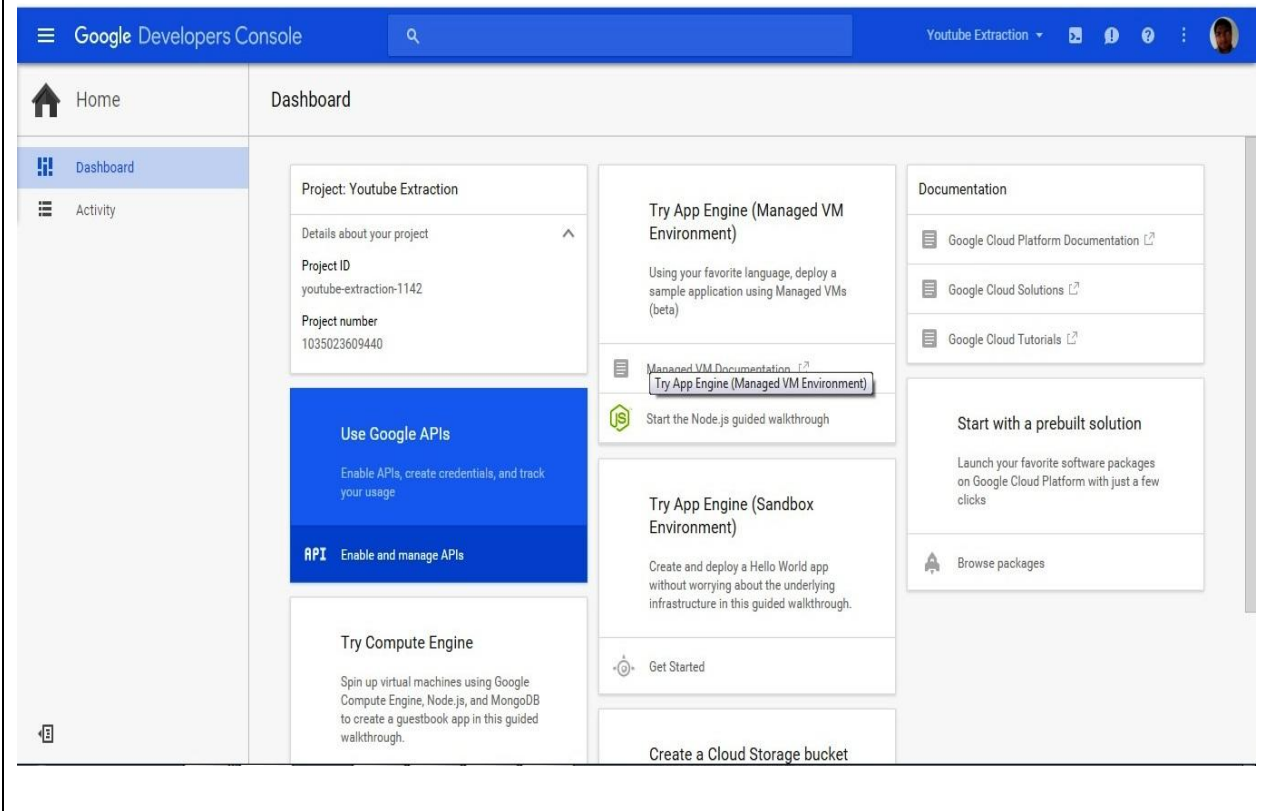
Design and Implement social web mining application using NoSQL databases, machine learning algorithm, Hadoop and Java/.Net

Aim	
Design and Implement social web mining application using NoSQL databases, machine learning algorithm, Hadoop and Java/.Net	

Objective(s)	
1	Develop mini project using frontend (Java)and backend (Cassandra) technologies

Theory
<p>Introduction:</p> <p>Social Media Mining is the process of representing, analyzing, and extracting actionable patterns from social media data. Social Media Mining introduces basic concepts and principal algorithms suitable for investigating massive social media data; it discusses theories and methodologies from different disciplines such as computer science, data mining, machine learning, social network analysis, network science, sociology, ethnography, statistics, optimization, and mathematics. It encompasses the tools to formally represent, measure, model, and mine meaningful patterns from large-scale social media data.</p> <p>Google Developer Console</p> <p>Google Developers (previously Google Code) is Google's site for software development tools,application programming interfaces(APIs), and technical resources. The site contains documentation on using Google developer tools and APIs—including discussion groups and blogs for developers using Google's developer products. There are APIs offered for almost all of Google's popular consumer products, like Google Maps, YouTube, Google Apps, and others. The site also features a variety of developer products and tools built specifically for developers. Google App Engine is a hosting service for web apps. Project Hosting gives users version control for open source code. Google Web Toolkit (GWT) allows developers to create Ajax applications in the Java programming language. The site contains reference information for community based developer products that Google is involved with like Android from the Open Handset Alliance and Open Social from the Open Social Foundation.</p> <p>YouTube API</p> <p>The YouTube Application Programming Interface, or the YouTube API, allows developers to access video statistics and YouTube channels' data via two types of calls, RESTand XML-RPC. Google describe the YouTube API Resources as 'APIs and Tools that let you bring the YouTube</p>

experience to your webpage, application or device.'This is one of the Google Developers



Conclusion

Develop mini project using frontend (Java)and backend (Cassandra) technologies