3.3 빅데이터 적재 실습하기

1) MongoDB 실습

실습1 MongoDB 설치 및 설정 실습하기

STEP_1. MongoDB repository 추가 후 설치

ㄴ https://www.mongodb.com 메인 Docs 카테고리 메뉴 하위 Server 메뉴 클릭

ㄴ Getting Started의 Installation Guides 클릭

ㄴ Install MongoDB에서 MongoDB Community Edition Installation Tutorials의 Install MongoDB Community Edition on Red Hat or CentOS 클릭

ㄴ Install MongoDB Community Edition에서 1단계 따라하기

```
#vi /etc/yum.repos.d/mongodb-org-4.2.repo
// 아래 내용 입력
[mongodb-org-4.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.2.asc
```

ㄴ Install MongoDB Community Edition에서 2단계 따라하기

```
#yum install -y mongodb-org
#vi /etc/yum.conf
// 맨 밑에 한줄로 아래 내용 입력
exclude=mongodb-org,mongodb-org-server,mongodb-org-shell,mongodb-org-mongos,
mongodb-org-tools
```

STEP_2. MongoDB 설정

ㄴ 외부 접속 허용 및 보안설정

```
#vi /etc/mongod.conf
// 29라인 수정
bindIp: 127.0.0.1 → 0.0.0.0

// 32라인 주석해제, 33라인 입력, 33라인 반드시 스페이스바로 들여쓰기 할 것
security:
 authorization: enabled
```

ㄴ SELINUX 해제

```
#vi /etc/selinux/config
// 7라인 수정
SELINUX=enforcing → disabled
```

ㄴ MongoDB 서비스 실행/자동실행/상태 확인

```
#systemctl start mongod
#systemctl enable mongod
#systemctl status mongod
```

ㄴ 외부접속을 위한 방화벽 허용(방화벽을 꺼놓을 경우 안해도 됨)

```
#firewall-cmd --permanent --zone=public --add-port=27017/tcp
#firewall-cmd --reload
```

STEP_3. MongoDB 접속/인증

ㄴ 최초 MongoDB를 설치하면 기본 계정은 존재하지 않고 생성해야 함. 또한 한 계정으로 여러 DB의 권한을 가질 수 없으며 1계정 - 1DB 원칙이 기존 RDBMS와 다름
로컬에서는 계정 정보 없이 mongo 명령으로 접속이 가능하지만 외부에서 접속인증을 통한 원격 접속을 위해서 계정을 생성 한다.

```
#mongo
```

ㄴ 관리자 DB 선택(Mongo 쉘에서 실행)

```
>use admin
```

ㄴ 슈퍼 관리자 생성 후 종료

```
>db.createUser({
   user: "admin",
   pwd: "1234",
   roles: [{role: "root", db: "admin"}]
 })

>exit
```

ㄴ admin 계정으로 다시 접속해서 일반 관리자 생성

```
#mongo -u admin -p
>use admin
>db.createUser({
       user:"chhak",  ← 일반 관리자로 사용할 아이디
       pwd:"1234",
       roles:["userAdminAnyDatabase","dbAdminAnyDatabase","readWriteAnyDatabase"]
   })
```

ㄴ test DB 생성, test 일반 사용자 생성

```
>use test    ← 테스트로 사용할 DB, 본인 계정 DB
>db.createUser({
       user:"test",  ← test DB 계정, 본인 DB 계정 생성
       pwd:"1234",
       roles:["dbAdmin","readWrite"]
   })
```

ㄴ test 사용자 삭제

```
>db.dropUser("test")
```

ㄴ MongoDB 종료

```
>exit
```

실습2 MongoDB 기본 쿼리 실습하기

STEP_1. MongoDB 접속

ㄴ test DB에 test 계정으로 접속

```
#mongo test -u test -p 1234
```

STEP_2. MongoDB 기본 쿼리연습

ㄴ 현재 DB, DB 리스트, Table 조회

```
>db
>show dbs
>show tables
```

ㄴ 데이터 입력

```
>db.user1.insert({uid:'A101', name:'김유신', hp:'010-1234-1111', age:25})
>db.user1.insert({uid:'A102', name:'김춘추', hp:'010-1234-2222', age:24})
>db.user2.insert({uid:'B101', name:'장보고', hp:'010-1234-3333', age:31})
>db.user2.insert({uid:'B102', name:'강감찬', hp:'010-1234-4444', age:36})
>db.user3.insert({uid:'C101', name:'이순신', hp:'010-1234-5555', age:47})
```

ㄴ 데이터 조회

```
>db.user1.find()
>db.user2.find()
>db.user3.find()
>db.user1.find({uid:'A101'})
>db.user2.find({age:{$gt:35}})
>db.user3.find({age:{$lt:50}},{uid:1, name:1})
```

ㄴ 데이터 수정

```
>db.user1.update({uid:'A101'},{$set:{age:29}})
>db.user3.update({age:{$gt: 40}},{$set:{hp:'010-1111-1234'}})
```

ㄴ 데이터 삭제

```
>db.user1.remove({uid:'A102'})
>db.user2.remove({age:{$gte:35}})
```

실습3 MongoDB GUI 클라이언트 실습하기

STEP_1. MongoDB Robo 설치하기

ㄴ https://robomongo.org 에서 Download Robo 3T Only 클릭

ㄴ Download Robo 3T 클릭

ㄴ Windows Robo 3T 1.x Installer 다운로드

ㄴ 설치 마법사 실행 후 설치하기

STEP_2. MongoDB 접속하기

ㄴ Robo 3T 실행 후 Create – Connection, Authentication 입력 후 Save 클릭

ㄴ MongoDB Connections에서 등록한 MongoDB 서버 선택 후 Connect 클릭

STEP_3. MongoDB 기본 쿼리 실습하기

ㄴ Collection : Member

ㄴ Field : uid, name, hp, pos, dep, rdate

ㄴ 데이터 입력, 조회, 수정, 삭제 실습하기

2) Hadoop 실습

실습1 Hadoop 설치 및 설정 실습하기

STEP_1. 가상 머신 환경설정

ㄴ 하둡 시스템 가상 머신 최소 3대 생성,

ㄴ 가상 머신 사양 하드디스크 최소 20GB, 메모리 최소 2GB 구성

STEP_2. 리눅스 설치 및 설정

ㄴ 가상 머신 3대 CentOS7 최소 설치, 'bigdata' 계정 생성

ㄴ 필수 애플리케이션 설치 및 설정

```
#yum -y update
#yum install -y vim wget net-tools java-1.8*
```

ㄴ ftp 설치 및 설정(CentOS 6 이상에서는 sftp 기본설치가 되어있음)

ㄴ ssh 설치 및 설정(CentOS 6 이상에서는 openSSH 기본설치가 되어있음)

ㄴ 네트워크 고정 ip설정(메뉴얼 참고)

ㄴ SELINUX 해제(모든 시스템 동일)

```
#vi /etc/selinux/config
7라인 : SELINUX=disabled
```

ㄴ NetworkManager 끄기(모든 시스템 동일)

```
#systemctl status NetworkManager
#systemctl stop NetworkManager
#systemctl disable NetworkManager
```

ㄴ 방화벽 해제(모든 시스템 동일)

```
#systemctl status firewalld
#systemctl stop firewalld
#systemctl disable firewalld
```

ㄴ host 설정(모든 시스템 동일)

```
#vi /etc/hostname
// 가상머신 이름과 동일하게 설정
hadoop101
hadoop102
hadoop103
```

ㄴ hosts 설정(모든 시스템 동일)

```
#vi /etc/hosts
// 기존 내용 모두 삭제 후 아래 내용 그대로 추가
192.168.56.101 hadoop101
192.168.56.102 hadoop102
192.168.56.103 hadoop103
```

ㄴ SSH 공개키 생성/배포(모든 시스템 동일)

```
#ssh-keygen -t rsa
#ssh-copy-id -i /root/.ssh/id_rsa.pub root@bigdata101
#ssh-copy-id -i /root/.ssh/id_rsa.pub root@bigdata102
#ssh-copy-id -i /root/.ssh/id_rsa.pub root@bigdata103
```

ㄴ 재부팅(모든 시스템 동일)

```
#reboot
```

STEP_3. Hadoop 설치/설정

ㄴ 하둡 2.x 버전 다운로드/압축해제/이동/링크생성(bigdata101, bigdata102, bigdata103 동일)

```
#wget http://mirror.apache-kr.org/hadoop/common/hadoop-2.x.x/hadoop-2.x.x.tar.gz
#tar xvfz hadoop-2.x.x.tar.gz
#mv hadoop-2.x.x /home/bigdata/
#cd /home/bigdata/
#ln -s hadoop-2.x.x/ hadoop
```

ㄴ 하둡 환경설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi ~/.bashrc
// 맨 아래에 하둡 환경변수 선언
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/home/bigdata/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export CLASS_PATH=$JAVA_HOME/lib:$CLASS_PATH
```

ㄴ 현재 쉘에 반영(bigdata101, bigdata102, bigdata103 동일)

```
#source ~/.bashrc
```

ㄴ 하둡 임시 디렉터리 생성(bigdata101, bigdata102, bigdata103 동일)

```
#mkdir $HADOOP_HOME/tmp
```

ㄴ core-site.xml 설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi $HADOOP_HOME/etc/hadoop/core-site.xml
<!-- 아래 내용 입력 -->
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://bigdata101:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/bigdata/hadoop-2.x.x/tmp</value>
        </property>
</configuration>
```

ㄴ hdfs-site.xml 설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi $HADOOP_HOME/etc/hadoop/hdfs-site.xml
<!-- 아래 내용 입력 -->
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
        <property>
                <name>dfs.permissions</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.webhdfs.enabled</name>
                <value>true</value>
        </property>
</configuration>
```

ㄴ mapred-site.xml template파일 복사 후 설정(bigdata101, bigdata102, bigdata103 동일)

```
#cp $HADOOP_HOME/etc/hadoop/mapred-site.xml.template $HADOOP_HOME
                                                      /etc
                                                      /hadoop
                                                      /mapred-site.xml
#vi $HADOOP_HOME/etc/hadoop/mapred-site.xml
<!-- 아래 내용 입력 -->
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

ㄴ yarn-site.xml 설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi $HADOOP_HOME/etc/hadoop/yarn-site.xml
<!-- 아래 내용 입력 -->
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>bigdata101</value>
    </property>
</configuration>
```

ㄴ masters 설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi $HADOOP_HOME/etc/hadoop/masters
//namenode로 지정할 서버 설정
bigdata101
```

ㄴ slaves 설정(bigdata101, bigdata102, bigdata103 동일)

```
#vi $HADOOP_HOME/etc/hadoop/slaves

//datanode로 지정할 서버 설정
bigdata101 (namenode, datanode 두 개의 역할)
bigdata102
bigdata103
```

ㄴ Hadoop Format(Namenode 서버인 bigdata101에서만 실행)

```
#hadoop namenode -format
```

ㄴ Hadoop 실행(namenode 서버인 bigdata101에서만 실행)

```
#start-all.sh

만약 위 명령문이 실행 안되면 아래 명령문으로 실행
#start-dfs.sh
#start-yarn.sh
```

ㄴ History 서버 실행(namenode 서버인 bigdata101에서만 실행)

```
#mr-jobhistory-daemon.sh start historyserver
```

ㄴ Hadoop 상태 확인(bigdata101에서 실행)

```
#jps
//아래 PID와 프로세스 확인, 순서 상관없음
2xxx NodeManager
1xxx NameNode
3xxx ResourceManager
2xxx DataNode
2xxx SecondaryNameNodee
1xxx JobHistoryServer
1xxx Jps
```

만약 DataNode 프로세스가 뜨지 않으면 Hadoop 임시 디렉터리 삭제/포맷/실행할 것

```
#rm -rf $HADOOP_HOME/tmp
#hadoop namenode -format
#start-all.sh
```

ㄴ Hadoop 상태 확인(bigdata102, bigdata103에서 실행)

```
#jps
//아래 PID와 프로세스 확인, 순서 상관없음
2xxx NodeManager
1xxx DataNode
1xxx Jps
```

ㄴ Hadoop 브라우저 확인

http://192.168.100.101:50070 → Hadoop Web UI

http://192.168.100.101:8088 → Hadoop Resource Manager

ㄴ Hadoop 종료(Namenode 서버인 bigdata101에서만 실행)

```
#stop-all.sh

만약 위 명령문이 실행 안되면 아래 명령문으로 실행
#stop-yarn.sh
#stop-dfs.sh
```

ㄴ History 서버 종료(namenode 서버인 bigdata101에서만 실행)

```
#mr-jobhistory-daemon.sh stop historyserver
```

실습2 Hadoop 기본 명령어 실습하기

STEP_1. HDFS 명령어 실습

ㄴ Filezilla FTP로 Namenode(bigdata101) 접속

ㄴ Namenode의 /home 디렉터리에 바탕화면/workspace/html 업로드

ㄴ Namenode 로컬 파일을 HDFS로 복사

```
#hdfs dfs -put /home/html /html
```

ㄴ http://192.168.xxx.101:50070 브라우저 - Utilities - Browse the file system

ㄴ /html 디렉터리 확인, 하위 디렉터리 확인

ㄴ HDFS 디렉터리|파일 조회

```
#hdfs dfs -ls /
#hdfs dfs -ls /html
#hdfs dfs -ls /html/Ch1
#hdfs dfs -ls /html/Ch2
#hdfs dfs -ls -R /html
```

ㄴ HDFS 파일 내용 미리보기

```
#hdfs dfs -cat /html/Ch1/hello.html
#hdfs dfs -cat /htm2/Ch2/2-1.html
```

ㄴ HDFS 디렉터리 생성

```
#hdfs dfs -mkdir /backup
#hdfs dfs -mkdir /backup/html
#hdfs dfs -mkdir /backup/html/Ch2
```

ㄴ HDFS 파일|디렉터리 복사

```
#hdfs dfs -cp /html/Ch1 /backup/html
#hdfs dfs -cp /html/Ch2/2-1.html /backup/html/Ch2
#hdfs dfs -cp /html/Ch2/2-2.html /backup/html/Ch2
```

ㄴ HDFS 파일|디렉터리 이동

```
#hdfs dfs -mv /html/Ch2/2-3.html /backup/html/Ch2
#hdfs dfs -mv /html/Ch2/2-4.html /backup/html/Ch2
#hdfs dfs -mv /html/Ch3 /backup/html
#hdfs dfs -mv /html/Ch4 /backup/html
```

ㄴ HDFS 파일|디렉터리 삭제

```
#hdfs dfs -rm /backup/html/Ch4/4-1.html
#hdfs dfs -rm /backup/html/Ch4/4-2.html
#hdfs dfs -rm -r /backup/html/Ch4
```

ㄴ HDFS 파일|디렉터리 로컬로 복사

```
#hdfs dfs -get /backup/html/Ch1/hello.html /root
#hdfs dfs -get /backup /root
```

3) Hbase 실습

실습1 HBase 설치/설정 실습하기

STEP_1. HBase 다운로드/설치/설정

ㄴ HBase 2.x 버전 다운로드/압축해제/이동(bigdata101에서 실행)

```
#wget http://mirror.apache-kr.org/hbase/2.x.x/hbase-2.x.x-bin.tar.gz
#tar xvfz hbase-2.x.x-bin.tar.gz
#mv hbase-2.x.x-bin /home/bigdata/
#cd /home/bigdata
#ln -s hbase-2.x.x-bin/ hbase
```

ㄴ HBase 환경설정(bigdata101에서 실행)

```
#vi ~/.bashrc

// 맨 아래에 HBase 환경변수 추가선언
export HBASE_HOME=/home/bigdata/hbase
export PATH=$PATH:$HBASE_HOME/bin
```

ㄴ 현재 쉘에 반영(bigdata101에서 실행)

```
#source ~/.bashrc
```

ㄴ hbase-env.sh 설정(bigdata101에서 실행)

```
#vi /home/bigdata/hbase/conf/hbase-env.sh

// 28라인 주석해제, JAVA 경로 수정
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk

// 89라인 주석해제, HBase 경로 수정
export HBASE_REGIONSERVERS=/home/bigdata/hbase/conf/regionservers

// 126라인 주석해제, Zookeeper 사용여부 true 설정
export HBASE_MANAGES_ZK=true
```

ㄴ hbase-site.xml 설정(bigdata101에서 실행)

```
#vi /home/bigdata/hbase-2.2.5/conf/hbase-site.xml
<!-- 아래 내용 입력 -->
<configuration>
        <property>
                <name>hbase.rootdir</name>
                <value>hdfs://192.168.xxx.101:9000/hbase</value>
        </property>
        <property>
                <name>hbase.master</name>
                <value>192.168.xxx.101:60010</value>
        </property>
        <property>
                <name>hbase.cluster.distributed</name>
                <value>true</value>
        </property>
        <property>
                <name>hbase.zookeeper.quorum</name>
                <value>localhost</value>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
        <property>
                <name>hbase.zookeeper.property.clientPort</name>
                <value>2181</value>
        </property>
        <property>
                <name>hbase.zookeeper.property.dataDir</name>
                <value>/home/bigdata/hbase/zookeeper_data</value>
        </property>
        <property>
                <name>hbase.unsafe.stream.capability.enforce</name>
                <value>false</value>
        </property>
</configuration>
```

STEP_2. HBase 실행

ㄴ Hadoop 실행(bigdata101에서 실행)

```
#start-all.sh
#mr-jobhistory-daemon.sh start historyserver

만약 이미 Hadoop이 실행되고 있으면 stop 후 다시 start
#stop-all
#mr-jobhistory-daemon.sh stop historyserver
```

ㄴ HBase 실행(bigdata101에서 실행)

```
#start-hbase.sh
```

ㄴ HBase 상태 확인(bigdata101에서 실행)

```
#jps
//아래 HMaster, HRegionServer PID와 프로세스 확인
2xxx NodeManager
2xxx HQuorumPeer
1xxx NameNode
3xxx ResourceManager
2xxx DataNode
5xxx HMaster
2xxx SecondaryNameNodee
1xxx JobHistoryServer
9xxx HRegionServer
1xxx Jps
```

ㄴ HBase 브라우저 확인

http://192.168.xxx.101:16010 → HBase Web UI

ㄴ HBase 종료(bigdata101에서 실행)

```
#stop-hbase.sh
```

실습2 HBase 기본 쿼리 실습하기

STEP_1. HBase 접속

ㄴ Hbase Shell 실행

```
#hbase shell
```

STEP_2. HBase 기본 쿼리연습

ㄴ Table 조회/생성

```
>list

// Table명이 'User1', column-family가 cf1인 Table 생성
>create 'User1', 'cf1'

// Table명이 'User2', column-family가 cf2인 Table 생성
>create 'User2', 'cf2'

// Table명이 'Member', column-family가 user, company인 Table 생성
>create 'Member', 'user', 'company'
```

ㄴ 데이터 입력

```
>put 'User1', 'row1', 'cf1:uid', 'A101'
>put 'User1', 'row1', 'cf1:name', 'Kim Yu Sin'
>put 'User1', 'row1', 'cf1:hp', '010-1234-1111'
>put 'User1', 'row1', 'cf1:age', 23

>put 'User1', 'row2', 'cf1:uid', 'A102'
>put 'User1', 'row2', 'cf1:name', 'Kim Chun Chu'
>put 'User1', 'row2', 'cf1:hp', '010-1234-2222'
>put 'User1', 'row2', 'cf1:age', 21

>put 'User2', 'a101', 'cf2:name', 'Jang Bo Go'
>put 'User2', 'a101', 'cf2:hp', '010-1234-3333'
>put 'User2', 'a101', 'cf2:age', 27

>put 'User2', 'a102', 'cf1:name', 'Lee Sun Sin'
>put 'User2', 'a102', 'cf1:hp', '010-1234-4444'
>put 'User2', 'a102', 'cf1:age', 21

>put 'Member', 'a101', 'user:name', 'Kim'
>put 'Member', 'a101', 'user:hp', '010-1234-1111'
>put 'Member', 'a101', 'company:pos', 'Manager'
>put 'Member', 'a101', 'company:dep', 101
>put 'Member', 'a101', 'company:rdate', '20-02-12'

>put 'Member', 'a102', 'user:name', 'Lee'
>put 'Member', 'a102', 'user:hp', '010-1234-2222'
>put 'Member', 'a102', 'company:pos', 'Director'
>put 'Member', 'a102', 'company:dep', 102
>put 'Member', 'a102', 'company:rdate', '20-03-10'
```

ㄴ 데이터 조회

```
>scan 'User1'
>scan 'User1', {COLUMNS => ['cf1:uid']}
>scan 'User1', {COLUMNS => ['cf1:uid', 'cf1:name']}

>get 'User2', 'a101'
>get 'User2', 'a101', 'cf2:name'
>get 'User2', 'a102', {COLUMN => ['cf2:name', 'cf2:hp']}
```