AY: 2022-23

Experiment 7

(No SQL Data Store)

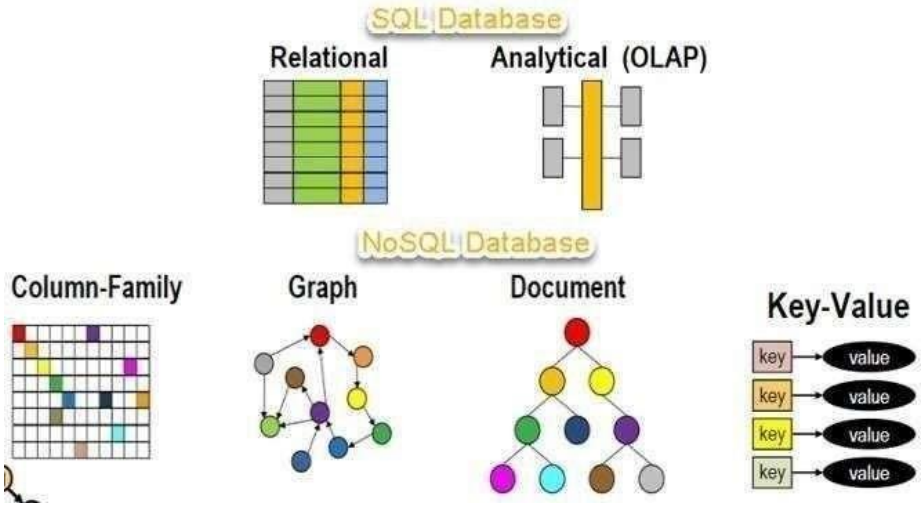
Ayush Vadalia
60009220096
D1-1

Aim: Implement No SQL Data Store using HBase.

Theory:

NoSQL:

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and realtime web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day. NoSQL database stands for "Not Only SQL" or "Not SQL."



 Difference between SQL and NoSQL data stores:

- SQL databases are relational, and NoSQL databases are non-relational.

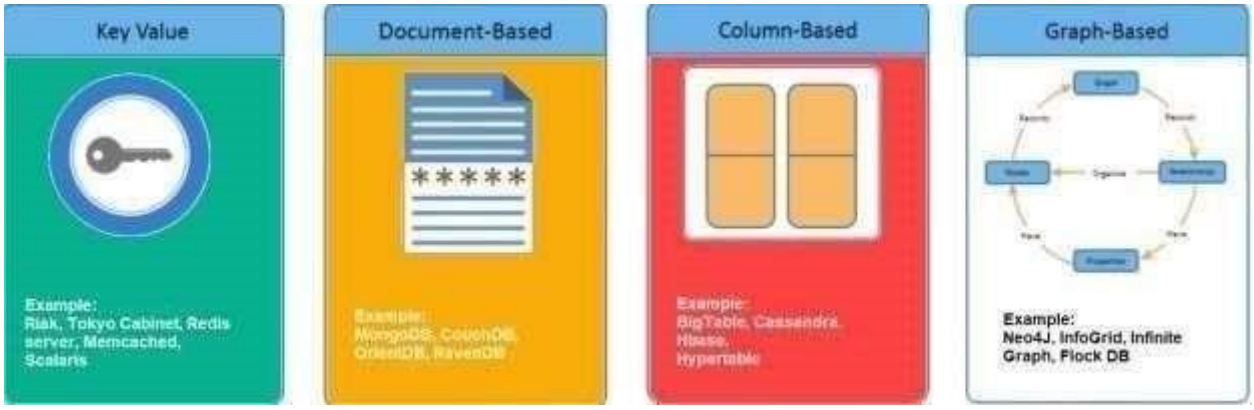
- SQL databases use structured query language (SQL) and have a predefined schema. NoSQL

- databases have dynamic schemas for unstructured data.
- SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
- SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
- SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs. Types of NoSQL Databases:

- Key-value Pair Based
- Column-oriented Graph
- Graphs based
- Document-oriented



Introduction to HBase

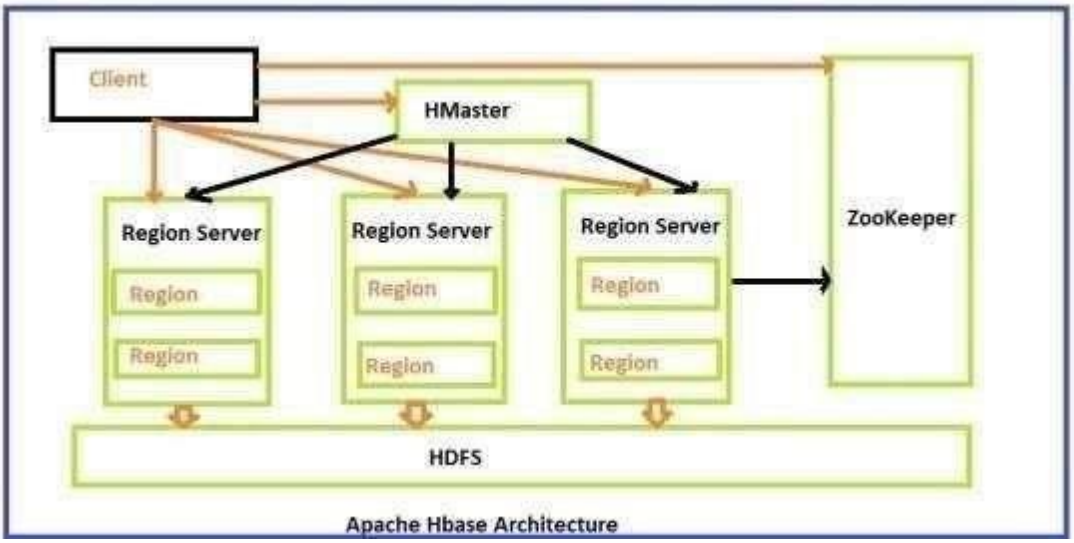
HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

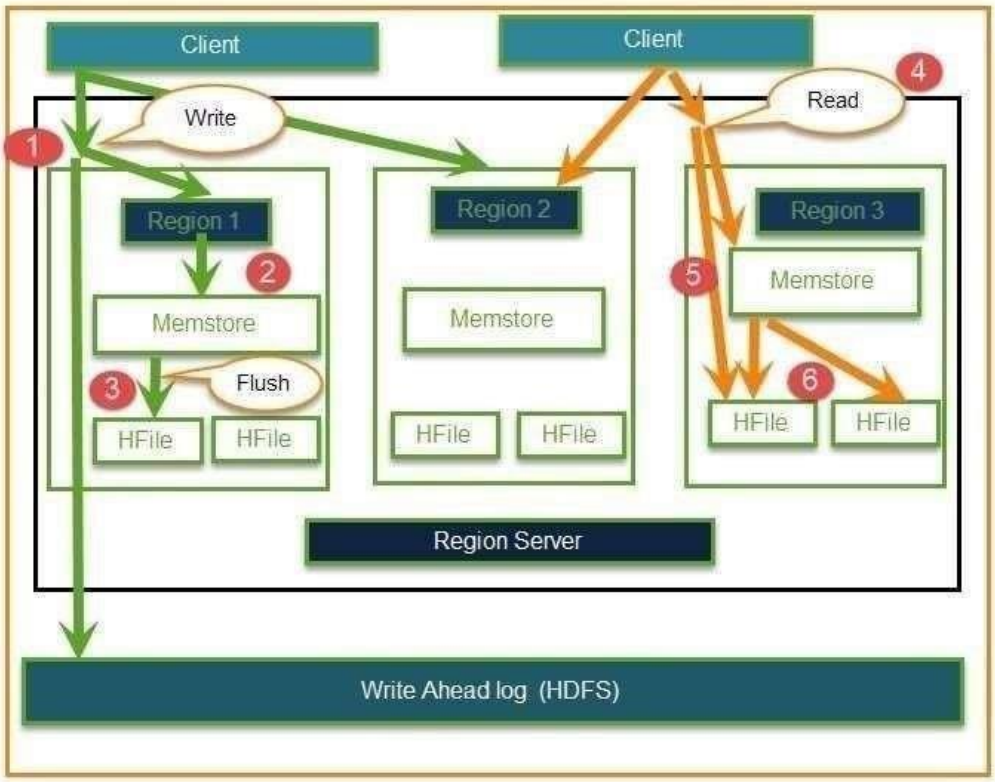
It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

HBASE Architecture

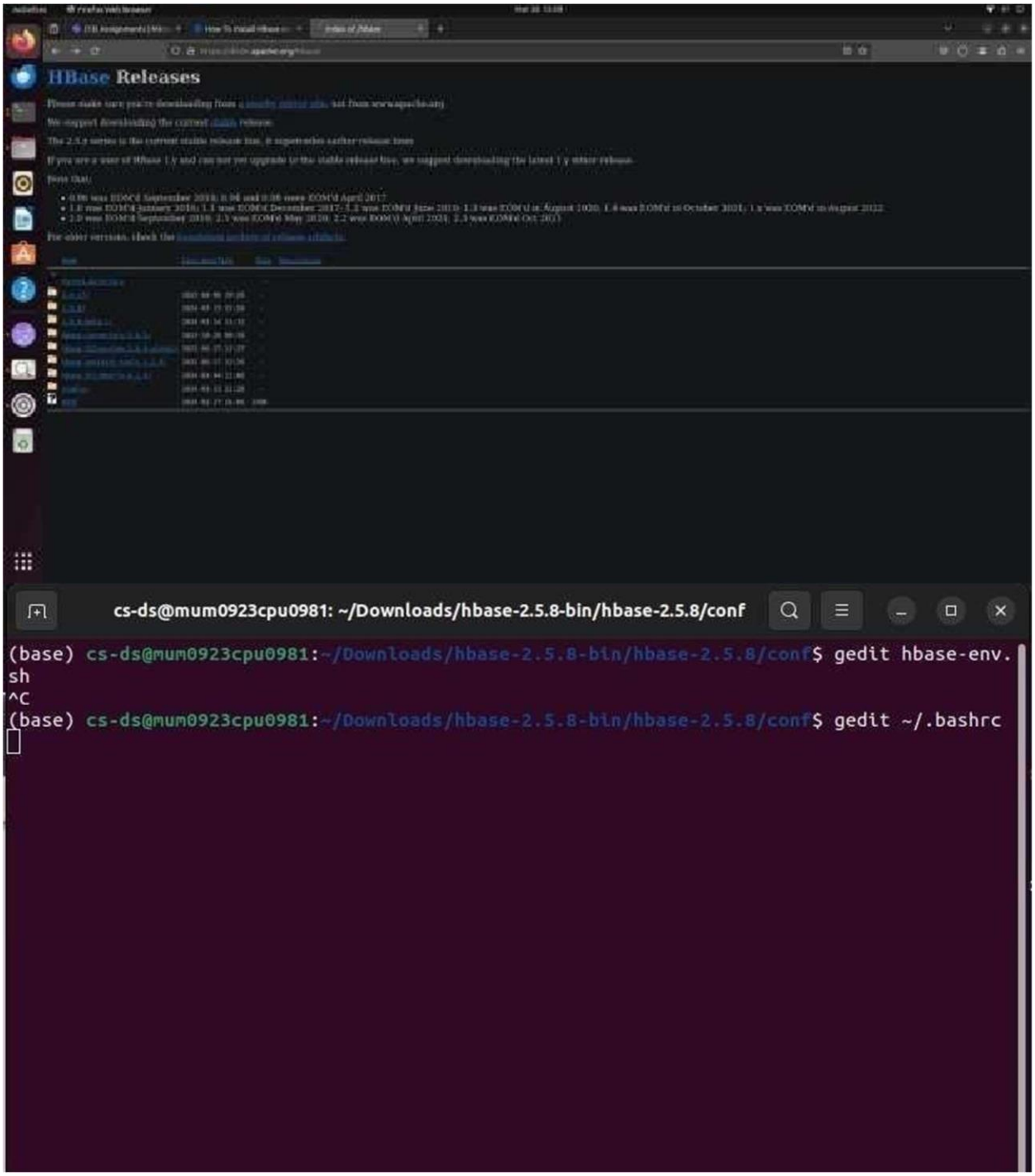


HBase Read and Write Data



Lab Assignment:

1. Installation of HBase on standalone mode.
2. Implementation of HBase Create Table with Java API & Shell.
3. Implement HBase Shell Commands and dynamic scaling:
    a. General commands
    b. Tables Managements commands
    c. Data manipulation commands
    d. Cluster Replication Commands


HBase Releases
cs-ds@mum0923cpu0981: ~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf
(base) cs-ds@mum0923cpu0981:~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf$ gedit hbase-env.sh
^C
(base) cs-ds@mum0923cpu0981:~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf$ gedit ~/.bashrc

Open | *hbase-site.xml ~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf | Save

*.bashrc | *hbase-site.xml

```
 */
-->
<configuration>
  <!--
    The following properties are set for running HBase as a single process on a
    developer workstation. With this configuration, HBase is running in
    "stand-alone" mode and without a distributed file system. In this mode, and
    without further configuration, HBase and ZooKeeper data are stored on the
    local filesystem, in a path under the value configured for `hbase.tmp.dir`.
    This value is overridden from its default value of `/tmp` because many
    systems clean `/tmp` on a regular basis. Instead, it points to a path within
    this HBase installation directory.

    Running against the `LocalFileSystem`, as opposed to a distributed
    filesystem, runs the risk of data integrity issues and data loss. Normally
    HBase will refuse to run in such an environment. Setting
    `hbase.unsafe.stream.capability.enforce` to `false` overrides this behavior,
    permitting operation. This configuration is for the developer workstation
    only and __should not be used in production!__

    See also https://hbase.apache.org/book.html#standalone_dist
  -->
  <property>
    <name>hbase.cluster.distributed</name>
    <value>false</value>
  </property>
  <property>
    <name>hbase.tmp.dir</name>
    <value>./tmp</value>
  </property>
  <property>
    <name>hbase.unsafe.stream.capability.enforce</name>
    <value>false</value>
  </property>
</configuration>


<property>

<name>hbase.rootdir</name>

<value>file:///home/hduser/HBASE/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/hduser/HBASE/zookeeper</value>

</property>
```

XML | Tab Width: 8 | Ln 71, Col 12 | INS

Open *.bashrc ~/ Save

```
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export PDSH_RCMD_TYPE=ssh
source <(kubectl completion bash)

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This load
bash_completion

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/cs-ds/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /dev
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/cs-ds/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/cs-ds/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/cs-ds/miniconda3/bin:$PATH"
    fi
fi
# Set HIVE_HOME

export HIVE_HOME="/home/Downloads/apache-hive-3.1.2-bin"
export PATH=$PATH:$HIVE_HOME/bin

export HBASE_HOME=/home/Downloads/hbase-2.5.8-bin/hbase-2.5.8
export PATH= $PATH:$HBASE_HOME/bin

unset __conda_setup
# <<< conda initialize <<<
```

```
cs-ds@mum0923cpu0981: ~/Downloads/hbase-2.5.8-bin/hbas...
(base) cs-ds@mum0923cpu0981:~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf$ gedit
hbase-env.sh
```

Open | *hbase-env.sh — ~/Downloads/hbase-2.5.8-bin/hbase-2.5.8/conf | Save

```
#!/usr/bin/env bash
#
#/**
# * Licensed to the Apache Software Foundation (ASF) under one
# * or more contributor license agreements.  See the NOTICE file
# * distributed with this work for additional information
# * regarding copyright ownership.  The ASF licenses this file
# * to you under the Apache License, Version 2.0 (the
# * "License"); you may not use this file except in compliance
# * with the License.  You may obtain a copy of the License at
# *
# *     http://www.apache.org/licenses/LICENSE-2.0
# *
# * Unless required by applicable law or agreed to in writing, software
# * distributed under the License is distributed on an "AS IS" BASIS,
# * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# * See the License for the specific language governing permissions and
# * limitations under the License.
# */

# Set environment variables here.

# This script sets variables multiple times over the course of starting an hbase process,
# so try to keep things idempotent unless you want to take an even deeper look
# into the startup scripts (bin/hbase, etc.)

# The java implementation to use.  Java 1.8+ required.
# export JAVA_HOME=/usr/java/jdk1.8.0/
export JAVA_HOME=//usr/lib/jvm/java-17-openjdk-amd64/jre

# Extra Java CLASSPATH elements.  Optional.
# export HBASE_CLASSPATH=

# The maximum amount of heap to use. Default is left to JVM default.
# export HBASE_HEAPSIZE=1G

# Uncomment below if you intend to use off heap cache. For example, to allocate 8G of
# offheap, set the value to "8G".
# export HBASE_OFFHEAPSIZE=1G

# Extra Java runtime options.
# Default settings are applied according to the detected JVM version. Override these default
# settings by specifying a value here. For more details on possible settings,
# see http://hbase.apache.org/book.html#_jvm_tuning
# export HBASE_OPTS

```