

# 빅데이터 처리시스템

## 수행모듈 개발하기

MapReduce 프로그램 개발 및 Hive 설치 및 수행

작성자: 이슬이

## 내용

빅데이터 처리시스템 수행모듈 개발하기.....	3
MapReduce(맵리듀스) .....	3
MapReduce 수행순서 .....	3
Hive(하이프).....	9
Hive 설치 .....	9
Naver 검색어 파일 키워드 집계 쿼리 수행 .....	13

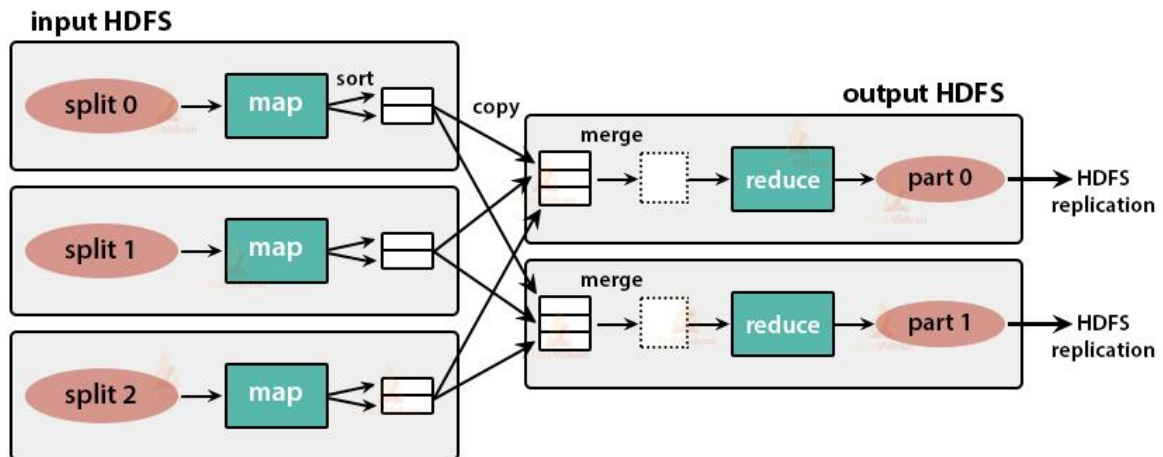
## 빅데이터 처리시스템 수행모듈 개발하기

### MapReduce(맵리듀스)

맵리듀스는 HDFS 에 분산 저장된 데이터에 접근하여 빠르게 분산 병렬 처리(필터링, 그룹핑, 정렬, 통합, 분리) 하도록 고안된 프로그래밍 프레임워크이다. 맵리듀스는 하나의 큰 데이터를 여러 개의 조각으로 나누어 처리하는 Map 단계와 처리된 결과를 취합하는 Reduce 단계로 구성된다. 맵리듀스의 분산 처리는 해당 데이터를 저장하고 있는 시스템에서 실행함으로써 데이터의 이동을 최소화해 데이터의 지역성(locality) 이점을 가진다.

다음은 맵리듀스의 아키텍처이다.

## Apache Hadoop MapReduce



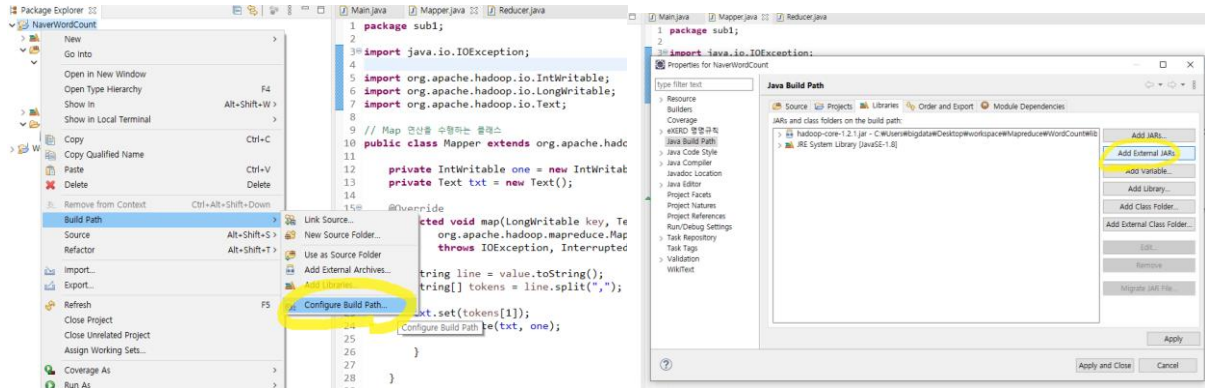
[그림 1] 맵리듀스의 아키텍처

출처: <https://techvidvan.com/tutorials/hadoop-architecture/>

### MapReduce 수행순서

1. 먼저 하둡 코어 파일을 다음 경로에서 jar 파일버전을 다운로드하여 NaverWordCount 프로젝트 안에 폴더 생성 후 (폴더명: lib) 폴더 안에 복사한다

(<https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1>). Configure Build Path을 통해 폴더 안에 저장된 jar 파일을 가져온다.



[그림 2] Configure build path 을 이용해 하둡을 라이브러리에 저장

2. MapReduce 자바 프로그래밍을 한다. 매퍼듀스 작업을 위해 세 가지 클래스, Main, Mapper, Reducer를 만든다. 이때 시스템 호환으로 인해 java compiler 버전을 1.8로 변경한다.

### ① Mapper

```
package sub1;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
// Map 연산을 수행하는 클래스
public class Mapper extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, IntWritable> {
    @Override
    protected void map(LongWritable key, Text value,
                       org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text,
IntWritable>.Context context)
        throws IOException, InterruptedException {
        StringTokenizer st = new StringTokenizer(value.toString());
        while (st.hasMoreElements()) {
            String word = st.nextToken();
            Text txt = new Text(word);
            IntWritable val = new IntWritable(1);
            // split 단어를 key, value 쌍으로 출력
            context.write(txt, val);
        }
    }
}
```

### ② Reducer

```
package sub1;
```

```

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
public class Reducer extends org.apache.hadoop.mapreduce.Reducer<Text, IntWritable, Text, IntWritable>{
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
                           org.apache.hadoop.mapreduce.Reducer<Text, IntWritable, Text,
IntWritable>.Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        IntWritable result = new IntWritable(sum);
        context.write(key, result);
    }
}

```

### ③ Main(Driver)

```

package sub1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Main {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "WordCount");
        job.setJarByClass(Main.class);
        job.setMapperClass(Mapper.class);
        job.setReducerClass(Reducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
    }
}

```

```

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
        System.out.println("WordCount 종료...");
    }
}

```

3. 프로젝트를 Export 하여 Java - Jar file로 Export한다. 바탕화면에 저장된 WordCount.jar 파일을 파일질라를 이용해 Bigdata101 루트폴더로 이동한다

```

[root@Bigdata101 ~]# ll
합 계 11480
-rw-r--r-- 1 root root 3904356 1월 12 15:11 NaverWordCount.jar
-rw-r--r-- 1 root root 14817 1월 11 14:31 User1.java
-rw-r--r-- 1 root root 3903478 1월 12 16:47 WeatherMapReduce.jar
-rw-r--r-- 1 root root 3902897 1월 11 17:12 WordCount.jar
-rw-r--r-- 1 root root 1387 1월 8 14:57 anaconda-ks.cfg
-rw-r--r-- 1 root root 38 1월 13 16:37 animal.txt
drwxr-xr-x 4 root root 39 1월 13 14:54 hive
drwxr-xr-x 17 root root 246 1월 12 14:38 naver
-rw-r--r-- 1 root root 28 1월 11 17:05 part-r-00000
-rw-r--r-- 1 root root 653 1월 14 14:43 sales2017.csv
-rw-r--r-- 1 root root 44 1월 11 16:55 sample.txt
-rw-r--r-- 1 root root 705 1월 14 14:43 user.csv
drwxr-xr-x 16 root root 258 1월 12 14:38 weather

```

[그림 3] NaverWordCount.jar 파일 화면

4. Hadoop을 실행하고 MapReduce 실행 명령어를 입력한다. 예제로 /naver 폴더에 저장된 2021년 1월 2일의 저장된 모든 데이터를 이용한다. 실행이 완료되면 'NaverWordCount 종료...' 메시지가 입력된다. (\*가상 머신 사양 하드디스크 최소 20GB, 메모리 최소 2GB 구성)

```

#start-all.sh

#yarn jar NaverWordCount.jar sub1.Main /naver/21-01-02/* /naver/result

```

The screenshot shows the Hadoop web interface with the 'All Applications' tab selected. A table lists various applications, with the first row highlighted in red:

ID	User	Name	Application Type	Queue	Priority	Start Time	Launch Time	Finish Time	State	Final Status	Running Containers	Allocated CPU Vcores	Allocated Memory	Allocated GPUs
application_1610677352369_0004	root	NaverWordCount	MAPREDUCE	default	0	Fri Jan 15 12:11:09 +0900 2021	Fri Jan 15 12:11:09 +0900 2021	N/A	RUNNING	UNDEFINED	1	1	2048	-1
application_1610677352369_0003	root	NaverWordCount	MAPREDUCE	default	0	Fri Jan 15 12:05:48 +0900 2021	Fri Jan 15 12:05:48 +0900 2021	12:06:00 +0900 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
application_1610677352369_0002	root	SELECT keyword, SUM(1) as total FROM ...DESC(Stage-2)	MAPREDUCE	default	0	Fri Jan 15 11:28:34 +0900 2021	Fri Jan 15 11:28:39 +0900 2021	11:29:00 +0900 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A
application_1610677352369_0001	root	SELECT keyword, SUM(1) as total FROM ...DESC(Stage-1)	MAPREDUCE	default	0	Fri Jan 15 11:27:57 +0900 2021	Fri Jan 15 11:27:58 +0900 2021	11:28:32 +0900 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A

[그림 4] 맵리듀스 작업 확인 화면

```

21/01/15 12:22:13 INFO mapreduce.Job: map 46% reduce 15%
21/01/15 12:22:14 INFO mapreduce.Job: map 46% reduce 15%
21/01/15 12:22:24 INFO mapreduce.Job: map 47% reduce 15%
21/01/15 12:22:25 INFO mapreduce.Job: map 47% reduce 16%
21/01/15 12:22:41 INFO mapreduce.Job: map 48% reduce 16%
21/01/15 12:22:51 INFO mapreduce.Job: map 49% reduce 16%
21/01/15 12:23:06 INFO mapreduce.Job: map 50% reduce 16%
21/01/15 12:23:08 INFO mapreduce.Job: map 50% reduce 17%
21/01/15 12:23:21 INFO mapreduce.Job: map 51% reduce 17%
21/01/15 12:23:32 INFO mapreduce.Job: map 52% reduce 17%
21/01/15 12:23:47 INFO mapreduce.Job: map 53% reduce 17%
21/01/15 12:23:50 INFO mapreduce.Job: map 53% reduce 18%
21/01/15 12:23:58 INFO mapreduce.Job: map 54% reduce 18%
21/01/15 12:24:14 INFO mapreduce.Job: map 55% reduce 18%
21/01/15 12:24:27 INFO mapreduce.Job: map 56% reduce 18%
21/01/15 12:24:32 INFO mapreduce.Job: map 56% reduce 19%
21/01/15 12:24:40 INFO mapreduce.Job: map 57% reduce 19%
21/01/15 12:24:55 INFO mapreduce.Job: map 58% reduce 19%
21/01/15 12:25:06 INFO mapreduce.Job: map 59% reduce 19%
21/01/15 12:25:08 INFO mapreduce.Job: map 59% reduce 20%
21/01/15 12:25:22 INFO mapreduce.Job: map 60% reduce 20%
21/01/15 12:25:33 INFO mapreduce.Job: map 61% reduce 20%
21/01/15 12:25:49 INFO mapreduce.Job: map 62% reduce 20%
21/01/15 12:25:50 INFO mapreduce.Job: map 62% reduce 21%
21/01/15 12:26:00 INFO mapreduce.Job: map 63% reduce 21%
21/01/15 12:26:15 INFO mapreduce.Job: map 64% reduce 21%

Total vcore-milliseconds taken by all reduce tasks=1282138
Total megabyte-milliseconds taken by all map tasks=26985713664
Total megabyte-milliseconds taken by all reduce tasks=1312909312

Map-Reduce Framework
  Map input records=15840
  Map output records=15840
  Map output bytes=284293
  Map output materialized bytes=324613
  Input split bytes=174240
  Combine input records=0
  Combine output records=0
  Reduce input groups=225
  Reduce shuffle bytes=324613
  Reduce input records=15840
  Reduce output records=225
  Spilled Records=31680
  Shuffled Maps =1440
  Failed Shuffles=0
  Merged Map outputs=1440
  GC time elapsed (ms)=268909
  CPU time spent (ms)=464010
  Physical memory (bytes) snapshot=330388021248
  Virtual memory (bytes) snapshot=3009518141440
  Total committed heap usage (bytes)=197936218112

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0

File Input Format Counters
  Bytes Read=458533
File Output Format Counters
  Bytes Written=3833
NaverWordCount 종료 ...

```

[그림 5] 맵리듀스 작업 확인 화면

## Browse Directory

/naver/result Go!

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	0 B	Jan 15 12:34	3	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	3.74 KB	Jan 15 12:34	3	128 MB	part-r-00000	

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2020.

[그림 6] HDFS 웹 페이지 디렉터리 확인 화면

5. 파일을 실행하기 위해 명령어를 입력한다. 실행결과는 다음과 같다

#hdfs dfs -cat /naver/result/part-r-00000

```
[root@Bigdata101 ~]# hdfs dfs -cat /naver/result/part-r-00000
제 목      1440
손 예 진   566
코 로 나   2.5단 계 연 장      558
현 빈      537
현 빈 손 예 진      513
곽 진 영   510
정 인 아 미 안 해      466
손 예 진 나 이      455
빈 지 노    317
사 랑 의 콜 센 터      277
사 랑 의 불 시 착      195
환 운 하    195
랜 인 블 랙 인 터 내 셔 널      187
랜 유      172
그 것 만 이 내 세 상      165
현 빈 나 이      160
블 랙 팬 서      155
랜 유 아 스 톤 빌 라      149
타 짜 3      149
하 림 각     148
현       137
이 서 진     132
비 나 이     124
김 하 영     122
이 영 하     122
영 화 런 결 말      119
신 과 함 께      117
불 어 라 미 풍 아      116
리 즈 유 나 이 티 드      116
인 터 스 텔 라      114
```

[그림 7] MapReduce 명령어 실행 확인 화면



## Hive(하이프)

페이스북에서 개발한 Hadoop 에 적재된 데이터를 분산 병렬 처리하기 위해 어려운 MapReduce 프로그래밍을 대신 HiveQL 이라 불리는 SQL 유사 언어를 이용해서 손쉽게 MapReduce 를 실행할 수 있는 빅데이터 탐색도구이다.

다음은 하이브의 아키텍처이다.



[그림 8] 스쿱 다운로드 화면








출처: NCS 학습모듈 빅데이터 처리시스템 개발

## Hive 설치

1. 아파치 하이브(hive)을 설치한다.

- ① 아파치 하이브 웹사이트(<https://hive.apache.org/>)에서 다운로드를 클릭하고 HTTP 아래에 적힌 주소를 클릭한다. 오른쪽 마우스를 클릭해 링크 주소 복사를 한 후 #wget 명령어를 이용해 설치한다.

## Index of /hive/hive-2.3.7

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">apache-hive-2.3.7-bin.tar.gz</a>	2020-07-03 04:34	222M	
 <a href="#">apache-hive-2.3.7-bin.tar.gz.asc</a>	2020-07-03 04:34	833	
 <a href="#">apache-hive-2.3.7-bin.tar.gz.sha256</a>	2020-07-03 04:34	95	
 <a href="#">apache-hive-2.3.7-src.tar.gz</a>	2020-07-03 04:34	21M	
 <a href="#">apache-hive-2.3.7-src.tar.gz.asc</a>	2020-07-03 04:34	833	
 <a href="#">apache-hive-2.3.7-src.tar.gz.sha256</a>	2020-07-03 04:34	95	

[그림 9] 하이브 다운로드 화면

```
#wget https://downloads.apache.org/hive/hive-2.3.7/apache-hive-2.3.7-bin.tar.gz
#tar xvf apache-hive-2.3.7-bin.tar.gz
#mv apache-hive-2.3.7-bin /home/bigdata/
#cd /home/bigdata/
#ln -s apache-hive-2.3.7-bin/ hive
```

### ② Hive 환경변수 설정(Namenode 실행)

하이브의 환경 변수인 \$HIVE\_HOME을 설정하고 PATH를 설정해 준다(셸 설정 파일인 .bashrc에 다음 내용을 넣어 주고 #source .bashrc을 실행해 반영해 준다.

```
#vi ~/.bashrc
// 맨 아래에 선언 추가
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
export HIVE_HOME=/home/bigdata/hive
export PATH=$PATH:$HIVE_HOME/bin
```

변경사항 현재 셸에 반영(Namenode 실행)

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

### ③ 필요한 jdbc 커넥터(드라이버)도 미리 설치되고 세팅되어 있어야 한다.

MySQL(<http://dev.mysql.com/downloads/connector/j/>)에서 리눅스용 mysql-connector-java-5.1.49-tar.gz 파일을 다운로드 한다. 압축해제 후 FileZilla FTP 로 mysql-connector-java-5.1.49-bin.jar 파일을 /root 경로에 업로드한 후 #mv 명령어로 hive/home/lib 으로 이동한다.

```
#mv mysql-co...-java-5.1.49/mysql-connector-java-5.1.49-bin.jar $HIVE_HOME/lib
```

```

-rw-r--r-- 1 root root 41760 2월 14 2017 maven-settings-3.1.1.jar
-rw-r--r-- 1 root root 41559 2월 14 2017 maven-settings-builder-3.1.1.jar
-rw-r--r-- 1 root root 18148 2월 14 2017 maxminddb-0.2.0.jar
-rw-r--r-- 1 root root 82123 12월 17 2016 metrics-core-2.2.0.jar
-rw-r--r-- 1 root root 111908 12월 17 2016 metrics-core-3.1.0.jar
-rw-r--r-- 1 root root 15823 12월 17 2016 metrics-json-3.1.0.jar
-rw-r--r-- 1 root root 35907 12월 17 2016 metrics-jvm-3.1.0.jar
-rw-r--r-- 1 root root 1006906 1월 12 17:06 mysql-connector-java-5.1.49-bin.jar
-rw-r--r-- 1 root root 7954 2월 14 2017 mysql-metadata-storage-0.9.2.jar
-rw-r--r-- 1 root root 1199572 12월 17 2016 netty-3.6.2.Final.jar
-rw-r--r-- 1 root root 2275047 9월 9 2018 netty-all-4.0.52.Final.jar
-rw-r--r-- 1 root root 191444 2월 14 2017 okhttp-1.0.2.jar
-rw-r--r-- 1 root root 19827 12월 17 2016 opencsv-2.3.jar
-rw-r--r-- 1 root root 733528 3월 10 2020 orc-core-1.3.4.jar
-rw-r--r-- 1 root root 25515 12월 17 2016 org.abego.treelayout.core-1.0.1.jar
-rw-r--r-- 1 root root 29555 12월 17 2016 paranamer-2.3.jar
-rw-r--r-- 1 root root 2902379 12월 17 2016 parquet-hadoop-bundle-1.8.1.jar
-rw-r--r-- 1 root root 48557 12월 17 2016 pentaho-aggregations-algorithm-5.1.

```

[그림 10] JDBC 드라이버 확인

- ④ Mysql(mariadb)에 Hive Metastore 데이터베이스 생성 및 hive 계정을 설정한다.  
mysql(mariadb)가 설치되어 있어야 한다.

```
#mysql -u root -p
```

```

mysql>CREATE DATABASE hive_metastore_db;
mysql>CREATE USER 'hive'@'localhost' IDENTIFIED BY '1234';
mysql>CREATE USER 'hive'@'%' IDENTIFIED BY '1234';
mysql>GRANT ALL ON *.* TO 'hive'@'localhost' IDENTIFIED BY '1234';
mysql>GRANT ALL ON *.* TO 'hive'@'%' IDENTIFIED BY '1234';
mysql>FLUSH PRIVILEGES;
mysql>exit

```

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' pane lists 'Bigdata101' and its sub-schemas, including 'hive\_metastore\_db' which is selected and shows a size of 1.8 MiB. The main area displays the 'Table' list for 'hive\_metastore\_db'. The table list includes columns for Name, Rows, Size, Created, Updated, Engine, Comment, and Type. The tables listed are AUX\_TABLE, BUCKETING\_COLUMNS, CDS, COLUMNS\_V2, COMPACTION\_STATISTICS\_TABLE, COMPLETED\_STATISTICS\_TABLE, COMPLETED\_STATISTICS\_TABLE, DATABASE\_PRIVILEGE\_TABLE, DBS, DB\_PRIVS, DELEGATION\_TABLE, FUNCS, FUNC\_RU, GLOBAL\_PRIVS, HIVE\_LOCKS, IDXS, and INDEX\_PARAMS.

이름	행	크기	생성됨	업데이트됨	엔진	코멘트	유형
AUX_TABLE	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
BUCKETING_COLUMNS	0	32.0 KiB	2021-01-12 17:0...		InnoDB		Table
CDS	3	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
COLUMNS_V2	11	32.0 KiB	2021-01-12 17:0...		InnoDB		Table
COMPACTION_STATISTICS_TABLE	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
COMPLETED_STATISTICS_TABLE	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
COMPLETED_STATISTICS_TABLE	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
DATABASE_PRIVILEGE_TABLE	0	32.0 KiB	2021-01-12 17:0...		InnoDB		Table
DBS	1	32.0 KiB	2021-01-12 17:0...		InnoDB		Table
DB_PRIVS	0	48.0 KiB	2021-01-12 17:0...		InnoDB		Table
DELEGATION_TABLE	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
FUNCS	0	48.0 KiB	2021-01-12 17:0...		InnoDB		Table
FUNC_RU	0	16.0 KiB	2021-01-12 17:0...		InnoDB		Table
GLOBAL_PRIVS	1	32.0 KiB	2021-01-12 17:0...		InnoDB		Table
HIVE_LOCKS	0	48.0 KiB	2021-01-12 17:0...		InnoDB		Table
IDXS	0	80.0 KiB	2021-01-12 17:0...		InnoDB		Table
INDEX_PARAMS	0	32.0 KiB	2021-01-12 17:0...		InnoDB		Table

[그림 11] MySQL 에서 데이터 베이스 확인

⑤ Hive 용 HDFS 디렉토리를 생성한다.

```
#hdfs dfs -mkdir /hive
#hdfs dfs -mkdir /hive/warehouse
```

⑥ Hive 설정 template 파일을 복사한다 (Namenode 실행)

```
#cd /home/bigdata/hive/conf
#cp hive-env.sh.template hive-env.sh
#cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties
#cp hive-log4j2.properties.template hive-log4j2.properties
#cp hive-default.xml.template hive-site.xml
```

⑦ hive-env.sh 설정을 한다 (Namenode 실행)

```
#vi /home/bigdata/hive/conf/hive-env.sh
48라인 주석해제, Hadoop 경로 입력
HADOOP_HOME=/home/bigdata/hadoop
```

⑧ hive-site.xml 설정을 한다. 기존 내용 모두 삭제 후 아래 내용을 입력한다. (Namenode 실행)

```
#vi /home/bigdata/hive/conf/hive-site.xml

<configuration>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/hive/warehouse</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost:3306/hive_metastore_db?createDatabaseIfNotExist=true</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>hive</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>1234</value>
  </property>
</configuration>
```

```
</configuration>
```

### ⑨ metastore 설정을 한다

```
#schematool -initSchema -dbType mysql
마지막 completed 확인 및 직접 metastore에 생성된 테이블 확인
...
Initialization script hive-schema-2.3.0.mysql.sql
Initialization script completed
schemaTool completed
```

### ⑩ hive 실행한다

```
#hive
```

```
[root@Bigdata101 ~]# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/apache-hive-2.3.7-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/home/bigdata/apache-hive-2.3.7-bin/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> █
```

[그림 12] JDBC 드라이버 확인

## Naver 검색어 파일 키워드 집계 쿼리 수행

1. 테이블 데이터를 입력한다. Hive 기본 내부테이블을 생성 한 후 LoadData 를 통해 데이터를 가져온다.

```
hive>CREATE TABLE `Naver_in` (
  > `rank` Int,
  > `keyword` String,
  > `rdate` String
  > )
  > row format delimited
  > fields terminated by ','
  > tblproperties("skip.header.line.count"="1");

hive>LOAD DATA INPATH '/naver/20-07-19/*' OVERWRITE INTO TABLE Naver_in;
```

## 2. 키워드를 집계하는 코드를 입력한 실행한다.

```
hive> SELECT keyword, SUM(1) as total FROM Naver_in
> GROUP BY keyword ORDER BY total DESC;
```

```
hive> SELECT keyword, SUM(1) as total FROM Naver_in
> GROUP BY keyword ORDER BY total DESC;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20210115112748_08514654-c496-4218-b357-e17ab9556325
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1610677352369_0001, Tracking URL = http://Bigdata101:8088/proxy/application_1610677352369_0001/
Kill Command = /home/bigdata/hadoop/bin/hadoop job -kill job_1610677352369_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-01-15 11:28:04,893 Stage-1 map = 0%, reduce = 0%
2021-01-15 11:28:21,530 Stage-1 map = 42%, reduce = 0%, Cumulative CPU 8.14 sec
2021-01-15 11:28:26,670 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 11.43 sec
2021-01-15 11:28:32,896 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 12.16 sec
MapReduce Total cumulative CPU time: 12 seconds 160 msec
Ended Job = job_1610677352369_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1610677352369_0002, Tracking URL = http://Bigdata101:8088/proxy/application_1610677352369_0002/
```

[그림 13] 하이브 명령어 실행 및 작업 화면

진 아 름	1057		
디 스 패 치		995	
현 빈 손 예 진		816	
현 빈	797		
손 예 진	794		
남 궁 민	771		
2020 kbs 연 기 대 상		691	
현 빈 나 이	589		
오 늘 해 뜨 는 시 간	385		
오 늘 일 출 시 간	341		
일 출 시 간	331		
인 하 대	290		
도 경 완	280		
스 친 송	211		
새 해 복 많 이 받 으 세 요		206	
해 듣 이	177		
나 나	168		
곽 진 영	147		
천 호 진	146		
사 면	145		
손 예 진 나 이	144		
양 자 물 리 학	142		
해 뜨 는 시 간	130		
괴 물	127		
시 등	125		
남 궁 민 여 자 친 구	114		
메 리 포 핀 스 리 턴 즈		112	
1월 1일 해 뜨 는 시 간		112	
박 은 빈	103		
김 하 영	102		
영 화 국 도 극 장	101		
2021년 새 해 인 사	99		
인 하 대 불	98		
나 뿐 녀 석 들	97		
인 하 대 학 교	96		
맨 인 블 랙 인 터 내 셔 널		93	
일 출	92		

[그림 14] 하이브 실행 결과 확인